Cinquièmes Journées de l'Intelligence Artificielle Fondamentale

Lyon, 8-10 juin 2011

Lyon 1

INSA LYON

LIRIS

Journées d'Intelligence Artificielle Fondamentale 2011 (IAF'11)

8-10 Juin 2011 - Lyon

Actes

Ces journées sont la principale animation du thème Intelligence Artificielle Fondamentale (thème 1) du GDR Information-Interaction-Intelligence.

Les journées sont composées d'exposés de synthèse et de contributions sélectionnées.

Contributions sélectionnées

- **Information flow in logic programming**
  Ali Awada, Philippe Balbiani, Antoun Yaacoub
- **Notions de dépendances pour les règles existentielles**
  Jean-François Baget, Marie-Laure Mugnier, and Michaël Thomazo
- **Théorie de méta-diagnostic : raisonnement sur les systèmes de diagnostic**
  Nuno Belard, Yannick Pencolé, Michel Combacau
- **Logique possibiliste à intervalles**
  Salem Benferhat, Julien Hué, Sylvain Lagrue, Julien Rossit
- **Caractérisation multiples minimales utilisant les formules booléennes partiellement définies**
  Fabien Chhel, Adrien Goëffon, Frédéric Lardeux, Frédéric Saubion
- **Cohérence de réseaux de contraintes qualitatives spatio-temporelles à partir de décompositions arborescentes**
  Jean-François Condotta
- **Sur le voisinage entre configurations qualitatives spatio-temporelles**
  Dominique D'Almeida et Nicolas Schwind
- **Modification de l'ATMS pour une prise en compte efficace de la justification d'hypothèses**
  Sylvain Dekoker, Amal El Fallah Seghrouchni, Patrick Taillibert
- **Utilisation de la logique des hypothèses pour la modélisation des voies de signalisation dans la cellule**
  Andrei Doncescu, Pierre Siegel
- **De l'organisation hexagonale des concepts de Blanché à l'analyse formelle de concepts et à la théorie des possibilités**
  Didier Dubois, Henri Prade
- **Revisiting Numerical Pattern Mining with Formal Concept Analysis**
  Mehdi Kaytoue, Sergei O. Kuznetsov, Amedeo Napoli
- **Rationalisation de bases de croyances pour la fusion en logique propositionnelle**
  Sébastien Konieczny, Pierre Marquis, Nicolas Schwind
- **Utilisation de la programmation par ensembles réponses (Answer Set Programming) sur de "petits" problèmes**
  Yves Moinard
- **Reconstruction de la sémantique d'admissibilité en défauts et programmation logique**
  Farid Nouioua, Vincent Risch
- **Vers une version possibiliste de la théorie des fonctions de croyance**
  Henri Prade, Agnès Rico

Exposés de synthèse

- **Théorie de la décision algorithmique et optimisation fondée sur les préférences**
  Patrice Perny
- **Formalisation de l'argumentation - quelques points de repères**
  Leila Amgoud, Philippe Besnard
- **Raisonner en présence d'ontologies**
  Jean-François Baget, Meghyn Bienvenu
- **Interrogation, induction et déduction automatiques pour le raisonnement non monotone - Hommage à Pascal Nicolas**
  Laurent Garcia
- **Raisonnement spatial**
  Jean-François Condotta, Philippe Balbiani

Comité de Programme

- Laurence Cholvy (ONERA, Toulouse)
- Marie-Odile Cordier (IRISA, Rennes)
- Tiago De Lima (CRIL, Lens)
- Laurent Garcia (LERIA, Angers)
- Philippe Jégou (LSIS, Marseille)
- Sébastien Konieczny (CRIL, Lens)
- Jérôme Lang (LAMSADE, Paris)
- Pierre Marquis (CRIL, Lens)
- Nicolas Maudet (LAMSADE, Paris)
- Marie Laure Mugnier (LIRMM, Montpellier)
- Amedeo Napoli (LORIA, Nancy)
- Odile Papini (LSIS, Marseille)
- Henri Prade (IRIT, Toulouse)
- Pierre Siegel (LSIS, Marseille)
- Laurent Simon (LRI, Paris)
- Vincent Vidal (ONERA, Toulouse)
- Christel Vrain (LIFO, Orleans)

Information flow in logic programming

Ali Awada[1], Philippe Balbiani[2], Antoun Yaacoub[2]

[1] Lebanese University

[2] Institut de Recherche en Informatique de Toulouse

Abstract : This paper proposes a theoretical foundation of what could be an information flow in logic programming. Several information flow definitions (based on success/failure, substitution answers, bisimulation between resolution trees of goals) are stated and compared. Decision procedures are given for each definition and complexity is studied for specific classes of logic programs.

Keywords : Logic programming, Information flow, Computational complexity

1 Introduction

Data security is the science and study of methods of protecting data in computer and communication systems from unauthorized disclosure and modification. One of the aspects of data security is the control of information flow in the system. In some sense, an information flow should describe controls that regulate the dissemination of information. These controls are needed to prevent programs from leaking confidential data, or from disseminating classified data to users with lower security clearances. The theory of information flow is well defined for imperative programming. Different models of information flow were proposed, namely, the Bell-LaPadula Model (Bell & LaPadula, 1973), nonlattice and nontransitive models (Foley, 1989; Denning, 1976) of information flow, and nondeducibility and noninterference (Goguen & Meseguer, 1982). Each model has rules about the conditions under which information can move throughout the system. For example, in the Bell-LaPadula Model which describes a lattice-based information flow policy, information can flow from an object in security level A to a subject in security level B if and only if B dominates A. Both compile-time mechanisms (Denning & Denning, 1977) and runtime mechanisms (Fenton, 1974) supporting the checking of information flows were also proposed.

Intuitively, information flows from an object x to an object y if the application of a sequence of commands causes the information initially in x to affect the information in y. For example, the sequence $tmp := x; y := tmp;$ has information flowing from x to y because the value of x at the beginning of the sequence is revealed when the value of y is determined at the end of the sequence. Several studies (Denning, 1982) addressed information flow in imperative programming, but none were concerned to bring answers of what could be an information flow in security systems for logic programming. In fact, logic programming is a well-known declarative method of knowledge representation and programming based on the idea that the language of first-order logic is

well-suited for both representing data and describing desired outputs. Logic programming was developed in the early 1970s based on work in automated theorem proving, in particular, on Robinson's resolution principle.

In this paper, we propose three definitions of information flows in logic programs. These definitions correspond to what can be observed by the user when a query $\leftarrow G(x, y)$ is run on a logic program P. In section 2 of this paper, we will present some basic notions about logic programming, syntax and semantics. In section 3, several definitions of information flow in logic programming are proposed relatively for a logic program P and a goal $\leftarrow G(x, y)$ of arity 2, (which stipulates the existence of a flow that passes from the variable x to the variable y in the goal $\leftarrow G(x, y)$). The implications between these definitions are then studied. Decision procedures are then given in section 4 for each of the previous definitions and computational issues studied for some types of logic programs.

2 Syntax and semantics

In this section, we introduce basic concepts of logic programming. See (Lloyd, 1984; Baral & Gelfond, 1994) for more details. In this article, we will use $p, q, \cdots$ for predicate symbols, $x, y, z, \cdots$ for variables, $f, g, h, \cdots$ for function symbols, and $a, b, c, \cdots$ for constants.The language L considered here is essentially that of first order predicate logic. It has countable sets of variables, function symbols and predicate symbols, these sets being mutually disjoint. Each function and predicate symbol is associated with a unique natural number called its *arity* ; a (function or predicate) symbol whose arity is n is said to be an *n-ary* symbol. A *0-ary* function symbol is referred to as a constant. A *term* is a variable, a constant, or a compound term $f(t_1, \cdots, t_n)$ where f is an *n-ary* function symbol and the t_i are terms, $1 \leq i \leq n$. A term is *ground* if no variable occurs in it. The *Herbrand universe* of L, denoted U_L, is the set of all ground terms that can be formed with the functions and constants in L. An *atom* is of the form $p(t_1, \cdots, t_n)$, where p is an *n-ary* predicate symbol and the t_i are terms, $1 \leq i \leq n$. An atom is *ground* if all t_i are ground. The *Herbrand base* of a language L, denoted B_L, is the set of all ground atoms that can be formed with predicates from L and terms from U_L. A *clause* is an expression of the form $A \leftarrow B_1, \cdots, B_n$ where $A, B_1, \cdots, B_n$ are atoms. A is called the head of the clause and $B_1, \cdots, B_n$ is called its body. A goal is an expression of the form $\leftarrow B_1, \cdots, B_n$. A clause r of the form $A \leftarrow$ (i.e., whose body is empty) is called a fact, and if A is a ground atom, then r is called a ground fact. The empty goal is denoted $\square$. A *predicate definition* is assumed to consist of a finite set (possibly ordered) of clauses defining the same predicate. A *logic program* consists of a finite set of predicate definitions. With each logic program P, we associate the language $L(P)$ that consists of the predicates, functions, and constants occurring in P. If no constant occurs in P, we add some constant to $L(P)$ to have a nonempty domain. A *substitution* is an idempotent mapping from a finite set of variables to terms. The identity substitution will be denoted ϵ. A substitution σ_1 is said to be more general than a substitution σ_2 if there is a substitution θ such that $\sigma_2 = \theta\sigma_1$. Two terms t_1 and t_2 are said to be *unifiable* if there exists a substitution σ such that $\sigma(t_1) = \sigma(t_2)$; in this case σ is said to be a unifier for the terms. If two terms t_1 and t_2 have a unifier, then they have a *most*

general unifier $mgu(t_1, t_2)$ that is unique up to variable renaming. In this paper, we will be interested in

- Datalog programs, i.e. logic programs without function symbols and where each variable appearing in the head of the clause, must also appear in its body.
- Binary programs, i.e. logic programs such that, the body of every program statement is composed of at most one atom.
- Hierarchical programs, i.e. logic program having a level mapping such that, in every program statement $A(t_1, \cdots, t_n) \leftarrow B$, the level of every predicate symbol in B is less than the level of A.

Note that the level mapping of a program is a mapping from its set of predicate symbols to the non-negative integers. We refer to the value of the predicate symbol under this mapping as the level of that predicate symbol. The operational behavior of logic programs can be described by means of *SLD-derivations*. An SLD-derivation for a goal $G = \leftarrow A_1, \cdots, A_n$ with respect to a program P is a sequence of goals $G_0, \cdots, G_i, G_{i+1}, \cdots$, such that $G_0 = G$, and if $G_i = \leftarrow B_1, \cdots, B_m$, then $G_{i+1} = \leftarrow \theta B_1, \cdots, \theta B_{i-1}, \theta B'_1, \cdots, \theta B'_k, \theta B_{i+1}, \cdots, \theta B_m$ such that $1 \leq i \leq m$; $B \leftarrow B'_1, \cdots, B'_k$ is a variant of a clause in P that has no variable in common with any of the goals $G_0, \cdots, G_i$; and $\theta = mgu(B_i; B)$. The goal G_{i+1} is said to be obtained from G_i by means of *resolution step*, and B_i is said to be the *resolved atom*. Let $G_0, \cdots, G_n$ be an SLD-derivation for a goal G with respect to a program P, and let θ_i be the unifier obtained when resolving the goal G_{i-1} to obtain G_i, $1 \leq i \leq n$. If this derivation is finite and maximal, i.e., one in which it is not possible to resolve the goal G_n with any of the clauses in P, then it corresponds to a terminating computation for G : in this case, if G_n is the empty goal then we say that $P?G$ succeeds and the computation is said to succeed with answer substitution θ, where θ is the substitution obtained by restricting the substitution $\theta_n \cdots \theta_1$ to the variables occurring in G; if G_n is not the empty goal, then the computation is said to fail. We say that $P?G$ fails if all computations from G in P fail. If the derivation is infinite, the computation does not terminate. Given a program P and a goal G, let $\Theta(P?G)$ be the set of all answer substitutions of G in P.

3 Information flow

As the theory of information flow is well studied for imperative programming, it is tempting to see what could be an information flow in logic programming, especially given the fact that there are no notions of assignment, or variable of a program. In fact, variables in logic programs behave differently from variables in conventional programming languages. They stand for an unspecified but single entity rather than for a store location in memory. The following three definitions for information flow in logic programming are based on the following principle. The information flow that occurs when the user ask a goal to logic programs depends mainly on what parts of the computation the user sees. In the first definition, the user only sees whether goals succeed or fail. In the second definition, the user has access to the set of substitution answers computed by the program. In the third definition, the user obtains the shape of the computation trees produced by the program.

3.1 Successes and failures

Let P be a logic program, and $G(x, y)$ be a two variables goal. We shall say that there is a flow from x to y in $G(x, y)$ with respect to successes and failures in P (in symbols $x \xrightarrow{SF}{}^{P}_{G} y$) iff there exists $a, b \in U_{L(P)}$ such that $P?G(a, y)$ succeeds and $P?G(b, y)$ fails.
Example 1 Let P_1 be the following program : $p(a, b) \leftarrow$ and let $G_1(x, y)$ be the following goal : $\leftarrow p(x, y)$. Since $P_1?G_1(a, y)$ succeeds and $P_1?G_1(b, y)$ fails, then $x \xrightarrow{SF}{}^{P_1}_{G_1} y$.

3.2 Substitution answers

Let P be a logic program, and $G(x, y)$ be a two variables goal. We shall say that there is a flow from x to y in $G(x, y)$ with respect to substitution answers in P (in symbols $x \xrightarrow{SA}{}^{P}_{G} y$) iff there exists $a, b \in U_{L(P)}$ such that $\Theta(P?G(a, y)) \neq \Theta(P?G(b, y))$.
Example 2 Let P_2 be the following program : $p(a, y) \leftarrow$, and let $G_2(x, y)$ be the following goal : $\leftarrow p(x, y)$. Since $\Theta(P_2?G_2(a, y)) = \{\epsilon\}$ and $\Theta(P_2?G_2(b, y)) = \emptyset$, then $x \xrightarrow{SA}{}^{P_2}_{G_2} y$.

3.3 Bisimulation

Our third definition of flow is based on the notion of bisimulation between goals. Let P be a logic program and $\mathcal{Z}$ be a binary relation between goals. We shall say that $\mathcal{Z}$ is a $P - bisimulation$ iff for all goals G, H, if $G\mathcal{Z}H$ then :
- for all goals $G' \in succ_P(G)$, there exists $H' \in succ_P(H)$, such that $G'\mathcal{Z}H'$.
- for all goals $H' \in succ_P(H)$, there exists $G' \in succ_P(G)$, such that $G'\mathcal{Z}H'$.
- $G = \square$ iff $H = \square$.

Above, $succ_P(G)$ denotes the set of all goals obtained from a goal G by means of a resolution step in the program P. Obviously,

Lemma 1 The relation identity Id between goals is a $P - bisimulation$.
Lemma 2 If $\mathcal{Z}$ is a $P - bisimulation$, then $\mathcal{Z}^{-1}$ is also a $P - bisimulation$.
Lemma 3 If $\mathcal{Z}_1$, $\mathcal{Z}_2$ are two $P - bisimulations$, then the composition $\mathcal{Z}_1\mathcal{Z}_2$ defined by $\mathcal{Z}_1\mathcal{Z}_2 = \{(G, H)/\exists I, G\mathcal{Z}_1 I \text{ and } I\mathcal{Z}_2 H\}$ is also a $P - bisimulation$.
Lemma 4 Let $(\mathcal{Z}_i)_{i\in I}$ be a family of $P - bisimulations$, then $\bigcup_{i\in I} \mathcal{Z}_i$ is also a $P - bisimulation$.

By lemma 4, there exists a maximal $P - bisimulation$, denoted $\mathcal{Z}_{max}$.

Example 3 Let P be the following program formed of :
$p(a, y) \leftarrow q(y)$, $p(b, y) \leftarrow r(y)$ and $p(b, y) \leftarrow s(y)$
and let G, H be respectively the following goals $\leftarrow p(a, y)$ and $\leftarrow p(b, y)$
Let Z be the binary relation between goals such that :
- $\leftarrow p(a, y)\ Z \leftarrow p(b, y)$

– $\leftarrow q(y)\ Z \leftarrow r(y)$
– $\leftarrow q(y)\ Z \leftarrow s(y)$

Obviously, Z is a $P - bisimulation$. Since $G\ Z\ H$, then $G\ \mathcal{Z}_{max}\ H$.
By lemmas 1, 2 and 3, we have
Lemma 5 $\mathcal{Z}_{max}$ is an equivalence relation.

Let P be a logic program, and $G(x, y)$ be a two variables goal. We shall say that there is a flow from x to y in $G(x, y)$ with respect to the bisimulation in P (in symbols $x \xrightarrow{BI}{}^{P}_{G}\ y$) iff there exists $a, b \in U_{L(P)}$ such that *not* $G(a, y)\mathcal{Z}_{max}G(b, y)$.

Example 4 Let P_3 be the following program : $p(x, a) \leftarrow$ and $p(a, b) \leftarrow q(a)$ and let $G_3(x, y)$ be the goal : $\leftarrow p(x, y)$. Let us prove *not* $\leftarrow p(a, y)\mathcal{Z}_{max} \leftarrow p(b, y)$. Suppose that $\leftarrow p(a, y)\mathcal{Z}_{max} \leftarrow p(b, y)$. Since $\leftarrow q(a) \in succ_{P_3}(\leftarrow p(a, y))$, then there should be $G_3 \in succ_{P_3}(\leftarrow p(b, y))$ such that $\leftarrow q(a)\mathcal{Z}_{max}G_3$. The problem is that the only goal in $succ_{P_3}(\leftarrow p(b, y))$ is the empty goal, which cannot be bisimilar to $\leftarrow q(a)$. Hence, *not* $\leftarrow p(a, y)\mathcal{Z}_{max} \leftarrow p(b, y)$. Therefore $x \xrightarrow{BI}{}^{P_3}_{G_3}\ y$.

3.4 Links between the definitions of information flow

The existence of a flow with respect to substitution answers does not entail the existence of a flow with respect to successes and failures. To see this, it suffices to consider the following example.
Example 5 Let P be the following program : $p(a, b) \leftarrow$ and $p(Z, c) \leftarrow$ and let $G(x, y)$ be the goal : $\leftarrow p(x, y)$. Since $\Theta(P?G(a, y)) = \{y/b, y/c\}$ and $\Theta(P?G(b, y)) = \{y/c\}$, then $x \xrightarrow{SA}{}^{P}_{G}\ y$. Since $P?G(a, y)$ and $P?G(b, y)$ both succeed, then $x \overset{SF}{\not\longrightarrow}{}^{P}_{G}\ y$.

However, one can establish the following result.

Lemma 6 Let P be a logic program and $G(x, y)$ be a two variables goal. If $x \xrightarrow{SF}{}^{P}_{G}\ y$ then $x \xrightarrow{SA}{}^{P}_{G}\ y$.
Proof 1 Suppose that $x \xrightarrow{SF}{}^{P}_{G}\ y$, then there exists $a, b \in U_{L(P)}$ such that $P?G(a, y)$ succeeds and $P?G(b, y)$ fails. Therefore, $\Theta(P?G(a, y)) \neq \emptyset$ and $\Theta(P?G(b, y)) = \emptyset$. Consequently, $x \xrightarrow{SA}{}^{P}_{G}\ y$. □

The existence of a flow with respect to bisimulation does not entail the existence of a flow with respect to successes and failures. The next example explains why.
Example 6 Let P_3 and G_3 be the program and goal considered in example 4. We know that $x \xrightarrow{BI}{}^{P_3}_{G_3}\ y$. Nevertheless, since all the goals of the form $G_3(a, y)$, with $a \in U_{L(P)}$, succeed, thus $x \overset{SF}{\not\longrightarrow}{}^{P_3}_{G_3}\ y$.

Nevertheless, it is worth noting at this point the following.

Lemma 7 Let P be a logic program and $G(x, y)$ be a two variables goal. If $x \xrightarrow{SF}{}^{P}_{G} y$ then $x \xrightarrow{BI}{}^{P}_{G} y$.

Proof 2 Suppose that $x \xrightarrow{SF}{}^{P}_{G} y$. Thus, there exists $a, b \in U_{L(P)}$ such that $P?G(a, y)$ succeeds and $P?G(b, y)$ fails. Suppose that $G(a, y)\mathcal{Z}_{max}G(b, y)$. Since $P?G(a, y)$ succeeds, then there exists an SLD-refutation $G_0, \cdots, G_n$ of $G(a, y)$ in P. That is to say, $G_0 = G(a, y)$, $G_n = \square$ and G_i is a successor of G_{i-1} in P for $i = 1 \cdots n$.
Since $G(a, y)\mathcal{Z}_{max}G(b, y)$ in P, thus $P?G(b, y)$ succeeds : a contradiction. Thus, *not* $G(a, y)\mathcal{Z}_{max}G(b, y)$ and $x \xrightarrow{BI}{}^{P}_{G} y$. $\square$

4 Decidability / Complexity

We now study the computational complexity of the following decision problems :

$$\pi_{SF} \begin{cases} \text{Input : A logic program } P\text{, a two variables goal } G(x, y) \\ \text{Output : Determine whether } x \xrightarrow{SF}{}^{P}_{G} y \end{cases}$$

$$\pi_{SA} \begin{cases} \text{Input : A logic program } P\text{, a two variables goal } G(x, y) \\ \text{Output : Determine whether } x \xrightarrow{SA}{}^{P}_{G} y \end{cases}$$

$$\pi_{BI} \begin{cases} \text{Input : A logic program } P\text{, a two variables goal } G(x, y) \\ \text{Output : Determine whether } x \xrightarrow{BI}{}^{P}_{G} y \end{cases}$$

4.1 Undecidability

In the general setting, our decision problems are undecidable.

Proposition 1 The three decision problems above are undecidable.

Proof 3 (π_{SF}) We will reduce the following undecidable decision problem π_1 (Devienne *et al.*, 1996) to π_{SF} :

$$\pi_1 \begin{cases} \text{Input : A logic program } P\text{, a ground goal } q(a) \\ \text{Output : } P?q(a) \text{ succeeds} \end{cases}$$

Let $(P, q(a))$ be an instance of π_1 and let $(P', G(x, y))$ be the instance of π_{SF} defined by : $P' = P \cup \{G(a, y) \leftarrow q(a)\}$, where G is a new predicate symbol of arity 2. We need to show that, $P?q(a)$ succeeds iff $x \xrightarrow{SF}{}^{P'}_{G} y$.
($\Rightarrow$) Suppose that $P?q(a)$ succeeds. Thus $P'?G(a, y)$ succeeds and $P'?G(b, y)$ fails, consequently $x \xrightarrow{SF}{}^{P'}_{G} y$.
($\Leftarrow$) Suppose that $x \xrightarrow{SF}{}^{P'}_{G} y$, then there exists $a', b' \in U_{L(P)}$ such that $P'?G(a', y)$ succeeds and $P'?G(b', y)$ fails. Thus, $a' = a$ and $b' \neq a$. Thus, $P?q(a)$ succeeds.

(π_{SA}) A similar proof applies here.
(π_{BI}) We will reduce the following undecidable decision problem (Devienne *et al.*, 1993) to π_{BI} :

$$\pi_2 \begin{cases} \text{Input : A binary logic program } P\text{, a ground goal } q(a) \\ \text{Output : The SLD-tree of } P?q(a) \text{ contains a failure branch} \end{cases}$$

Let $(P, q(a))$ be an instance of π_2 and let $(P', G(x, y))$ be the instance of π_{BI} defined by :

$$P' = P \cup \begin{cases} G(a, y) \leftarrow q(a) \\ G(b, y) \leftarrow G(b, y) \text{ for all } b \text{ in } L(P) \text{ such that } a \neq b \\ G(f(x_1, \cdots, x_n), y) \leftarrow G(f(x_1, \cdots, x_n), y) \text{ for all } f \text{ in } L(P) \end{cases}$$

Remark that for all $a' \in U_{L(P)}$, the computation tree of $P'?G(a', y)$ consists of a unique infinite branch. We need to show that the SLD-tree of $P?q(a)$ contains a failure branch iff $x \xrightarrow{BI}{}^{P'}_{G} y$.
($\Rightarrow$) Suppose that the SLD-tree of $P?q(a)$ contains a failure branch. Thus the SLD-tree of $P'?G(a, y)$ will eventually contains this failure branch while the SLD-tree of $P'?G(b, y)$ will have infinite branche(s). Consequently $x \xrightarrow{BI}{}^{P'}_{G} y$.
($\Leftarrow$) Suppose that $x \xrightarrow{BI}{}^{P'}_{G} y$, then there exists $a', b' \in U_{L(P)}$ such that *not* $P'?G(a', y)\mathcal{Z}_{max}P'?G(b', y)$. Hence, either a' or b' is equal to a. Thus (in the case of $a' = a$) the SLD-tree of $P?q(a)$ contains a failure branch. □

4.2 Decidability

If one restricts the language to Datalog programs and goals then determining existence of information flows becomes decidable.

Proposition 2 π_{SF} is EXPTIME-complete for Datalog programs.
Proof 4 (*Membership*) The following algorithm decides the existence of the information flow in Datalog programs.

Require: A Datalog program P, a goal $\leftarrow G(x, y)$, finite Herbrand Universe $U_{L(P)} = \{a_1, \cdots, a_n\}$
Ensure: $x \xrightarrow{SF}{}^{P}_{G(x,y)} y$ for the Datalog program P and the goal g

```
answer = false ; i = 0 ;
while i < n and not answer do
    i = i + 1 ; j = i ;
    while j < n and not answer do
        j = j + 1 ;
        if (p?G(a_i, y) succeeds and p?G(a_j, y) fails) or (p?G(a_i, y) fails and
        p?G(a_j, y) succeeds) then
            answer = true ;
        end if
```

```
    end while
end while
return answer ;
```

This algorithm is deterministic and using the fact that Datalog is program complete for EXPTIME (Vardi, 1982; Immerman, 1986), it follows that it can be executed in EXPTIME.
(*Hardness*) In order to prove EXPTIME-hardness, we consider the following decision problem known to be EXPTIME-hard (Vardi, 1982) :

$$\pi_3 \begin{cases} \text{Input : A Datalog program } P\text{, a ground atom } A \\ \text{Output : } P?A \text{ (} A \text{ is a logical consequence of } P\text{)} \end{cases}$$

Let (P, A) an instance of π_3 and let $(P', g(x, y))$ be the instance of π_{SF} defined by $P' = P \cup \{g(a, y) \leftarrow A\}$, where g is a new predicate symbol. Thus $P?A$ iff $x \xrightarrow{SF}{}^{P'}_{g} y$.
($\Rightarrow$) Suppose that A is a logical consequence of P, thus $P'?g(a, y)$ succeeds and $P'?g(b, y)$ fails. Consequently $x \xrightarrow{SF}{}^{P'}_{g} y$.
($\Leftarrow$) Suppose that $x \xrightarrow{SF}{}^{P'}_{g} y$. Then there exists $a', b' \in U_{L(P)}$ such that $P'?g(a', y)$ succeeds and $P'?g(b', y)$ fails. Hence, it follows that $a' = a$ and $b' \neq a$. Thus, $P?A$. □

Proposition 3 π_{SA} is EXPTIME-complete for Datalog programs.
Proof 5 A proof similar to the previous one applies here. □

Concerning π_{SF}, determining existence of flows is even in $\Sigma_2 P$ if one considers binary hierarchical Datalog programs.

Proposition 4 π_{SF} is in $\Sigma_2 P$ for binary hierarchical Datalog programs.
Proof 6 Let us consider the following nondeterministic algorithm with oracle :
Procedure SF(P,$G(x, y)$**)**
Require: A binary hierarchical Datalog program P, a goal $G(x, y)$.
Ensure: $x \xrightarrow{SF}{}^{P}_{G} y$

```
choose a, b in U_{L(P)}
if (P?G(a, y) ∈ SUCCESSES and P?G(b, y) ∈ FAILURES) then
    Accept
else
    Reject
end if
```

The oracle SUCCESSES consists in the set of all pairs (P, G) such that G succeeds in P. Restricting P to binary hierarchical programs, one can show that SUCCESSES belongs to NP. The oracle FAILURES consists in the set of all pairs (P, G) such that G fails in P. Restricting P to binary hierarchical programs, one can show that FAILURES belongs to co-NP. Hence π_{SF} is in $\Sigma_2 P$. □

At the time of writing, we do not know if π_{SA} is in $\Sigma_2 P$ too for binary hierarchical programs. Now, let us address the complexity of deciding the existence of flows with

TABLE 1 – Complexity results

	General setting	Datalog programs	Binary hierarchical Datalog programs
π_{SF}	Undecidable	EXPTIME-complete	in $\Sigma_2 P$
π_{SA}	Undecidable	EXPTIME-complete	in EXPTIME
π_{BI}	Undecidable	?	in EXPTIME

respect to our third definition.

Proposition 5 π_{BI} is in EXPTIME for hierarchical binary Datalog programs.
Proof 7 Since $EXPTIME = APSPACE$, then it suffices to demonstrate that π_{BI} is in $APSPACE$ for binary hierarchical Datalog programs. In this respect, we consider the following alternating algorithm :
Procedure bisim(P,G_1,G_2)
Require: A hierarchical Datalog program P, two goals G_1 and G_2.
Ensure: Deciding whether $G_1 \mathcal{Z}_{max} G_2$

case $(succ(P, G_1), succ(P, G_2))$
- $(true, true)$ **:**
  (∀) choose $i, j \in \{1, 2\}$ such that $i \neq j$
  (∀) choose a successor G'_i of G_i in P
  (∃) choose a successor G'_j of G_j in P
  (.) call **bisim**(P, G'_i, G'_j)
- $(true, false)$ **:** reject
- $(false, true)$ **:** reject
- $(false, false)$ **:**
  if $(G_1 = \square$ iff $G_2 = \square)$ **then**
    accept
  else
    reject
  end if
endcase

The subprocedure $succ(.,.)$ produces, given a program P and a goal G a Boolean value. More precisely, $succ(P, G)$ is true iff there exists a goal G' such that G' is derived from G and P. Obviously, $succ(.,.)$ can be implemented in deterministic linear time. Concerning the procedure **bisim**, seeing that P is hierarchical, it accepts its inputs P, G_1, G_2 iff $G_1 \mathcal{Z}_{max} G_2$. Moreover, seeing that P is binary, **bisim** can be implemented in polynomial space. □

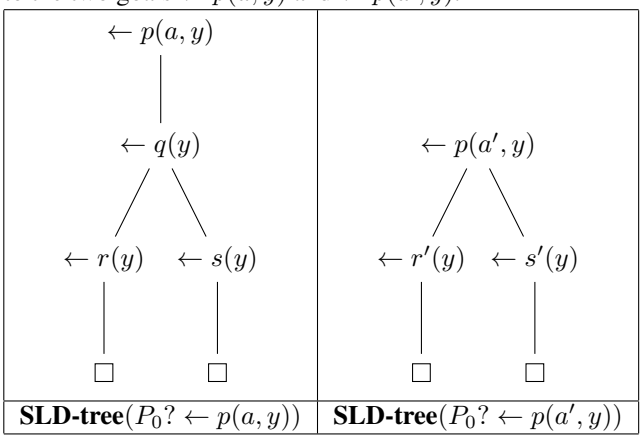
5 Conclusion

In this paper, we have proposed three definitions of information flow in logic programs. As proved in section 4.1, determining whether there exists an information flow is undecidable in the general setting. Hence, a natural question was to restrict the language of logic programming as done in section 4.2. Table 1 contains the results we

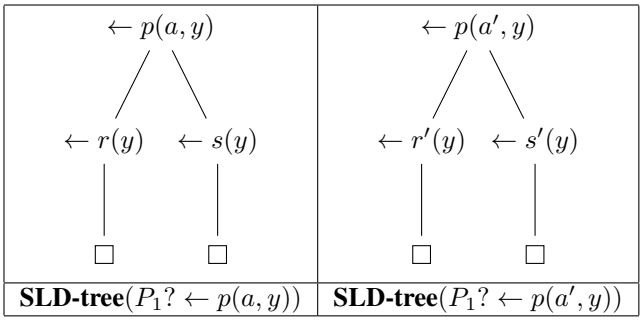
have obtained so far. Much remains to be done. Firstly, in the setting of Datalog programs, the main difficulty concerning π_{BI} comes from loops or infinite branches in SLD-refutation trees. Therefore, in order to determine, given a Datalog program P and two Datalog goals G_1 and G_2, whether $G_1 \mathcal{Z}_{max} G_2$, one can think about using loop checking techniques and considering either restricted programs, or nvi programs or svo programs. See (Bol, 1995) and (Bol *et al.*, 1991) for details. Secondly, considering the unfold/fold transformations introduced by Tamaki and Sato (Tamaki & Sato, 1984) within the context of logic programs optimization, one can ask whether these transformations introduce or eliminate information flows. Obviously, since folding or unfolding clauses in logic programs change neither its successes, nor its failures (Tamaki & Sato, 1984), nor its substitution answers (Kawamura & Kanamori, 1990), the information flows based either on successes and failures or on substitution answers are preserved after applying the transformations of Tamaki and Sato. The same cannot be said for information flows based on bisimulation. For example, let P_0 be the logic program containing the following clauses :

$C_1 : p(a, y) \leftarrow q(y)$	$C_4 : r(y) \leftarrow$	$C_7 : p(a', y) \leftarrow s'(y)$
$C_2 : q(y) \leftarrow r(y)$	$C_5 : s(y) \leftarrow$	$C_8 : r'(y) \leftarrow$
$C_3 : q(y) \leftarrow s(y)$	$C_6 : p(a', y) \leftarrow r'(y)$	$C_9 : s'(y) \leftarrow$

and let G be the goal $\leftarrow p(x, y)$. It is easy to verify that $x \xrightarrow{BI}{}^{P_0}_{G} y$. To see this, we sketch, by omitting the different substitutions, the SLD-refutation trees corresponding to the two goals $\leftarrow p(a, y)$ and $\leftarrow p(a', y)$.



Obviously, as *not* $\leftarrow p(a, y) \mathcal{Z}_{max} \leftarrow p(a', y)$, $x \xrightarrow{BI}{}^{P_0}_{G} y$. By unfolding C_1, the program P_1 is obtained from P_0 by replacing C_1 with the following clauses :
$C_{10} : p(a, y) \leftarrow r(y) \quad C_{11} : p(a, y) \leftarrow s(y)$
In the new transformed program P_1, the two SLD-refutation trees of the goals $\leftarrow p(a, y)$ and $\leftarrow p(a', y)$ are bisimilar as shown in the figure on the next page. Thus $x \not\xrightarrow{BI}{}^{P_1}_{G} y$. Hence, a general question concerns the definition of transformations of logic programs that never introduce or eliminate information flows.



Références

BARAL C. & GELFOND M. (1994). Logic programming and knowledge representation. *Journal of Logic Programming*, **19-20**, 73–148.

BELL D. & LAPADULA L. (1973). Secure computer systems : Mathematical foundations and model. *The MITRE Corporation Bedford MA Technical Report M74244*, **1**(M74-244), 42.

BOL R. (1995). *Loop Checking in Logic Programming*. Amsterdam, the Netherlands : Dissertation of the CWI (Centre for Mathematics and Computer Science).

BOL R., APT K. & KLOP J. (1991). An analysis of loop checking mechanisms for logic programs. *Theoretical Computer Science*, **86**, 35–79.

DENNING D. (1976). A lattice model of secure information flow. *Commun. ACM*, **19**, 236–243.

DENNING D. (1982). *Cryptography and Data Security*. Addison-Wesley.

DENNING D. & DENNING P. (1977). Certification of programs for secure information flow. *Commun. ACM*, **20**, 504–513.

DEVIENNE P., LEBÈGUE P., PARRAIN A., ROUTIER J.-C. & WÜRTZ J. (1996). Smallest Horn clause programs. *The Journal of Logic programming*, **27**, 227–267.

DEVIENNE P., LEBÈGUE P. & ROUTIER J.-C. (1993). Halting problem of one binary Horn clause is undecidable. *STACS 93*, **665**, 48–57.

FENTON J. (1974). Memoryless subsystems. *The Computer Journal*, **17**, 143–147.

FOLEY S. (1989). A model for secure information flow. In *Security and Privacy, 1989. Proceedings., 1989 IEEE Symposium on*, p. 248–258 : IEEE Computer Society.

GOGUEN J. & MESEGUER J. (1982). Security policies and security models. *Security and Privacy, IEEE Symposium on*, **0**, 11.

IMMERMAN N. (1986). Relational queries computable in polynomial time. *Information and Control*, **68**, 86–104.

KAWAMURA T. & KANAMORI T. (1990). Preservation of stronger equivalence in unfold/fold logic program transformation. *Theoretical Computer Science*, **75**, 139–156.

LLOYD J. (1984). *Foundations of logic programming*. Springer-Verlag.

TAMAKI H. & SATO T. (1984). Unfold/fold transformations of logic programs. In *In S.-Å. Tärnlund, editor, Proceedings of The Second International Conference on Logic Programming*, p. 127–139.

VARDI M. Y. (1982). The complexity of relational query languages (extended abstract). In *Proceedings of the fourteenth annual ACM symposium on Theory of computing*, STOC '82, p. 137–146 : ACM.

Notions de dépendances pour les règles existentielles

Jean-François Baget, Marie-Laure Mugnier, and Michaël Thomazo

Université Montpellier 2
LIRMM
INRIA - GraphIK

Résumé : Nous nous intéressons à un problème de décision consistant à déterminer s'il existe une réponse à une requête conjonctive dans une base de connaissances composée d'une base de données et d'une ontologie décrite par un ensemble de règles contenant des variables existentielles en conclusion, équivalentes aux Tuple-Generating Dependencies (TGD) en bases de données. Ce problème indécidable impacte de nombreux domaines (logiques de description, bases de données, web sémantique...). Dans Baget (2004) et Baget *et al.* (2009) il a été proposé une notion de dépendance entre règles pour définir de nouveaux cas décidables. Cependant, si cette notion de dépendance est optimale à un instant donné du raisonnement, elle ne prend pas en compte l'historique de celui-ci. Dans cet article, nous proposons un raffinement de cette notion, la k-dépendance, qui tient compte d'un historique de taille bornée.
Mots-clés : Règles existentielles, TGD, Datalog+/-, Dépendance

1 Introduction

Un thème de recherche à la croisée des bases de données et de la représentation des connaissances reçoit actuellement une attention croissante : il s'agit de l'interrogation de bases de connaissances munies d'ontologies expressives par des requêtes conjonctives. Dans ce domaine, le langage Datalog+/- (Calì *et al.* (2009)) code l'ontologie par des contraintes, des EGDs (Equality Generating Dependencies) et des TGDs (Tuple Generating Dependencies). Ce sont ces derniers objets que nous appelons règles existentielles, et que nous étudions sous leur forme logique, qui sont des formules closes du type $\forall x_1 \dots x_p(H \rightarrow (\exists z_1 \dots z_q C))$, où H et C sont des conjonctions d'atomes. Le problème fondamental est celui de la conséquence logique : étant donnée une base de connaissances $K = (F, \mathcal{R})$ et Q, où F et Q sont deux conjonctions d'atomes existentiellement closes, appelés respectivement le fait et la requête, et $\mathcal{R}$ est un ensemble de règles, est-il vrai que $K \models Q$?

Les connaissances de nature ontologique sont traditionnellement représentées par des logiques de description. Cependant, la nécessité de prendre en compte des requêtes conjonctives (et non plus seulement des requêtes réduites à un atome) a profondément modifié le paysage de ces logiques. De nouvelles logiques, moins expressives,

sont étudiées afin de répondre à ce problème avec un meilleur compromis expressivité/complexité.

L'approche présentée ici provient d'un courant parallèle de recherche, où certaines contraintes inhérentes aux logiques de description (arité des prédicats inférieure ou égale à deux, pas de cycle dans les règles) n'ont plus lieu d'être. Cependant, sans contraintes sur les règles, la réponse à des requêtes conjonctives est un problème indécidable. Il est donc important de définir des cas particuliers pour lesquels ce problème devient décidable. La plupart des cas décidables connus reposent sur un processus de marche avant, appelé "chase" en base de données (Maier *et al.* (1979)). A l'aide des règles, on opère une saturation de la base de faits, c'est-à-dire que l'on ajoute des informations aux faits en appliquant les règles, tant que ces informations sont nouvelles. Lorsque ce processus s'arrête (Fagin *et al.* (2005), Baget & Mugnier (2002), Baget (2004), Spezzano & Greco (2010)), la conséquence logique est décidable. Si ce processus ne s'arrête pas, mais génère des faits dont la structure est quasi-arborescente (Calì *et al.* (2008), Calì *et al.* (2009), Baget *et al.* (2010)), le problème est également décidable. Enfin, une autre condition de décidabilité repose sur la finitude d'un mécanisme de marche arrière. Malheureusement, décider si un ensemble de règles vérifie l'un des trois critères sus-cités est un problème indécidable. Une zoologie de classes de règles décidables ayant ces propriétés a donc été proposée.

Cependant, ces classes concrètes ne se comportent pas bien les unes par rapport aux autres. Si $\mathcal{R}_1$ et $\mathcal{R}_2$ sont des classes décidables, il est généralement faux que $\mathcal{R}_1 \cup \mathcal{R}_2$ soit décidable. De manière à pouvoir décomposer un ensemble de règles en sous-ensembles plus simples étudiables de manière (presque) indépendante, une notion de dépendance a été introduite dans Baget (2004) et Baget *et al.* (2009). Intuitivement, une règle R_2 dépend d'une règle R_1 s'il existe un fait F tel qu'une application de R_1 sur F peut permettre une application de R_2 qui n'était pas possible sur F uniquement. Toutefois, si cette notion de dépendance est optimale à un instant donné du raisonnement, elle ne prend pas en compte l'historique de celui-ci. Dans cet article, nous proposons un raffinement de cette notion, la k-dépendance, qui tient compte d'un historique de taille bornée par k.

En section 2, nous formalisons le problème de conséquence logique pour les règles existentielles, et introduisons les principales définitions. En section 3, nous rappelons la définition de dépendance et certaines de ses propriétés. Nous étudions les k-dépendances en section 4. Nous concluons et présentons quelques pistes à explorer en section 5.

2 Préliminaires

On considère un langage logique du premier ordre avec constantes mais sans autre symbole de fonction. Un atome est donc de la forme $p(t_1, \ldots, t_k)$ où p est un prédicat d'arité k et t_i sont des variables ou des constantes. Un fait F est une conjonction d'atomes fermée existentiellement, que l'on verra comme un ensemble d'atomes. Les variables d'une conjonction d'atomes seront notées $\mathrm{var}(F)$ et ses termes $\mathrm{term}(F)$. Une requête conjonctive a la même forme qu'un fait, et l'on identifiera donc les deux notions. Une règle existentielle (et simplement règle par la suite) est une formule close de la forme $\forall x_1 \ldots \forall x_p (H \rightarrow (\exists z_1 \ldots \exists z_q C))$ où H et C sont deux conjonctions non

vides d'atomes appelées respectivement l'hypothèse et la conclusion de la règle R. Par la suite, les quantificateurs seront implicites. Une base de connaissances est un couple composé d'un ensemble de faits et d'un ensemble de règles. L'union de deux faits étant un fait, on considérera l'ensemble des faits comme un seul fait. On note donc $(F, \mathcal{R})$ une base de connaissances, où F est un fait et $\mathcal{R}$ un ensemble de règles. Nous nous intéressons au problème suivant : "étant donnés une base de connaissances $K = (F, \mathcal{R})$ et un fait (aussi appelé requête conjonctive) Q, est-il vrai que $F, \mathcal{R} \models Q$, i.e. que Q est conséquence logique de la conjonction des formules composant F et $\mathcal{R}$?"

Substitution et homomorphisme. Etant donné un ensemble de variables V et un ensemble de termes T, une *substitution* de V par T est une application de V dans T. Soit $\sigma : V \to T$ une substitution, et F un fait. $\sigma(F)$ est le fait obtenu à partir de F en remplaçant chaque occurrence de $x \in V \cap \text{term}(F)$ par $\sigma(x)$. Une substitution est dite *sûre* s'il s'agit d'une bijection entre V et un ensemble de variables *fraîches* (c'est-à-dire n'apparaissant nulle part dans les formules du raisonnement). Etant donnés deux faits F et Q, un homomorphisme π de Q vers F est une substitution de $\text{var}(Q)$ par $\text{term}(F)$ telle que $\pi(Q) \subseteq F$. L'homomorphisme est correct et complet par rapport à la conséquence logique dans le fragment des faits : étant donné deux faits F et Q, $F \models Q$ ssi il y a un homomorphisme de Q dans F.

Pour la facilité de lecture, nous rappelons quelques définitions provenant de Baget *et al.* (2011a).

Définition 1 (Application d'une règle)
Soit F un fait et $R = (H, C)$ une règle. R est dite applicable *à F s'il existe un homomorphisme π de H dans F. Dans ce cas, l'application de R à F selon π produit un fait $\alpha(F, R, \pi) = F \cup \pi^s(C)$, où $\pi^s = \pi$ sur le domaine de π, et est une substitution sûre des autres variables. Les* atomes créés *par l'application de R à F selon π sont les éléments de $\pi^s(C) \backslash F$. Un atome créé par R est dit d'*origine *R.*

Exemple 1
Soit $F = \{r(a, b), r(b, c)\}$ un fait, $R_1 : r(x, y) \wedge r(y, z) \to r(x, z)$ et $R_2 : r(x, y) \to r(y, z)$ deux règles. R_1 s'applique par $\pi(x) = a, \pi(y) = b, \pi(z) = c$. Le fait obtenu est alors $F' = \{r(a, b), r(b, c), r(a, c)\}$. R_2 s'applique à F' de trois manières différentes : par $\pi_1(x) = a, \pi_1(y) = b$, $\pi_2(x) = b, \pi_2(y) = c$ et $\pi_3(x) = a, \pi_3(y) = c$. Si l'on applique R_1 à F' par π_3, on obtient $F'' = \{r(a, b), r(b, c), r(a, c), r(c, x_1)\}$, x_1 étant existentiellement quantifiée.

Nous considérons la saturation d'un fait, qui est un mécanisme de marche avant en largeur. En notant F_0 le fait initial, F_i étant le fait à l'étape i, on obtient F_i à partir de F_{i-1} en calculant tous les homomorphismes d'une hypothèse de règle dans F_{i-1} qui n'ont pas déjà été appliqués, puis en les appliquant.

Définition 2 (Saturé d'ordre k, ordre et origine d'un atome)
Soit F un fait et $\mathcal{R}$ un ensemble de règles existentielles. $\Pi(\mathcal{R}, F)$ désigne l'ensemble des homomorphismes de l'hypothèse d'une règle de $\mathcal{R}$ dans F : $\Pi(\mathcal{R}, F) = \{(R, \pi) | R = (H, C)$ et π est un homomorphisme de H dans $F\}$. La saturation *de F par $\mathcal{R}$ à l'ordre k est notée $F^k = \alpha_k(F, \mathcal{R})$ et est définie récursivement comme suit :*

- $\alpha_0(F, \mathcal{R}) = F$
- $\forall i > 0, F^i = F^{i-1} \cup_{(R=(H,C),\pi)\in\Pi(F^{i-1},\mathcal{R})\setminus\Pi(F^{i-2},\mathcal{R})} \pi^s(C)$, *où par convention,* $F^{-1} = \emptyset$.

*Un atome est dit d'*ordre k *(par rapport à* $F, \mathcal{R}$*) s'il est créé par une application de règle appartenant à* $F^k \setminus F^{k-1}$.

On note $\alpha_\infty(F, \mathcal{R}) = \cup_{k\in\mathbb{N}} \alpha_k(F, \mathcal{R})$.

Théorème 1
Soit F *et* Q *deux faits et* $\mathcal{R}$ *un ensemble de règles.* $F, \mathcal{R} \models Q$ *si et seulement s'il existe* $k \in \mathbb{N}$ *et un homomorphisme de* Q *dans* $\alpha_k(F, \mathcal{R})$.

Définition 3 (Ensemble à expansion finie)
Un ensemble de règles $\mathcal{R}$ *est dit à expansion finie si et seulement s'il existe un entier* k *tel que* $F^k = \alpha_k(F, \mathcal{R}) \equiv \alpha_{k+1}(F, \mathcal{R})$.

Avec un ensemble de règles à expansion finie, le problème de conséquence logique devient décidable. L'algorithme 1 est en effet assuré de terminer avec un ensemble $\mathcal{R}$ à expansion finie.

Données: $\mathcal{R}$ un ensemble de règles, F un fait et Q une requête
Résultat: Vrai si Q est conséquence logique de $F, \mathcal{R}$, faux ou ne s'arrête pas sinon
$F_c = F$;
tant que $\alpha_(F_c, \mathcal{R}) \not\equiv F_c$ **faire**
  $F_c = \alpha(F_c, \mathcal{R})$;
  si $F_c \models Q$;
  alors
    retourner *Vrai*;
  fin
fin
retourner *Faux*;

Algorithm 1: L'algorithme générique de marche avant

En effet, il suffit d'appliquer des étapes de saturation de F, c'est-à-dire calculer $\alpha_k(F, \mathcal{R})$ jusqu'à ce que $\alpha_k(F, \mathcal{R}) \equiv \alpha_{k+1}(F, \mathcal{R})$. Un tel k existe par hypothèse : il suffit alors de tester la conséquence logique entre faits.

3 Rappels sur la notion de dépendance

La recherche de cas décidables pour la conséquence logique, c'est-à-dire de contraintes sur les ensembles de règles assurant la décidabilité a mené à la définition de nombreuses classes de règles. Malheureusement, ces différents critères sont généralement incompatibles, dans la mesure où si $\mathcal{R}_a$ et $\mathcal{R}_b$ appartiennent à des classes décidables, il est généralement faux que $\mathcal{R}_a \cup \mathcal{R}_b$ appartienne également à une classe décidable (Baget *et al.* (2010)). Cette propriété empêche la décomposition d'un ensemble de règles en sous-ensembles plus simples, que l'on souhaiterait étudier de manière séparée.

Une réponse partielle à ce problème a été proposée dans Baget (2004) et Baget *et al.* (2009) : la notion de dépendance. Intuitivement, une règle R_j dépend d'une règle R_i si une application de R_i est susceptible de déclencher une nouvelle application de R_j. Cette notion est formalisée ci-dessous. Les définitions et propriétés de cette section sont tirées de Baget *et al.* (2009).

3.1 Définitions et propriétés

Définition 4 (Dépendance de R_j par rapport à R_i)
Soit R_i, R_j deux règles. $R_j = (H_j, C_j)$ dépend de $R_i = (H_i, C_i)$ s'il existe un fait F, un homomorphisme $\pi : H_i \to F$ et un homomorphisme $\pi' : H_j \to \alpha(F, R_i, \pi)$ tel que $\pi'(H_j) \not\subseteq F$.

Exemple 1 (suite)
L'application π_3 de R_2 n'était pas possible dans F, mais a été rendue possible dans F' grâce à une application de R_1. R_2 dépend donc de R_1. Dans F'' obtenu en appliquant R_2, il y a une nouvelle application de R_1 rendue possible. R_1 dépend donc de R_2.

Cette notion abstraite de dépendance peut se calculer effectivement par un test d'unification entre H_j et C_i (cette notion étant relativement complexe, nous invitons le lecteur intéressé à se reporter à Baget *et al.* (2009)). Ce test est NP-complet, mais cette étape peut être considérée comme étant faite hors-ligne dans le cas de réponses à des requêtes conjonctives. Les relations de dépendance sont alors compilées dans le *graphe de dépendance* des règles.

Définition 5 (Graphe de dépendance)
Soit $\mathcal{R}$ un ensemble de règles. Le graphe de dépendance de règles de $\mathcal{R}$ est le graphe ayant pour sommets les éléments de $\mathcal{R}$, et qui possède un arc de R_i vers R_j ssi R_j dépend de R_i. Il est noté $GRD(\mathcal{R})$ (de l'anglais Graph of Rule Dependencies*).*

Avoir un graphe de dépendance sans circuit est une condition suffisante pour être à expansion finie :

Propriété 1 (GRD sans circuit)
Soit $\mathcal{R}$ un ensemble de règles. Si $GRD(\mathcal{R})$ est sans circuit, alors $\mathcal{R}$ est à expansion finie.

La propriété clé suivante permet (potentiellement) de *décomposer* un ensemble de règles en sous-ensembles plus petits, qui peuvent être étudiés séparément.

Propriété 2 (Décomposition en composantes fortement connexes)
Soit $\mathcal{R}$ un ensemble de règles. Si toutes les composantes fortement connexes de $GRD(\mathcal{R})$ sont à expansion finie, alors $\mathcal{R}$ est à expansion finie.

3.2 Limites de cette notion de dépendance

Nous présentons maintenant plusieurs limites de cette notion de dépendance. Pour des raisons historiques, il est parfois demandé que les règles aient des conclusions atomiques. C'était en effet le cas dans le Datalog [1] de base, où cela peut être supposé sans perte de généralité, car toute règle Datalog avec plusieurs atomes en conclusion peut être décomposée en un ensemble de règles avec même hypothèse et un seul atome en conclusion. Dans le cas des règles existentielles, la transformation est également possible, mais fait appel à un prédicat auxiliaire par règle, comme présenté ci-dessous.

Définition 6 (Décomposition atomique $\mathcal{T}_a$)
Soit $R = (H, C)$ une règle possédant k atomes en conclusion. $\mathcal{T}_a(R)$ est un ensemble de $k + 1$ règles. Une première règle a pour hypothèse H et pour conclusion $p_R(x_1, \ldots, x_n)$, où var$(C) = \{x_i\}_i$, et p_R est un prédicat frais. Les k autres règles ont pour hypothèse $p_R(x_1, \ldots, x_n)$, et pour conclusion l'un des atomes qui est en conclusion de R (une règle par atome). On définit enfin $\mathcal{T}_a(\mathcal{R}) = \cup_{R \in \mathcal{R}} \mathcal{T}_a(R)$.

Exemple 2
Les règles ci-dessous montrent la décomposition de $\mathcal{R} = \{R_1, R_2\}$ en $\mathcal{T}_a(\mathcal{R}) = \{R_1^0, R_1^1, R_1^2, R_1^3, R_2^0, R_2^1\}$:

$R_1 : p(x) \rightarrow r(x,y) \wedge r(y,z) \wedge r(z,x)$	$R_2 : r(x,y) \wedge r(y,x) \rightarrow p(y)$
$R_1^0 : p(x) \rightarrow p_{R_1}(x,y,z)$	$R_2^0 : r(x,y) \wedge r(y,x) \rightarrow p_{R_2}(x,y)$
$R_1^1 : p_{R_1}(x,y,z) \rightarrow r(x,y)$	$R_2^1 : p_{R_2}(x,y) \rightarrow p(y)$
$R_1^2 : p_{R_1}(x,y,z) \rightarrow r(y,z)$	
$R_1^3 : p_{R_1}(x,y,z) \rightarrow r(z,x)$	

Propriété 3
Soit $K = (F, \mathcal{R})$ une base de connaissances et Q une requête conjonctive. $\mathcal{T}_a(\mathcal{R})$ est tel que $\mathcal{R}, F \models Q$ ssi $\mathcal{T}_a(\mathcal{R}), F \models Q$.

Cependant, les graphes de dépendance de ces deux ensembles de règles peuvent avoir des comportements différents : $GRD(\mathcal{R})$ peut être sans circuit alors que $GRD(\mathcal{T}_a(\mathcal{R}))$ en possède, comme le montre l'exemple suivant.

Exemple 2 (suite)
En reprenant l'exemple précédent, les graphes de dépendance de $\mathcal{R}$ et $\mathcal{T}_a(\mathcal{R})$ sont visibles dans la Figure 1. Du fait de la dépendance de R_2^0 envers R_1^1, R_1^2 et R_1^3, on a maintenant un cycle dans le GRD. En découpant la conclusion de R_1, on a ainsi introduit de nouvelles dépendances.

Plus généralement, dans le cas où R_c dépend de R_b qui dépend elle-même de R_a, il n'est pas certain qu'une application de R_c puisse être déclenchée par une application de R_b qui a été déclenchée par une application de R_a. Ce cas de figure est illustré par l'exemple 2, après décomposition atomique (en considérant par exemple $R_a = R_1^0, R_b = R_1^1, R_c = R_2^0$). Alors que la dépendance calcule très exactement la

1. On a alors des règles sans variable existentielle en conclusion, i.e., pour toute $R = (H, C)$, var$(C) \subseteq$ var(H).

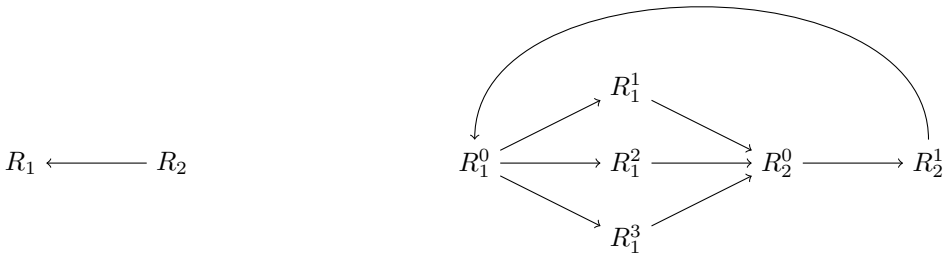


FIGURE 1 – Les graphes de dépendances de $\mathcal{R}$ et de $\mathcal{T}_a(\mathcal{R})$

possibilité pour une règle de déclencher l'application d'une autre sur un fait dont on ne sait rien, l'utilisation de la dépendance à une étape $k > 1$ de saturation n'est plus optimale, car nous avons une information supplémentaire sur le fait : il a été produit par $k - 1$ étapes de saturation. C'est cette nouvelle notion de dépendance que nous nous proposons d'étudier dans la suite de cet article.

4 Une nouvelle famille de dépendances

Pour cela, nous proposons dans cette section un raffinement de la notion de dépendance : la dépendance à l'ordre k ($k \geq 1$), encore appelée k-dépendance. L'intuition derrière cette notion est d'exclure des cas de dépendance celles qui font "fortement" appel au fait initial, au sens où elles ne peuvent avoir lieu "tard", c'est-à-dire sur un fait qui est un saturé à l'ordre k d'un autre fait. Il devient donc nécessaire (même si la notation n'en fait pas explicitement mention) de prendre en compte l'ensemble $\mathcal{R}$ tout entier, et plus seulement deux règles pour décider de la dépendance.

Définition 7 (k-dépendance)
Soit $\mathcal{R}$ un ensemble de règles, $k \geq 1$, et $R_i, R_j \in \mathcal{R}$. On dit que $R_j = (H_j, C_j)$ dépend à l'ordre k (ou k-dépend) de $R_i = (H_i, C_i)$ s'il existe F et un homomorphisme $\pi : H_j \to \alpha_k(F, \mathcal{R})$ tel qu'un atome de H_j ait pour image par π un atome d'ordre k et d'origine R_i. Un tel fait F est appelé un k-témoin de la dépendance

Comme pour la dépendance usuelle, nous synthétisons les dépendances au sein d'un graphe de dépendance, le $GRD_k(\mathcal{R})$:

Définition 8 ($GRD_k(\mathcal{R})$)
Soit $\mathcal{R}$ un ensemble de règles. Le graphe de dépendance à l'ordre k de $\mathcal{R}$ a pour sommets les éléments de $\mathcal{R}$, et un arc de R_i vers R_j ssi R_j dépend à l'ordre k de R_i. Il est noté $GRD_k(\mathcal{R})$.

Nous commençons par démontrer que l'objet que nous définissons est bien "plus petit" que le graphe usuel, ce qui était notre but initial :

Propriété 4
Soit $\mathcal{R}$ un ensemble de règles. On a, pour $1 \leq k \leq k'$, les inclusions suivantes :

$$GRD_{k'}(\mathcal{R}) \subseteq GRD_k(\mathcal{R}) \subseteq GRD_1(\mathcal{R}) = GRD(\mathcal{R})$$

Preuve : Soit $1 \leq k \leq k'$. Comme le saturé à l'ordre k' de F est égal au saturé à l'ordre k de $\alpha_{k'-k}(F, \mathcal{R})$, nous en déduisons que $GRD_{k'}(\mathcal{R}) \subseteq GRD_k(\mathcal{R})$.

Dans quels cas le $GRD_k(\mathcal{R})$ est-il strictement plus petit que le $GRD(\mathcal{R})$? Pour en obtenir une intuition, il est intéressant de considérer l'exemple suivant : $p(x) \wedge r(x, y) \rightarrow r(y, z)$. Cette règle dépend d'elle même, mais ne 2-dépend pas d'elle même (car elle ne produit pas d'atome de prédicat p). Cet exemple est généralisable, en considérant des chaînes d'atomes de prédicat r de longueur k en hypothèse, la conclusion apportant un $k+1^{\text{ème}}$ élément à la chaîne. Cette règle serait k dépendante d'elle-même, mais pas $k + 1$-dépendante.

Nous montrons maintenant que les propriétés du $GRD(\mathcal{R})$ concernant la marche avant peuvent être généralisées au $GRD_k(\mathcal{R})$.

Propriété 5 (Acyclicité de $GRD_k(\mathcal{R})$)
Soit $\mathcal{R}$ un ensemble de règles. S'il existe k tel que $GRD_k(\mathcal{R})$ soit sans circuit, $\mathcal{R}$ est à expansion finie.

Preuve : Soit F un fait quelconque. Soit F^k le saturé à l'ordre k de F. Soit $\mathcal{R}_1$ l'ensemble des règles qui s'appliquent de façon nouvelle à F^k. Les règles susceptibles de s'appliquer à F^{k+l} sont celles avec un chemin de longueur l partant d'une règle de $\mathcal{R}_1$. Comme $GRD_k(\mathcal{R})$ est sans circuit, il existe m tel que m soit la longueur d'un plus long chemin dans $GRD_k(\mathcal{R})$. On a alors $F_{k+m} = F_{k+m+1}$, et $\mathcal{R}$ est donc à expansion finie.

Propriété 6 (Décomposition en c.f.c.)
Soit $\mathcal{R}$ un ensemble de règles. Si toutes les composantes fortement connexes de $GRD_k(\mathcal{R})$ sont à expansion finie, alors $\mathcal{R}$ est à expansion finie.

Preuve : Similaire à celle de la propriété 5.

Il existe également des propriétés du GRD_k qui ne sont pas directement le pendant d'une propriété du GRD. En effet, la notion de k-dépendance possède une certaine résistance aux transformations syntaxiques, par exemple à la décomposition atomique présentée dans la section précédente :

Propriété 7 (Résistance à la décomposition atomique)
Soit $\mathcal{R}$ un ensemble de règles. Si $GRD_k(\mathcal{R})$ est sans circuit, alors $GRD_{2k}(\mathcal{T}_a(\mathcal{R}))$ est sans circuit.

Preuve : Voir Baget *et al.* (2011b).

L'intérêt de la notion de k-dépendance ne se limite pas à la définition de classes décidables plus générales qu'auparavant. Nous présentons ici de possibles optimisations pratiques que peut apporter la connaissance du $GRD_k(\mathcal{R})$, avec la présentation de l'algorithme 2. En effet, par définition de la k-dépendance, une fois effectuées k étapes

de saturation, il n'est pas nécessaire de tester les applications potentielles de toutes les règles, mais uniquement de celles étant successeurs dans le $GRD_k(\mathcal{R})$ d'une règle appliquée au rang précédent.

Données: F un fait, $\mathcal{R}$ une base de connaissances, Q une requête, $GRD_k(\mathcal{R})$
Résultat: Vrai si Q est conséquence logique de $F, \mathcal{R}$, faux ou ne s'arrête pas sinon
$F_b = \alpha_{k-1}(F, \mathcal{R})$;
$F_c = \alpha(F_b, \mathcal{R})$;
tant que $F_b \not\equiv F_c$ **faire**
  $F_b = F_c$;
  $\mathcal{R}_b =$ l'ensemble des règles ayant été appliquées à F_b;
  $\mathcal{R}_c =$ l'ensemble des successeurs des règles de $\mathcal{R}_b$ dans $GRD_k(\mathcal{R})$;
  $F_c = \alpha(F_c, \mathcal{R}_c)$;
  si $F_c \models Q$ **alors**
    retourner *Vrai*;
  fin
fin
retourner *Faux;*

Algorithm 2: L'algorithme de marche avant avec un graphe de k-dépendance précompilé.

Enfin, nous mentionnons un résultat de complexité. Soit la famille de problèmes de décision DÉDUCTION$_k$, paramétrée par $k \geq 1$:

– ENTRÉE : un ensemble de règles $\mathcal{R}$, $R_i, R_j \in \mathcal{R}$
– QUESTION : R_j k-dépend de R_i ?

On peut montrer (Baget *et al.* (2011b)) que pour tout $k \geq 1$, le problème DÉDUCTION$_k$ est NP-complet.

5 Conclusion et travaux ultérieurs

Nous nous sommes intéressés au problème de l'interrogation de bases de connaissances munies de règles expressives à l'aide de requêtes conjonctives. Pour ce problème indécidable dans le cas général, il a été proposé dans les nombreuses publications récentes, un grand nombre de classes décidables de règles. Cependant, ces différentes classes de règles sont incompatibles, au sens où l'union de deux ensembles de règles décidables n'est plus nécessairement décidable. Le seul outil à la connaissance des auteurs permettant de combiner des résultats de décidabilité est le graphe de dépendance des règles, qui souffre cependant de certaines faiblesses. Nous avons proposé un raffinement de la notion de dépendance, à savoir la k-dépendance, qui possède de meilleures propriétés. Il reste cependant des pistes à explorer. Premièrement, l'algorithme (non présenté ici, voir Thomazo (2010),Baget *et al.* (2011b)) pour caractériser la k-dépendance entre deux règles repose sur de la force brute. Deuxièmement, il serait bon d'adapter les résultats obtenus également dans le cas de la marche arrière. Pour ces deux pistes, étudier les liens entre la k-dépendance et les unificateurs (qui servent pour la marche

arrière et pour caractériser la dépendance usuelle) est la démarche envisagée. Enfin, l'étude de la k-dépendance invite à considérer des réécritures de règles, de manière à obtenir des ensembles de règles équivalentes ayant de bonnes propriétés syntaxiques.

Références

BAGET J.-F. (2004). Improving the forward chaining algorithm for conceptual graphs rules. In *KR*, p. 407–414.

BAGET J.-F., LECLÈRE M. & MUGNIER M.-L. (2010). Walking the decidability line for rules with existential variables. In *Proceedings of the Twelfth International Conference on Principles of Knowledge Representation and Reasoning, Toronto, Canada* : AAAI Press.

BAGET J.-F., LECLÈRE M., MUGNIER M.-L. & SALVAT E. (2009). Extending decidable cases for rules with existential variables. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI 2009), Pasadena, California, USA,*, p. 677–682.

BAGET J.-F., LECLÈRE M., MUGNIER M.-L. & ÉRIC SALVAT (2011a). On rules with existential variables : Walking the decidability line. *To appear in Artificial Intelligence Journal*.

BAGET J.-F. & MUGNIER M.-L. (2002). The Complexity of Rules and Constraints. *J. Artif. Intell. Res. (JAIR)*, **16**, 425–465.

BAGET J.-F., MUGNIER M.-L. & THOMAZO M. (2011b). *Towards Farsighted Dependencies for Existential Rules*. LIRMM Research Report 11016, LIRMM.

CALÌ A., GOTTLOB G. & KIFER M. (2008). Taming the infinite chase : Query answering under expressive relational constraints. In *Proceedings of the Eleventh International Conference on the Principles of Knowledge Representation and Reasoning (KR 2008), Sydney, Australia*, p. 70–80.

CALÌ A., GOTTLOB G. & LUKASIEWICZ T. (2009). A general datalog-based framework for tractable query answering over ontologies. In *Proceedings of the 28th symposium on Principles of database systems (PODS'09)*, p. 77–86, New York, NY, USA : ACM.

FAGIN R., KOLAITIS P. G., MILLER R. J. & POPA L. (2005). Data exchange : semantics and query answering. *Theor. Comput. Sci.*, **336**(1), 89–124.

MAIER D., MENDELZON A. O. & SAGIV Y. (1979). Testing implications of data dependencies. *ACM Trans. Database Syst.*, **4**, 455–469.

SPEZZANO F. & GRECO S. (2010). Chase termination : A constraints rewriting approach. *PVLDB*, **3**(1), 93–104.

THOMAZO M. (2010). Interrogation de bases de connaissances avec des règles expressives : décidabilité et dépendance entre règles. *Rapport de M2 au MPRI*.

Théorie de méta-diagnostic : raisonnement sur les systèmes de diagnostic

Nuno Belard[1,2,3], Yannick Pencolé[2,3], Michel Combacau[2,3]

[1] Airbus France; 316 route de Bayonne; 31060 Toulouse, France
[2] LAAS-CNRS; 7 Avenue du Colonel Roche; F-31077 Toulouse, France
[3] Université de Toulouse; UPS, INSA, INP, ISAE; LAAS; 7 Avenue du Colonel Roche, F-31077 Toulouse, France
nuno.belard@airbus.com, ypencole@laas.fr, combacau@laas.fr

Résumé : Le principe du diagnostic à base de modèle est mis en œuvre à l'aide d'un algorithme qui, s'appuyant sur un modèle du système sous-jacent et de son observation, génère des hypothèses de diagnostic. Dans les applications réelles et contrairement aux hypothèses classiques, il se peut que ni le modèle, ni les observations, ni même l'algorithme implanté ne remplissent les pré-requis nécessaires à la bonne marche du système de diagnostic. Le problème de méta-diagnostic consiste à déterminer les anomalies du système de diagnostic utilisé. Dans cet article, nous proposons dans un premier temps, une théorie générale du méta-diagnostic avec une sémantique clairement définie pour résoudre ce problème. Puis, nous établissons une liste des pré-requis au bon fonctionnement d'un système de diagnostic et leurs relations. Enfin, à l'aide du cadre théorique proposé, nous proposons et résolvons des exemples de problèmes de méta-diagnostic s'appuyant sur les pré-requis définis précédemment.

Mots-clés : Diagnostic à base de modèle, représentation des connaissances

1 Introduction

Le raisonnement diagnostique consiste à déterminer les composants normaux et anormaux d'un système réel. Dans le cadre du diagnostic à base de modèle (Reiter, 1987), (de Kleer & Williams, 1987), on dispose d'un algorithme générique produisant les diagnostics en s'appuyant sur une connaissance dite « profonde » du système et sur les observations disponibles. Dans cet article, ce modèle de connaissance est appelé « représentation du système » [1] et nous définissons un *système de diagnostic* par le triplet : *représentation du système*, *observations*, *algorithme de diagnostic*.

Sur des applications réelles, il n'est malheureusement pas rare de constater que les systèmes de diagnostic ne vérifient pas les prérequis usuels pour leur bon fonctionnement tel que, par exemple, l'utilisation d'une représentation correcte (c.-à-d. ontologiquement vraie) du système. Dans le cadre aéronautique, par exemple, cette garantie est nécessaire, sinon le système de diagnostic peut conduire à des opérations de maintenance incorrectes provoquant des délais non anticipés avec de sérieuses conséquences économiques. Néanmoins, disposer d'une représentation ontologiquement vraie n'est

1. Le mot « modèle » est, quant à lui, réservé à son usage classique dans le contexte de la théorie des modèles.

pas simple et de nombreux ingénieurs s'emploient à trouver des moyens d'en détecter et d'en corriger les anomalies.

Nous proposons d'appeler *méta-diagnostic* le problème de la détermination des anomalies dans un système de diagnostic. Notre première contribution, en section 3, est la définition d'une théorie logique de méta-diagnostic pour la résolution de ce type de problème. Cette théorie offre de nombreux avantages : elle dispose d'une sémantique clairement définie par la logique, c'est une théorie générale et unifiée afin de raisonner sur tout type de système de diagnostic. De plus, cette théorie de méta-diagnostic rend possible l'utilisation successive d'un ensemble de cas de tests (ensembles d'observations sur le système de diagnostic) afin de raffiner le méta-diagnostic. Cette possibilité de raffinement est particulièrement utile dans le cadre aéronautique où l'on dispose d'un ensemble important de cas de tests du système de diagnostic étudié ce qui permet une automatisation du processus d'isolation des anomalies dans le système de diagnostic.

La section 5 décrit la modélisation et la résolution de deux problèmes importants de méta-diagnostic. Ces problèmes de méta-diagnostic s'appuient sur des propriétés généralement requises pour le bon fonctionnement de système de diagnostic et qui sont présentées dans la section 4.

2 Rappels

Nous supposons ici que les bases de la théorie des modèles (structure, modèle et extensions) sont familières au lecteur (Hodges, 1993) ainsi que le cadre logique classique du diagnostic à base de modèle (Reiter, 1987)(de Kleer & Williams, 1987).

2.1 Système réel/représentation du système

Le diagnostic à base de modèle (DBM) est un problème de raisonnement dont le but est de déterminer les anomalies d'un système étant donnés sa description (souvent noté DS) et un ensemble d'observations OBS. L'hypothèse cruciale sur laquelle s'appuient les approches DBM est que le couple $(\mathsf{DS},\mathsf{OBS})$ représente *correctement* la réalité sous-jacente (c.-à-d. le système réel et les observations). Plus formellement, la réalité n'est représentable qu'à travers une structure, notons-la Ψ, d'informations brutes qui pourraient être accessibles (comportements et observations) si les capacités d'ingénierie et de calcul étaient illimitées. L'hypothèse des approches DBM sous-entend ainsi qu'il existe toujours une structure s disponible, modèle de $\mathsf{DS}\cup\mathsf{OBS}$ (c.-à-d. $\mathsf{s}\in\mathsf{Mod}(\mathsf{DS}\cup\mathsf{OBS})$) pouvant être étendue à une structure t (c.-à-d. $\mathsf{s}\subseteq\mathsf{t}$) isomorphe à $\Psi : t \rightleftharpoons \Psi$. Utilisant les mots de Tarski (Tarski, 1936), les approches DBM s'appuient sur le fait que $\mathsf{DS}\cup\mathsf{OBS}$ est une théorie *ontologiquement vraie*, une théorie « *which says that the state of affairs is so and so, and the state of affairs is indeed so and so* ». Dans cet article, nous appelons *système réel* un ensemble R d'entités élémentaires (ou unités) communicantes et remplaçables (UCR), sur lesquelles s'effectue toute action de maintenance issue d'un diagnostic. La connaissance associée à R sera appelée *représentation du système*. Le système de diagnostic étant défini par l'utilisateur, la notion d'anomalie du système réel est donc lié à son point de vue :

Définition 1 (Anomalies d'une entité)
Une entité $c \in R$ est dite anormale *(ou subit une* anomalie*) si au moins une de ses caractéristiques, vitale pour l'application considérée, a subi une transformation irréversible.*

Enfin, le système de diagnostic a pour objectif de déterminer *l'état de santé réel* $\sigma_{\text{réel}}$ constitué de l'ensemble des entités anormales.

2.2 Diagnostic à base de modèle

En s'appuyant sur les notions et les notations décrites ci-dessus, nous rappelons brièvement le cadre classique du diagnostic à base de modèle qui est utilisé dans cet article (de Kleer & Williams, 1987) (Reiter, 1987).

Définition 2 (représentation du système)
La représentation d'un système est un couple (DS,COMPS) où :

1. *DS, la description du système dans cette représentation, est un ensemble de phrases du premier ordre.*
2. *COMPS, est un ensemble fini de constantes modélisant les composants du système réel.*

Nous supposons ici l'existence d'un isomorphisme entre les ensembles R et COMPS du fait que R est un ensemble défini par l'utilisateur et par les actions de maintenance sur les entités de R. Le prédicat Ab(c) (resp. $\neg$Ab(c)) signifie que le composant $c \in$ COMPS est anormal (resp. normal).

Les observations font partie des quelques liens entre le système réel et sa représentation. Intuitivement, les observations sont des captures d'information issues du système réel effectuées à l'aide d'un ensemble O de *capteurs* mesurant la valeur v(p) d'un paramètre réel p. Une fois acquises, ces observations sont confrontées à la représentation du système dans la forme dans laquelle v(p) et p sont effectivement représentés afin d'effectuer le diagnostic.

Définition 3 (Observations)
Les observations OBS forment un ensemble de phrases du premier ordre.

L'exemple suivant illustre un système réel, sa contrepartie qui est modélisée ainsi que des observations. Cet exemple servira de base tout au long de cet article.

Exemple 1
On considère le circuit de la figure 1 de Davis (Davis, 1984).

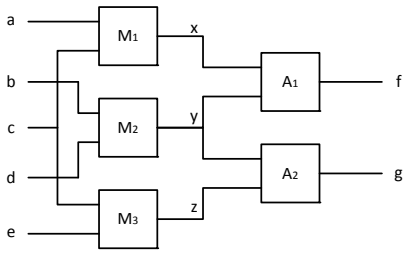


FIGURE 1 – Un circuit avec 3 multiplicateurs (M_1,M_2, M_3) *et 2 additionneurs* (A_1,A_2).

Supposons que ce circuit soit représenté par COMPS = {M_1,M_2,M_3,A_1,A_2} *et* DS[2]

2. Remarquons que, dans cet exemple, la première phrase de DS *est manifestement incorrecte car elle ne représente pas le comportement d'un multiplicateur réel.*

$M_1desc : \neg\, Ab(M_1) \Rightarrow (v(x) = (v(a) + 1) * v(c))$
$M_2desc : \neg\, Ab(M_2) \Rightarrow (v(y) = v(b) * v(d))$
$M_3desc : \neg\, Ab(M_3) \Rightarrow (v(z) = v(c) * v(e))$
$A_1desc : \neg\, Ab(A_1) \Rightarrow (v(f) = v(x) + v(y))$
$A_2desc : \neg\, Ab(A_2) \Rightarrow (v(g) = v(y) + v(z))$

étendu avec les axiomes appropriés pour l'arithmétique en particulier. On suppose également que les paramètres a, b, c, d, e, f *et* g *sont observés avec pour valeurs 1, 2, 3, 4, 5, 11 et 22 respectivement.*

Étant donnés une représentation (DS,COMPS) et des observations OBS, il est possible désormais d'introduire la notion de *problème de diagnostic*.

Définition 4 (Problème de diagnostic)
Un problème de diagnostic est un triplet (DS,COMPS,OBS).

Finalement, la définition sémantique du diagnostic repose sur le concept d'*état de santé* s'appuyant sur la représentation du système $\sigma(\Delta,\mathsf{COMPS}\backslash\Delta) = [\bigwedge_{c\in\Delta}\mathsf{Ab}(c)]\wedge[\bigwedge_{c\in(\mathsf{COMPS}\backslash\Delta)}\neg\mathsf{Ab}(c)]$.

Définition 5 (Diagnostic (sémantique))
Soit $\Delta \subseteq COMPS$. Un diagnostic, $D_{T\text{-}M}$, pour (DS,COMPS,OBS) est l'ensemble des candidats $\sigma(\Delta,COMPS\backslash\Delta)$ tels que $DS\cup OBS\cup\sigma(\Delta,COMPS\backslash\Delta)$ est satisfiable.

La définition 5 fournit une sémantique parfaitement définie pour la notion de diagnostic. Néanmoins, comme elle est donnée en terme de modèles, elle offre peu d'intérêts au niveau algorithmique. La théorie de la démonstration offre une approche syntaxique et un environnement algorithmique plus attractif. C'est pourquoi nous considérons qu'il existe un démonstrateur automatique sous-jacent dans chaque *algorithme de diagnostic* $\mathcal{A}$. L'algorithme de (Reiter, 1987) utilise un tel démonstrateur de façon explicite.

Définition 6 (Diagnostic (syntaxique))
Un diagnostic (syntaxique) $D_{T\text{-}D}$ est l'ensemble de candidats calculé par l'algorithme de diagnostic $\mathcal{A}$ ($\vdash_{\mathcal{A}}$).

3 Caractérisation du méta-diagnostic

La section 2 a présenté les éléments servant à définir un système de diagnostic.

Définition 7 (Système de diagnostic)
Un système de diagnostic est un quadruplet (DS,COMPS,OBS,$\mathcal{A}$).

Comme mentionné dans l'introduction, les systèmes de diagnostic peuvent être eux-mêmes anormaux, par rapport à certaines propriétés souhaitées pour de nombreuses raisons : des erreurs de modélisation, des erreurs de perception (observations incertaines), des erreurs algorithmiques (l'algorithme ne produisant pas les candidats de diagnostic issus de la définition sémantique). Sachant que dans les applications réelles de telles anomalies sont loin d'être rares, il est nécessaire d'établir une théorie de méta-diagnostic dont la finalité est de raisonner sur le système de diagnostic lui-même. Afin d'obtenir une théorie générale, nous nous appuyons ici sur la logique du premier ordre. [3]

3. Les résultats de ce papier sont également valides pour un ensemble d'autres logiques.

3.1 Méta-système

Au niveau méta-diagnostic, les *méta-composants* sont associés aux éléments du système de diagnostic dont le comportement peut être suspecté comme étant normal ou anormal. Le comportement de ces composants, de même que leurs interactions sont décrits dans un méta-système à l'aide du prédicat unaire $\mathsf{M\text{-}Ab}(\cdot)$ sémantiquement associé à la présence d'une anomalie dans un méta-composant. Intuitivement, un méta-système rassemble donc la connaissance statique utilisée par le méta-diagnostic pour raisonner sur le système de diagnostic sous-jacent.

Définition 8 (Méta-système)
Un méta-système est un couple (DM-S,M-COMPS) où :

1. *DM-S, la description du méta-système, est un ensemble de phrases du premier ordre.*
2. *M-COMPS l'ensemble des méta-composants identifés par des constantes.*

Le choix des méta-composants dépend de l'objectif du raisonnement et des hypothèses de départ.

1. Il n'est pas nécessaire de considérer chaque élément d'un système de diagnostic comme un méta-composant.
2. Les méta-composants peuvent être définis à différents niveaux d'abstraction.

Par exemple, si pour un problème donné, nous supposons que les algorithmes de diagnostic sont corrects et les observations normales, alors, afin de déterminer dans la représentation du système si chaque phrase décrit correctement le système réel, il suffit d'associer à chacune de ces phrases un méta-composant et uniquement à ces phrases. Ceci est illustré ci-dessous en s'appuyant sur l'exemple 1. Une possibilité pour DM-S avec M-COMPS={$\mathsf{M_1desc}$, $\mathsf{M_2desc}$, $\mathsf{M_3desc}$, $\mathsf{A_1desc}$, $\mathsf{A_2desc}$, Alg} serait[4] :

$$\neg\mathsf{M\text{-}Ab}(\mathsf{M_1desc}) \Rightarrow [\neg\mathsf{Ab}(\mathsf{M_1}) \Rightarrow (\mathsf{v(x)} = (\mathsf{v(a)}+1)*\mathsf{v(c)})]$$
$$\neg\mathsf{M\text{-}Ab}(\mathsf{M_2desc}) \Rightarrow [\neg\mathsf{Ab}(\mathsf{M_2}) \Rightarrow (\mathsf{v(y)} = \mathsf{v(b)} * \mathsf{v(d)})]$$
$$\neg\mathsf{M\text{-}Ab}(\mathsf{M_3desc}) \Rightarrow [\neg\mathsf{Ab}(\mathsf{M_3}) \Rightarrow (\mathsf{v(z)} = \mathsf{v(c)} * \mathsf{v(e)})]$$
$$\neg\mathsf{M\text{-}Ab}(\mathsf{A_1desc}) \Rightarrow [\neg\mathsf{Ab}(\mathsf{A_1}) \Rightarrow (\mathsf{v(f)} = \mathsf{v(x)} + \mathsf{v(y)})]$$
$$\neg\mathsf{M\text{-}Ab}(\mathsf{A_2desc}) \Rightarrow [\neg\mathsf{Ab}(\mathsf{A_2}) \Rightarrow (\mathsf{v(g)} = \mathsf{v(y)} + \mathsf{v(z)})]$$

L'idée derrière la description DM-S est que si une DS-phrase n'est pas anormale, alors elle décrit correctement ce qui se passe en réalité.

3.2 Méta-observations

Les méta-observations sont utilisées en liaison avec les méta-systèmes afin de déterminer les anomalies des méta-composants.

Définition 9 (Méta-observations)
Les méta-observations M-OBS forment un ensemble fini de phrases du premier ordre.

La disponibilité des méta-observations peut avoir plusieurs sources : il peut s'agir d'informations issues de cas de tests dans lesquels l'état de santé réel $\sigma_{\mathsf{réel}}$ est connu, des résultats de diagnostic calculés par le système de diagnostic, des observations du système, etc. Les méta-observations regroupent ainsi toutes les observations disponibles au niveau système et au niveau méta-système. Dans l'exemple précédent, si l'état de santé

4. La section 5 présente plus clairement la méthode pour décrire un méta-système.

réel est connu et a produit l'ensemble d'observations OBS alors les méta-observations disponibles M-OBS sont :

1. $\sigma_{\text{réel}} \equiv \neg Ab(M_1) \wedge \neg Ab(M_2) \wedge \neg Ab(M_3) \wedge \neg Ab(A_1) \wedge Ab(A_2)$,
2. $OBS \equiv v(a) = 1 \wedge v(b) = 2 \wedge v(c) = 3 \wedge v(d) = 4 \wedge v(e) = 5 \wedge v(f) = 11 \wedge v(g) = 22$.

3.3 Méta-diagnostic

Un méta-système et un ensemble de méta-observations forment un problème de *méta-diagnostic*.

Définition 10 (Problème de méta-diagnostic)
Un problème de méta-diagnostic est un triplet (M-DS,M-COMPS,M-OBS).

La résolution d'un problème de méta-diagnostic consiste à déterminer les anomalies des méta-composants M-COMPS en s'appuyant sur les méta-observations.

Définition 11 (État de méta-santé)
Soit $\Phi \subseteq$ M-COMPS un ensemble de méta-composants, l'état de méta-santé $\pi(\Phi, M\text{-}COMPS \setminus \Phi)$ est la conjonction : $\bigwedge_{mc \in \Phi} M\text{-}Ab(mc)] \wedge [\bigwedge_{mc \in (M\text{-}COMPS \setminus \Phi)} \neg M\text{-}Ab(mc)$.

Définition 12 (Méta-diagnostic)
Le méta-diagnostic M-D, pour le problème (M-DS,M-COMPS,M-OBS) est l'ensemble des candidats $\pi(\Phi, M\text{-}COMPS \setminus \Phi)$ tels que $M\text{-}DS \cup M\text{-}OBS \cup \pi(\Phi, M\text{-}COMPS \setminus \Phi)$ est satisfiable.

Au final, le lecteur pourra constater que les définitions 5 et 12 sont identiques. En effet, un problème de méta-diagnostic est un problème de diagnostic dont l'objet diagnostiqué est un système de diagnostic, ce qui conduit entre autres à ce que :

1. chaque algorithme de diagnostic existant peut être utilisé comme un algorithme de méta-diagnostic s'il est correct et complet au sens de la sémantique sous-jacente (Hodges, 1993).
2. chaque méthode pouvant réduire la complexité pratique du problème de diagnostic peut s'appliquer au problème de méta-diagnostic.

4 Propriétés des systèmes de diagnostic et de leur résultats

La section précédente décrit une caractérisation du méta-diagnostic suffisamment générique pour résoudre différents problèmes de méta-diagnostic. Avant d'en décrire certains en section 5, il est nécessaire de présenter un certain nombre de propriétés sur les systèmes de diagnostic dont l'absence est considérée comme anormale. Une liste plus exhaustive est proposée dans (Belard *et al.*, 2010).

4.1 Propriétés sur les résultats de diagnostic

La qualité d'un diagnostic peut être évaluée à l'aide de deux propriétés : la *validité* et la *certitude*.

Définition 13 (Validité)
Soit $\sigma_{réel}$ un état de santé tel que, pour chaque $c \in COMPS$, si c est l'image de $r \in R$: 1) si r est anormal, $\neg Ab(c) \wedge \sigma_{réel} \models \perp$; et 2) si r est normal, $Ab(c) \wedge \sigma_{réel} \models \perp$. Un résultat de diagnostic, D, est valide si $\sigma_{réel} \in D$; et invalide sinon.

Définition 14 (Certitude)
Un résultat de diagnostic, D, est certain s'il ne contient qu'un seul candidat, c.-à-d. $\#D=1$; et incertain sinon.

Disposer de résultats valides et certains est extrêmement intéressant, notamment dans le monde aéronautique où les résultats invalides peuvent conduire les équipes de maintenance d'un appareil à déposer des composants en parfait état de fonctionnement et où des résultats incertains peuvent entrainer des tâtonnement impactant de manière inacceptable la disponibilité de l'avion.

4.2 Propriétés sur les observations

Dans cet article, nous ne citons qu'une propriété sur les observations : la véracité. Informellement, des observations sont dites *ontologiquement vraies* si la perception d'un paramètre réel est correct, autrement dit, si la mesure de la valeur du paramètre p est x alors celle-ci est réellement sa valeur. Formellement :

Définition 15 (Véracité)
Soit Ω l'ensemble de toutes les structures et $\Psi \in \Omega$ l'ensemble des informations brutes sur la réalité du système[5]*. Les observations OBS sont ontologiquement vraies ssi $\exists_{s \in Mod(OBS)} \exists_{t \in \Omega} (s \subseteq t) \wedge (t \rightleftharpoons \Psi)$ (Tarski, 1936).*

Il faut insister ici sur la différence entre la vérité logique et la vérité ontologique (véracité), la première étant fondée sur les axiomes d'une théorie et la deuxième s'appuyant sur une correspondance entre les phrases de la théorie et la réalité (Tarski, 1936). Sans observations ontologiquement vraies, les résultats de diagnostic au sens de la définition 5 ne sont pas garantis d'être valides au sens de la définition 13 comme nous le démontrerons plus tard à travers le théorème 1.

4.3 Propriétés sur la représentation du système

On identifie deux propriétés associées à la représentation du système : la véracité et la diagnosticabilité. Intuitivement si la représentation du système est ontologiquement vraie alors toute phrase X de DS énonce que X a effectivement lieu en réalité. Formellement,

Définition 16 (Véracité de la représentation du système)
Soit Ω l'ensemble de toutes les structures et $\Psi \in \Omega$ l'ensemble des informations brutes disponibles sur la réalité. Une représentation du système est ontologiquement vraie ssi, pour tout ensemble ontologiquement vrai OBS, $\exists_{s \in Mod(DS \cup OBS)} \exists_{t \in \Omega} (s \subseteq t) \wedge (t \rightleftharpoons \Psi)$. Si la représentation du système est ontologiquement vraie alors toutes ses phrases le sont.

Comme pour la véracité des observations, la véracité de la représentation du système est essentielle pour garantir que tout diagnostic au sens de la définition 5 soit valide comme nous le montrerons plus tard.

Concernant la diagnosticabilité de la représentation du système, nous reprenons la définition de (Console *et al.*, 2000) avec nos termes :

Définition 17 (Diagnosticabilté)
La représentation du système, associée à un ensemble de capteurs O, est diagnosticable ssi pour tout ensemble d'observations ontologiquement vraies, il existe toujours un et un seul candidat de diagnostic au sens de la définition 5.

5. Pour une discussion détaillée, voir la sous-section 2.1.

L'intérêt de la diagnosticabilité est motivé par le besoin de fournir des diagnostics certains.

4.4 Propriétés sur l'algorithme de diagnostic

Comme on l'a vu, un algorithme de diagnostic peut être vu comme un démonstrateur de théorèmes, deux propriétés lui sont associées : la correction et la complétude.

Définition 18 (Correction et complétude)
Soit T une théorie logique et φ une phrase dans un langage $\mathcal{L}$ pour une sémantique donnée. Un algorithme de diagnostic $\mathcal{A}$ est correct ssi : si ($T \vdash_{\mathcal{A}} \varphi$), alors ($T \models_{\mathcal{L}} \varphi$). Il est complet ssi : si ($T \models_{\mathcal{L}} \varphi$), alors ($T \vdash_{\mathcal{A}} \varphi$).

Par définition, un algorithme de diagnostic correct et complet rend équivalent les notions de diagnostic sémantique et de diagnostic syntaxique (voir définition 6). C'est la raison essentielle de ces propriétés qui sont requises, par exemple, dans le démonstrateur sur lequel s'appuie l'approche de Reiter (Reiter, 1987). On pourrait se demander quel est l'intérêt de méta-diagnostiquer si un algorithme est correct ou pas, complet ou pas, sachant qu'il suffirait de garantir par construction les propriétés d'un tel algorithme. Là encore, le problème est lié aux applications réelles dont la complexité fait que certains algorithmes de diagnostic peuvent difficilement être modélisés dans un cadre logique ou bien encore, le fonctionnement même de l'algorithme n'est pas connu de l'utilisateur final (l'intégrateur en aéronautique par exemple) pour des raisons de confidentialité.

5 Quelques problèmes de méta-diagnostic

Avec la caractérisation présentée en section 3 et les propriétés décrites en section 4, il est possible de modéliser et de résoudre des problèmes de méta-diagnostic typiques. Le processus de modélisation est constitué de 4 étapes :

1. définition des méta-composants en s'appuyant sur les hypothèses du problème et le niveau de détail de la solution ;
2. définition des propriétés dont l'absence est considérée anormale ;
3. définition des comportements normaux et anormaux des méta-composants ;
4. définition des méta-observables.

Pour résoudre les problèmes suivants, nous avons directement utilisé GDE (*General Diagnostic Engine*) (de Kleer & Williams, 1987)(Forbus & de Kleer, 1993). Ceci a été possible du fait que tout algorithme de diagnostic correct et complet peut devenir un algorithme de méta-diagnostic, ce qui est le cas de GDE.

5.1 Le problème de la représentation erronée d'un système

Posons le problème suivant : « soit le système de diagnostic (DS, COMPS, OBS, $\mathcal{A}$), supposons que OBS soit ontologiquement vrai et que $\mathcal{A}$ soit correct et complet, déterminer des phrases ontologiquement fausses dans DS ». La résolution de ce problème requiert le théorème suivant :

Théorème 1
Si (DS,COMPS) est ontologiquement vrai, alors pour tout problème (DS, COMPS, OBS) avec des observations ontologiquement vraies, le diagnostic sémantique $D_{T\text{-}M}$ est valide.

Preuve Démontrons par contrapposition. Supposons que $\sigma_{\text{réel}}$, l'état de santé réel, ne soit pas un candidat de diagnostic au sens de la définition 5 : $\mathsf{DS} \cup \mathsf{OBS} \cup \sigma_{\text{reél}}$ est donc non-satisfiable, il n'a pas de modèle. L'état $\sigma_{\text{réel}}$ étant issu de la réalité, il doit avoir un modèle s tel que $\exists_{\mathsf{t} \in \Omega}\ (\mathsf{s} \subseteq \mathsf{t}) \wedge (\mathsf{t} \rightleftharpoons \Psi)$ et par conséquent $\neg(\exists_{\mathsf{s} \in \mathsf{Mod}(\mathsf{DS} \cup \mathsf{OBS})}\ \exists_{\mathsf{t} \in \Omega}\ (\mathsf{s} \subseteq \mathsf{t}) \wedge (\mathsf{t} \rightleftharpoons \Psi))$. Comme les observations sont ontologiquement vraies, la représentation du système ne l'est donc pas.

Imaginons l'instance du problème suivant issu de l'exemple 1.

1. Les méta-composants sont les phrases de DS.
2. La propriété requise est la véracité de chaque phrase de DS.
3. Considérons la phrase A de DS représentée par le méta-composant mc_1 : soit A est ontologiquement vraie, soit mc_1 est anormal, c.-à-d. $\mathsf{M\text{-}Ab}(\mathsf{mc}_1) \vee \mathsf{A}$. Ainsi le comportement normal de cette phrase est $\neg\mathsf{M\text{-}Ab}(\mathsf{mc}_1) \Rightarrow \mathsf{A}$.
4. À partir du théorème 1 et des hypothèses de ce problème, on sait que si chaque méta-composant est normal alors le diagnostic sémantique est valide. Il suffit donc de méta-observer l'état de santé réel et les observations du système. [6]

Suivant ce processus de modélisation, on obtient le $\mathsf{DM\text{-}S}$ et les $\mathsf{M\text{-}COMPS}$ de l'exemple 1. Si on suppose que les $\mathsf{M\text{-}OBS}$ sont ceux de ce même exemple, on obtient par GDE le méta-diagnostic suivant constitué de 3 diagnostics minimaux (noyaux) : $\{\mathsf{M}_1\mathsf{desc}\}$; $\{\mathsf{M}_2\mathsf{desc}\}$; $\{\mathsf{A}_1\mathsf{desc}\}$. Bien qu'ambigu, ce méta-diagnostic inclut bien la non-véracité de la phrase $\mathsf{M}_1\mathsf{desc}$.

5.2 Le problème de la diagnosticabilité de la représentation du système, de la correction et de la complétude des algorithmes de diagnostic

Supposons maintenant que la phrase $\mathsf{M}_1\mathsf{desc}$ ait été corrigée, nous posons désormais l'hypothèse que la représentation du système est ontologiquement vraie. Investigons le problème suivant : « déterminer si la représentation du système associée aux capteurs O est diagnosticable et si l'algorithme de diagnostic est correct et complet ». Là encore, un certain nombre de théorèmes sont nécessaires pour résoudre ce problème.

Théorème 2
Si $\mathcal{A}$ est correct et complet et si (DS,COMPS) est ontologiquement vrai, alors pour tout problème (DS,COMPS,OBS) où OBS est ontologiquement vrai, le diagnostic $D_{T\text{-}D}$ est valide.

Preuve $\mathcal{A}$ étant correct et complet, $\mathsf{D}_{\mathsf{T\text{-}D}}$ et $\mathsf{D}_{\mathsf{T\text{-}M}}$ sont équivalents, d'où le résultat par le théorème 1.

Théorème 3
Si $\mathcal{A}$ est correct et complet et si (DS,COMPS) est ontologiquement vrai et de plus diagnosticable à partir des capteurs O, alors, pour tout problème (DS,COMPS,OBS) où OBS est ontologiquement vrai, tout diagnostic $D_{T\text{-}D}$ est valide et certain.

Preuve Résultat issu de la définition 17 et du théorème 2.

Reprenons une fois encore l'exemple 1 mais avec la version corrigée de $\mathsf{M}_1\mathsf{desc}$ et modélisons le problème.

6. Dans de nombreuses applications réelles, il est possible, a posteriori, d'observer avec certitude l'état de santé réel ou du moins une partie. Dans le domaine aéronautique, par exemple, cette observation se fait au cours de (voire après) la réparation où des tests systématiques et poussés aboutissent à cette observation.

1. Méta-composants : l'algorithme Alg et la description DS.
2. Propriétés étudiées : diagnosticabilité de DS muni des capteurs O, correction et complétude de Alg.
3. Le théorème 2 donne le bon comportement de Alg à savoir $\neg\mathsf{M\text{-}Ab}(\mathsf{Alg}) \Rightarrow (\sigma_{\text{réel}} \Rightarrow \mathsf{Dis}(\mathsf{D}_{\text{T-D}}))$ où $\mathsf{Dis}(\mathsf{D}_{\text{T-D}})$ est la disjonction représentant l'ensemble $\mathsf{D}_{\text{T-D}}$. Le théorème 3 donne le bon comportement de DS : $\neg\mathsf{M\text{-}Ab}(\mathsf{Alg}) \wedge \neg\mathsf{M\text{-}Ab}(\mathsf{DS}) \Rightarrow (\#\mathsf{D}_{\text{T-D}} = 1)$.
4. Les méta-observables sont $\mathsf{D}_{\text{T-D}}$ et l'état de santé réel $\sigma_{\text{réel}}$.

Maintenant, imaginons que $\sigma_{\text{réel}} = \neg\mathsf{Ab}(\mathsf{M}_1) \wedge \neg\mathsf{Ab}(\mathsf{M}_2) \wedge \neg\mathsf{Ab}(\mathsf{M}_3) \wedge \neg\mathsf{Ab}(\mathsf{A}_1) \wedge \mathsf{Ab}(\mathsf{A}_2)$ et que le diagnostic $\mathsf{D}_{\text{T-D}}$ issu de l'algorithme $\mathcal{A}$ soit l'ensemble des candidats couverts par les diagnostics noyaux suivants : $\{\mathsf{A}_2\}$, $\{\mathsf{A}_1 \wedge \mathsf{M}_2\}$, $\{\mathsf{M}_3\}$ and $\{\mathsf{M}_1 \wedge \mathsf{M}_2\}$. GDE retourne alors les deux diagnostics noyaux suivants : $\{\mathsf{Alg}\}$; $\{\mathsf{DS}\}$.

Ce premier résultat de méta-diagnostic nous informe d'une anomalie sur les méta-composants mais n'est pas satisfaisant. Sachant que le méta-diagnostic est un raisonnement monotone ($\mathsf{DM\text{-}S}$ étant issu de théorèmes mathématiques, et $\mathsf{M\text{-}OBS}$ étant ontologiquement vrai), on peut utiliser d'autres cas de tests pour raffiner le résultat. Supposons par exemple, le nouveau jeu de méta-observations suivant : $\sigma_{\text{réel}} = \neg\mathsf{Ab}(\mathsf{M}_1) \wedge \neg\mathsf{Ab}(\mathsf{M}_2) \wedge \neg\mathsf{Ab}(\mathsf{M}_3) \wedge \neg\mathsf{Ab}(\mathsf{A}_1) \wedge \mathsf{Ab}(\mathsf{A}_2)$ et le diagnostic $\mathsf{D}_{\text{T-D}}$ est l'ensemble des candidats couverts par les diagnostics noyaux suivants : $\{\mathsf{A}_1 \wedge \mathsf{M}_2\}$, $\{\mathsf{M}_3\}$ and $\{\mathsf{M}_1 \wedge \mathsf{M}_2\}$. Il en résulte le méta-diagnostic $\{\mathsf{Alg}\}$ montrant un problème d'incorrection ou d'incomplétude dans l'algorithme et non pas un problème de diagnosticabilité dans DS.

6 Conclusions

Dans cet article, nous avons proposé une théorie générale de méta-diagnostic et avons modélisé et résolu deux problèmes communs de méta-diagnostic. Un problème de méta-diagnostic est posé comme un problème de diagnostic si bien que les outils de diagnostic disponibles et les techniques pour gérer la complexité algorithmique sont les mêmes, comme l'utilisation de GDE le montre ici. Le méta-diagnostic a de nombreuses applications, tant dans le monde académique qu'industriel. Citons par exemple le cas de la maintenance en aéronautique où la détection d'erreurs et la validation automatique d'algorithmes « boîtes noires » sont des enjeux cruciaux.

Références

BELARD N., PENCOLÉ Y. & COMBACAU M. (2010). Defining and exploring properties in diagnostic systems. In *DX-10 21th International Workshop on Principles of Diagnosis*.

CONSOLE L., PICARDI C. & RIBAUDO M. (2000). Diagnosis and diagnosability analysis using pepa. In *ECAI*.

DAVIS R. (1984). Diagnostic reasoning based on structure and behavior. *Artif. Intell.*, **24**(1-3).

DE KLEER J. & WILLIAMS B. C. (1987). Diagnosing multiple faults. *Artif. Intell.*, **32**(1).

FORBUS K. D. & DE KLEER J. (1993). *Building Problem Solvers*. M.I.T. University Press.

HODGES W. (1993). *Model Theory*. Number 42 in Encyclopedia of Mathematics and its Applications. Cambridge University Press.

REITER R. (1987). A theory of diagnosis from first principles. *Artif. Intell.*, **32**(1).

TARSKI A. (1936). The concept of truth in formalized languages. In *Logic, Semantics, Metamathematics*. Oxford : Oxford University Press.

Logique Possibiliste à Intervalles

Salem Benferhat[1], Julien Hué[1], Sylvain Lagrue[1], Julien Rossit[2]

[1] CRIL – CNRS UMR 8188 – Université d'Artois
Faculté des sciences Jean Perrin - Rue Jean Souvraz, SP 18
62307 lens Cedex, France
{benferhat,hue,lagrue}@cril.univ-artois.fr

[2] LIPADE – Université Paris Descartes
45 rue des Saints Pères
75270 Paris Cedex 06, France
julien.rossit@parisdescartes.fr

Résumé : La logique possibiliste est un formalisme permettant de raisonner à partir d'informations incertaines représentées à l'aide de bases de croyances pondérées potentiellement incohérentes. Elle associe pour cela à des formules de la logique propositionnelle des degrés strictement positifs issus de l'intervalle [0,1]. Cependant, en pratique, il peut être difficile pour un expert de fournir les degrés exacts à associer aux formules d'une base de croyances.

Cet article propose une représentation flexible de l'information incertaine dans laquelle les poids associés aux formules sont fournis sous forme d'un intervalle. Nous étudions en premier lieu une approche pour raisonner à partir de bases de croyances possibilistes reposant sur les intervalles, en étendant les principaux concepts de la logique possibiliste tels que les mesures de possibilité ou de nécessité. Nous fournissons ensuite une caractérisation d'une base de croyance possibiliste reposant sur les intervalles, par le biais du concept des bases de croyances possibilistes compatibles.

En particulier, nous montrons que la logique possibiliste reposant sur les intervalles étend la logique possibiliste classique lorsque les intervalles sont des singletons. Finalement, nous fournissons un résultat de complexité calculatoire concernant l'obtention des conclusions plausibles d'une base de croyances possibiliste à intervalles. En particulier, nous montrons que considérer cette flexibilité dans la représentation de l'information incertaine ne nécessite aucun coût calculatoire supplémentaire.

1 Introduction

La logique possibiliste (Dubois *et al.*, 1994; Dubois & Prade, 2004) est un formalisme permettant de représenter et de raisonner à partir d'informations incertaines et incomplètes. D'un point de vue syntaxique, l'incertitude est représentée par un ensemble de formules pondérées de la forme $K = \{\langle \varphi_i, \alpha_i \rangle : i = 1, .., n\}$ où φ_i représente une formule propositionnelle, et α_i un nombre réel appartenant à l'intervalle [0,1]. De cette

façon, la paire $\langle \varphi_i, \alpha_i \rangle$ signifie que φ_i est certaine (ou importante) à un degré d'au moins α_i.

Un mécanisme d'inférence a été proposé dans Lang (2000) afin de dériver les conclusions plausibles d'une base de croyances possibiliste, qui nécessite $log_2(m)$ appels au test de satisfiabilité d'un ensemble de clauses propositionnelles (SAT), où m est le nombre de degrés différents employés par K. D'autre part, l'incertitude est représentée au niveau sémantique en associant un degré de possibilité à chaque monde possible (ou interprétation).

Différentes extensions de la logique possibiliste ont été proposée, en particulier pour remplacer l'intervalle unitaire $[0, 1]$ par un treillis complet ou encore un pré-ordre partiel. Par exemple, dans Lafage *et al.* (1999), un ensemble d'hypothèses supportant une formule est utilisé en lieu et place d'un nombre positif réel. Dans Dubois *et al.* (1992b), les degrés sont remplacés par un ensemble de valeurs positives (pas nécessairement incluses dans $[0, 1]$) représentant une fenêtre de temps dans laquelle les formules sont vraies. Dans Benferhat *et al.* (2004); Benferhat & Prade (2006), une extension est proposée afin de traiter des informations partiellement ordonnées. Cependant, ces extensions peuvent augmenter la complexité calculatoire (en particulier dans le cas d'informations partiellement ordonnées), ou encore ne généralisent pas (au sens strict) la logique possibiliste.

Cet article s'inscrit dans l'esprit de ces extensions de la logique possibiliste. Il étudie les fondements théoriques, fournissant une analyse des problèmes calculatoires d'une logique possibiliste basée sur les intervalles. La question abordée par cet article est de savoir si il est possible d'étendre et d'accroitre la puissance d'expressivité de la logique possibiliste standard en représentant l'incertitude associée aux formules par les moyens d'intervalles, sans pour autant augmenter la complexité calculatoire du processus de raisonnement.

Le formalisme considéré dans cet article est celui de la logique possibiliste basée sur les intervalles. Au niveau syntaxique, les informations sont représentées par une base de croyances possibilistes à intervalles, de la forme $IK = \{\langle \varphi_i, I_i \rangle : i = 1, .., n\}$, où $I_i = [\alpha_i, \beta_i]$ est un sous-intervalle fermé de $]0, 1]$. La paire $\langle \varphi_i, I_i \rangle$, appelée formule pondérée par un intervalle, signifie que le poids associé à φ_i est un des éléments de I_i. Cette interprétation disjonctive de $\langle \varphi_i, I_i \rangle$ ne doit pas être confondue avec l'interprétation conjonctive présentée dans Dubois *et al.* (1992b), qui correspond au fait que pour tout $\alpha_i \in I_i$, $\langle \varphi_i, \alpha_i \rangle$ est vraie. Cette interprétation conjonctive a du sens lorsque l'on considère des intervalles de temps, lorsque $\langle \varphi_i, I_i \rangle$ signifie que φ_i est vraie sur l'intégralité de l'intervalle de temps I_i.

De façon similaire, la sémantique de la logique possibiliste basée sur les intervalles est une distribution de possibilités à intervalles, où un sous-intervalle de [0,1] est associé à chaque interprétation. À l'inverse de la logique possibiliste classique, une distribution de possibilités à intervalles n'induit qu'un préordre *partiel* sur l'ensemble des interprétations.

Sur la base d'une interprétation disjonctive de la notion d'intervalle, nous proposons de voir une base de croyances à intervalles comme une famille de bases de croyances possibilistes standards compatibles. Une base possibiliste compatible est obtenue en considérant un unique poids possible de l'intervalle associé à chaque formule. Un pre-

mier résultat intéressant est que raisonner avec l'ensemble de toutes les bases possibilistes compatibles est équivalent à définir le niveau d'incohérence d'une base de croyances possibiliste à intervalles. Nous montrons également que raisonner à partir de bases de croyances possibilistes à intervalles n'est pas plus coûteux que de raisonner à partir de la logique possibiliste standard : nous étendons cette dernière sans coût calculatoire supplémentaire. Nous fournissons également une caractérisation sémantique où une base de croyances possibiliste à intervalles induit un unique préordre partiel sur l'ensemble des interprétations.

La Section 2 présente une brève introduction à la logique possibiliste. La Section 3 ainsi que la Section 4 étudient les représentations sémantique puis syntaxique de la logique possibiliste basée sur les intervalles. La Section 5 étudie le processus d'inférence et présente les résultats en termes de complexité calculatoire qui lui sont associées. Enfin, la Section 6 discute de travaux apparentés.

2 Quelques rappels sur la logique possibiliste

2.1 Distributions de possibilité

Nous considérons un langage propositionnel $\mathcal{L}$ utilisant un nombre *fini* de variables propositionnelles. Nous notons Ω l'ensemble fini des interprétations de $\mathcal{L}$ et ω désigne un élément de Ω.

Une distribution de possibilité, notée π, est une fonction de Ω vers $[0, 1]$. $\pi(\omega)$ représente le degré de compatibilité (ou cohérence) d'une interprétation ω avec les connaissances disponibles. $\pi(\omega) = 1$ signifie que ω est totalement compatible avec les connaissances disponibles, là où $\pi(\omega) = 0$ signifie que ω est impossible. $\pi(\omega) > \pi(\omega')$ signifie simplement que ω est plus compatible que ω'. Une distribution de possibilité π est dite normalisée si il existe une interprétation ω telle que $\pi(\omega) = 1$. Celle-ci est dite sous-normalisée sinon. Les distributions de possibilité sous-normalisées encodent des ensembles de croyances incohérents.

Une distribution de possibilité permet de définir deux fonctions de $\mathcal{L}$ vers $[0, 1]$ appelées mesures de possibilité et de nécessité, notée Π et N, définies par :

$$\Pi(\varphi) = \max\{\pi(\omega) : \omega \in \Omega,\ \omega \models \varphi\},\ \text{ et}$$

$$N(\varphi) = 1 - \Pi(\neg\varphi).$$

$\Pi(\varphi)$ indique dans quelle mesure la formule φ est compatible avec l'ensemble des connaissances disponibles, alors que $N(\varphi)$ apprécie dans quelle mesure celle-ci peut être déduite.

Étant donnée une distribution de possibilité π encodant un ensemble de connaissances disponibles, une formule φ est dite conséquence de π, noté $\pi \models_\pi \varphi$, si et seulement si $\Pi(\varphi) > \Pi(\neg\varphi)$. Intuitivement, φ est conséquence de π si les meilleurs modèles de φ (les modèles de φ associés aux degrés les plus forts) sont plus plausibles (ou préférés) que les meilleurs modèles de $\neg\varphi$.

2.2 Base de croyances possibiliste

Une formule possibiliste est un tuple $\langle\varphi,\alpha\rangle$ où φ est un élément de $\mathcal{L}$ et $\alpha \in (0,1]$ est une valuation de φ représentant $N(\varphi)$. Remarquons qu'il est possible de ne considérer aucune formule de la forme $\langle\varphi, 0\rangle$, dans la mesure où celles-ci n'apportent aucune information. Une base de croyances $K = \{\langle\varphi_i, \alpha_i\rangle, 1 \leq i \leq n\}$ est un ensemble de formules possibilistes.

Une notion importante, jouant un rôle central dans le processus d'inférence, est celle d'α-coupe stricte. Une α-coupe stricte, notée K_α, est l'ensemble des formules propositionnelles définies par $K_\alpha = \{\varphi : \langle\varphi,\beta\rangle \in K$ et $\beta > \alpha\}$.

L'α-coupe stricte est une notion utile pour mesurer le degré de cohérence de K, définie par $Inc(K) = \max\{\alpha : K_\alpha$ en cas d'incohérence, $\alpha = 0\}$ sinon.

Si $Inc(K) = 0$, alors K est dite totalement cohérente. Si une base possibiliste est partialement incohérente, alors $Inc(K)$ peut être vu comme un seuil à partir duquel toute formule est considérée comme non suffisamment fondée pour être prise en compte dans le processus d'inférence. Plus précisément, nous définissons la notion de noyau d'une base de croyances comme étant composé des formules associées à un degré de certitude supérieur à $Inc(K)$, soit formellement :

$$\mathcal{C}ore(K) = K_{Inc(K)} = \{\varphi : \langle\varphi,\alpha\rangle \in K \text{ et } \alpha > Inc(K)\}$$

Une formule φ est une conséquence d'une base possibiliste K, dénoté $K \vdash_\pi \varphi$, si et seulement si $\mathcal{C}ore(K) \vdash \varphi$.

Une base de croyances possibiliste est une représentation compacte répandue d'une distribution de possibilité. Étant donnée une base possibiliste K, nous pouvons générer une unique distribution possibiliste où chaque interprétation ω satisfaisant toute formule propositionnelle de K est associée au degré le plus haut possible, soit $\pi(\omega) = 1$ (puisqu'elles sont totalement cohérentes), là où les autres sont ordonnées vis-à-vis de la formule la plus importante qu'elles falsifient. Plus formellement :

$$\forall\omega \in \Omega, \pi_K(\omega) = \begin{cases} 1 \text{ si } \forall\langle\varphi,\alpha\rangle \in K, \omega \models \varphi \\ 1 - max\{\alpha_i : \langle\varphi_i, \alpha_i\rangle \in K, \omega \not\models \varphi_i\} \text{ sinon.} \end{cases}$$

Le résultat de complétude et d'adéquation suivant est vérifié :

$$K \vdash_\pi \varphi \text{ ssi } \pi_K \models_\pi \varphi.$$

3 Distributions de possibilités à intervalles

L'objectif de cette section est l'étude d'un cadre plus générale, où l'incertitude n'est pas encodée au moyen d'une simple valeur de nécessité mais par le biais d'un intervalle de degrés possibles. Ce cadre permet l'introduction d'une imprécision sur la priorité associée aux croyances. Nous utilisons des intervalles de nombres réels de la forme $I = [\alpha,\beta] \subseteq [0,1]$ afin d'encoder l'incertitude associée aux formules. Nous notons $\mathcal{I}$ l'ensemble de tous les intervalles inclus dans $[0,1]$.

Opérations sur les intervalles Étant donnés $I_1 = [\alpha_1, \beta_1]$ et $I_2 = [\alpha_2, \beta_2]$ deux intervalles, nous définissons les opérations suivantes qui seront utilisées dans la suite, et

garantissent la cohérence de cet article. En particulier, elles permettent l'extension de la logique possibiliste standard lorsque les intervalles sont des singletons de la forme $[\alpha,\alpha]$.

- Maximum d'intervalles : étant donné un ensemble d'intervalles $I_i = [\alpha_i, \beta_i]$ $\mathcal{M}\{I_1, \ldots, I_n\} = [\max\{\alpha_1, \ldots, \alpha_n\}, \max\{\beta_1, \ldots, \beta_n\}]$
- Inverse d'un intervalle : $1 \ominus I_1 = [1 - \beta_1, 1 - \alpha_1]$
- Comparaison d'intervalles : $I_1 \lhd I_2$ ssi $\beta_1 < \alpha_2$

Intuitivement, quand I_1 et I_2 sont associés à deux formules φ_1 et φ_2, alors $I_1 \lhd I_2$ signifie que la formule φ_2 est strictement préférée à φ_1. Cette attitude est clairement une interprétation sûre (mais prudente) de la notion de préférence. En effet, en utilisant la relation $\lhd$, $\langle \varphi_2, I_2 \rangle$ est préférée à $\langle \varphi_1, I_1 \rangle$ si quelque soir le degré assigné à φ_2 en provenance de I_2, il sera plus grand que le degré associé à φ_1 obtenue de I_1.

Distributions de possibilités à intervalles et compatibles Une distribution de possibilités à intervalles, notée $\pi_{\mathcal{I}}$, est une fonction de Ω vers $\mathcal{I}$. $\pi_{\mathcal{I}}(\omega) = I$ signifie que le degré de possibilité associé à ω est un des éléments de I. $\pi_{\mathcal{I}}$ n'induit seulement qu'un préordre partiel sur les interprétations, défini par $\omega < \omega'$ (ω' est préférée à ω) si et seulement si $\pi_{\mathcal{I}}(\omega) \lhd \pi_{\mathcal{I}}(\omega')$. Puisque $\lhd$ est un préordre partiel, une distribution de possibilités à intervalles n'induit qu'un préordre partiel sur les interprétations. Une distribution de possibilités à intervalles peut être vue comme une famille de distributions de possibilités standard compatibles définie par :

Définition 1
Soit $\pi_{\mathcal{I}}$ une distribution de possibilité à intervalles. Une distribution de possibilité π est dite compatible avec $\pi_{\mathcal{I}}$ si et seulement si $\forall \omega \in \Omega,\ \pi(\omega) \in \pi_{\mathcal{I}}(\omega)$.

Il n'existe pas une distribution compatible unique. Nous notons $\mathcal{C}mp(\pi_{\mathcal{I}})$ l'ensemble de toutes les distributions de possibilités compatibles avec $\pi_{\mathcal{I}}$. Une distribution de possibilités à intervalles est normalisée s'il existe une interprétation ω, telle que $\pi_{\mathcal{I}}(\omega) = [1, 1]$. Il peut être constater que $\pi_{\mathcal{I}}$ est normalisée si et seulement si toutes ses distributions compatibles sont normalisées.

Mesures de nécessité et de possibilité Une façon naturelle de définir la contrepartie des mesures de possibilité et de nécessité associées à une formule φ à partir d'une distribution de possibilité à intervalles est de prendre en compte l'ensemble de toutes les distributions compatibles, c'est à dire :

Définition 2
Soit $\pi_{\mathcal{I}}$ une distribution de possibilité à intervalles et soit φ une formule. Alors :

$$\Pi_{\mathcal{I}}(\varphi) = [\min_{\pi \in \mathcal{C}mp(\pi_{\mathcal{I}})} \Pi(\varphi), \max_{\pi \in \mathcal{C}mp(\pi_{\mathcal{I}})} \Pi(\varphi)],\ \textit{et}$$

$$N_{\mathcal{I}}(\varphi) = [\min_{\pi \in \mathcal{C}mp(\pi_{\mathcal{I}})} N(\varphi), \max_{\pi \in \mathcal{C}mp(\pi_{\mathcal{I}})} N(\varphi)]$$

L'intervalle des degrés de possibilité associé à une formule représente toutes les valeurs possibles qui peuvent être obtenues à partir de l'ensemble des distributions compatibles. La proposition suivante montre que les degrés de possibilité et de nécessité peuvent être calculés directement et de façon équivalente en utilisant les distributions de possibilités à intervalles :

Proposition 1
Soit $\pi_{\mathcal{I}}$ une distribution de possibilités à intervalles, et soit φ une formule. Alors :

$$\Pi_{\mathcal{I}}(\varphi) = \mathcal{M}\{\pi_{\mathcal{I}}(\omega) : \omega \in \Omega, \omega \models \varphi\} \text{ et}$$

$$N_{\mathcal{I}}(\varphi) = 1 \ominus \Pi_{\mathcal{I}}(\neg\varphi)$$

avec $1\ominus$ et $\mathcal{M}$ respectivement les opérations inverse et maximum sur les intervalles définies précédemment.

Il est possible de montrer que dans des cas particuliers où les intervalles fournis par les distributions de possibilités sont des singletons, notre approche coïncide avec les définitions standard des mesures de la logique possibiliste. Formellement :

Proposition 2
Dans les cas où les intervalles de $\pi_{\mathcal{I}}$ sont seulement des singletons (en d'autres termes, quels que soient $\omega, \pi_{\mathcal{I}}(\omega) = [\alpha, \alpha]$) alors :

(i) *$\pi_{\mathcal{I}}$ possède une unique distribution de possibilités compatibles π et*

(ii) *$\Pi_{\mathcal{I}}(\varphi) = [\Pi(\varphi), \Pi(\varphi)]$ et $N_{\mathcal{I}}(\varphi) = [N(\varphi), N(\varphi)]$ où N et Π sont les mesures possibilistes standards.*

Définition 3
Étant donnée une distribution de possibilités à intervalles $\pi_{\mathcal{I}}$, une formule φ est dite acceptée, ou conséquence de $\pi_{\mathcal{I}}$, noté $\pi_{\mathcal{I}} \models_{\mathcal{I}} \varphi$, si et seulement si $\Pi_{\mathcal{I}}(\neg\varphi) \lhd \Pi_{\mathcal{I}}(\varphi)$.

Il peut être montré que le Définition 3 peut être reformulée au moyen de distributions de possibilité compatibles, formellement :

$$\pi_{\mathcal{I}} \models_{\mathcal{I}} \varphi \text{ ssi } \forall \pi \in \mathcal{C}mp(\pi_{\mathcal{I}}), \Pi(\varphi) > \Pi(\neg\varphi).$$

D'un point de vue sémantique, l'utilisation de distributions de possibilités compatibles représente une justification solide des concepts principaux de la logique des possibilités à intervalles : les mesures de possibilité et de nécessité à intervalles, et la condition de normalisation. La section suivante montre que ceci se tient au niveau syntaxique de la représentation de la logique possibiliste à intervalles.

4 Base possibiliste à intervalles

Nous étudions dans cette section la représentation syntaxique de la logique possibiliste basée sur les intervalles. Pour cela, nous généralisons la notion de base possibiliste à celle de base de croyances possibiliste à intervalles.

Définition 4 (Base possibiliste à intervalles)
Une base possibiliste à intervalles, notée IK, est un multi-ensemble de formules associées à des intervalles :

$$IK = \{\langle \varphi, I \rangle, \varphi \in \mathcal{L} \text{ et } I \in \mathcal{I}\}$$

Cette représentation intuitive de $\langle \varphi, I \rangle$ signifie que le degré associé à φ est un des éléments de $I = [\alpha,\beta]$. De façon similaire à la logique possibiliste classique, nous ne permettons pas à α de valoir 0. De cette façon, nous ne représentons explicitement que les informations acceptées. L'utilisation d'intervalles ouverts, ou d'intervalles incluant 0 fera l'objet de futures recherches.

Exemple 1
Nous utiliserons la base de croyance possibiliste à intervalles suivante pour illustrer les concepts principaux de cet article :

$$IK = \{\langle a, [.7, .9]\rangle, \langle a \wedge b, [.55, .8]\rangle, \langle \neg a, [.5, .6]\rangle, \langle \neg b, [.1, .3]\rangle\}$$

4.1 Bases compatibles

Une base possibiliste à intervalles IK peut être vue comme une famille de bases possibilistes standard, appelées bases compatibles. Une base possibiliste à intervalles K est dite compatible avec IK si et seulement si il existe une bijection de IK vers K telle que pour chaque formule associée avec un intervalle I de IK, le degré de cette formule dans K est un élément de I. Plus formellement :

Définition 5 (Base possibiliste à intervalles)
Une base possibiliste K est dite compatible avec une base possibiliste à intervalles IK si et seulement si il existe une bijection f de IK vers K telle que $\{f(\langle \varphi, [\alpha,\beta]\rangle) = \langle \varphi,\delta \rangle \in K$ t.q. $\alpha \leq \delta \leq \beta\}$

En d'autres termes, les bases possibilistes compatibles sont obtenues à partir des bases possibilistes à intervalles en remplaçant chaque formule possibiliste à intervalles $\langle \varphi, I\rangle$ par une formule possibiliste classique $\langle \varphi,\delta \rangle$ où $\delta \in I$. Chaque base possibiliste compatible est telle que $K = \{\langle \varphi,\delta \rangle \ : \ \langle \varphi, I\rangle \in IK$ et $\delta \in I\}$.

Nous notons $\mathcal{C}mp(IK)$ l'ensemble infini des bases possibilistes compatibles associées à une base possibiliste à intervalles IK.

Étant donnée une base possibiliste à intervalles IK, nous définissons deux bases compatibles particulières, notées IK_{lb} et IK_{ub}, en sélectionnant respectivement soit la borne inférieure de l'intervalle (point de vue pessimiste), soit la borne supérieure (point de vue optimiste) :

(i) $IK_{lb} = \{\langle \varphi,\alpha \rangle \ : \ \langle \varphi, [\alpha,\beta]\rangle \in IK\}$

(ii) $IK_{ub} = \{\langle \varphi,\beta \rangle \ : \ \langle \varphi, [\alpha,\beta]\rangle \in IK\}$

Exemple 2
Le Tableau 1 présente cinq exemples de bases possibilistes compatibles. La valeur de nécessité associée à chaque formule de chacune de ces bases standard appartient à l'intervalle qui leur est respectivement associé dans IK. Les bases compatibles K_1 et K_5 fournies par le Tableau 1 correspondent respectivement à IK_{lb} et IK_{ub}.

IK		K_1	K_2	K_3	K_4	K_5
a	[.7,.9]	.7	.7	.8	.8	.9
$a \wedge b$	[.55,.8]	.55	.6	.6	.6	.8
$\neg a$	[.5,.6]	.5	.55	.56	.58	.6
$\neg b$	[.1,.3]	.1	.2	.2	.2	.3

TABLE 1 – Exemples de bases compatibles

4.2 Inférence à partir des bases compatibles

La relation d'inférence à partir d'une base possibiliste à intervalles peut être définie à partir de l'ensemble des bases possibilistes compatibles. En d'autres termes, une formule φ est une conclusion plausible d'une base possibiliste à intervalles IK si et seulement si elle peut être déduite de chaque base possibiliste appartenant à $\mathcal{C}mp(IK)$. Formellement :

$$IK \vdash_c \varphi \text{ ssi } \forall K \in \mathcal{C}mp(IK), K \vdash_\pi \varphi$$

où $\vdash_\pi$ est défini Section 2.2.

4.3 Des bases possibilistes à intervalles aux distributions possibilistes à intervalles

De même que dans le cadre des bases possibilistes standard, une base de croyances possibiliste à intervalles IK permet également la représentation compacte d'une distribution de possibilités à intervalles π_{IK}. Cette distribution de possibilités peut être obtenue par le biais : i) d'une extension de la définition de π_K donnée par la Section 2.2 afin de manipuler les intervalles, ii) des distributions de possibilités obtenues des bases compatibles, et iii) des deux bases compatibles particulières IK_{lb} et IK_{ub}. Ceci est résumé par la Définition 6 et la Proposition 3

Définition 6 (Distribution de possibilités à intervalles)
Soit IK une base possibiliste à intervalles. Alors :

$$\pi_{IK}(\omega) = [\min_{K \in \mathcal{C}mp(IK)} \pi_K(\omega), \max_{K \in \mathcal{C}mp(IK)} \pi_K(\omega)]$$

où π_K est la distribution possibiliste standard associée à la base compatible K.

Proposition 3
La définition de π_{IK} de la Définition 6 est équivalente à :

i) $\pi_{IK}(\omega) = [\pi_{IK_{ub}}(\omega), \pi_{IK_{lb}}(\omega)]$, *en d'autres termes $\pi_{IK}(\omega)$ est bornées par les poids associés à ω respectivement par les bases particulières IK_{lb} et IK_{ub}*

ii) $\pi_{IK}(\omega) = \begin{cases} [1,1] \text{ si } \forall(\varphi, I) \in IK, \omega \models \varphi \\ 1 - \mathcal{M}\{I : (\varphi, I) \in IK, \omega \not\models \varphi\} \text{ sinon.} \end{cases}$

Un résultat important se trouve dans le fait que la relation d'inférence à partir de π_{IK} est équivalente à considérer la relation d'inférence basée sur l'ensemble des bases compatibles. En d'autres termes, le résultat de cohérence et de complétude est également vérifié dans le cadre de la logique possibiliste basée sur les intervalles :

Proposition 4
Soit IK une base possibiliste à intervalles. Alors :

$$IK \models_c \varphi \quad \textit{ssi} \quad \Pi_{IK}(\neg\varphi) \lhd \Pi_{IK}(\varphi).$$

Exemple 3
L'Exemple 1, nous donne $IK_{lb} = \{\langle a, .7\rangle, \langle a \wedge b, .55\rangle, \langle \neg a, .5\rangle, \langle \neg b, .1\rangle\}$ et $IK_{ub} = \{\langle a, .9\rangle, \langle a \wedge b, .8\rangle, \langle \neg a, .6\rangle, \langle \neg b, .3\rangle\}$. Le Tableau 2 fournit les distributions de possibilités induites par IK_{lb} et IK_{ub}.

$\omega_i \in \Omega$	a	b	$\pi_{IK_{lb}}(\omega_i)$	$\pi_{IK_{ub}}(\omega_i)$	$\pi_{IK}(\omega_i)$
ω_0	0	0	.3	.1	[.1,.3]
ω_1	0	1	.3	.1	[.1,.3]
ω_2	1	0	.45	.2	[.2,.45]
ω_3	1	1	.5	.4	[.4,.5]

TABLE 2 – Distributions de possibilités à intervalles

D'après le Tableau 2, nous avons, par exemple, $\pi_{IK} \models a$ puisque $\Pi_{IK}(\neg a) = [.5, .6] \lhd \Pi_{IK}(a) = [.7, .9]$.

5 Analyse d'un point de vue calculatoire

5.1 Degré d'incohérence

La section précédente fournit la définition d'une relation d'inférence à partir d'une base de croyances possibiliste à intervalles, utilisant le concept de base compatible ou de la distribution de possibilités qui lui est associée. Cette section se concentre sur les problèmes calculatoires en proposant une caractérisation syntaxique de cette relation d'inférence.

Cette caractérisation est basée sur une extension naturelle de la notion de degré d'incohérence, et de celle du noyau d'une base de croyance possibiliste utilisé par la logique possibiliste classique (voir Section 2).

À nouveau, une façon de définir le degré d'incohérence associé à une base de croyances possibiliste à intervalles est de considérer l'ensemble des degrés d'incohérences associés à chaque base possibiliste compatible avec IK. Formellement :

Définition 7 (Degré d'incohérence basé sur les intervalles)
Soit IK une base possibiliste à intervalles. Alors :

$$Inc(IK) = \{Inc(K) \ : \ K \in \mathcal{C}mp_{IK}\}$$

La proposition suivante montre que $Inc(IK)$, l'ensemble des degrés d'incohérence associés à chaque base compatible, constitue un intervalle. Formellement :

Proposition 5
Soit IK une base possibiliste à intervalles. Alors : $Inc(IK) = [Inc(IK_{lb}), Inc(IK_{ub})]$.

5.2 Noyau d'une base possibiliste à intervalles

Le noyau de IK est intuitivement l'ensemble des formules propositionnelles dont les intervalles associés sont plus grand que $Inc(IK)$ au sens de $\triangleleft$ fourni par la Section 4.2. Formellement :

Définition 8
Soit IK une base possibiliste à intervalles. Alors :

$$\mathcal{C}ore(IK) = \{\varphi \ : \ \langle \varphi, I \rangle \in IK \text{ et } Inc(IK) \triangleleft I\}$$

La Proposition 6 ci-après montre que $\mathcal{C}ore(IK)$ est cohérent et est inclus dans le noyau de chaque base compatible, soit encore :

Proposition 6
Soit IK une base possibiliste à intervalles. Alors :

i) $\mathcal{C}ore(IK)$ *est cohérent ;*

ii) $\forall K \in \mathcal{C}mp_{IK}, \mathcal{C}ore(IK) \subseteq \mathcal{C}ore(K)$.

Finalement, la proposition suivante montre que les conclusions plausibles dérivées de $\mathcal{C}ore(IK)$ sont les mêmes que celles obtenues de l'intégralité de l'ensemble des bases compatibles. En corollaire, elles sont également les mêmes que celles fournies au niveau sémantique.

Proposition 7
Soit IK une base possibiliste à intervalles. Alors : $\forall \psi \in \mathcal{L}$:

$$IK \models_c \psi \text{ ssi } \mathcal{C}ore(IK) \vdash \psi \text{ ssi } \forall K \in \mathcal{C}mp_{IK}, \mathcal{C}ore(K) \vdash \psi$$

Exemple 4
Considérant à nouveau l'exemple 1, nous avons $IK_{lb} = \{\langle a, .7\rangle, \langle a \wedge b, .55\rangle, \langle \neg a, .5\rangle, \langle \neg b, .1\rangle\}$ et $IK_{ub} = \{\langle a, .9\rangle, \langle a \wedge b, .8\rangle, \langle \neg a, .6\rangle, \langle \neg b, .3\rangle\}$. Ainsi, nous obtenons $Inc(IK_{lb}) = .5$ et $Inc(IK_{ub}) = .6$, et ainsi $Inc(IK) = [.5, .6]$.

De cette façon, $\mathcal{C}ore(IK) = \{a\}$. Il est possible de vérifier que les conclusions obtenues de l'Exemple 3 sont les mêmes que celles obtenues en considérant $\mathcal{C}ore(IK)$.

5.3 Complexité calculatoire

Cette sous-section analyse le complexité calculatoire du processus d'inférence introduit dans cet article. Nous rappelons que le problème de décision « Une formule ψ est elle la conséquence d'une base possibiliste standard ? » est un problème Δ_p^2-complet (Nebel (1994)). Puisque lorsque les intervalles sont des singletons, l'inférence à partir de bases possibilistes à intervalles correspond à celle utilisé dans le cadre de la logique possibiliste standard, notre relation d'inférence est au moins Δ_p^2. En fait, celle-ci est exactement $\Delta_p^2 - complet$:

Proposition 8
Le problème de décision « Une formule ψ est-elle la conséquence d'une base possibiliste à intervalles ? « est un problème $\Delta_p^2 - complet$.

La Proposition 8 montre que les bases possibilistes à intervalles offrent un cadre plus flexible pour la représentation des informations incertaines sans coût calculatoire supplémentaire. La complexité calculatoire du raisonnement à partir de telles bases est en effet du même ordre de magnitude que celle observée dans le cas de bases possibilistes classiques.

Algorithme 1 Calculer les conséquences d'une base possibiliste à intervalles

ENTRÉES: IK, ψ
SORTIES: Vrai si ψ est une conséquence de IK. Faux sinon.
1: Calculer IK_{lb}
2: Calculer $Inc(IK_{lb})$
3: Calculer IK_{ub}
4: Calculer $Inc(IK_{ub})$
5: $Inc(IK) \leftarrow [Inc(IK_{lb}), Inc(IK_{ub})]$
6: $\mathcal{C}ore(IK) \leftarrow \emptyset$
7: **pour tout** $\langle \varphi, I \rangle \in IK$ **faire**
8: **si** $Inc(IK) \triangleleft I$ **alors**
9: $\mathcal{C}ore(IK) \leftarrow \mathcal{C}ore(IK) \cup \{\varphi\}$
10: **finsi**
11: **fin pour**
12: **renvoyer** $\mathcal{C}ore_{IK} \vdash \psi$

Détaillons maintenant l'Algorithme 1). Celui-ci est de complexité $O(log_2(n) \times SAT)$ où SAT est le test de satisfiabilité propositionnelle et $n = \max(m_{lb}, m_{ub})$, où m_{lb} (resp. m_{ub}) est le nombre de degré différents dans IK_{lb} (resp. IK_{ub}). L'algorithme calcule d'abord $\mathcal{C}ore(IK)$. Cette étape est accomplie en déterminant IK_{lb} et IK_{ub} et leur degré d'incohérence associés (Étapes 1-5). Ces étapes nécessitent $2 \times log_2(n)$ appels à un oracle SAT. Les Étapes 6-10 calculent $\mathcal{C}ore(IK)$ depuis IK en considérant les formules propositionnelles φ dont l'intervalle associé est plus grand que $Inc(IK)$. Cette étape est de complexité $O(|IK|)$ où $|IK|$ est le nombre de formules de IK. La dernière étape nécessite un unique appel à SAT afin de vérifier si ψ est une conséquence de $\mathcal{C}ore(IK)$. Ainsi, dériver les conclusions plausible peut être accompli en $O(log_2(N))$ appels à la résolution d'un problème SAT.

6 Travaux associés

Dans Dubois *et al.* (1992b), une extension de la logique possibiliste, *Timed Possibilistic Logic* (TPL), a été proposée afin de gérer des informations temporelles. Les formules sont pondérées grâce à des ensembles finis de valeurs dont les bornes sont des nombres réels quelconques (les intervalles n'appartiennent pas forcément à $[0, 1]$). Elles représentent des trames temporelles où les formules sont vraies, autrement dit la valeur de nécessité de la formule est positive pour la formule dans l'intervalle de temps associé. La sémantique sous-jacente diffère de la notre dans le sens où nous considérons une interprétation disjonctive en ne considérant comme vrai qu'un seul élément de l'intervalle. Par conséquent, les outils et les opérateurs utilisés dans Dubois *et al.* (1992b) (comme les opérateurs d'union et d'intersection) diffèrent de ceux que nous avons utilisés.

D'un point de vue plus technique (et indépendamment des problème de représentation) notre approche étend la logique possibiliste classique, ce qui n'est pas forcément le cas de TPL. Par exemple, si ω_1 et ω_2 sont les seuls modèles d'une formule φ, et si $\pi(\omega_1) = [.3, .3]$, $\pi(\omega_2) = [.4, .4]$ (c'est-à-dire que les intervalles sont réduits à des singletons), alors avec notre approche nous obtenons $\Pi(\varphi) = [.4, .4]$, alors qu'en utilisant celle décrite dans Dubois *et al.* (1992b) nous obtenons $\Pi(\varphi) = [.3, .4]$, qui n'est plus un singleton. Néanmoins, les différences entre nos approches respectives s'expliquent par l'interprétation différente que nous associons aux intervalles : TPL est définie pour des informations temporelles alors que la notre est plus orientée vers des situations où l'incertitude associée aux formules est imprécise.

La *logique des supporters* proposée dans Lafage *et al.* (1999) est très proche de celle donnée dans Dubois *et al.* (1992b). La différence principale se situe une nouvelle fois dans la signification des intervalles : ils sont ici considérés en tant que justification (dans le sens d'un ATMS) d'une formule. Cette approche est également conjonctive ne permets pas réellement de traiter des imprécisions.

Dans un article plus récent, Marquis & Öztürk (2009) ont proposés de représenter les préférences grâce à des formules pondérées par des intervalles. Les intervalles ne représentent pas un degré associé à la formule mais une connaissance sur une valeur numérique quelconque. Par exemple, une formule $\langle hauteur, \geq, 6, \leq, 7\rangle$ représente la connaissance qu'une valeur $hauteur$ est incluse entre 6 et 7. L'objectif de cet article est de représenter de manière compacte les ordres d'intervalles associés aux interprétations en utilisant ces formules pondérées. Ils ont étudiés certaines familles d'opérateurs d'agrégation et fournis plusieurs résultats concernant la complexité. Leur travail est orienté vers le problème de décision et de représentation compacte de préférence ; ils ne traitent pas de problèmes d'inférence.

Conclusion

Cet article propose les fondations pour le raisonnement à l'aide de bases possibilistes à intervalles. La méthode proposée étend la logique possibiliste classique dans le cas où les intervalles sont des singletons. La flexibilité dans la représentation des informations incertaines est obtenues en utilisant le concept de bases compatibles. Nous avons

montré que les mesures de possibilité et de nécessité ainsi que la distribution de possibilité peuvent être calculés simplement et sans coût calculatoire supplémentaire. Ainsi, le problème de décision dans le cadre de la logique possibiliste à intervalles est Δ_p^2-complet. Finalement, lorsque les intervalles sont des singletons, toutes les relations de conséquences données dans cet article sont équivalentes à celle de la logique possibiliste classique.

Ce travail ouvre différentes perspectives. Par exemple, il a été brièvement proposé dans Dubois & Prade (2004) une extension du principe de résolution aux intervalles. Il serait intéressant de comparer cette définition à notre cadre. Il serait également intéressant d'étudier ce principe de résolution par rapport à différentes extensions basées sur d'autres ordres sur les intervalles et proposées dans Benferhat *et al.* (2011) (dont d'ailleurs ce présent article reprend les principaux résultats). Un autre problème intéressant serait d'établir les liens entre notre formalisme et celui de la logique possibiliste multi-source présentée dans Dubois *et al.* (1992a). Enfin, un travail futur consisterait en l'étude de la fusion, de la révision et du conditionnement dans le cadre de la logique possibiliste à intervalles.

Références

BENFERHAT S., HUÉ J., LAGRUE S. & ROSSIT J. (2011). Interval-based possibilistic logic. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11)*. To appear.

BENFERHAT S., LAGRUE S. & PAPINI O. (2004). Reasoning with partially ordered information in a possibilistic framework. *Fuzzy Sets and Systems*, **144**, 25–41.

BENFERHAT S. & PRADE H. (2006). Compiling possibilistic knowledge bases. In *Proc. of ECAI'06*, p. 337–341.

DUBOIS D., LANG J. & PRADE H. (1992a). Dealing with multi-source information in possibilistic logic. In *Proceedings of the 10th European Conference on Artificial Intelligence (ECAI'92)*, p. 38–42.

DUBOIS D., LANG J. & PRADE H. (1992b). Timed possibilistic logic. *Fundamenta Informaticae*, **XV**, 211–234.

DUBOIS D., LANG J. & PRADE H. (1994). Possibilistic logic. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, p. 439–513.

DUBOIS D. & PRADE H. (2004). Possibilistic logic : a retrospective and prospective view. *Fuzzy sets and systems*, **144**, 3–23.

LAFAGE C., LANG J. & SABBADIN R. (1999). A logic of supporters. In *Information, uncertainty and fusion*, p. 381–392. Springer.

LANG J. (2000). Possibilistic logic : complexity and algorithms. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 5, p. 179–220. Kluwer Academic.

MARQUIS P. & ÖZTÜRK M. (2009). Representing interval orders by weighted bases : Some complexity results. *Mathematical Social Sciences*, **57**, 367–388.

NEBEL B. (1994). Base revision operations and schemes : semantics, representation and complexity. In *Proc. of ECAI'94*, p. 341–345.

Caractérisation multiples minimales utilisant les formules booléennes partiellement définies

Fabien Chhel, Adrien Goëffon, Frédéric Lardeux, and Frédéric Saubion

LERIA, Université d'Angers, France
prenom.nom@univ-angers.fr

Résumé : Dans cet article, nous proposons de calculer un ensemble de formules booléennes partiellement définies qui correspond à la recherche de caractérisations de groupes de vecteurs booléens selon un ensemble de caractères (variables). Ce problème peut être utile pour de nombreuses applications pratiques où les données sont des résultats expérimentaux de tests de présence/absence. Ce problème a été largement étudié, mais généralement dans un cas plus limité, car ici nous avons à calculer simultanément un ensemble complet de formules de caractérisation minimale. Nous proposons donc deux approches différentes qui nous permettent d'étudier la satisfiabilité et la complexité sous-jacente de ce problème. Afin de mettre en évidence l'applicabilité de notre travail, nous fournissons aussi des résultats expérimentaux que nous avons obtenus à partir de données biologiques.

1 Introduction

L'apprentissage automatique de fonctions booléennes à partir d'exemples a été largement étudié pendant de nombreuses années, des travaux initiaux de Valiant (1984), Natarajan (1987), à d'autres plus récents comme Gavaldà & Thérien (2009). L'acquisition de connaissances comme des ensembles de vecteurs booléens est en effet un moyen facile de collecter les résultats d'expériences dans divers domaines d'application. Comme il sera illustré dans le présent document, les entités qui sont modélisées comme des vecteurs de booléens correspondent aux résultats d'une série de tests de diagnostic où la présence ou l'absence de caractères (les variables du problème) sont observées. L'objectif général est alors de fournir une formule booléenne qui est satisfaite par un ensemble d'entités (les exemples positifs) et qui est falsifiée par un autre ensemble (les exemples négatifs). La formule qui en résulte peut être exprimée en des syntaxes différentes (CNF, DNF, fonctions booléennes, BDD, ...). Les deux ensembles constituent une définition partielle de la formule. Bien entendu, de nombreuses autres techniques provenant de l'apprentissage peuvent être utilisées pour apprendre à partir d'exemples, telles que les classificateurs (par exemple, Support Vecteur Machine de Meyer *et al.* (2003)) mais elles n'apportent pas toujours des résultats précis et il est difficile de les utilisées sur un grand ensemble de variables. Le système FOIL de Quinlan (1990) a été conçu pour apprendre des clauses de Horn à partir d'exemples et peut

apparaître bien adapté à notre problème, mais la notion de minimalité n'est pas traitée avec ce système.

Dans cet article, nous nous concentrons sur une extension de ce problème initial en considérant simultanément plusieurs ensembles d'entités. Cette extension induit un supplément de niveau combinatoire. Informellement, on considère un ensemble $\mathcal{G}$ de n ensembles (appelés groupes) g_i d'entités. Chaque g_i doit être satisfait par une formule booléenne, et falsifié par toutes les autres entités qui sont dans $\mathcal{G} \setminus g_i$. Cela peut être interprété comme une caractérisation multiple exclusive des n groupes d'entités. En outre, nous exigeons que le nombre de variables utilisées dans chaque formule soit minimisé. La minimisation des expressions booléennes a sont importance, par exemple dans la conception de circuit électronique, et a sollicité beaucoup de travail afin de fournir des techniques d'optimisation efficaces. Néanmoins, notez que notre problème diffère du principal objectif de la conception de circuits électroniques dont le but est de réduire au maximum les composants logiques du circuit par rapport à un nombre fixe d'entrées et sorties (McCluskey (1956)). En effet, notre problème reviendrait à minimiser le nombre d'entrées en gardant le nombre de composants logiques fixe.

Nous étudions ici deux façons de traiter le problème qui correspondent à deux possibles formulations comme illustré par la figure 1. Le problème initial est défini par une matrice booléenne. Dans la partie gauche, nous abordons le problème par des techniques dont les formules booléennes sont le support. Nous étudierons d'abord la satisfiabilité du problème et sa complexité. Dans la partie droite, le problème se traduit par un problème de projection, basé sur l'approche de Makino *et al.* (1999), où les fonctions booléennes partiellement défini (*partially defined Boolean formula - pdBf*) sont exposées et dûment étudiées. Cela nous fournit une autre façon de calculer des solutions plus compactes.

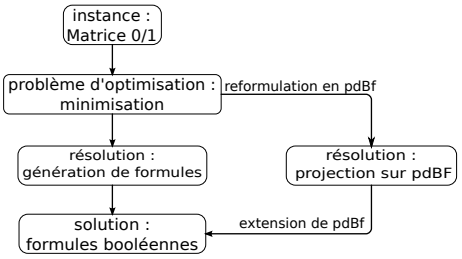


FIGURE 1 – Présentation générale des approches proposées.

La transformation entre les deux approches est polynomiale. Dans la première approche, les solutions sont exprimées sous forme de formules de la logique propositionnelle, qui induisent plusieurs difficultés liées à leur taille. En particulier, il parait difficile de contrôler les redondances et les tautologies lors de la recherche de solutions. En outre, l'espace de recherche peut être infini, sans définir des règles de réduction convergentes vers des formes normales. Au contraire, le second formalisme utilisant des projections fournit une représentation plus compacte des solutions. Néanmoins, notre première modélisation nous permet de bien étudier la satisfiabilité du problème et de

fournir une solution canonique.

Dans la dernière partie du document, nous montrons comment ce problème est appliquée à la biologie végétale avec des résultats expérimentaux. Ces expériences ont été menées avec des biologistes et ont conduit à l'élaboration d'un kit de diagnostic breveté pour l'identification des maladies bactériennes.

2 Le problème de caractérisation multiple

La description du problème de caractérisation multiple (PCM), dans le sens de la présence/absence de plusieurs caractères, nous permet naturellement d'utiliser le formalisme de la logique propositionnelle. Nous considérons qu'une matrice de booléens correspond à n caractères et m entités.

$$A \equiv \begin{pmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \vdots & \vdots \\ a_{m1} & \dots & a_{mn} \end{pmatrix}$$

Chaque ligne de cette matrice représente une entité, caractérisée par la présence ou l'absence d'un ensemble de caractères. Nous considérons ensuite les caractères comme des variables booléennes et les entités comme des affectations booléennes.

Pour chaque colonne indexées par $j \in \{1, \dots, n\}$, nous définissons une variable propositionnelles x_i qui correspond à un caractère. X est l'ensemble initial des variables propositionnelles. Pour chaque ligne $i \in \{1, \dots, m\}$, nous considérons une entité e_i comme une interprétation booléenne, c'est à dire une application de X à $\{0, 1\}$ (faux, vrai), tel que $\forall i, j, e_i(x_j) = a_{ij}$. Nous notons $\mathcal{E}$ l'ensemble de toutes les entités. Pour une formule propositionnelle ϕ sur X et $e \in \mathcal{E}$, nous notons $e \models \phi$ le fait que l'interprétation e satisfasse la formule ϕ.

Remarquons que dans des cas réels, des techniques de pré-traitements sont applicables pour réduire la taille de la matrice originale. L'ensemble des variables qui est utilisé pour la caractérisation peut donc être plus petit que le jeu original. En particulier, s'il y a un caractère j tel que $\forall i, k \in \{1, \dots, m\}, a_{ij} = a_{kj}$, alors la colonne correspondante j est retirée de la matrice puisque ce caractère ne peut évidemment pas être utilisé pour distinguer les entités.

La définition des groupes correspond à des ensembles disjoints de lignes de la matrice A. S'il y a deux lignes identiques appartenant au même groupe, on peut supprimer l'une des eux. Dans ce cas, chaque groupe est alors un sous-ensemble des entités de $\mathcal{E}$. Notons que deux entités identiques peuvent appartenir à deux groupes différents. Nous étudierons ce dernier aspect en ce qui concerne la satisfiabilité du problème.

Un littéral est une variable $x \in X$ ou sa négation, notée $\neg x$. Une clause est une disjonction de littéraux. Nous notons $\mathcal{L}$ l'ensemble des littéraux construit sur X. Une formule ϕ est dite sous forme normale conjonctive (CNF) si elle est une conjonction de clauses. Une formule est dite en forme normale disjonctive (DNF) si elle est une disjonction de conjonctions de littéraux.

Nous allons maintenant définir une instance du problème PCM.

Définition 1 (Instance du PCM)
Une instance de PCM est définie par un tuple $(X, \mathcal{E}, G)$ où X est un ensemble de variables propositionnelles, $\mathcal{E}$ un ensemble d'entités défini sur X et $G \subseteq 2^{\mathcal{E}}$.

Nous nous focalisons maintenant sur la définition de la caractérisation d'un groupe qui doit permettre de reconnaître ses propres entités et à la discrimination (c.-à-d. de ne pas accepter) des entités d'autres groupes (d'un point de vue logique cela correspond donc à la satisfaction ou la réfutation de formules).

Définition 2 (Caractérisation de Groupe)
Soit une instance $(X, \mathcal{E}, G)$, une formule ϕ_g caractérise un groupe $g \in G$ ssi :
$\forall e \in g, e \models \phi_g$ *(accepte les entités du groupe)*
et
$\forall g' \in G \setminus \{g\}, \forall e' \in g', e' \not\models \phi_g$ *(discrimine les entités des autres groupes).*

Par extension, nous notons $g \models \phi_g$ le fait que ϕ_g caractérise g selon la définition précédente. $Sol(g)$ est l'ensemble de toutes les caractérisations du groupe g et $Sol(g) = \{\phi_g | g \models \phi_g\}$.

Définition 3 (Solution du PCM)
Soit une instance $P \equiv (X, \mathcal{E}, G)$, une solution admissible du PCM est un $|G|$-tuple de formules $\Phi = (\phi_1, \cdots, \phi_{|G|})$ tel que $\forall i \in 1..|G|, g_i \in G, g_i \models \phi_i$.

$SOL(P)$ est l'ensemble de toutes les caractérisations multiples pour tous les groupes. $SOL(P) = Sol(g_1) \times \cdots \times Sol(g_{|G|})$. Soit un tuple de formules $\Phi = (\phi_1, \cdots, \phi_{|G|})$ et un ensemble de groupe G, nous notons par extension $G \models \Phi$ le fait que $\forall i \in 1..|G|, g_i \in G, g_i \models \phi_i$.

Définition 4 (Satisfiabilité du PCM)
Une instance $(X, \mathcal{E}, G)$ est satisfiable (resp. insatisfiable) ssi $\forall g \in G, Sol(g) \neq \emptyset$ (resp. $\exists g \in G, Sol(g) = \emptyset$).

Nous rappelons quelques définitions sur la taille et la longueur de formules.

Définition 5 (Taille et longueur d'une formule)
Soit ϕ une formule et $var(\phi)$ l'ensemble des variables de ϕ.
- *la taille de ϕ est $|var(\phi)|$;*
- *la longueur de ϕ, notée $len(\phi)$, est le nombre de feuilles du terme ϕ.*

Exemple 1
Soit $\phi = a \wedge b \wedge (\neg a \vee c)$, nous avons $|var(\phi)| = 3$ et $len(\phi) = 4$.

Définition 6 (Taille d'un tuple de formules)
Pour un tuple de formules $\Phi = (\phi_1, \cdots, \phi_n)$ nous avons $|\Phi| = |\bigcup_{\phi_i} var(\phi_i)|$.

3 L'approche par formules booléennes

Intuitivement, l'approche par formules booléennes semble la plus adaptée à résoudre le PCM.

3.1 Satisfiabilité

Notre premier but est d'étudier la satisfiabilité d'une instance du PCM (problème de décision). Nous proposons de construire une formule canonique qui peut être une solution pour n'importe quel problème satisfiable. Nous utilisons cette formule canonique pour exhiber une condition nécessaire et suffisante pour le problème de satisfiabilité.

Pour chaque entité e dans g, nous construisons la formule $\phi_e^+ \equiv \bigwedge_{x \in X} \delta(e,x)$ où $\delta : \mathcal{E} \times X \to \mathcal{L}$ est une fonction telle que $\delta(e,x) = \neg x$ si $e \models \neg x$ et $\delta(e,x) = x$ sinon. Nous notons $\phi_g^+ \equiv \bigvee_{e \in g} \phi_e^+$. Nous remarquons que ϕ_g^+ est en DNF. De la même façon nous avons $\phi_e^- \equiv \bigvee_{x \in X} \neg\delta(e,x)$ et $\phi_g^- \equiv \bigwedge_{g' \in G \setminus \{g\}} \bigwedge_{e' \in g'} \phi_{e'}^-$ (qui est en CNF). Nous notons $\phi_g \equiv \phi_g^+ \wedge \phi_g^-$.

Proposition 1 (Satifisfiabilité d'une instance)
Soit une instance $P \equiv (X, \mathcal{E}, G)$, nous avons :

$$SOL(P) \neq \emptyset \text{ ssi } \forall g \in G \begin{cases} \forall e \in g, e \models \phi_g \\ \text{et} \\ \forall g' \in G \setminus \{g\}, \forall e' \in g', e' \not\models \phi_g \end{cases}$$

Preuve 1
Nous devons prouver que $L \Leftrightarrow R$ avec L et R respectivement la première partie et la seconde partie de la proposition.

$L \Leftarrow R$:
Si R nous avons immédiatement que l'ensemble $\{\phi_g\}_{g \in G}$ est une solution de P puisque chaque formule ϕ_g caractérise g (définition 2). Donc $SOL(P) \neq \emptyset$.

$L \Rightarrow R$:
Nous devons prouver que $\neg R \Rightarrow SOL(P) = \emptyset$. Considérons $\neg R$:

$$\begin{aligned} &\neg[\forall g \in G\big((\forall e \in g, e \models \phi_g) \wedge (\forall g' \in G \setminus \{g\}, \forall e' \in g', e' \not\models \phi_g)\big)] \\ &\Leftrightarrow (\exists g \in G, \exists e \in g, e \not\models \phi_g) \vee (\exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', e' \models \phi_g) \end{aligned}$$

Nous devons prouver que $(\exists g \in G, \exists e \in g, e \not\models \phi_g) \Rightarrow SOL(P) = \emptyset$ et $(\exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', e' \models \phi_g) \Rightarrow SOL(P) = \emptyset$.

Commençons avec $\exists g \in G, \exists e \in g, e \not\models \phi_g \Rightarrow SOL(P) = \emptyset$. Nous avons $\phi_g \equiv \phi_g^+ \wedge \phi_g^-$. Par construction, $\forall g \in G, \forall e \in g, e \models \phi_g^+$. Alors $\exists g \in G, \exists e \in g, e \not\models \phi_g \Rightarrow SOL(P) = \emptyset$ ssi $\exists g \in G, \exists e \in g, e \not\models \phi_g^- \Rightarrow SOL(P) = \emptyset$.

$$\begin{aligned} &\exists g \in G, \exists e \in g, e \not\models \phi_g^- \\ &\Leftrightarrow \exists g \in G, \exists e \in g, e \not\models \bigwedge_{g' \in G \setminus \{g\}} \bigwedge_{e' \in g'} \phi_{e'}^- \\ &\Leftrightarrow \exists g \in G, \exists e \in g, \exists g' G \setminus \{g\}, \exists e' \in g', e \not\models \phi_{e'}^- \\ &\Leftrightarrow \exists g \in G, \exists e \in g, \exists g' G \setminus \{g\}, \exists e' \in g', e \not\models \bigvee_{x \in X} \neg\delta(e', x) \\ &\Leftrightarrow \exists g \in G, \exists e \in g, \exists g' G \setminus \{g\}, \exists e' \in g', e \models \bigwedge_{x \in X} \delta(e', x) \end{aligned}$$

Nous avons besoin du lemme suivant.

Lemme 1
$\forall e, e' \in \mathcal{E}, e \models \bigwedge_{x \in X} \delta(e', x) \Leftrightarrow e = e'$

Nous avons l'équivalence $\exists g \in G, \exists e \in g, \forall g' \in G \setminus \{g\}, \exists e' \in g', e = e'$ *ce qui implique* $SOL(P) = \emptyset$.

Considérons la seconde partie de $\neg R$. *Nous devons prouver que* $\exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', e' \models \phi_g \Rightarrow SOL(P) = \emptyset$.

De la même manière par construction, $\forall g' \in G \setminus \{g\}, \forall e' \in g', e' \not\models \phi_g^-$ *par* $\forall e' \in g', \phi_{e'}^- \equiv \bigvee_{x \in X} \neg\delta(e', x)$. *Considérons la partie correspondante à* ϕ_g^+ *dans* ϕ_g, *qui est :*

$$\begin{aligned}
&\exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', e' \models \phi_g^+ \\
&\Leftrightarrow \exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', e' \models \textstyle\bigvee_{e \in g} \phi_e^+ \\
&\Leftrightarrow \exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', \exists e \in g, e' \models \phi_e^+ \\
&\Leftrightarrow \exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', \exists e \in g, e' \models \textstyle\bigwedge_{x \in X} \delta(e, x) \\
&\Leftrightarrow \exists g \in G, \exists g' \in G \setminus \{g\}, \exists e' \in g', \exists e \in g, e = e' \\
&\Rightarrow SOL(P) = \emptyset
\end{aligned}$$

Nous avons $\neg R \Rightarrow SOL(P) = \emptyset$.

Corollaire 1

Soit $P \equiv (X, \mathcal{E}, G)$ *:*

$$\exists g, g' \in G, g \neq g', \exists e \in g, \exists e' \in g', e = e' \Leftrightarrow SOL(P) = \emptyset$$

Remarquons que le corollaire précédent prouve que si G est une partition de $\mathcal{E}$, alors il existe toujours une solution. Bien sûr, il était manifeste qu'une instance possédant deux entités dans deux groupes différents n'était pas satisfiable mais les résultats précédents garantissent que c'est l'unique raison. Ce qui peut être reformulé par :

Corollaire 2

Soit $P \equiv (X, \mathcal{E}, G)$ *:*

$$P\ est\ satisfiable \Leftrightarrow \forall g, g' \in G, g \neq g', g \cap g' = \emptyset$$

Nous exhibons alors une solution canonique Φ_{Max}^P pour n'importe quel problème satisfiable P :

$$\Phi_{Max}^P \equiv (\phi_{g_1}, \ldots, \phi_{g_{|G|}})$$

Cette solution canonique est aussi une solution maximale.

Proposition 2

Soit $P \equiv (X, \mathcal{E}, G)$, *nous avons les propriétés suivantes :*

1. $SOL(P) \neq \emptyset \Leftrightarrow G \models \Phi_{Max}^P$
2. $\forall \Phi \in SOL(P), \Phi \leftrightarrow \Phi_{Max}^P$
3. $\forall \Phi \in SOL(P), |\Phi_{Max}^P| \geq |\Phi|$

Un algorithme par génération de formules booléennes a été développé mais il ne sera pas présenté dans ce papier. Nous exposons par contre les deux difficultés rencontrées par cette approche.

– La taille de la formule peut être infinie si nous nous basons sur les règles de la logique propositionnelle.

– La détection, lors de la construction,d'un terme redondant (commutativité des opérateurs logiques,distributions, ...) ou logiquement inutile (tautologie,insatisfiable) est très coûteuse.

Malheureusement ces problèmes ne sont pas totalement surmontés en se restreignant à une syntaxe de forme normale (CNF). Pour preuve, la génération des CNF sans redondances est semblable au problème de détection des clauses redondantes qui est coNP-complet (Boufkhad & Roussel (2000); Fourdrinoy *et al.* (2007)).

3.2 Caractérisation minimale

Avant d'étudier une autre formulation du PCM, nous introduisons un critère d'optimalité qui est de minimiser la taille de la solution. Étant donné un PCM P, nous cherchons d'abord à calculer une solution $(\phi_1, \cdots, \phi_n)$ telle que chaque ϕ_i est un élément minimal de $Sol(g_i)$. Remarquons qu'une solution minimale globale est un élément minimal de $SOL(P)$ (en ce qui concerne la définition 6) dont le calcul est beaucoup plus coûteux. En effet, une solution ne contenant que des formules minimales pour chacun des groupes ne garantit en rien l'obtention d'une solution minimale globale.

Nous nous intéressons maintenant à la caractérisation pour un groupe. Nous savons que les bornes sont données par :

– inférieure : $|min(SOL(P))| \geq max(|min(Sol(g_1))|, \cdots, |min(Sol(g_n)|)$.
– supérieure : $|min(SOL(P))| \leq |\Phi|$ avec $\Phi = (min(Sol(g_1)), \cdots, min(Sol(g_n))$

Sur la complexité, le problème de caractérisation minimale pour un groupe semblerait être NP-difficile, et le PCM minimale serait tout aussi difficile mais nous ne pouvons pas actuellement en apporter la preuve formelle. Mais en considérant ceux-ci comme un problème du choix d'une partie de X, l'idée est de se ramener au problème de couverture dont le problème de décision est bien connu pour être NP-complet (Garey & Johnson (1985) - le problème de minimisation est NP-difficile).

4 L'approche par *pdBf*

Dans cette section, nous proposons maintenant d'encoder le problème d'une manière différente afin d'exclure les problèmes majeurs à l'approche par formules booléennes (la possibilité de construire des termes infinis et la difficulté à éviter les redondances et tautologies) mais aussi de calculer directement une solution globale. Pour présenter l'approche par fonctions booléennes partiellement définies (*partially defined Boolean formula - pdBf*), nous rappelons quelques notations et concepts liés aux *pdBf* (voir Makino *et al.* (1999) pour plus de détails) et nous montrons l'équivalence entre les deux approches.

Définition 7 (Fonction booléenne)
Une fonction booléenne f est une application $f : \mathcal{B}^n \mapsto \mathcal{B}$ d'arité n dont $\mathcal{B}$ désigne l'ensemble des booléens.

Définition 8 (Fonction booléenne partiellement définie)
Une fonction booléenne partiellement définie pf est définie comme un sous-ensemble $\mathcal{C}^+ \cup \mathcal{C}^- \subseteq \mathcal{B}^n$, où $\forall e \in \mathcal{C}^+$ (resp. $\forall e' \in \mathcal{C}^-$), nous avons $pf(e) = \top$ (resp. $pf(e') =$

$\perp$) et $\mathcal{C}^+$ (resp. $\mathcal{C}^-$) est appelé l'ensemble des exemples positifs (resp. négatifs). Une $pdBf_g$ est définie comme une pdBf *avec $\mathcal{C}^+ = g$ et $\mathcal{C}^- = G \setminus \{g\}$*

Nous étendons la notion de consistance de la logique propositionnelle aux *pdBf*.

Proposition 3 (Consistance et inconsistance d'une pdBf)
Une pdBf *est consistante (resp. inconsistante) ssi $\mathcal{C}^+ \cap \mathcal{C}^- = \emptyset$ (resp. $\mathcal{C}^+ \cap \mathcal{C}^- \neq \emptyset$).*

Définition 9 (Extension d'une pdBf)
Une extension d'une pdBf *est une formule ϕ, telle que ϕ est satisfaite (resp. falsifiée) par tous les éléments dans $\mathcal{C}^+$ (resp. $\mathcal{C}^-$)*

Dans la définition ci-dessus, on remarque que l'arité d'une *pdBf* est égale à la taille de son extension. Dans Eiter *et al.* (2002), les auteurs développent plusieurs techniques en temps polynomial pour trouver des extensions pour de nombreuses catégories définies. Nous introduisons maintenant la notion de projection qui est un concept clé pour calculer des représentations compactes de solutions.

Définition 10 (Projection)
Une projection est une fonction $\pi : \mathcal{B}^n \mapsto \mathcal{B}^{\,k}$ où $k \leq n$.

k est la dimension de la projection. π_{pdBf_g} est la projection associée à la $pdBf_g$. Nous notons $g \models \pi$ (i.e., π_{pdBf_g} est consistante) le fait que nous ayons :

$$\cup_{e \in \mathcal{C}^+} \pi_{pdBf_g}(e) \bigcap \cup_{e \in \mathcal{C}^-} \pi_{pdBf_g}(e) \neq \emptyset$$

Dans le but de préserver les résultats des sections précédentes, nous avons à prouver que les approches par formules et par projections (sur les *pdBf*) sont équivalentes.

Proposition 4 (Projetabilité et satisfiabilité)
$P \equiv (X, \mathcal{E}, G)$ est satisfiable ssi P est projetable sur G, c-à-d $\forall g \in G, \exists \phi, g \models \phi \Leftrightarrow \exists \pi, g \models \pi$

Preuve 2
$L \Rightarrow R$: En réutilisant la définition nous pouvons écrire $g \models \phi \Rightarrow \forall e \in g, e \models \phi_g \wedge \forall e' \notin g, e' \not\models \phi$ et en raisonnant par récurrence sur la longueur de ϕ, nous avons :

- *cas de base : $len(\phi) = 1$*
  Soit $\phi = x_i$ (resp. $\phi = \neg x_i$) et si $g \models \phi$ alors la seule pdBf *consistante x_i est $\forall e \in g, e(x_i) = \top$ (resp. $e(x_i) = \perp$) et $\forall e' \in G \setminus \{g\}, e'(x_i) = \perp$ (resp. $e(x_i) = \top$). Il existe alors π telle que $g \models \pi_{x_i}$.*
- *cas d'induction : Pour n>1, $len(\phi) = n - 1$ et $g \models \phi \Rightarrow g \models \pi$*
  Nous pouvons toujours construire $\phi' = \phi \otimes x_i$ avec $\otimes \in \{\wedge, \vee\}$, telle que $g \models \phi'$ (selon le principe du cas de base). Nous avons $\exists \pi', g \models \pi'$.

$R \Leftarrow L$: A partir d'une projection sur une $pdBf_g$, nous pouvons toujours trouver une solution via une extension (formule canonique par exemple).

Remarque 1

- *Une même projection qui est valide pour chaque groupe est donc une solution pour le PCM.*
- *Pour chaque groupe, l'extension trouvée peut-être de longueur non minimale mais nous garantissons par contre la minimalité de sa taille.*

L'algorithme EXACT-PROJ-CAR calcule des projections de la plus petite à la plus grande et retourne une extension pour la première projection trouvée (solution optimale).

5 Application : la minimisation de données biologiques

Le PCM est d'un grand intérêt pour de nombreux biologistes. La caractérisation précise des collections de souches bactériennes est un enjeu scientifique majeur, car les bactéries sont responsables d'importantes maladies sur les plantes, et donc soumise à des procédures officielles de contrôle (par exemple, en Europe, la directive 2000/29/CE). Le développement de tests de diagnostic est donc un enjeu important pour systématiquement identifier les souches de ces espèces. Dans ce contexte, le problème de la caractérisation correspond à l'identification d'un groupe de souches vis à vis d'autres groupes, basée sur la présence ou l'absence de certains caractères particuliers. Une souche est donc un vecteur de valeurs binaires qui reflète la présence (valeur 1) ou l'absence (valeur 0) de ces caractères. En outre, pour réduire le coût prohibitif des tests biologiques, nous nous intéressons à minimiser les solutions (le nombre de tests associés pour chaque identification) que ce soit pour un PCM ou pour un test plus spécifique pour un groupe donnée.

Les biologistes nous ont fourni les 3 instances suivantes :

- **A** : 8 groupes, 108 entités (de 2 à 54 par groupe), 155 caractères.
- **B** : 4 groupes, 112 entités (de 5 à 69 par groupe), 155 caractères.
- **C** : 7 groupes, 112 entités (de 2 à 40 par groupe), 155 caractères.

A, B et C sont des exemples provenant de l'API BioMérieux basée sur des propriétés bio-chimiques de l'espèce *Ralstonia*. Dans le tableau 1, nous présentons les résultats expérimentaux que nous avons obtenus avec notre algorithme. Les quatre premières lignes fournissent les caractéristiques de l'instance. La cinquième ligne donne la caractérisation minimale obtenue par EXACT-PROJ-CAR ("-" si inconnue) pour chacun des groupes. Nous n'avons aucune solution globale pour chacune de ces instances.

Instances	A								B				C						
Entités Totales	108								113				112						
Groupes	1	2	3	4	5	6	7	8	1	2	3	4	1	2	3	4	5	6	7
Entités par groupes	21	5	3	54	9	8	7	2	31	69	8	5	38	15	5	6	2	40	6
EXACT-PROJ-CAR (taille)	2	2	2	4	4	3	3	3	3	6	5	2	-	-	5	4	2	5	4

TABLE 1 – Application sur des instances réelles.

Les solutions de tailles inférieures à 3 sont instantanément trouvées. Nous constatons une augmentation exponentielle du temps de calcul qui peut aller de quelques secondes pour des tailles de 5, jusqu'à plusieurs heures pour des tailles supérieures.

Ces résultats, même partiels, ont permis d'apporter des solutions aux biologistes qui ne disposaient pas encore de telles méthodes pour analyser, traiter, et garantir la minimalité des caractérisations trouvées. De nouveaux tests de diagnostic ont ainsi pu être produits et vérifiés expérimentalement.

6 Conclusion et travaux futurs

Dans ce document, nous avons défini le problème de la caractérisation multiple et nous avons proposé deux formalismes afin d'étudier ses propriétés et de calculer une solution minimale. Malheureusement, l'approche complète associée aux *pdBf* pour un PCM échoue. Toutefois à partir du problème de caractérisation minimale des différents groupes, nous pouvons établir des bornes. Dans l'avenir, nous pensons nous orienter vers des méthodes de recherches locales ou évolutionnaires afin de trouver et améliorer les solutions de ce problème combinatoire et ainsi compléter l'approche par force brute de EXACT-PROJ-CAR. Une application aux données biologiques a été décrite pour mettre en évidence les utilisations possibles de ce problème d'acquisition de connaissances générales. Du point de vue des utilisateurs, le formalisme semble être utile, car il précise l'information (quels caractères sont importants et quelles formules sont à mettre en œuvre pour un test de diagnostic). Il serait également très utile d'être en mesure de calculer plusieurs solutions optimales différentes et les biologistes sont également intéressés par le découverte de relations entre les caractères. Les conséquences logiques seraient un moyen d'y parvenir. Nous pensons donc orienter nos prochains travaux vers des techniques issues de la compilation de connaissances.

Références

BOUFKHAD Y. & ROUSSEL O. (2000). Redundancy in random SAT formulas. In *Proc. of the 7th National Conference on AI and 20th Conference on Innovative Applications of AI*, p. 273–278 : AAAI Press / The MIT Press.

EITER T., IBARAKI T. & MAKINO K. (2002). Decision lists and related boolean functions. *Theor. Comput. Sci.*, **270**, 493–524.

FOURDRINOY O., GRÉGOIRE E., MAZURE B. & SAÏS L. (2007). Eliminating redundant clauses in SAT instances. In P. V. HENTENRYCK & L. A. WOLSEY, Eds., *The 4th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, p. 71–83 : Springer.

GAREY M. R. & JOHNSON D. S. (1985). *Computers and Intractability : a guide to the theory of NP-Completeness*. New York : Freeman.

GAVALDÀ R. & THÉRIEN D. (2009). An algebraic perspective on boolean function learning. In *Algorithmic Learning Theory, 20th International Conference, ALT 2009,Proceedings*, volume 5809 of *LNCS*, p. 201–215 : Springer.

MAKINO K., ICHI HATANAKA K. & IBARAKI T. (1999). Horn extensions of a partially defined boolean function. *SIAM J. Comput.*, **28**(6), 2168–2186.

MCCLUSKEY E. (1956). Minimization of boolean functions. *Bell System Tech. J*, **35**, 1417–1444.

MEYER D., LEISCH F. & HORNIK K. (2003). The support vector machine under test. *Neurocomputing*, **55**(1-2), 169–186.

NATARAJAN B. K. (1987). On learning boolean functions. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, p. 296–304 : ACM.

QUINLAN J. R. (1990). Learning logical definitions from relations. *Machine Learning*, **5**, 239–266.

VALIANT L. G. (1984). A theory of the learnable. *Commun. ACM*, **27**(11), 1134–1142.

Cohérence de réseaux de contraintes qualitatives spatio-temporelles à partir de décompositions arborescentes

Jean-François Condotta

Université Lille-Nord de France, Artois CRIL-CNRS, UMR 8188
rue de l'université, SP 16, F62307 Lens
condotta@cril.fr

Résumé : Dans le cadre d'une traduction des réseaux de contraintes temporelles de l'Algèbre des Intervalles en problèmes SAT, Li et al. [7] ont récemment proposé une nouvelle décomposition de ces réseaux de contraintes afin de réduire le nombre de clauses SAT à résoudre. Dans cet article, nous montrons que ces décompositions sont équivalentes à des décompositions arborescentes. Dans un deuxième temps, nous considérons la propriété de fermeture par faible composition sur laquelle sont basées les méthodes de filtrage pour la résolution de réseaux de contraintes qualitatives (RCQ). Nous montrons que pour certaines classes de RCQ, la satisfaction de cette propriété par les clusters issus d'une décomposition arborescente est suffisante pour décider de la cohérence. À partir de ce résultat, nous proposons un nouvel algortithme permettant de décider du problème de la cohérence des RCQ à l'aide d'une classe traitable et d'une décomposition arborescente.

1 Introduction

Le raisonnement sur des informations temporelles et spatiales est une tâche cruciale dans certaines applications informatiques, en particulier dans des domaines tels que les SIG (Systèmes d'Informations Géographiques), le traitement du langage naturel, la planification temporelle/spatiale. Le raisonnement spatio-temporel qualitatif est un domaine dans lequel sont proposés des formalismes permettant de représenter et de traiter qualitativement les informations sur des configurations d'entités spatiales/temporelles. Un formalisme qualitatif considère des éléments particuliers pour représenter les différentes entités et un ensemble fini de relations de base sur ces entités. Chaque relation de base est représentative d'une position relative particulière entre des entités, et correspond à une abstraction de configurations pouvant être caractérisées de manière numérique. Ces dernières années de nombreux formalismes ont été proposés et étudiés, pour le raisonnement temporel nous pouvons par exemple citer l'Algèbre des Intervalles proposée par Allen [1]. Ce formalisme considère comme entités les intervalles de la droite et treize relations de base représentant toutes les configurations possibles des quatre

bornes de deux intervalles (voir la figure 1). L'algèbre des Intervalles a inspiré la définition de nombreux formalismes qualitatifs pour le temps [9, 10, 14]. Concernant le raisonnement spatial, le plus connu et étudié des formalismes qualitatifs est certainement RCC [15, 16]. RCC permet de représenter qualitativement des configurations de régions à l'aide de relations de base d'ordre topologique.

Dans le cadre des formalismes qualitatifs, l'ensemble des informations temporelles ou spatiales est habituellement représenté par des réseaux de contraintes particuliers appelés réseaux de contraintes qualitatives (RCQ en abrégé). Dans un RCQ, chaque variable représente une entité spatiale/temporelle du système et une contrainte définit les configurations possibles entre des entités à l'aide d'un ensemble de relations de base permises. Étant donné un RCQ, le principal problème se posant est de déterminer s'il admet une solution (le problème de la cohérence). Ce problème est en règle général un problème NP–Complet, de nombreuses études ont été réalisées afin de proposer des méthodes pour résoudre efficacement en pratique ce problème. Deux types d'approches sont suivies pour l'élaboration de ces méthodes.
La première approche, initiée par Nebel [11], consiste en la définition d'algorithmes de recherche de type backtrack combinés avec une propagation locale de contraintes basée sur des classes de relations dites traitables et la méthode de fermeture par faible composition. Cette dernière permet de maintenir le RCQ $\diamond$–cohérent (propriété proche de la chemin-cohérence). À chaque étape de la recherche, une contrainte est découpée en sous-relations de la classe traitable utilisée. Puis, pour chacune des sous-relations issues de ce découpage, la contrainte est remplacée par la sous-relation et un filtrage est réalisé à l'aide de la fermeture par faible composition. Ce filtrage permet de supprimer des relations de base non satisfiables et de réduire l'espace de recherche. L'utilisation d'une classe traitable dans ce processus permet de réduire le facteur de branchement lors de la recherche. Les solveurs de contraintes qualitatives utilisent en général cette approche, en particulier GQR* [5] qui est actuellement le plus performant.

Une autre approche plus récente [13] consiste en la traduction des RCQ en SAT (problème de satisfiabilité propositionnelle). Typiquement, les problèmes SAT issus de telles traductions contiennent d'une part des clauses modélisant les contraintes explicites du RCQ, et d'autre part des clauses modélisant les configurations possibles compte tenu de l'opération de faible composition sur laquelle est basée la fermeture par faible composition. L'avantage d'une telle approche est que les instances SAT obtenues peuvent être résolues de manière efficace par les solveurs SAT actuels. Un inconvénient majeur est que ces instances SAT sont de tailles très importantes. Récemment, pour tenter de pallier cet inconvénient dans le cadre de RCQ de l'Algèbre des Intervalles, Li *et al*. [7] ont proposé une méthode décomposant récursivement à chaque étape un RCQ en deux RCQ équivalents portant chacun sur moins de variables. Les contraintes présentes dans le RCQ initial et non présentes dans l'un des deux RCQ sont caractérisées comme non nécessaires à la recherche d'une solution et ne sont donc pas considérées lors d'un processus de traduction en SAT. Les instances SAT obtenues en utilisant cette méthode sont de moindre tailles comparativement à des traductions complètes. De plus, Li *et al*. ont montré expérimentalement que ces instances étaient résolues plus rapidement.

Dans cet article, nous définissons formellement des décompositions de RCQ appelées décompositions de type PartRec. Ces décompositions particulières permettent de re-

présenter les décompositions de RCQ issues de l'approche suivie par Li *et al*. dans leur processus de traduction SAT. Nous montrons que les décompositions de type PartRec peuvent de manière équivalente se définir à l'aide de décompositions arborescentes de RCQ, décompositions intensivement étudiées dans le cadre des CSP discrets. Dans un second temps, nous étudions le problème de la cohérence des RCQ au travers des décompositions arborescentes. Nous montrons notamment qu'étant donnée une décomposition arborescente d'un RCQ définie sur un classe traitable possédant certaines propriétés, la fermeture par faible composition restreinte aux contraintes appartenant aux différents clusters est suffisante pour décider du problème de la cohérence du RCQ. Fort de ce résultat, nous proposons un algorithme permettant de décider du problème de la cohérence des RCQ à l'aide d'une classe traitable et d'une décomposition arborescente.

2 Préliminaires sur les formalismes qualitatifs

2.1 Les relations

Un formalisme qualitatif considère un ensemble fini B de relations appelées relations de base ou encore relations atomiques. Ces relations sont définies sur un domaine D et nous supposerons dans la suite que ces relations sont binaires. Il est à noter que peu de formalismes qualitatifs considèrent des relations de base d'arité supérieure à deux. Les éléments de D sont utilisés pour représenter les entités temporelles ou spatiales. L'ensemble B correspond à toutes les configurations qualitatives possibles entre deux entités temporelles ou spatiales et satisfait des propriétés particulières [17] : (1) B forme une partition de $\mathsf{D} \times \mathsf{D}$, (2) B contient la relation identité sur D et, (3) B est fermé pour l'inverse, *i.e*. la relation inverse d'une relation de base de B appartient également à B. Une relation (complexe) d'un formalisme qualitatif correspond à l'union de relations de base. Une relation est représentée par l'ensemble des relations de base incluses dans la relation. Ainsi, l'ensemble 2^{B} représentera l'ensemble des relations du formalisme qualitatif considéré. Parmi les relations de 2^{B} nous dénoterons par Ψ la relation contenant toutes les relations de base de B. Comme ensemble d'ensembles, 2^{B} est muni des opérations ensemblistes d'intersection et d'union. Comme ensemble de relations binaires il est également muni de l'opération inverse ($^{-1}$). L'inverse d'une relation $r \in 2^{\mathsf{B}}$ est l'union des relations de base inverses des relations de base appartenant à r. L'ensemble 2^{B} est aussi muni de l'opération binaire de faible composition, notée dans la suite par $\diamond$. Cette opération peut être définie de la manière suivante : $a \diamond b = \{c \in \mathsf{B} : \exists x, y, z \in \mathsf{D}$ avec $x\ a\ z, z\ b\ y$ et $x\ c\ y\}$, avec $a, b \in \mathsf{B}, r \diamond s = \bigcup_{a \in r, b \in s} \{a \diamond b\}$, avec $r, s \in 2^{\mathsf{B}}$. Remarquons que la relation $r \diamond s$ peut être également définie par la plus petite (pour l'inclusion) relation appartenant à 2^{B} incluant la composition relationnelle habituelle $r \circ s = \{(x, y) \in \mathsf{D} \times \mathsf{D} : \exists x, y, z \in \mathsf{D}$ avec $x\ a\ z, z\ b\ y$ et $x\ c\ y\}$. Pour certains formalismes qualitatifs $r \circ s$ et $r \diamond s$ correspondent à la même relation. Une classe de relations $\mathcal{C}$ est un sous-ensemble de 2^{B} fermé pour les opérations d'inverse, d'intersection et de faible composition, contenant les relations singletons de 2^{B} et la relation Ψ. Étant données une relation $r \in 2^{\mathsf{B}}$ et une classe de relations $\mathcal{C}$, la plus petite (pour l'inclusion) relation de $\mathcal{C}$ contenant r sera dénotée par $r^{\mathcal{C}}$ et sera appelée la fermeture de r par rapport à $\mathcal{C}$.

Comme illustration, considérons le formalisme qualitatif temporel bien connu appelé Algèbre des Intervalles [1]. Pour ce formalisme, les intervalles de la droite des rationnels sont utilisés pour représenter les entités temporelles. Ainsi, D est l'ensemble $\{(x^-, x^+) \in \mathbb{Q} \times \mathbb{Q} : x^- < x^+\}$. L'ensemble des relations de base consiste en un ensemble de treize relations binaires, chacune correspondant à une position relative particulière de deux intervalles de la droite. Ces relations sont illustrées dans la figure 1. Nous avons $\mathsf{B} = \{eq, p, pi, m, mi, o, oi, s, si, d, di, f, fi\}$. Chacune de ces relations de base peut être définie par des contraintes sur les bornes de deux intervalles. Par exemple, la relation de base *meets* (m) peut être définie de la manière suivante, $m = \{((x^-, x^+), (y^-, y^+)) \in \mathsf{D} \times \mathsf{D} : x^+ = y^-\}$. À titre d'exemple, l'ensemble $\{p, m\} \in 2^{\mathsf{B}}$ correspond à la relation $p \cup m$. Nous avons, $\{p, m\}^{-1} = \{p^{-1}, m^{-1}\} = \{pi, mi\}$ et $\{p, m\} \diamond \{m, s, eq\} = \{p, m\}$.

Relation	Symbole	Inverse	Illustration
precedes	p	pi	
meets	m	mi	
overlaps	o	oi	
starts	s	si	
during	d	di	
finishes	f	fi	
equals	eq	eq	

FIG. 1 – Les relations de base de l'Algèbre des Intervalles.

2.2 Les réseaux de contraintes qualitatives

Un réseau de contraintes qualitatives (RCQ) est un couple composé d'un ensemble de variables représentant les entités temporelles du système et d'un ensemble de contraintes défini par une application associant à chaque couple de variables un ensemble de relations de base (les configurations possibles). Formellement, un RCQ est défini de la manière suivante :

Définition 1 *Un* RCQ $\mathcal{N}$ *est un couple* (V, C) *où :*

- $V = \{v_1, \ldots, v_m\}$ *est un ensemble fini de* m *variables avec* m *un entier positif ;*
- C *est une application associant à chaque couple* (v_i, v_j) *de* $V \times V$ *un ensemble* $C(v_i, v_j)$ *de relations de base :* $C(v_i, v_j) \in 2^{\mathsf{B}}$. *Dans la suite,* $C(v_i, v_j)$ *sera également dénoté par* $\mathcal{N}[v_i, v_j]$. C *est telle que* $C(v_i, v_i) \subseteq \{\mathsf{Id}\}$, *avec* Id *la relation de base correspondant à la relation d'identité sur* D *et* $C(v_i, v_j) = C_(v_j, v_i)^{-1}$ *pour tout* $v_i, v_j \in V$.

Dans la figure 2(a) est représenté un RCQ $\mathcal{N}$ de l'Algèbre des Intervalles. Sur la figure ne sont par représentées les contraintes définies par la relation Ψ et la contrainte entre deux variables v_i et v_j lorsque $v_i = v_j$ ou lorsque la contrainte entre v_j et v_i est déjà présente. Concernant un RCQ $\mathcal{N} = (V, C)$ nous avons les notations et définitions suivantes : Une *instanciation partielle* de $\mathcal{N}$ sur $V' \subseteq V$ est une application s de V' vers D. Une *solution partielle* de $\mathcal{N}$ sur $V' \subseteq V$ est une instanciation partielle sur V' telle que $(s(v_i), s(v_j))$ satisfait $C(v_i, v_j)$ pour tout $v_i, v_j \in V'$, *i.e.* il existe une relation de base $b \in C(v_i, v_j)$ telle que $(s(v_i), s(v_j)) \in b$ pour tout $v_i, v_j \in V'$. Une *solution* de $\mathcal{N}$ est une solution partielle de $\mathcal{N}$ sur V. $\mathcal{N}$ est cohérent si, et seulement si, $\mathcal{N}$ admet une solution. $\mathcal{N}$ est dit trivialement incohérent dans le cas où il existe deux variables $v, v' \in V$ telles que $\mathcal{N}[v, v'] = \emptyset$. $\mathcal{N}$ est globalement cohérent si, et seulement si, toute solution partielle de $\mathcal{N}$ peut être étendue en une solution de $\mathcal{N}$. La projection du RCQ $\mathcal{N}$ sur V' avec $V' \subseteq V$, dénotée par $\mathcal{N}_{V'}$, est le RCQ (V', C_{proj}) avec C_{proj} la restriction de C sur V'. Un sous-RCQ $\mathcal{N}'$ de $\mathcal{N}$ est un RCQ (V, C') tel que $C'(v_i, v_j) \subseteq C(v_i, v_j)$, pour tout $v_i, v_j \in V$. Soient deux RCQ $\mathcal{N}^1$ et $\mathcal{N}^2$ définis respectivement sur les ensembles de variables V^1 et V^2, avec pour chaque couple de variables $v, v' \in V^1 \cap V^2, \mathcal{N}^1[v, v'] = \mathcal{N}^2[v, v']$. Le RCQ $\mathcal{N}^1 \cup \mathcal{N}^2$ est l'unique RCQ $\mathcal{N}$ défini sur l'ensemble de variables $V^1 \cup V^2$ tel que $\mathcal{N}[v, v'] = \mathcal{N}^1[v, v']$ pour tout $v, v' \in V^1, \mathcal{N}[v, v'] = \mathcal{N}^2[v, v']$ pour tout $v, v' \in V^2, \mathcal{N}[v, v'] = \Psi$ pour tout $v \in V^2 \setminus V^1$ et $v' \in V^1 \setminus V^2$.

Un RCQ $\mathcal{N} = (V, C)$ est dit $\diamond$-cohérent ou fermé par faible composition si, et seulement si, $C(v_i, v_j) \subseteq C(v_i, v_k) \diamond C(v_k, v_j)$ pour tout $v_i, v_j, v_k \in V$. La fermeture par faible composition du RCQ $\mathcal{N}$, dénotée par $\diamond(\mathcal{N})$ est le plus grand (pour $\subseteq$) $\diamond$-cohérent sous-RCQ de $\mathcal{N}$. Cette fermeture peut-être obtenue par itération de l'opération de triangulation : $C(v_i, v_j) \leftarrow C(v_i, v_j) \cap (C(v_i, v_k) \diamond C(v_k, v_j))$ pour tout $v_i, v_j, v_k \in V$ jusqu'à ce qu'un point fixe soit atteint. Cette méthode peut-être implémentée avec des algorithmes de complexité temporelle $O(m^3)$, avec m le nombre de variables. Pour certaines classes de relations, le problème de la cohérence d'un RCQ peut-être résolu par la fermeture par faible composition. Pour l'Algèbre des Intervalles, nous avons par exemple tout RCQ $\mathcal{N}$ $\diamond$-cohérent (et non trivialement incohérent) dont les contraintes sont définies par des relations de la classe des relations préconvexes, appelées également relations ORD-Horn [12, 8], qui est cohérent. Pour la classe des relations convexes de l'Algèbre des Intervalles nous avons une propriété plus forte, en effet tout RCQ $\mathcal{N}$ $\diamond$-cohérent défini par des relations de cette classe est globalement cohérent [2].

Pour clore cette section nous donnons quelques notations concernant les arbres. Étant donné un arbre (un graphe acyclique connecté) enraciné $T = (X, F)$ et un noeud $X_i \in X$, nous dénoterons par $\mathsf{desc}(X_i)$ (resp. $\mathsf{asc}(X_i)$) l'ensemble des descendants de X_i (resp. des ascendants de X_i). Notons que $\mathsf{desc}(X_i)$ et $\mathsf{asc}(X_i)$ contiennent X_i. Étant donné X' un sous-ensemble non vide de noeuds de X, $\mathsf{lca}(X')$ dénotera le noeud de T qui est le plus bas ascendant commun des noeuds appartenant à X'. L'ensemble $\mathsf{leaves}(T)$ correspond à l'ensemble des feuilles de T. Et enfin, étant donné $X_i \in X$, T_{X_i} dénotera le sous-arbre de T de racine X_i.

3 Décompositions de RCQ

3.1 Décompositions arborescentes

La relation Ψ est composée de toutes les relations de base possibles de B et est satisfaite par toute paire d'éléments du domaine D. La définition d'une contrainte entre deux variables d'un RCQ à l'aide de la relation Ψ permet de spécifier que localement il n'y a pas de contrainte concernant la position relative des deux entités représentées par les deux variables. Ainsi, de manière naturelle nous définissons le graphe de contraintes d'un RCQ $\mathcal{N} = (V, C)$, par le graphe non orienté $\mathsf{G}(\mathcal{N}) = (V, E)$ avec $(v, v') \in E$ si, et seulement si, $\mathcal{N}[v, v'] \neq \Psi$ et $v \neq v'$. Dans la suite nous supposons qu'étant donné un RCQ $\mathcal{N}$, $\mathsf{G}(\mathcal{N})$ est connecté. Dans le cas contraire, $\mathcal{N}$ peut être trivialement partagé en deux RCQ indépendants sans variable commune. Comme dans le cadre des CSP discrets nous définissons une décomposition arborescente d'un RCQ comme une décomposition arborescente de son graphe de contraintes [18] :

Définition 2 *Soit $\mathcal{N} = (V, C)$ un* RCQ *et $\mathsf{G}(\mathcal{N}) = (V, E)$ son graphe de contraintes. Une décomposition arborescente de $\mathcal{N}$ est un arbre $T = (X = \{X_0, \ldots, X_n\}, F)$ avec n un entier positif, où X est une famille de sous-ensembles de variables de V ($X_i \subseteq V$), telle que :*
(1) $\bigcup\{X_i \in X\} = V$,
(2) $\forall (v, v') \in E$, il existe $X_i \in X$ tel que $v, v' \in X_i$,
(3) pour tout $X_i, X_j, X_k \in X$, si X_j est sur l'unique chemin entre X_i et X_k alors $X_i \cap X_k \subseteq X_j$.

Étant donnée une décomposition arborescente $T = (X = \{X_0, \ldots, X_n\}, F)$ d'un RCQ, la largeur de T est égale à $max\{|X_i| - 1 : X_i \in X\}$. De plus, chaque ensemble de variables X_i est appelé cluster. Dans la figure 2(c) se trouve représentée une décomposition arborescente du RCQ $\mathcal{N}$ de la figure 2(a).

3.2 Les décompositions de type PartRec

Récemment, Li *et al*. [7] ont proposé une méthode permettant de traduire un RCQ de l'Algèbre des Intervalles en problème SAT. Cette méthode décompose récursivement à chaque étape un RCQ $\mathcal{N} = (V, C)$ en deux RCQ $\mathcal{N}^1$ et $\mathcal{N}^2$ tels que $\mathcal{N} = \mathcal{N}^1 \cup \mathcal{N}^2$. Les contraintes définies par la relation Ψ dans le RCQ $\mathcal{N}$ non présentes dans $\mathcal{N}^1$ et $\mathcal{N}^2$ sont caractérisées comme non nécessaires à la recherche d'un scénario cohérent du RCQ global initial et donc non traduites en SAT. L'avantage d'une telle traduction est que l'instance SAT obtenue par cette méthode est de plus petite taille qu'une instance issue d'une traduction complète de l'ensemble des contraintes. En nous inspirant de cette méthode, nous définissons des décompositions de RCQ particulières, appelées dans la suite décompositions PartRec (pour partitionnement récursif), de la manière suivante :

Définition 3 *Soit $\mathcal{N}$ un* RCQ $= (V, C)$. *Une décomposition* PartRec *de $\mathcal{N}$ est un arbre enraciné $T = (X = \{X_0, \ldots, X_n\}, F)$, avec n un entier positif, où X est une famille de sous-ensembles de V ($X_i \subseteq V$) et $X_r = V$ avec X_r la racine de T. De plus, chaque*

noeud de T a deux fils ou aucun et étant donnés $X_i, X_j, X_k \in X$ tels que X_j et X_k sont fils de X_i, T satisfait les propriétés suivantes :

(1) $\mathcal{N}_{X_i} = \mathcal{N}_{X_j} \cup \mathcal{N}_{X_k}$, $X_j \setminus X_k \neq \emptyset$ *et* $X_k \setminus X_j \neq \emptyset$.

(2) *Pour tout* $X_m \in \mathsf{desc}(X_i)$, *si* $X_m \cap (X_j \cap X_k) \neq \emptyset$ *alors* $(X_j \cap X_k) \subseteq X_m$.

Intuitivement, d'après la condition (1) nous avons les contraintes pertinentes sur l'ensemble des variables X_i qui appartiennent à l'ensemble des contraintes sur X_j ou à celui des contraintes sur X_k, ou aux deux. De plus, l'ensemble X_j ou l'ensemble X_k ne regroupe pas la totalité des variables de X_i. La condition (2) stipule que toute contrainte entre deux variables appartenant à l'intersection de deux RCQ issus d'un découpage ne pourra pas être scindée lors d'un nouveau découpage. Dans la figure 2(b) est représentée une décomposition PartRec du RCQ $\mathcal{N}$ de la figure 2(a). Notons que le traitement récursif employé dans le cadre de la traduction des RCQ en clauses SAT proposée par Li *et al*. [7] définit implicitement une décomposition $T = (X, T)$ de type PartRec du RCQ $\mathcal{N}$ considéré. Les ensembles de variables $X_i \in X \setminus \mathsf{leaves}(T)$ correspondent aux ensembles de variables partagés au cours du traitement. Les ensembles de variables non partagés pour lesquels le traitement s'arrête correspondent aux éléments $X_i \in \mathsf{leaves}(T)$. Seules les contraintes portant sur des variables appartenant à un même ensemble $X_i \in \mathsf{leaves}(T)$ sont traduites en clauses SAT. Li *et al*. donne un algorithme polynomial permettant de calculer une partition de type PartRec . En pratique, lors des découpages récursifs, un équilibre concernant le nombre des variables des deux RCQ obtenus et une minimisation des contraintes appartenant à l'intersection de ces deux RCQ permettent de maximiser le nombre de contraintes à laisser de côté.
Nous définissons la largeur d'une décomposition PartRec $T = (X, F)$ par $max\{|X_i| - 1 : X_i \in \mathsf{leaves}(T)\}$ et appelons cluster de T tout élément $X_i \in \mathsf{leaves}(T)$. Remarquons que du fait de la propriété (1) de la définition précédente, $X_i \neq \emptyset$ pour tout $X_i \in X$. De plus, une décomposition PartRec satisfait les propriétés suivantes :

Proposition 1 *Soit* $T = (X = \{X_0, \ldots, X_n\}, F)$ *une décomposition* PartRec *d'un* RCQ $\mathcal{N} = (V, C)$. *Pour tout* $X_i \in X$, *nous avons :*

(1) Pour tout $X_j \in X$ *tel que* $i \neq j$, $X_i \neq X_j$ *et de plus, si* $X_j \in \mathsf{desc}(X_i)$ *alors* $X_j \subset X_i$ *;*

(2) T_{X_i} *est une décomposition* PartRec *de* $\mathcal{N}_{X_i}$ *;*

(3) $\mathcal{N}_{X_i} = \bigcup\{\mathcal{N}_{X_j} : X_j \in \mathsf{leaves}(T_{X_i})\}$ *;*

(4) $X_i = \bigcup\{X_j \in \mathsf{leaves}(T_{X_i})\}$.

Les preuves de ces propriétés sont omises et peuvent être directement établies à partir de la définition 3. Notons que du fait de la propriété (1), nous avons pour tout $X_i, X_j \in X$ les ensembles X_i et X_j qui sont différents si, et seulement si, $i \neq j$. Ainsi, pour tout $X_i \in X$, l'ensemble des variables de X_i caractérise un et un seul noeud de X. Soit une décomposition PartRec $T = (X, F)$ d'un RCQ $\mathcal{N} = (V, C)$. Étant donné $X_i \in X$, $\overline{X_i}$ dénotera l'ensemble X_i dans le cas où $X_i \in \mathsf{leaves}(T)$, l'ensemble $X_j \cap X_k$ avec X_j et X_k les deux fils de X_i dans le cas où $X_i \notin \mathsf{leaves}(T)$. Notons que $\overline{X_i} \subseteq X_i$ puisque $X_j \subseteq X_i$ et $X_k \subseteq X_i$. De plus, nous avons les propriétés suivantes :

Proposition 2 *Soit une décomposition* PartRec $T = (X = \{X_0, \ldots, X_n\}, F)$ *d'un* RCQ $\mathcal{N} = (V, C)$.

(1) Pour tout $X_i \in X$, $\overline{X_i} \neq \emptyset$.
(2) Soient $X_i, X_j, X_k \in X$ *avec* $X_k = \mathsf{lca}(\{X_i, X_j\})$. *Si* $X_k \neq X_i$ *et* $X_k \neq X_j$ *alors* $X_i \cap X_j \subseteq \overline{X_k}$.
(3) Pour tout $X_i \in X$, *il existe* $X_j \in \mathsf{leaves}(X)$ *tel que* $\overline{X_i} \subseteq X_j$.

Pour une meilleure lisibilité, la preuve de cette proposition et celles des suivantes sont données en Annexe.

3.3 Des décompositions PartRec vers les décompositions arborescentes

Dans cette section nous allons montrer qu'à partir d'une décomposition PartRec T d'un RCQ nous pouvons définir une décomposition arborescente de même largeur qui, comme nous le verrons par la suite, considère les mêmes regroupements de variables. Avant cela, nous introduisons une nouvelle notation : étant donnés $X_i, X_j \in X$ avec X_j un noeud fils de X_i, $\widetilde{X_i X_j}$ dénotera l'ensemble $\{X_k : X_k \in \mathsf{desc}(X_j) \text{ et } \overline{X_i} \subseteq \overline{X_k}\}$. Concernant l'ensemble $\widetilde{X_i X_j}$, nous avons les propriétés suivantes.

Proposition 3 *Soit une décomposition* PartRec $T = (X, F)$ *d'un* RCQ $\mathcal{N} = (V, C)$ *et* $X_i, X_j \in X$ *avec* X_j *fils de* X_i. *Nous avons : (1)* $\widetilde{X_i X_j} \neq \emptyset$, *(2)* $\mathsf{lca}(\widetilde{X_i X_j}) \in \widetilde{X_i X_j}$.

Maintenant, nous définissons à partir d'une décomposition PartRec T d'un RCQ $\mathcal{N}$, un arbre dénoté par $\overline{T}$. Nous montrerons par la suite que cet arbre est une décomposition arborescente de $\mathcal{N}$ possédant certaines particularités.

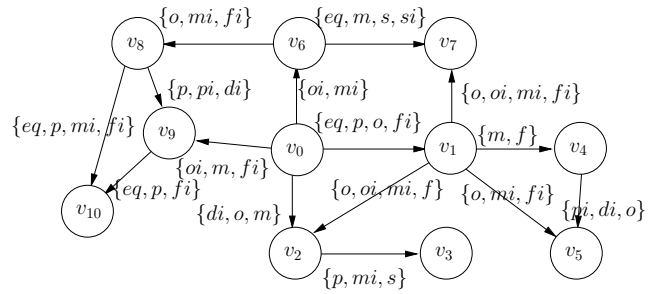
Définition 4 *Soient* $\mathcal{N}$ *un* RCQ *et une décomposition* PartRec $T = (X = \{X_0, \ldots, X_n\}, F)$ *de* $\mathcal{N}$. *À partir de* T *et d'un élément* $X_i \in X$, *nous définissons inductivement un arbre* $\overline{T}_{X_i} = (X_{X_i}, F_{X_i})$ *de racine* $\overline{X_i}$ *de la manière suivante :*
- *Cas de base :* $X_i \in \mathsf{leaves}(T)$, $\overline{T}_{X_i} = (\{\overline{X_i}\}, \emptyset)$.
- *Cas Inductif :* $X_i \notin \mathsf{leaves}(T)$. *Dans* T, *soient* X_j *et* X_k *les deux noeuds fils de* X_i, $X_l = \mathsf{lca}(\widetilde{X_i X_j})$ *et* $X_m = \mathsf{lca}(\widetilde{X_i X_k})$. X_{X_i} *et* F_{X_i} *sont définis par* $X_{X_i} = X_{X_j} \cup X_{X_k} \cup \{\overline{X_i}\}$, $F_{X_i} = F_{X_j} \cup F_{X_k} \cup \{(\overline{X_i}, \overline{X_l}), (\overline{X_i}, \overline{X_m})\}$.

$\overline{T}$ *est défini par l'arbre de racine* $\overline{T}_{X_r}$ *avec* $X_r \in X$ *la racine de* T.

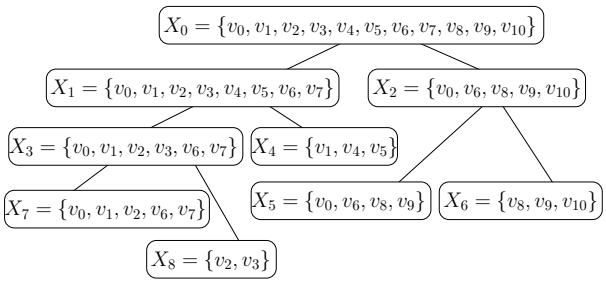
L'arbre $\overline{T}$ est une décomposition arborescente du RCQ $\mathcal{N}$ pour lequel T est une décomposition PartRec :

Proposition 4 *Soient* $\mathcal{N} = (V, C)$ *un* RCQ *et une décomposition* PartRec $T = (X = \{X_0, \ldots, X_n\}, F)$ *de* $\mathcal{N}$. *Nous avons :* $\overline{T} = (\overline{X} = (\overline{X} = \{\overline{X_0}, \ldots, \overline{X_n}\}, \overline{F})$ *est une décomposition arborescente de* $\mathcal{N}$ *telle que pour tout* $\overline{X_i} \in \overline{X}$ *il existe* $X_j \in \mathsf{leaves}(T)$ *avec* $\overline{X_i} \subseteq X_j$ *et telle que pour tout* $X_j \in \mathsf{leaves}(T)$ *il existe* $\overline{X_i} \in \overline{X}$ *avec* $\overline{X_i} = X_j$.

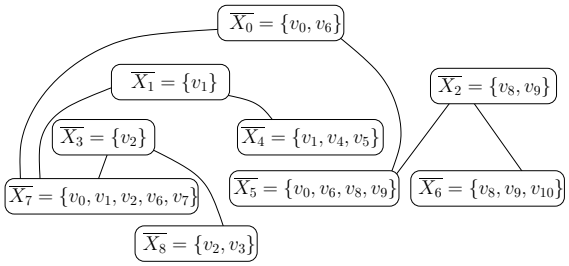
Ainsi, à toute décomposition T de type PartRec peut être associée une décomposition arborescente dont les clusters considèrent les mêmes regroupements de variables que ceux issus de $\mathsf{leaves}(T)$. Sous ce point de vue, une décomposition de type PartRec peut donc être considérée comme une décomposition arborescente particulière. À titre



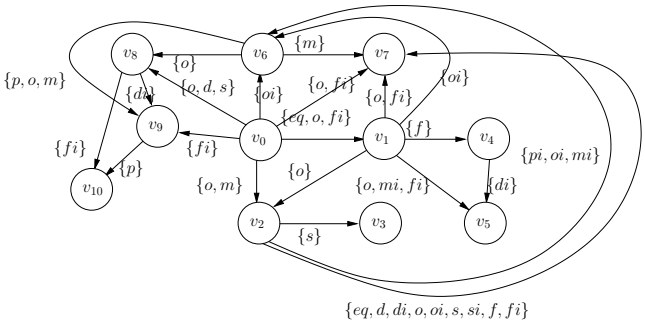
(a)



(b)



(c)



(d)

FIG. 2 – (a) un RCQ $\mathcal{N} = (V, C)$ de l'Algèbre des Intervalles, (b) une décomposition PartRec $T = (X, F)$ de $\mathcal{N}$, (c) une décomposition arborescente de $\mathcal{N}$ correspondant à $\overline{T}$, (d) un sous-RCQ de $\mathcal{N}$ ORD-Horn et ${}^{\diamond}_{X}$–cohérent.

d'illustration, considérons le RCQ $\mathcal{N} = (V, C)$ de la figure 2(a) et $T = (X, F)$ une de ses décompositions de type PartRec illustrée dans la figure 2(b). La décomposition arborescente $\overline{T}$ est représentée dans la figure 2(c).

4 Décompositions arborescentes et cohérences de RCQ

Dans cette section, nous montrons que pour certains RCQ, une décomposition arborescente permet de réduire le nombre de contraintes à considérer dans la cadre de la résolution du problème de la cohérence. En particulier, nous montrons que pour certaines classes de relations, la fermeture par faible composition restreinte aux clusters issus d'une décomposition arborescente est complète pour le problème de la cohérence. Tout d'abord, nous introduisons une nouvelle cohérence locale pour les RCQ correspondant à la propriété de $\diamond$-cohérence restreinte à des ensembles de variables d'un RCQ :

Définition 5 *Soient $\mathcal{N} = (V, C)$ un RCQ et $X = \{X_0, \ldots, X_n\}$ une famille de sous-ensembles de V. $\mathcal{N}$ est ${}^{\diamond}_{X}$-cohérent si, et seulement si, pour tout $X_i \in X$, le RCQ $\mathcal{N}_{X_i}$ est un RCQ $\diamond$-cohérent.*

Notons que la ${}^{\diamond}_{X}$-cohérence est proche de la chemin-cohérence partielle proposée dans [3]. Étant donnés un RCQ $\mathcal{N} = (V, C)$ et $X = \{X_0, \ldots, X_n\}$ une famille de sous-ensembles de V, nous dénoterons par ${}^{\diamond}_{X}(\mathcal{N})$ le plus grand (pour $\subseteq$) ${}^{\diamond}_{X}$-cohérent sous-RCQ de $\mathcal{N}$.
Le résultat suivant concerne des RCQ dont les contraintes sont définies par des relations issues d'une classe $\mathcal{C}$ pour laquelle tout RCQ fermé par faible composition est globalement cohérent. À titre d'illustration, nous pouvons citer la classe des relations convexes de l'Algèbre des Intervalles pour laquelle les RCQ possèdent une telle propriété.

Théorème 1 *Soit $\mathcal{N} = (V, C)$ un RCQ défini sur une classe de relations $\mathcal{C}$ pour laquelle tout RCQ $\diamond$-cohérent est globalement cohérent et soit $T = (X = \{X_0, \ldots, X_n\}, F)$ une décomposition arborescente de $\mathcal{N}$. Si $\mathcal{N}$ est un RCQ ${}^{\diamond}_{X}$-cohérent et non trivialement incohérent alors $\mathcal{N}$ est cohérent.*

Nous allons caractériser un résultat similaire pour la classe des relations ORD-Horn de l'Algèbre des Intervalles. Étant donné un RCQ $\mathcal{N}$ fermé par faible composition défini par des relations de cette classe, dans le cas général $\mathcal{N}$ n'est pas forcément globalement cohérent. Malgré tout, nous avons une propriété proche de la globale cohérence qui est satisfaite : toute solution partielle de maximale de $\mathcal{N}$ peut être étendue en une solution maximale de $\mathcal{N}$ (voir Proposition 6 dans [8]). Dans [8], Ligozat attribue une dimension (un entier compris entre 0 et 2) à chacune des relations de base de l'Algèbre des Intervalles. Une solution partielle de dimension maximale est une solution satisfaisant pour chaque couple de variables une relation de base de dimension maximale par rapport aux dimensions des autres relations de base présentes dans la contrainte. Ainsi, nous pouvons établir la propriété suivante :

Théorème 2 *Soit $\mathcal{N} = (V, C)$ un RCQ défini sur la classe des relations ORD-Horn de l'Algèbre des Intervalles et soit $T = (X = \{X_0, \ldots, X_n\}, F)$ une décomposition*

arborescente de $\mathcal{N}$. Si $\mathcal{N}$ est un RCQ $\overset{\diamond}{\underset{X}{}}$*-cohérent et non trivialement incohérent alors $\mathcal{N}$ est cohérent.*

Preuve La preuve est similaire à la preuve de la proposition 5 si ce n'est que les instanciations partielles manipulées sont des instanciations partielles maximales. $\dashv$

Nous avons démontré dans la section précédente qu'étant donné un RCQ $\mathcal{N}$ et une décomposition PartRec $T = (X = \{X_0, \ldots, X_n\}, F)$ d'un RCQ nous pouvions définir une décomposition arborescente $T' = \overline{T} = (X' = \{X'_0, \ldots, X'_n\}, F)$ telle que pour tout $X'_i \in X'$, il existe $X_j \in \mathsf{leaves}(T)$ avec $X'_i \subseteq X_j$. À partir de cela et des deux théorèmes précédents découlent les propriétés suivantes.

Corollaire 1 *Soient $\mathcal{N} = (V, C)$ un* RCQ *défini sur une classe de relations $\mathcal{C}$ et $T = (X = \{X_0, \ldots, X_n\}, F)$ une décomposition* PartRec *de $\mathcal{N}$.*

- *Si $\mathcal{C}$ est telle que tout* RCQ *$\diamond$-cohérent défini sur $\mathcal{C}$ est globalement cohérent et si $\mathcal{N}$ est $\overset{\diamond}{\underset{\mathsf{leaves}(T)}{}}$-cohérent alors $\mathcal{N}$ est cohérent.*
- *Si $\mathcal{C}$ est la classe des relations ORD-Horn de l'Algèbre des Intervalles et si $\mathcal{N}$ est $\overset{\diamond}{\underset{\mathsf{leaves}(T)}{}}$-cohérent alors $\mathcal{N}$ est cohérent.*

5 Algorithme

Dans cette section nous proposons un algorithme permettant de résoudre le problème de la cohérence d'un RCQ $\mathcal{N} = (V, C)$ à partir d'un ensemble X de clusters issus d'une décomposition arborescente $T = (X, F)$ de $\mathcal{N}$ et d'une classe de relations $\mathcal{C}$. La classe $\mathcal{C}$ est supposée être la classe des relations ORD-Horn de l'Algèbre des Intervalles ou bien une classe pour laquelle tout RCQ $\diamond$-cohérent est globalement cohérent. À l'aide d'une recherche exhaustive de type backtrack, l'algorithme proposé tente de caractériser un sous-RCQ de $\mathcal{N}$ $\overset{\diamond}{\underset{X}{}}$-cohérent et défini sur $\mathcal{C}$. L'intérêt de l'algorithme proposé réside d'une part dans le fait que seules les contraintes définies par des variables d'un même cluster seront considérées lors de la recherche. Et d'autre part, lors de la recherche, la fermeture par faible composition est réalisée uniquement au travers d'opérations de triangulation impliquant des triplets de variables issues d'un même cluster.

Présentons en détails l'algorithme proposé. La fonction principale Cohérence débute son traitement en dupliquant le RCQ $\mathcal{N}$ et le poursuit en réalisant un pré-traitement sur $\mathcal{N}$ à l'aide d'un appel de la fonction preTreatment. Cette fonction non définie explicitement doit retourner un sous-RCQ équivalent à $\mathcal{N}$. En règle générale, la méthode par faible composition est utilisée comme pré-traitement du fait de sa rapidité. Malgré tout, des méthodes avec des forces de filtrage plus élevées telles que celles proposées dans [4] peuvent être utilisées. Si le pré-traitement conduit à un RCQ trivialement incohérent l'algorithme se termine (lignes 4-5). À la ligne 6, est initialisé l'ensemble E par l'ensemble des couples de variables appartenant à au moins un des clusters de X. L'ensemble E délimite d'une part les contraintes sur lesquelles est réalisée la recherche (fonction CohérenceAux, ligne 5). D'autre part, E permet de délimiter les contraintes sur lesquelles doivent être réalisées les opérations de triangulation pour maintenir le

RCQ $\diamond_X$-cohérent, traitement réalisé par la fonction $\diamond_X$–Cohérence. La recherche effective est réalisée par la fonction CohérenceAux. En premier lieu, cette fonction maintient $\mathcal{N}$ $\diamond_X$–Cohérent (ligne 2). Dans le cas où le RCQ est détecté non cohérent, un retour arrière est réalisé (lignes 3-4). Dans le cas contraire, la prochaine contrainte à traiter est sélectionnée (ligne 5). La fermeture par rapport à $\mathcal{C}$ de la relation définissant la contrainte sélectionnée ne doit pas être incluse dans la relation définissant initialement cette même contrainte. Nous verrons par la suite que cette condition est suffisante pour assurer l'arrêt de la recherche (ligne 6-7). La relation $\mathcal{N}[v, v']$ correspondant à la contrainte sélectionnée est scindée en sous-relations $r_1, \ldots, r_k$ appartenant à $\mathcal{C}$ (ligne 8). Ainsi, nous avons $\mathcal{N}[v, v'] = r_1 \cup \ldots \cup r_k$. Puis, en remplaçant alternativement la contrainte sélectionnée par chacune de ces sous-relations r_i, un sous-RCQ de $\mathcal{N}$ cohérent est recherché à l'aide de l'appel de CohérenceAux à la ligne 11. Ce sous-RCQ aura pour particularité d'avoir comme relation définissant la contrainte entre les variables v et v' une relation $r \subseteq r_i$. Notons que $r_i \in \mathcal{C}$ et est forcément une sous-relation de $\mathcal{N}_{Init}[v, v']$, ainsi nous pouvons affirmer que la contrainte entre v et v' ne sera plus sélectionnée (ligne 5) lors des appels suivants de CohérenceAux. Ceci garanti l'arrêt de notre algorithme. Dans le cas où chacune des sous-relations $r_1, \ldots, r_k$ ne conduit pas à la caractérisation d'un sous-RCQ cohérent, l'incohérence de $\mathcal{N}$ est détectée.

Function `Cohérence(`$\mathcal{N}$,X,$\mathcal{C}$`)` : Booléen

in : $\mathcal{N} = (V, C)$, un RCQ, X un ensemble d'ensembles de variables de V, $\mathcal{C}$ une classe de relations.

output : $true$ si $\mathcal{N}$ est cohérent, $false$ sinon.

```
1 begin
2     N_Init ← N;
3     N ← preTreatment(N);
4     if N = ⊥ then
5         return false ;
6     E ← {(v, v') : ∃X_i ∈ X, avec v, v' ∈ X_i};
7     return CohérenceAux(N) ;
8 end
```

Théorème 3 *Soit $\mathcal{N} = (V, C)$ un* RCQ *défini sur une classe de relations $\mathcal{C}$. $\mathcal{C}$ est la classe des relations $OrdHorn$ de l'algèbre des intervalles ou bien est telle que tout* RCQ *$\diamond$-cohérent défini sur $\mathcal{C}$ est globalement cohérent. Soit $T = (X, F)$ une décomposition arborescente de $\mathcal{N}$. L'algorithme* Cohérence$(\mathcal{N}, X, \mathcal{C})$ *résout le problème de la cohérence de $\mathcal{N}$.*

Preuve. À partir de la description précédente de l'algorithme nous pouvons affirmer que l'algorithme s'arrête et que de plus, lorsqu'une incohérence est détectée, $\mathcal{N}$ est bien incohérent. Concentrons nous sur le cas où l'algorithme Cohérence détecte la cohérence de $\mathcal{N}$ et montrons que dans ce cas là, $\mathcal{N}$ est bien un RCQ cohérent. Dans la suite, $\mathcal{N}_D$ dénotera le RCQ $\mathcal{N}$ initial et nous dénoterons par $\mathcal{N}_F$ le RCQ correspondant au RCQ $\mathcal{N}$ lors de l'arrêt de Cohérence. Nous pouvons affirmer que $\mathcal{N}_F$ est un sous-RCQ de

Function `CohérenceAux(`$\mathcal{N}$`)` : Booléen

in : $\mathcal{N} = (V, C)$, un RCQ.
output : $true$ si $\mathcal{N}$ est cohérent, $false$ sinon.

1 **begin**
2 $\mathcal{N} \leftarrow \overset{\diamond}{_X}$–Cohérence($\mathcal{N}, E$);
3 **if** $\mathcal{N} = \perp$ **then**
4 **return** *false* ;
5 Sélectionne $(v, v') \in E$ tel que $(\mathcal{N}[v, v'])^{\mathcal{C}} \not\subseteq \mathcal{N}_{Init}[v, v']$;
6 **if** *un tel couple n'existe pas* **then**
7 **return** *true* ;
8 Partager $\mathcal{N}[v, v']$ en sous-relations $r_1, \ldots, r_k \in \mathcal{C}$;
9 **foreach** $i \in 1, \ldots, k$ **do**
10 $\mathcal{N}[v, v'] \leftarrow r_i$; $\mathcal{N}[v', v] \leftarrow r_i^{-1}$;
11 **if** CohérenceAux($\mathcal{N}$) **then**
12 **return** *true* ;
13 **return** *false*
14 **end**

Function $\overset{\diamond}{_X}$`-Cohérence(`$\mathcal{N}$`)`

in : $\mathcal{N} = (V, C)$.
output : $\mathcal{N}$ est $\diamond$-cohérent par rapport à E.

1 **begin**
2 changement $\leftarrow$ true ;
3 **while** *changement* **do**
4 changement $\leftarrow$ false ;
5 **foreach** *triplet* $(v_i, v_j), (v_i, v_k), (v_k, v_j) \in E$ **do**
6 $rel_{ij} \leftarrow \mathcal{N}[v_i, v_j] \cap (\mathcal{N}[v_i, v_k] \diamond \mathcal{N}[v_k, v_j])$;
7 **if** $\mathcal{N}[v_i, v_j] \not\subseteq rel_{ij}$ **then**
8 changement $\leftarrow$ true ;
9 $\mathcal{N}[v_i, v_j] \leftarrow rel_{ij}$;
10 **return** $\mathcal{N}$
11 **end**

$\mathcal{N}_I$ puisque toutes les opérations de modification du RCQ consistent en des suppressions de relations de base présentes dans ses contraintes. Soit $\mathcal{N}' = (V, C')$ défini par $\mathcal{N}'[v, v'] = (\mathcal{N}_F[v, v'])^{\mathcal{C}}$, pour tout v, v'. Du fait de la ligne 5 de CohérenceAux, nous pouvons affirmer que $\mathcal{N}' \subseteq \mathcal{N}_D$. Considérons maintenant le RCQ $\overset{\diamond}{_X}(\mathcal{N}')$, du fait que $\mathcal{N}'$ est défini sur la classe $\mathcal{C}$ nous pouvons affirmer que $\overset{\diamond}{_X}(\mathcal{N}')$ est également un RCQ défini sur $\mathcal{C}$ (ceci provient du fait que $\mathcal{C}$ est un ensemble stable pour les opérations d'intersection, d'inverse et de composition faible). Par définition de $\mathcal{N}'$ nous avons $\mathcal{N}_F \subseteq \mathcal{N}'$, il suit que $\overset{\diamond}{_X}(\mathcal{N}_F) \subseteq \overset{\diamond}{_X}(\mathcal{N}')$. Du fait de la ligne 2 de CohérenceAux, nous pouvons également affirmer que $\mathcal{N}_F = \overset{\diamond}{_X}(\mathcal{N}_F)$. Ainsi, $\mathcal{N}_F \subseteq \overset{\diamond}{_X}(\mathcal{N}') \subseteq \mathcal{N}_D$. De plus, $\overset{\diamond}{_X}(\mathcal{N}')$ n'est pas trivialement incohérent car $\mathcal{N}_F$ n'est pas trivialement incohé-

rent. $\overset{\diamond}{_X}(\mathcal{N}')$ est donc un RCQ non trivialement incohérent, défini sur $\mathcal{C}$ et $\overset{\diamond}{_X}$-cohérent. D'après les théorèmes 1 et 2 nous pouvons en déduire que $\overset{\diamond}{_X}(\mathcal{N}')$ est cohérent. $\overset{\diamond}{_X}(\mathcal{N}')$ étant un sous-RCQ de $\mathcal{N}_D$ nous pouvons en conclure que $\mathcal{N}_D$ est cohérent. $\dashv$

L'application de l'algorithme Cohérence nécessite au préalable le calcul d'une décomposition arborescente $T = (X, F)$ de $\mathcal{N}$. Notons qu'il n'est pas nécessaire de calculer une décomposition arborescente optimale (de largeur minimale) qui est un problème NP-difficile. Lors d'une étude expérimentale préliminaire, nous avons considéré des RCQ générés aléatoirement de manière similaire à celle utilisée dans [4]. À partir de décompositions de type PartRec calculées selon la méthode proposée dans [7], nous avons pu constater que le temps de calcul de ces décompositions est négligeable compte tenu du temps effectif mis lors de la recherche. D'autre part, l'utilisation de ces décompositions permet d'éliminer en moyenne un peu moins de 40% des triplets à considérer lors de l'application de la fermeture par faible composition et un peu plus de 20% des contraintes lors de la recherche pour les instances difficiles se trouvant à une densité de relations non triviales de 10.25. Ceci se traduisant par un gain de temps lors de la résolution de ces instances.

6 Conclusion et perspectives

Dans cet article, nous avons défini et étudié les décompositions de RCQ de type PartRec. Ces décompositions correspondent à celles utilisées par *Li et al*. [7] dans un processus de décomposition de RCQ proposé dans le cadre d'une traduction de RCQ en problèmes SAT. Nous avons montré que les décompositions de type PartRec peuvent s'exprimer de manière équivalente par des décompositions arborescentes. D'autre part, nous avons étudié le problème de la cohérence d'un RCQ en lien avec une décomposition arborescente, une classe traitable et la fermeture par faible composition. Nous avons montré que pour certaines classes de relations, en particulier pour la classe des relations ORD-Horn de l'algèbre des intervalles, l'obtention de la $\diamond$–cohérence sur les contraintes appartenant aux différents clusters de la décomposition arborescente est suffisante pour affirmer la cohérence du réseau de contraintes qualitatives. Ce résultat généralise des propriétés établis dans [7] puisque nous considérons d'une part des décompositions arborescentes et d'autre part des classes traitables incluant l'ensemble des relations singletons. Nous avons également proposé un algorithme permettant de décider du problème de la cohérence des RCQ en tirant profit d'une classe traitable et d'une décomposition arborescente. Une évaluation expérimentale de cet algorithme est en cours de réalisation. Elle consiste en l'application de cet algorithme sur des RCQ de l'Algèbre des Intervalles en utilisant la classe des relations ORD-Horn et des décompositions PartRec calculées d'une manière similaire à celle utilisée par *Li et al*. [7]. Il est également envisagé d'utiliser des décompositions arborescentes autres que les décompositions de type PartRec. Une autre perspective de recherches est la définition et l'étude d'algorithmes inspirés de l'algorithme BTD [6] afin de profiter des structures des décompositions arborescentes utilisées lors de la résolution des RCQ.

Références

[1] J-F. Allen. An interval-based representation of temporal knowledge. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI'81)*, pages 221–226, 1981.

[2] Peter Van Beek and Robin Cohen. Exact and approximate reasoning about temporal relations. *Computational Intelligence*, 6 :133–44, 1990.

[3] Christian Bliek and Djamila Sam-Haroud. Path consistency on triangulated constraint graphs. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 456–461, 1999.

[4] Jean-François Condotta and Christophe Lecoutre. A class of df-consistencies for qualitative constraint networks. In *Proceedings of 12th International Conference on Principles of Knowledge Representation and Reasoning (KR'10)*, pages 319–328, 2010.

[5] Z. Gantner, M. Westphal, and S. Wolfl. GQR a fast reasoner for binary qualitative constraint calculi. In *Proceedings of AAAI'08 Workshop on Spatial and Temporal Reasoning*, 2008.

[6] Philippe Jégou and Cyril Terrioux. Hybrid backtracking bounded by tree-decomposition of constraint networks. *Artificial Intelligence*, 146(1) :43–75, 2003.

[7] Jason Jingshi Li, Jinbo Huang, and Jochen Renz. A divide-and-conquer approach for solving interval algebra networks. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI'09)*, pages 572–577, 2009.

[8] Gérard Ligozat. A New Proof of Tractability for ORD-Horn Relations. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI'96)*, volume 1, pages 395–401, 1996.

[9] Gérard F. Ligozat. On Generalized Interval Calculi. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI'91)*, pages 234–240, Menlo Park/Cambridge, July 1991. American Association for Artificial Intelligence, AAAI Press/The MIT Press.

[10] Robert A. Morris, William D. Shoaff, and Lina Al-Khatib. Path consistency in a network of non-convex intervals. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI'93)*, 1993.

[11] B. Nebel. Solving hard qualitative temporal reasoning problems : Evaluating the efficienty of using the ORD-Horn class. In *Proceedings of ECAI'96*, pages 38–42, 1996.

[12] Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning About Temporal Relations : A Maximal Tractable Subclass of Allen's Interval Algebra. *Journal of the ACM*, 42(1) :43–66, 1995.

[13] Duc Nghia Pham, John Thornton, and Abdul Sattar. Modelling and solving temporal reasoning as propositional satisfiability. *Artif. Intell.*, 172(15) :1752–1782, 2008.

[14] A.K. Pujari, G.V. Kumari, and A. Sattar. Indu : An interval and duration network. In *Proceedings of Australian Joint Conference on Artificial Intelligence*, pages 291–303, 1999.

[15] D.A. Randell, Z. Cui, and A.G. Cohn. A spatial logic based on regions and connection. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, pages 165–176, 1992.

[16] J. Renz. Maximal tractable fragments of the region connection calculus : a complete analysis. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 448–455, 1999.

[17] Jochen Renz and Gérard Ligozat. Weak composition for qualitative spatial and temporal reasoning. In *Proceedings of the Eleventh International Conference on Principles and Practice of Constraint Programming (CP'05)*, 2005.

[18] Neil Robertson and Paul D. Seymour. Graph minors. ii. algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3) :309–322, 1986.

Annexe

Preuve (Proposition 2).

(1) Considérons le cas où $X_i \notin \mathsf{leaves}(T)$ (pour le cas où $X_i \in \mathsf{leaves}(T)$ la propriété est trivialement vraie puisque $\overline{X_i} = X_i$ et $X_i \neq \emptyset$). Soient $X_i, X_j, X_k \in X$ tels que X_i est le père de X_j et X_k. Supposons par l'absurde que $X_j \cap X_k = \emptyset$. D'après la définition 3, nous avons $\mathcal{N}_{X_i} = \mathcal{N}_{X_j} \cup \mathcal{N}_{X_k}$. Ainsi, pour tout $v \in X_j$ et $v' \in X_k$, nous pouvons affirmer que $\mathcal{N}[v, v'] = \Psi$ et que (v, v') n'est pas une arête de $\mathsf{G}(\mathcal{N})$. De plus, nous avons précédemment effectuer l'hypothèse que pour tout RCQ $\mathcal{N}$ considéré $\mathsf{G}(\mathcal{N})$ est connecté. Ainsi, pour tout $v \in X_j$ et $v' \in X_k$, il existe un chemin dans $\mathsf{G}(\mathcal{N})$ entre v et v' et ce chemin passe forcément par deux arêtes (v, v'') et (v, v''') avec $v'' \in V \setminus \{X_j \cup X_k\}$ et $v''' \in V \setminus \{X_j \cup X_k\}$. Puisque $X_j \cup X_k = X_i$ nous avons $v'' \notin X_i$. Soit X_l le plus proche ascendant de X_i tel que $v'' \in X_l$. Soit X_m et X_o les deux fils de X_l avec $X_m \in \mathsf{asc}(X_i)$. Nous avons $v'' \notin X_m$ et $v \in X_m$. De plus, $v'' \in X_o$ puisque $X_o \cup X_m = X_l$. De plus, nous savons que $\mathcal{N}_{X_l} = \mathcal{N}_{X_m} \cup \mathcal{N}_{X_o}$. Il s'ensuit que $\mathcal{N}[v, v''] = \Psi$. Or, (v, v'') est une arête de $\mathsf{G}(\mathcal{N})$. Il y a une contradiction. Nous pouvons conclure $X_j \cap X_k \neq \emptyset$.

(2) Soient $X_i, X_j, X_k \in X$ avec $X_k = \mathsf{lca}(\{X_i, X_j\})$. Supposons que $X_k \neq X_i$ et $X_k \neq X_j$. Nous avons forcément $X_k \notin \mathsf{leaves}(T)$, dénotons par X_l et X_m les deux fils de X_k. Nous avons $X_i \in \mathsf{desc}(X_l)$ et $X_j \in \mathsf{desc}(X_m)$ ou bien, $X_i \in \mathsf{desc}(X_m)$ et $X_j \in \mathsf{desc}(X_l)$. Ainsi d'après la proposition 1 (1), nous avons $X_i \subseteq X_l$ et $X_j \subseteq X_m$ ou $X_i \subseteq X_m$ et $X_j \subseteq X_l$. Il suit que $X_i \cap X_j \subseteq X_m \cap X_l$. Nous pouvons en conclure que $X_i \cap X_j \subseteq \overline{X_k}$.

(3) Soit $X_i \in X$. Considérons tout d'abord le cas où $X_i \in \mathsf{leaves}(X)$. Par définition, $\overline{X_i} = X_i$, ainsi la propriété est satisfaite. Supposons maintenant que $X_i \notin \mathsf{leaves}(X)$ et soient X_k et X_l les deux fils de X_i. Nous avons $X_k \cap X_l = \overline{X_i} \neq \emptyset$. Soit $v \in \overline{X_i}$, nous avons $v \in X_i$. D'après la proposition 1 (4), il existe $X_j \in \mathsf{leaves}(T) \cap \mathsf{desc}(X_i)$, tel que $v \in X_j$. D'après la propriété (2) de la définition 3, nous avons $X_k \cap X_l \subseteq X_j$ puisque $(X_k \cap X_l) \cap X_j \neq \emptyset$.

$\dashv$

Preuve (Proposition 3).

- Nous savons que $\overline{X_i} \neq \emptyset$ (proposition 2 (1)), il existe donc une variable $v \in \overline{X_i}$. Par définition de $\overline{X_i}$, nous savons que $v \in X_j$. D'après la proposition 1 (4), il existe $X_k \in \mathsf{leaves}(T) \cap \mathsf{desc}(X_j)$ tel que $v \in X_k$. D'après la propriété (3) de la définition 3, nous pouvons affirmer que $\overline{X_i} \subseteq X_k$. Ainsi, $X_k \in \widetilde{X_i X_j}$ et nous pouvons en conclure que $\widetilde{X_i X_j} \neq \emptyset$.
- Soit $X_l = \mathsf{lca}(\widetilde{X_i X_j})$. Il existe $X_k, X_m \in \widetilde{X_i X_j}$ tels que $X_l = \mathsf{lca}(\{X_k, X_m\})$. Dans le cas où $X_l = X_k$ ou $X_l = X_m$ la propriété est trivialement vraie. Dans le

cas où, $X_l \neq X_k$ et $X_l \neq X_m$ nous avons d'après la proposition 2 (2), $X_k \cap X_m \subseteq \overline{X_l}$. Comme $\overline{X_i} \subseteq \overline{X_k}$ et $\overline{X_i} \subseteq \overline{X_m}$ nous pouvons en déduire que $\overline{X_i} \subseteq \overline{X_l}$. De plus, comme $X_l = \mathsf{lca}(\{X_k, X_m\})$, $X_k \in \mathsf{desc}(X_j)$ et $X_m \in \mathsf{desc}(X_j)$, nous avons $X_l \in \mathsf{desc}(X_j)$. Nous pouvons en conclure que $X_l \in \widetilde{X_i X_j}$.

$\dashv$

Preuve (Proposition 4). Les propriétés (1) et (2) de la définition 2 découlent du fait que pour tout $X_i \in \mathsf{leaves}(X)$, $\overline{X_i} = X_i$ et de la proposition 1. Montrons maintenant que la propriété (3) de la définition 2 est satisfaite par $\overline{T}$. Soient $\overline{X_i}, \overline{X_j}, \overline{X_k} \in \overline{X}$ avec $\overline{X_j}$ sur l'unique chemin entre $\overline{X_i}$ et $\overline{X_k}$ dans $\overline{T}$. Montrons que si $v \in \overline{X_i}$ et $v \in \overline{X_k}$ alors $v \in \overline{X_j}$. Si la longueur du chemin entre $\overline{X_i}$ et $\overline{X_k}$ est 0 la propriété est trivialement vraie. Supposons que la propriété soit vraie pour tout chemin de longueur $l \geq 0$ et montrons par induction qu'elle est vraie pour tout chemin entre $\overline{X_i}$ et $\overline{X_k}$ de longueur $l + 1$. Supposons dans un premier temps que $X_i \notin \mathsf{desc}(X_k)$ et $X_k \notin \mathsf{desc}(X_i)$ dans T. En examinant la définition 4 nous remarquons que dans $\overline{T}$ un chemin entre $\overline{X_i}$ et $\overline{X_k}$ passe forcément par $\overline{X_l}$ avec $X_l = \mathsf{lca}(\{X_i, X_k\})$. Nous avons d'après la proposition 2 (2), $X_i \cap X_k \subseteq \overline{X_l}$. Par conséquent, $v \in \overline{X_l}$. En considérant les deux chemins $\overline{X_i}, \ldots, \overline{X_l}$ et $\overline{X_l}, \ldots, \overline{X_j}$ qui ont une longueur inférieure ou égale à l, et en utilisant l'hypothèse d'induction nous pouvons affirmer que tous les noeuds du chemin $\overline{X_i}, \ldots, \overline{X_l}$ et ceux du chemin $\overline{X_l}, \ldots, \overline{X_k}$ contiennent v puisque $\overline{X_i}, \overline{X_l}$ et $\overline{X_k}$ contiennent v. Supposons maintenant que $X_i \in \mathsf{desc}(X_k)$ dans T (le cas $X_k \in \mathsf{desc}(X_i)$ peut être traité de manière similaire). Le cas $X_i = X_k$ est trivial, supposons que $X_i \neq X_k$. Nous avons $X_i \in \mathsf{desc}(X_l)$ avec X_l un des deux fils de X_k dans T. En examinant la définition 4 nous remarquons que dans $\overline{T}$, un chemin entre $\overline{X_k}$ et $\overline{X_i}$ passe forcément par $\overline{X_m}$ avec $X_m = \mathsf{lca}(\widetilde{X_k X_l})$. D'après la proposition 3 (2), nous savons que $\overline{X_k} \subseteq \overline{X_m}$. Il en résulte que $v \in \overline{X_m}$. En considérant les deux chemins $\overline{X_i}, \ldots, \overline{X_m}$ et $\overline{X_m}, \ldots, \overline{X_k}$ qui ont une longueur inférieure ou égale à l, et en utilisant l'hypothèse d'induction nous pouvons affirmer que tous les noeuds du chemin $\overline{X_i}, \ldots, \overline{X_m}$ et ceux du chemin $\overline{X_m}, \ldots, \overline{X_k}$ contiennent v puisque $\overline{X_i}, \overline{X_m}$ et $\overline{X_l}$ contiennent v.
Le fait que pour tout $\overline{X_i} \in \overline{X}$, il existe $X_j \in \mathsf{leaves}(T)$ tel que $\overline{X_i} \subseteq X_j$ découle, de manière directe de la proposition 2 (3). $\dashv$

Preuve (Proposition 1). Nous supposons sans perte de généralité que T possède une racine. Soient un noeud $X_i \in X$ et $T_{X_i} = (X_{X_i}, F_{X_i})$ le sous-arbre de T de racine X_i. Étant donnée une instanciation partielle s sur V' avec $V' \cap X'_i \subseteq X_i$ pour tout $X'_i \in X_{X_i}$ et telle que pour tout $X_j \in X$ avec $X_j \subseteq V'$, s_{X_j} est solution de $\mathcal{N}_{X_j}$, nous allons montrer la propriété suivante. s peut-être étendue en une instanciation partielle s' sur $V'' = V' \cup \bigcup\{X'_i \in X_{X_i}\}$ telle que que pour tout $X_j \in X$ avec $X_j \subseteq V''$, s'_{X_j} est solution de $\mathcal{N}_{X_j}$. Nous allons effectuer une preuve par induction sur la taille de X_{X_i}.

- Cas de base : $|X_{X_i}| = 1$. Nous avons $X_{X_i} = \{X_i\}$. Puisque $\mathcal{N}$ est $\diamond_X$-cohérent, nous avons le RCQ $\mathcal{N}_{X_i}$ qui est un RCQ $\diamond$-cohérent et donc globalement cohérent. $s_{V' \cap X_i}$ est une solution partielle de $\mathcal{N}_{X_i}$ pouvant être étendue en une solution s'' de $\mathcal{N}_{X_i}$. Nous définissons par s' l'instanciation partielle sur $V' \cup X_i$ de la manière suivante : si $v \in V'$ alors $s'(v) = s(v)$ sinon $s'(v) = s''(v)$. Nous avons s'_{X_i} qui est une solution de $\mathcal{N}_{X_i}$ et plus généralement s'_{X_k} est solution de $\mathcal{N}_{X_k}$ pour tout $X_k \in X$ et $X_k \subseteq V' \cup X_i$.
- Cas d'induction : $|X_{X_i}| > 1$. Nous supposons la propriété vraie pour tout sous-

arbre $T_{X_j} = (X_{X_j}, F_{X_j})$ avec $|X_{X_j}| < |X_{X_i}|$. Comme dans le cas précédent nous étendons s en une instanciation partielle s' sur $V' \cup X_i$ telle que s'_{X_k} est solution de $\mathcal{N}_{X_k}$ pour tout $X_k \in X$ et $X_k \subseteq V' \cup X_i$. Par hypothèse d'induction cette instanciation partielle s' peut-être étendue à toutes les variables appartenant aux descendants de X_i. En effet, considérons X_l un fils de X_i. Nous remarquons tout d'abord qu'en dénotant par $T_{X_l} = (X_{X_l}, F_{X_l})$ le sous-arbre de T de racine X_l, $|X_{X_l}| < |X_{X_i}|$. De plus comme T est une décomposition arborescente, nous avons pour tout $X_m \in X_{X_l}$, $X_m \cap (V' \cup X_i) \subseteq X_l$ (d'après la propriété (3) de la définition 2). Ainsi, l'hypothèse d'induction peut s'appliquer sur $T_{X_l} = (X_{X_l}, F_{X_l})$.

En appliquant la propriété précédente sur X_r racine de T, nous savons qu'il existe une instanciation partielle s sur l'ensemble des variables $\bigcup\{X_i : X_i \in X\}$ telle que s_{X_i} est solution du RCQ $\mathcal{N}_{X_i}$ pour tout $X_i \in X$. Notons tout d'abord que $V = \bigcup\{X_i : X_i \in X\}$ d'après la propriété (1) de la définition 2. Ainsi, s est une instanciation sur V. De plus nous pouvons montrer que s est solution de $\mathcal{N}$. En effet, soient $v, v' \in V$, si $\mathcal{N}[v, v'] = \Psi$ nous avons $s(v)$ et $s(v')$ qui satisfont clairement la contrainte $\mathcal{N}[v, v']$. Maintenant supposons que $\mathcal{N}[v, v'] \neq \Psi$. D'après la propriété (2) de la définition 2, il existe $X_i \in X$ tel que $v \in X_i$ et $v' \in X_i$. Nous savons que $s(v)$ et $s(v')$ satisfont $\mathcal{N}_{X_i}[v, v']$. Comme, $\mathcal{N}_{X_i}[v, v'] = \mathcal{N}[v, v']$ nous en déduisons que $s(v)$ et $s(v')$ satisfont $\mathcal{N}[v, v']$. Nous pouvons conclure que s est une solution de $\mathcal{N}$ $\dashv$

Sur le voisinage entre configurations qualitatives spatio-temporelles

Dominique D'Almeida et Nicolas Schwind

CRIL – CNRS
Université d'Artois
Lens, France
{dalmeida, schwind}@cril.univ-artois.fr

Résumé : La fusion et la révision de connaissances temporelles ou spatiales sous la forme de réseaux de contraintes qualitatives (RCQ) sont motivées par de nombreuses applications. En particulier, déterminer la distance ou le coût éventuel qui induit la modification d'un scénario est une tâche essentielle. Pour cela, la notion de voisinage conceptuel entre relations de base de certains formalismes qualitatifs a été introduite dans la littérature. Actuellement, l'approche utilisée pour le calcul de la distance possède l'inconvénient d'être trop dépendante de l'aspect structurel du RCQ, car elle est réalisée localement, de manière "syntaxique" sur les contraintes. Dans cet article, nous proposons une nouvelle définition du voisinage conceptuel entre relations de base pour de nombreux formalismes qualitatifs. En nous appuyant sur les configurations qualitatives spatio-temporelles, nous donnons une définition de voisinage entre scénarios pertinente et naturelle, nous permettant ainsi d'y associer la notion de distance.

Mots-clés : formalismes qualitatifs, réseaux de contraintes qualitatives, voisinage conceptuel

1 Introduction

La représentation des connaissances est un domaine majeur en intelligence artificielle. En particulier, la représentation de connaissances temporelles ou spatiales prend une part très importante dans ce domaine, et les applications sont nombreuses (e.g., systèmes d'informations géographiques, navigation de robots, planification). L'approche la plus communément utilisée est une approche qualitative, dans laquelle les entités temporelles et spatiales ne sont pas associées à des données « numériques », mais à des notions symboliques. Ces informations peuvent être représentées par un formalisme qualitatif, défini par un ensemble fini de symboles, appelés *relations de base*, caractérisant la position relative entre des entités temporelles ou spatiales. Parmi les formalismes qualitatifs les plus célèbres, l'algèbre des intervalles (Allen, 1981) permet de modéliser des informations temporelles, décrites par des intervalles sur la droite du temps, et l'algèbre RCC8 (Randell *et al.*, 1992) permet la représentation d'informations spa-

tiales/topologiques. Dans le cas général, décider de la cohérence d'informations qualitatives modélisées à l'aide de tels formalismes est un problème NP-complet.

Face à des connaissances erronées ou incomplètes, Freksa (1992) propose d'adapter l'algèbre des intervalles en s'intéressant à la notion de *voisinage conceptuel* entre relations de base. Cette approche permet un raisonnement sur des connaissances fines ou grossières en exploitant les propriétés « physiques » de ce formalisme. Ceci a pour but d'améliorer la puissance du processus d'inférence par abstraction ou relaxation sur les connaissances. Gooday & Cohn (1994) adaptent les travaux de Freksa au formalisme RCC-8 afin d'améliorer l'efficacité calculatoire. Cependant, définir le voisinage conceptuel entre relations de base n'est pas trivial. Cette notion est dépendante des formalismes qualitatifs et de la façon dont les entités temporelles ou spatiales peuvent se déplacer dans leur domaine. La notion de voisinage est donc régie par une *loi de transformation*, basée sur la notion de continuité dans un espace topologique.

Le voisinage conceptuel est également étudié par Egenhofer & Al-Taha (1992), en identifiant les *déformations topologiques* des relations de base du formalisme RCC-8, dans le but de déterminer l'évolution de certaines connaissances ou de manière plus fine les étapes par lesquelles passe cette évolution. Cette approche relève des domaines de la révision et de la fusion de connaissances temporelles ou spatiales, complètes ou incomplètes, et l'étude du voisinage conceptuel est un élément primordial pour définir une *distance* entre les connaissances. Pour cela, la manipulation des connaissances temporelles ou spatiales est facilitée par une représentation sous la forme de *réseaux de contraintes qualitatives* (RCQ en abrégé). On parle alors de distance entre *configurations* - un état figé des entités dans leur domaine - ou *scénarios* - un ensemble de configurations défini par les relations entre entités - qui sont les ingrédients majeurs des processus cités précédemment (Condotta *et al*., 2008, 2010). Dans la littérature, le calcul de ces distances relève d'une analyse « syntaxique » des contraintes du RCQ, et est réalisé localement à partir de l'agrégation des distances entre relations de base. En d'autres termes, cette distance dépend de la structure qui décrit les connaissances et non des configurations qualitatives qu'elles représentent. Notre approche permet de définir un voisinage « sémantique » entre scénarios cohérents plus naturel et représentatif du domaine, au moyen du voisinage entre les configurations de ces scénarios. Ce voisinage induit naturellement une nouvelle définition de distance entre scénarios cohérents, pour de nombreux formalismes qualitatifs.

La suite de l'article s'organise comme suit. Dans la section 2, nous introduisons quelques notions préliminaires sur les formalismes qualitatifs et la représentation sous la forme de RCQ. Nous y introduisons également un exemple « fil conducteur » sur l'algèbre des intervalles nous permettant d'illustrer nos définitions. Nous donnons également quelques rappels de topologie et introduisons les notations utilisées. Dans la section 3, nous définissons la notion de « loi de transformation », illustrée en adaptant les lois de transformations de Freksa à notre cadre. Nous définissons ensuite le voisinage conceptuel entre relations de base, puis nous illustrons le graphe de voisinage des relations pour deux lois de transformations. Dans la section 4, nous définissons le voisinage entre scénarios cohérents ainsi que la notion de distance entre scénarios, avant de conclure dans la dernière section.

2 Notions préliminaires

2.1 Formalismes qualitatifs

Un formalisme qualitatif considère un ensemble fini B de relations de base binaires définies sur un domaine D. Les éléments de D représentent les entités temporelles ou spatiales considérées. Chaque relation de base $b \in \mathsf{B}$ représente une position relative particulière pouvant être satisfaite par deux éléments de D. L'ensemble B satisfait également les trois propriétés suivantes : (i) B forme une partition de $\mathsf{D} \times \mathsf{D}$, autrement dit tout couple de $\mathsf{D} \times \mathsf{D}$ satisfait une et une seule relation de base de B ; (ii) la relation identité sur D, notée eq, appartient à l'ensemble B ; (iii) pour toute relation de base $b \in \mathsf{B}$, la relation inverse de b, notée b^{-1}, appartient également à l'ensemble B. Ces trois propriétés font de B un *schéma de partition* (Ligozat & Renz, 2004).

Par exemple, considérons le formalisme d'Allen, dit également l'algèbre des intervalles (Allen, 1981), qui permet de décrire des positions relatives entre des intervalles de temps non ponctuels, sans notion de mesure sur ces intervalles. Ce formalisme considère un ensemble B_{int} de treize relations de base définies sur le domaine des intervalles fermés de la droite $\mathsf{D}_{int} = \{(x^-, x^+) \in \mathbb{R}^2 \ : x^- < x^+\}$. Les relations de base de $\mathsf{B}_{int} = \{eq, p, pi, m, mi, o, oi, s, si, d, di, f, fi\}$ sont illustrées dans le tableau 1.

Relation	Illustration	Relation inverse
X précède Y p	X Y	Y succède X pi
X rencontre Y m	X Y	Y est rencontré par X mi
X chevauche Y o	X Y	Y est chevauché par X oi
X commence Y s	X Y	Y est commencé par X si
X est contenu par Y d	X Y	Y contient X di
X fini Y f	X Y	Y est fini par X fi
X est égal à Y eq	X Y	Y est égal à X eq

TABLE 1 – Illustration des relations de base de l'algèbre des intervalles

Les entités peuvent être mises en relation par une relation dite *complexe*. Une relation complexe est l'union de relations de base, que l'on représente par l'ensemble des relations de base qu'elle contient. (dans la suite, nous omettons le qualificatif « complexe »). Par exemple, dans le cadre du formalisme d'Allen, deux intervalles x et y satisfont la relation $\{p, m, eq\}$, noté $x \ \{p, m, eq\} \ y$, si et seulement si l'intervalle x *précède*, *rencontre* ou *est égal à* l'intervalle y. L'ensemble 2^{B}, l'ensemble des sous-ensembles de B, forme l'ensemble des relations. 2^{B} est muni, entre autres, des opérations ensemblistes usuelles et de l'opération *inverse* définie par $R^{-1} = \{b^{-1} \mid b \in R\}$. Dans la suite, étant donné un ensemble B de relations de base sur un domaine D, 2^{B} dénotera le formalisme qualitatif considéré.

2.2 Réseaux de contraintes qualitatives

Il est courant de représenter un ensemble de configurations qualitatives à propos d'un ensemble d'entités spatiales ou temporelles à l'aide d'un graphe de contraintes, appelé *réseau de contraintes qualitatives* (RCQ en abrégé). Les nœuds d'un tel graphe représentent les entités en question et ses arcs représentent des contraintes définies pour chacune d'elles par un ensemble de relations de base « permises » entre les entités reliées par la contrainte. Une structure de RCQ constitue un moyen très naturel pour représenter un ensemble de croyances ou de préférences sur un ensemble d'entités, et fournit également les outils algorithmiques nécessaires pour le raisonnement et la déduction sur les informations qualitatives. Formellement, étant donné un formalisme qualitatif 2^{B} défini sur le domaine D, un RCQ sur 2^{B} est un couple (V, C) où $V = \{v_1, ..., v_n\}$ est un ensemble fini de variables représentant les entités, et C est une application qui associe à tout couple de variables (v_i, v_j) une relation $N[v_i, v_j]$ de 2^{B}, notée également $N[i, j]$. C est telle que pour tout couple de variables (v_i, v_j), $N[i, i] = \{eq\}$ et $N[i, j] = N[j, i]^{-1}$.

Étant donné un RCQ $N = (V, C)$, le RCQ $N' = (V, C')$ est un *sous*-RCQ de N, noté $N' \subseteq N$, si et seulement si pour tout $v_i, v_j \in V$, $N'[i, j] \subseteq N[i, j]$. Un RCQ est atomique si et seulement si chaque contrainte est définie par une relation singleton de 2^{B}. Un *scénario* de N est un sous-RCQ atomique de N. Une *instanciation* est une application de V dans D. Une instanciation α est appelée *solution* d'un RCQ $N = (V, C)$ si et seulement si pour tout couple de variables v_i, v_j, $(\alpha(v_i), \alpha(v_j)) \in N[i, j]$. Un RCQ est *cohérent* si et seulement si il admet une solution. Dans la suite de l'article, $[N]$ dénotera l'ensemble des scénarios cohérents de N et $[\![N]\!]$ l'ensemble de ses solutions.

Exemple 1 (Réseau de contraintes qualitatives)
Pour illustrer les RCQ*, considérons l'exemple suivant : la présentation de travaux scientifiques doit être réalisée durant un certain temps (T). La présentation se compose d'un exposé (E), lui-même terminé par une session dédiée aux questions (Q), puis elle est suivie d'une phase de délibération (D). Le* RCQ *$N = (V, C)$ correspondant, défini sur le formalisme d'Allen $2^{\mathsf{B}_{int}}$ est représenté sur la figure 1. On a $V = \{T, E, Q, D\}$ et C défini par les contraintes décrites ci-dessus*[1]*. Par exemple, l'exposé (E) précède ou rencontre la phase de délibération (D).*

La figure 2 illustre, quant à elle, un scénario cohérent σ de N ainsi qu'une solution α de ce scénario.

Dans le cas général, le problème de la cohérence d'un RCQ sur un formalisme qualitatif quelconque, i.e., le problème de la recherche d'une solution de ce RCQ, est un problème indécidable (Hirsch, 1999). Néanmoins, nous considérons d'une part que résoudre le problème de la cohérence d'un RCQ N revient à déterminer si N admet un scénario cohérent, et d'autre part nous supposons que le problème de la cohérence d'un scénario est décidable en temps polynomial, ce qui est le cas à notre connaissance pour l'ensemble des formalismes qualitatifs d'intérêt qui ont été définis dans la littérature.

1. Par souci de clarté, pour tout couple de variables (v_i, v_j), on ne représente pas la contrainte $N[i, j]$ lorsque $N[i, j] = \mathsf{B}$ (i.e., lorsqu'aucune information n'est spécifiée quant à la position relative entre les entités représentées par les variables v_i et v_j), lorsque $N[j, i]$ est représentée ou lorsque $i = j$.

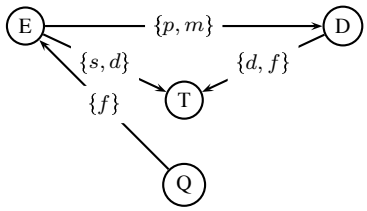


FIGURE 1 – N, un RCQ défini sur $2^{\mathsf{B}_{int}}$

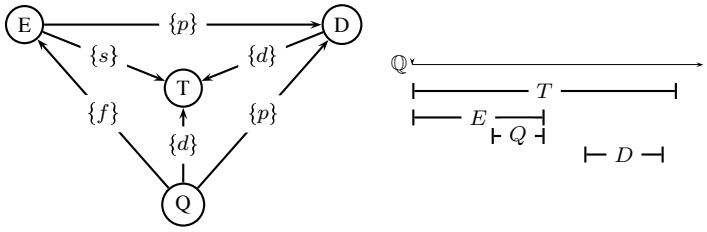


FIGURE 2 – σ, un scénario cohérent de N et α, une solution de σ.

Par conséquent, le problème de la cohérence d'un RCQ appartient à la classe NP de problèmes de décision. Pour la plupart des formalismes qualitatifs d'intérêt, ce problème est NP-complet.

2.3 Rappels de topologie et notations

Nous rappelons maintenant quelques notions de base sur les topologies. Considérons D un ensemble fixé. Une topologie sur D est un couple $(\mathsf{D}, Vois_{\mathcal{T}})$ où $Vois_{\mathcal{T}}$ est une application qui associe à tout élément x de D un ensemble non vide $Vois_{\mathcal{T}}(x)$ de sous-ensembles de D appelés *voisinages* de x, et qui vérifie les conditions suivantes, $\forall x \in \mathsf{D}$, $\forall W \in Vois_{\mathcal{T}}(x)$:

- $x \in W$;
- $\forall W' \subseteq \mathsf{D}$, si $W \subseteq W'$, alors $W' \in Vois_{\mathcal{T}}(x)$;
- $\forall W' \in Vois_{\mathcal{T}}(x), W \cap W' \in Vois_{\mathcal{T}}(x)$;
- $\exists W' \in Vois_{\mathcal{T}}(x)\ \forall y \in W', W \in Vois_{\mathcal{T}}(y)$.

Un *ouvert* O d'une topologie $\mathcal{T}$ sur D est un sous-ensemble de D qui est voisinage de chacun de ses points. Une topologie peut être également définie à partir des ouverts, et dans ce cas le voisinage d'un élément x de D correspond à l'ensemble des sous-ensembles de D qui contiennent un ouvert contenant x.

Soit $E \subseteq \mathsf{D}$. Étant donné une topologie particulière $\mathcal{T}$ sur D, l'*adhérence* de E, notée $adh_{\mathcal{T}}(E)$, est définie par $adh_{\mathcal{T}}(E) = \{x \in \mathsf{D} \mid \forall W \in Vois_{\mathcal{T}}(x), W \cap E \neq \emptyset\}$; l'*intérieur* de E, noté $int_{\mathcal{T}}(E)$, est défini par $int_{\mathcal{T}}(E) = \{x \in \mathsf{D} \mid E \in Vois_{\mathcal{T}}(x)\}$; la *frontière* de E, notée $front_{\mathcal{T}}(E)$, est définie par $front_{\mathcal{T}}(E) = adh_{\mathcal{T}}(E) \backslash int_{\mathcal{T}}(E)$.

3 Loi de transformation et voisinage conceptuel entre relations

Une notion de voisinage conceptuel peut être définie entre les relations de base d'un schéma de partition B. Freksa (1992) est le premier à évoquer une telle notion dans le cadre du formalisme d'Allen. Parallèlement, Egenhofer & Al-Taha (1992) ont proposé une notion similaire de voisinage, appelée *distance topologique*, entre des relations de type topologique, correspondant aux relations de base du formalisme RCC8 (Randell *et al.*, 1992). Quel que soit le formalisme qualitatif étudié, la notion de voisinage conceptuel entre relations est elle-même basée sur une notion sous-jacente de *loi de transformation* des entités dans leur domaine.

3.1 Loi de transformation

Une loi de transformation décrit l'ensemble des manières possibles pour une entité du domaine de se « transformer » au sein de son domaine, autrement dit, l'ensemble des valeurs d'une entité qui succèdent directement une valeur initiale de l'entité, lorsque l'on souhaite « transformer » l'entité selon la loi correspondante.

La définition d'une loi de transformation reste arbitraire. Tout d'abord, elle dépend du type de domaine considéré. Par exemple, pour le domaine correspondant à l'ensemble des sous-ensembles fermés d'un espace topologique (par exemple, un segment du plan, une région de points fermée du plan, ...), Kurata (2009) considère une loi de transformation d'entité correspondant à la « déformation » de l'entité de manière continue dans le domaine sans modifier sa structure topologique. Un autre exemple est celui où le domaine correspond à un espace affine réel de dimension finie, la notion de transformation d'une entité ne correspond plus alors à celle de sa déformation, mais plutôt à celle de son « déplacement » dans l'espace. Par ailleurs, au sein d'un même domaine une loi de transformation peut être définie de différentes manières : dans le cas des points d'un espace affine réel de dimension finie, une loi de transformation peut correspondre au « déplacement » d'une entité ponctuelle librement sur le plan, i.e., dans n'importe quelle direction autour de lui, ou elle peut restreindre ses possibilités de déplacement en suivant l'un des axes du repère définissant le plan.

Par exemple pour le formalisme d'Allen, Freksa (1992) identifie trois lois de transformation des intervalles dans leur domaine D_{int} :

- la première loi de transformation (dite *A-transformation*) considère un contexte où une transformation d'un intervalle correspond au déplacement de l'une de ses deux bornes vers la gauche ou vers la droite ; dans un tel contexte, on considère par exemple que le décalage d'un intervalle vers la gauche ou vers la droite s'effectue en deux temps, par un déplacement indépendant de chacune de ses deux bornes ;
- pour la deuxième loi de transformation (*B-transformation*), un intervalle peut être déplacé vers la gauche ou vers la droite en déplaçant simultanément ses deux bornes dans la même direction ;
- enfin, la troisième loi de transformation identifiée par Freksa (*C-transformation*) correspond au rétrécissement ou à l'élargissement d'un intervalle, i.e., en déplaçant chacune de ses deux bornes dans des directions opposées.

Nous proposons une définition très générale de la notion de loi de transformation qui peut s'appliquer à n'importe quel domaine D pour les entités considérées.

Définition 1 (Loi de transformation)
Une loi de transformation τ *sur* D *est un ensemble de topologies sur* D.

La topologie est le socle conceptuel des notions de continuité (et donc de transformations continues), ce qui constitue un cadre suffisamment général pour s'appliquer à un grand nombre d'applications. En effet, en considérant pour support une topologie particulière sur D, on peut représenter une transformation d'entité particulière de la loi de transformation comme un *continuum linéaire* sur D, i.e., un sous-ensemble de D muni d'une relation d'ordre totale et dense : un tel continuum linéaire peut être caractérisé par une application continue sur D. Ainsi, une topologie sur D caractérise un ensemble de continuums linéaires qui représente l'ensemble des possibilités pour une entité de D de se « déplacer » ou se « déformer » de manière ordonnée et continue dans son domaine. Enfin, une loi de transformation est formée de manière générale de plusieurs topologies, ceci permettant de représenter des situations où une entité peut se déplacer de plusieurs manières différentes. Par exemple, la loi représentant un déplacement d'un point du plan suivant l'un des deux axes générant le plan est définie par deux topologies, chacune correspondant à la topologie usuelle sur $\mathbb{R}$ que l'on associe à chaque axe.

Nous illustrons cette notion au travers des lois de transformation τ_A et τ_C, correspondant respectivement aux A-transformation et C-transformation décrites précédemment. Chacune de ces lois de transformation est formée de deux topologies sur D_{int}. Formellement :

Exemple 2 (Lois de transformation τ_A et τ_C)
- $\tau_A = \{\mathcal{T}_A, \mathcal{T}'_A\}$, *où pour tout intervalle fermé* $x \in \mathsf{D}_{int}$ *:*
  - $Vois_{\mathcal{T}_A}(x)$ *est l'ensemble des sous-ensembles de* D_{int} *qui contiennent un ouvert* $O(x, X)$ *de* $\mathcal{T}_A$*, où* $O(x, X)$ *est un sous-ensemble de* D_{int} *de la forme* $O(x, X) = \{\{x^-\} \times X \mid X$ *un intervalle ouvert contenant* $x^+\}$ *;*
  - $Vois_{\mathcal{T}'_A}(x)$ *est l'ensemble des sous-ensembles de* D_{int} *qui contiennent un ouvert* $O(x, Y)$ *de* $\mathcal{T}'_A$*, où* $O(x, Y)$ *est un sous-ensemble de* D_{int} *de la forme* $O(x, Y) = \{Y \times \{x^+\} \mid Y$ *un intervalle ouvert contenant* $x^-\}$ *;*
- $\tau_C = \{\mathcal{T}_C, \mathcal{T}'_C\}$, *où pour tout intervalle fermé* $x \in \mathsf{D}_{int}$ *:*
  - $Vois_{\mathcal{T}_C}(x)$ *est l'ensemble des sous-ensembles de* D_{int} *qui contiennent un ouvert* $O(x, X, Y)$ *de* $\mathcal{T}_C$*, où* $O(x, X, Y)$ *est un sous-ensemble de* D_{int} *non singleton de la forme* $O(x, X, Y) = \{x\} \cup \{X \times Y \mid X$ *un intervalle ouvert tel que* $X^- = x^-, Y$ *un intervalle ouvert tel que* $Y^+ = x^+\}$ *;*
  - $Vois_{\mathcal{T}'_C}(x)$ *est l'ensemble des sous-ensembles de* D_{int} *qui contiennent un ouvert* $O(x, X', Y')$ *de* $\mathcal{T}'_C$*, où* $O(x, X', Y')$ *est un sous-ensemble de* D_{int} *non singleton de la forme* $O(x, X', Y') = \{x\} \cup \{X' \times Y' \mid X'$ *un intervalle ouvert tel que* $X'^+ = x^-, Y'$ *un intervalle ouvert tel que* $Y'^- = x^+\}$.

Les figures 3 et 4 représentent schématiquement des sous-ensembles $O(x, X)$, $O(x, Y)$, $O(x, X, Y)$, $O(x, X', Y')$ de D_{int}, étant donné un intervalle de référence $x \in \mathsf{D}_{int}$. Par exemple, un voisinage d'un intervalle $x \in \mathsf{D}_{int}$ selon la topologie $\mathcal{T}_A$ est un sur-ensemble d'un ensemble d'intervalles de D_{int} de la forme $O(x, X)$.

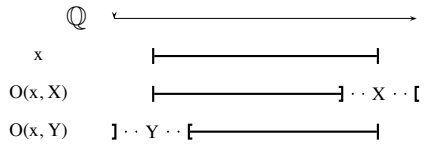


FIGURE 3 – Loi $\tau_A = \{\mathcal{T}_A, \mathcal{T}'_A\}$

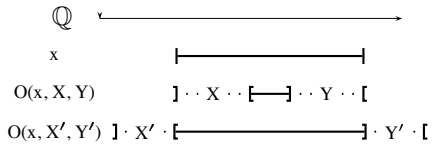


FIGURE 4 – Loi $\tau_C = \{\mathcal{T}_C, \mathcal{T}'_C\}$

3.2 Voisinage conceptuel entre relations

Étant donnée une loi de transformation sur D, on peut définir une notion de voisinage entre deux relations binaires disjointes sur D. De manière informelle, étant donnée une topologie particulière sur D, une relation binaire $R_1 \subseteq \mathsf{D} \times \mathsf{D}$ est dite voisine d'une autre relation binaire $R_2 \subseteq \mathsf{D} \times \mathsf{D}$ si l'on peut passer directement d'un couple de valeurs de R_1 à un couple de valeurs de R_2 par une transformation de la seconde valeur du couple (Freksa, 1992; Condotta *et al.*, 2008; Kurata, 2009), la transformation étant régie par la topologie correspondante. Formellement :

Définition 2 (Voisinage faible entre relations)
Soient $\mathcal{T}$ une topologie sur D *et R_1, R_2 deux relations binaires disjointes sur* D*. R_2 est dite* faiblement voisine de *R_1 pour $\mathcal{T}$ ssi $\exists (x, y) \in R_1$, $\exists W \subseteq \mathsf{D}$ tel que $y \in front_{\mathcal{T}}(W)$ et tel que ($\forall y' \in D$, si $y' \in int_{\mathcal{T}}(W)$ alors $(x, y') \in R_2$).*

Une notion plus restrictive de voisinage entre relations peut être définie : une relation $R_1 \subseteq \mathsf{D} \times \mathsf{D}$ est dite fortement voisine d'une autre relation $R_2 \subseteq \mathsf{D} \times \mathsf{D}$ si *pour tout* couple de valeurs de R_1, on peut passer directement à un couple de valeurs de R_2 par une transformation de la seconde valeur du couple. Formellement,

Définition 3 (Voisinage fort entre relations)
Soient $\mathcal{T}$ une topologie sur D*, et R_1, R_2 deux relations binaires disjointes sur* D*. R_2 est dite* fortement voisine de *R_1 pour $\mathcal{T}$ ssi $\forall (x, y) \in R_1, \exists W \subseteq \mathsf{D}$ tel que $y \in front_{\mathcal{T}}(W)$ et tel que ($\forall y' \in D$, si $y' \in int_{\mathcal{T}}(W)$, alors $(x, y') \in R_2$).*

Dans la suite de cet article, nous considérons les formalismes qualitatifs pour lesquels les notions de voisinage faible et fort entre relations coïncident sur l'ensemble des relations de base du schéma de partition associé B. Cette restriction n'est pas contraignante, puisqu'elle est vérifiée pour la plupart des formalismes qualitatifs proposés dans la littérature : c'est le cas en particulier du formalisme d'Allen, de l'algèbre des points ou des directions cardinales [2]. Par conséquent, dans la suite de l'article nous omettons le qualificatif « faible » ou « fort ».

Nous étendons maintenant la notion de voisinage entre relations à celle basée sur un ensemble de topologies, c'est-à-dire à une loi de transformation. La notion correspon-

2. La propriété qui caractérise de tels formalismes reste inconnue à ce jour, mais il reste cependant aisé de vérifier *ad hoc* si les deux notions coïncident pour un formalisme qualitatif donné.

dante définit une relation de voisinage *conceptuel* entre relations [3] :

Définition 4 (Voisinage conceptuel entre relations)
Soient τ une loi de transformation et R_1, R_2 deux relations binaires disjointes sur D. On dit que R_2 est conceptuellement voisine de R_1 pour la loi τ, noté $R_1 \xrightarrow{\tau} R_2$ ssi il existe une topologie $\mathcal{T} \in \tau$ telle que R_2 est voisine de R_1 pour $\mathcal{T}$.

Étant donné un schéma de partition B sur D et une loi de transformation τ, une relation de voisinage conceptuel entre les relations de base de B peut alors être représentée au moyen d'un graphe de voisinage conceptuel sur B, i.e., un graphe orienté dont chaque noeud représente un élément de B. Dans un tel graphe, une arête relie un nœud n_a correspondant à une relation de base $a \in \mathsf{B}$ à un autre nœud n_b correspondant à une relation de base $b \in \mathsf{B}$ si et seulement si $a \xrightarrow{\tau} b$.

Exemple 3 (Voisinage conceptuel sur B_{int} pour les lois τ_A et τ_C)
Les figures 5 et 6 illustrent respectivement les graphes de voisinage conceptuel sur B_{int} pour les lois de transformation τ_A et τ_C. Les figures 7 et 8 montrent chacune une instance de voisinage conceptuel entre deux relations de B_{int} pour les lois τ_A et τ_C. Ainsi pour la loi τ_A, on a par exemple $eq \xrightarrow{\tau_A} si$ (cf. figure 7), et pour la loi τ_C, on a par exemple $eq \xrightarrow{\tau_C} di$ (cf. figure 8).

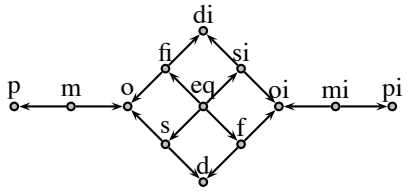


FIGURE 5 – Graphe de voisinage conceptuel sur B_{int} pour la loi τ_A.

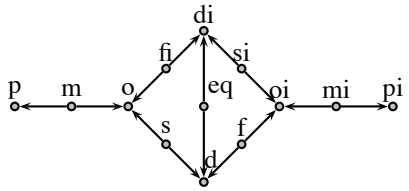


FIGURE 6 – Graphe de voisinage conceptuel sur B_{int} pour la loi τ_C.

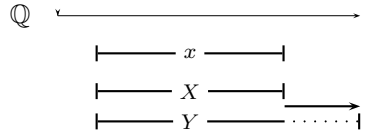


FIGURE 7 – si est conceptuellement voisine de eq pour la loi τ_A.

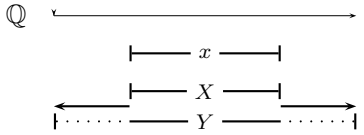


FIGURE 8 – di est conceptuellement voisine de eq pour la loi τ_C.

3. La relation de voisinage conceptuel entre relations introduite ici n'est pas symétrique. La fermeture symétrique de cette relation donne lieu à la notion de voisinage conceptuel entre relations initialement proposée dans (Condotta *et al.*, 2008).

4 Voisinage entre scénarios cohérents

Le problème du changement de croyances ou de préférences représentées au moyen de RCQ a suscité l'intérêt de plusieurs chercheurs ces dernières années, notamment celui de la fusion de RCQ (Condotta *et al.*, 2008, 2010) ou la réparation de RCQ incohérents (Dylla & Wallgrün, 2007). Afin d'élaborer le processus de fusion ou de réparation de RCQ, il est très souvent nécessaire de disposer d'une notion de *distance* entre RCQ, plus précisément d'une distance entre scénarios. Une telle distance peut par ailleurs être exploitée pour représenter une notion de « coût » pour passer d'une configuration qualitative à une autre. Considérons par exemple le scénario σ représenté sur la figure 2. Supposons que nous souhaitions optimiser l'utilisation du temps dont on dispose (T) pour la présentation de l'exposé (E) et la phase de délibération (D), sans perturber le séminariste, i.e., en conservant un planning proche de l'original. Le scénario σ' représenté sur la figure 9 caractérise le scénario « idéal ». On suppose également que les modifications des créneaux sont régies par la loi de transformation τ_C (cf. figure 4), autrement dit un créneau peut être soit élargi, soit réduit. La question qui se pose alors est la suivante *« Dans quelle mesure faut-il modifier le scénario initial σ pour obtenir le scénario idéal σ' ? »* . La définition d'une distance entre les deux scénarios nous permettrait alors de répondre à cette question.

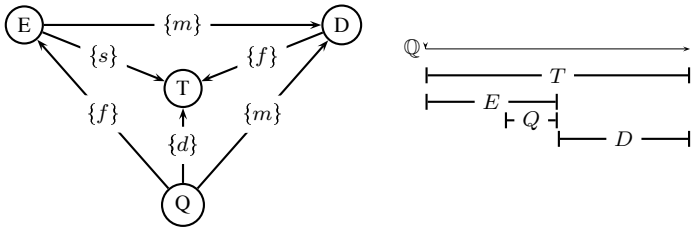


FIGURE 9 – σ', un scénario cohérent de N et α', une solution de σ'.

Condotta *et al.* (2008, 2010) ont proposé une notion de distance entre scénarios qui est définie de la manière suivante. Pour tout scénario σ,σ' sur le même formalisme qualitatif 2^{B} et sur le même ensemble de variables V la distance entre σ et σ', notée d est définie par :

$$d(\sigma,\sigma') = f\{d_{\mathsf{B}}(\sigma[i,j], \sigma'[i,j]) \mid v_i, v_j \in V, i < j\},$$

où d_{B} est une pseudo-distance[4] sur B et f une fonction d'agrégation[5] symétrique en chacun de ses arguments.

Cette définition de distance entre scénarios exploite la notion de voisinage conceptuel entre relations de base de B. En effet, d_{B} peut être définie pour toute relation de base $a, b \in \mathsf{B}$ comme étant la longueur de la chaîne la plus courte dans le graphe de voisinage

4. Une pseudo-distance d_E sur E est une application de $E \times E$ dans $\mathbb{R}^+$ qui vérifie $\forall e_1, e_2 \in E$, $d_E(e_1, e_2) = 0$ si et seulement si $e_1 = e_2$ et $d_E(e_1, e_2) = d_E(e_2, e_1)$.

5. Une fonction d'agrégation est une application qui associe à un vecteur de nombres réels positifs un nombre réel positif.

conceptuel sur B en question reliant les nœuds n_a et n_b correspondant aux relations de base a et b. Ainsi, pour le formalisme d'Allen, si l'on considère le graphe de voisinage conceptuel sur B_{int} pour la loi τ_C (cf. figure 5), on a par exemple $d_{\mathsf{B}}(o, si) = 3$. Lorsqu'une distance d_{B} sur B est définie, la fonction d'agrégation f est utilisée pour agréger les distances calculées localement au niveau des contraintes entre les scénarios, et par conséquent obtenir une distance entre les deux scénarios. Le choix pour la fonction d'agrégation f dépend du contexte. Par exemple pour $f = \sum$, les distances d_{B} sur toutes les contraintes sont sommées (Condotta *et al.*, 2008).

Bien qu'une telle distance exploite la notion de voisinage conceptuel entre relations de base, elle peut donner lieu à des résultats contre-intuitifs. Considérons de nouveau les scénarios cohérents σ et σ' représentés sur les figures 2 et 9. Intuitivement, ces deux scénarios devraient être considérés comme étant « voisins » selon la loi de transformation τ_C (cf. figure 4) puisqu'il suffit, à partir d'une solution de σ (cf. figure 2), de réduire la durée de l'intervalle représenté par la variable D pour directement obtenir une solution de σ' (cf. figure 9). Cependant, il s'avère que les scénarios diffèrent au niveau de plusieurs paires de variables ((D, E), (D, T) et (D, Q)). Ainsi, lorsque $f = \sum$, on peut vérifier dans l'exemple que $d(\sigma,\sigma') = 3$.

De manière générale, la transformation d'une entité au sein d'une solution d'un scénario entraîne souvent une modification simultanée de plusieurs contraintes de ce scénario, et c'est ici que cette distance rencontre ses limites : sa définition s'effectue en deux temps, d'abord de manière locale où une distance entre relations de base est calculée contrainte par contrainte, puis passe par une étape d'agrégation de ces valeurs locales. Par conséquent, elle dépend très fortement de la structure des RCQ et non pas des configurations qualitatives que ces derniers représentent.

Nous proposons ici une approche différente pour définir la distance entre scénarios cohérents ; celle-ci est basée sur une notion sous-jacente de voisinage entre scénarios cohérents, qui se veut être une extension naturelle du voisinage entre relations vers le voisinage entre ensembles d'instanciations du domaine. Ainsi, la définition du voisinage entre scénarios cohérents ne dépend plus de la structure de ces derniers, mais des solutions qu'ils représentent. La définition du voisinage entre deux scénarios cohérents σ,σ' définis sur un ensemble de variables V se base sur deux paramètres : une loi de transformation d'entité, et un sous-ensemble V_* de V qui définit l'ensemble des variables dont les entités représentées peuvent se transformer dans leur domaine suivant la loi correspondante, et ce, de manière *successive*. Cette définition est générique au sens où l'on peut considérer selon le contexte que la modification d'une configuration qualitative s'effectue par la transformation d'une seule entité, de plusieurs entités ou même de l'ensemble des entités représentées par V. Intuitivement, pour une loi de transformation τ et un sous-ensemble V_* de V fixé, deux scénarios σ et σ' sont V_*-voisins pour la loi τ si et seulement si l'on peut, à partir d'une solution α de σ, construire directement une solution α' de σ' en modifiant α par transformation successive des entités représentées par V_* pour la loi τ. Formellement,

Définition 5 (Voisinage entre scénarios cohérents)
Soient τ une loi de transformation sur D, B *un schéma de partition, σ,σ' deux scénarios*

cohérents sur 2^{B} *et* V, *et* $V_* \subseteq V$ *où les variables de l'ensemble* V_* *sont renommées pour former l'ensemble* $V_* = \{v^1, \ldots, v^m\}$. σ' *est* V_*-voisin de σ pour la loi τ *ssi* $\exists \alpha \in [\![\sigma]\!], \exists (W_1, \mathcal{T}_1), \ldots, (W_m, \mathcal{T}_m) \subseteq \mathsf{D} \times \tau$, $(\forall v^j \in V_*, \alpha(v^j) \in adh_{\mathcal{T}_j}(W_j))$ *et* $(\forall \alpha' \in \mathsf{D}^n, ((\forall v_i \in V \backslash V_*, \alpha'(v_i) = \alpha(v_i))$ *et* $(\forall v^j \in V_*, \alpha'(v^j) \in int_{\mathcal{T}_j}(W_j)))$ *ssi* $\alpha' \in [\![\sigma']\!])$.

La définition 5 formalise la notion de voisinage entre scénarios cohérents, mais ne fournit pas de méthode algorithmique pour déterminer si deux scénarios cohérents donnés sont voisins ou non. Le théorème suivant nous permet de pallier le problème.

Théorème 1
Soient τ *une loi de transformation sur* D, B *un schéma de partition,* σ,σ' *deux scénarios cohérents sur* 2^{B} *et* V, *et* $V_* \subseteq V$. *Soit* V' *l'ensemble* $\{v'_i \mid v_i \in V_*\}$. *On définit le* RCQ $N = N(\sigma,\sigma', V_*, \tau)$ *sur* 2^{B} *et l'ensemble de variables* $V \cup V'$ *par :*
- $\forall (v_i, v_j) \in V \times V, N[v_i, v_j] = \sigma[v_i, v_j]$;
- $\forall (v'_i, v'_j) \in V' \times V', N[v'_i, v'_j] = \sigma'[v_i, v_j]$;
- $\forall (v'_i, v_j) \in V' \times V \backslash V_*, N[v'_i, v_j] = \sigma'[v_i, v_j]$;
- $\forall (v'_i, v_j) \in V' \times V_*, i \neq j, N[v'_i, v_j] = \mathsf{B}$;
- $\forall v'_i \in V'$, $N[v'_i, v_i]$ *est défini par l'ensemble des relations de base voisines de la relation* eq *pour la loi* τ.

σ' *est* V_**-voisin de* σ *pour la loi* τ *si et seulement si* N *est cohérent.*

Déterminer si deux scénarios donnés sont voisins (pour un certain paramétrage de ce voisinage) revient à décider de la cohérence d'un RCQ. Étant donné que le problème de la cohérence d'un RCQ est un problème NP-complet (donc décidable), nous avons à disposition une méthode algorithmique afin de décider du voisinage entre deux scénarios cohérents.

Exemple 4 (Voisinage entre scénarios cohérents)
Considérons de nouveau les scénarios σ *et* σ' *(cf. figures 2 et 9).* σ *est* $\{D\}$*-voisin de* σ' *pour la loi* τ_C, *puisque d'après la définition 5, on peut à partir d'une solution* α' *de* σ', *construire directement une solution* α *de* σ *en modifiant* α' *par un « rétrécissement » de l'intervalle* D.

La figure 10 illustre le RCQ $N(\sigma,\sigma', \{D\}, \tau_C)$ *construit selon la définition donnée dans le théorème 1. Ce* RCQ *est cohérent, nous pouvons déduire du théorème 1 que les scénarios cohérents* σ, σ' *sont* $\{D\}$*-voisins pour la loi de transformation* τ_C.

Nous pouvons à présent construire une distance entre scénarios cohérents, basée sur la notion de voisinage entre scénarios cohérents que nous venons de définir et de caractériser de manière algorithmique. Pour cela, nous introduisons la notion de k-voisinage entre scénarios cohérents, pour tout entier $k > 0$. Deux scénarios cohérents sont k-voisins si et seulement si l'on peut trouver un sous-ensemble V_* de variables de taille au plus k tel que les deux scénarios soient V_*-voisins :

Définition 6 (k-voisinage entre scénarios cohérents)
Soient τ *une loi de transformation sur* D, B *un schéma de partition,* σ,σ' *deux scénarios cohérents sur* 2^{B} *et* V, *et* k *un entier positif.* σ' *est* k*-voisin de* σ *selon la loi* τ *ssi il existe* $V_* \subseteq V$, $|V_*| \leqslant k$ *tel que* σ *est* V_**-voisin de* σ' *pour la loi* τ.

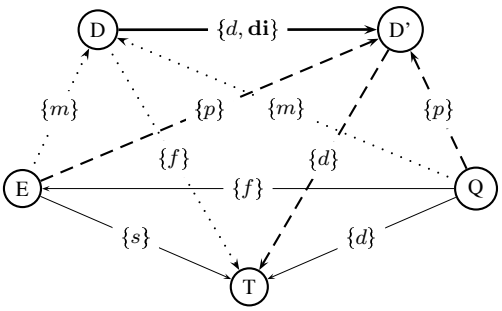


FIGURE 10 – Le RCQ $N(\sigma,\sigma', \{D\}, \tau_C)$.

Le choix du paramètre k dépend du contexte. Il correspond au nombre maximum d'entités qui ont la possibilité de se transformer successivement pour former une étape unique de transformation d'une configuration qualitative représentée par un scénario. On peut en effet remarquer de manière évidente que pour tout entier $k > 0$, le k-voisinage entre scénarios cohérents implique leur k'-voisinage, pour tout entier positif $k' \leqslant k$.

Étant donné un schéma de partition B sur D, un ensemble de variables V, une loi de transformation et un entier $k > 0$ fixé, une relation de k-voisinage entre scénarios cohérents (à l'instar d'une relation de voisinage conceptuel entre relations de base, voir section précédente) peut être représentée au moyen d'un graphe de voisinage sur l'ensemble des scénarios cohérents sur 2^{B} et V, i.e., un graphe orienté dont chaque noeud représente un scénario cohérent sur 2^{B} et V. Dans un tel graphe, une arête relie un nœud n_σ correspondant à un scénario cohérent σ à un autre nœud $n_{\sigma'}$ correspondant à un scénario cohérent σ' si et seulement si σ' est k-voisin de σ pour la loi τ. Une notion de k-distance entre scénarios cohérents peut alors être dérivée de ce graphe.

Définition 7 (k-distance entre scénarios cohérents)
Soient τ une loi de transformation sur D, B un schéma de partition, σ,σ' deux scénarios cohérents sur 2^{B} et V, et k un entier positif. La k-distance entre σ et σ' est la longueur de la chaîne la plus courte reliant les deux nœuds représentant σ et σ' dans le graphe de voisinage dérivé du k-voisinage entre scénarios cohérents correspondant.

Nous avons donc à disposition une distance entre scénarios cohérents générique et pertinente puisqu'elle est indépendante de l'aspect structurel de ces scénarios.

5 Conclusion

Dans cet article, nous avons proposé une définition générique de la notion de loi de transformation, qui peut être appliquée à n'importe quel domaine D (muni d'une topologie) pour les entités spatiales ou temporelles. Cette définition s'appuie sur la notion de continuité induite des espaces topologiques. À titre d'exemple, nous avons importé dans notre cadre deux lois de transformation particulières de l'algèbre des intervalles initialement proposées par Freksa. À partir de ces lois de transformations, nous avons défini la notion de voisinage entre relations de base d'un formalisme qualitatif. Cette

notion donne naturellement naissance à un graphe de voisinage conceptuel entre relations de base. Nous avons également proposé une nouvelle définition du voisinage entre scénarios, dont nous avons fourni une caractérisation algorithmique. Nous avons également proposé par extension une notion de distance entre scénarios cohérents, ingrédient majeur des opérateurs de changements de croyances ou de préférences décrites par des RCQ. La distance ainsi définie est plus pertinente et naturelle que les distances actuellement proposées dans la littérature.

En perspective, une étude expérimentale est prévue afin d'évaluer la complexité de calcul d'une distance entre deux scénarios cohérents donnés. Une méthode « brutale » consiste à choisir l'un des deux scénarios donnés, à générer l'ensemble de ses scénarios cohérents voisins, puis pour chacun de ces scénarios voisins obtenus à générer l'ensemble de leurs voisins, et ainsi de suite jusqu'à atteindre l'autre scénario ; la profondeur de l'arbre de recherche finalement obtenu correspond alors à la distance entre les deux scénarios initiaux. Cet algorithme paraît très coûteux, et c'est pourquoi une recherche approfondie doit être menée dans le but de permettre l'élagage de certaines branches de l'arbre de recherche lors de son développement. En outre, il serait intéressant d'étudier dans quelle mesure la notion de distance introduite dans cet article peut être adaptée d'autres formalismes de représentation d'informations. Par exemple, dans le cadre de la logique propositionnelle, de nombreux opérateurs de fusion de bases de croyances sont construits à partir d'une notion de distance entre interprétations (Konieczny *et al.*, 2004). Une distance typiquement choisie par ces opérateurs est la distance de Hamming (Dalal, 1988). L'utilisation de cette distance nécessite d'admettre une hypothèse d'indépendance entre les variables propositionnelles. Or, lorsque l'on est en présence de contraintes d'intégrité (représentées par une formule propositionnelle) qui codent la structure du problème représenté, i.e., des lois « physiques » et indéniables, l'indépendance entre les variables est remise en cause et la distance de Hamming est alors un choix inapproprié (Konieczny *et al.*, 2011). Par conséquent, à l'instar de notre approche qui consiste à construire une distance basée sur les lois du domaine et du formalisme qualitatif considéré, un travail futur sera (dans le cadre de la logique propositionnelle en particulier), d'élaborer de nouvelles distances, plus pertinentes que la distance de Hamming, et qui dépendent de ces contraintes d'intégrité.

Références

ALLEN J. F. (1981). An interval-based representation of temporal knowledge. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI'81)*, p. 221–226.

CONDOTTA J.-F., KACI S., MARQUIS P. & SCHWIND N. (2010). A syntactical approach to qualitative constraint networks merging. In *Proceedings of the 17th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR'10)*, p. 233–247.

CONDOTTA J.-F., KACI S. & SCHWIND N. (2008). A Framework for Merging Qualitative Constraints Networks. In *Proceedings of the 21st International FLAIRS Conference (FLAIRS'08)*, p. 586–591.

DALAL M. (1988). Investigations into a theory of knowledge base revision : preliminary report. In *Proceedings of the 7th National Conference on Artificial Intelligence (AAAI'88)*, p. 475–479.

DYLLA F. & WALLGRÜN J. O. (2007). Qualitative spatial reasoning with conceptual neighborhoods for agent control. *Journal of Intelligent and Robotic Systems*, **48**(1), 55–78.

EGENHOFER M. J. & AL-TAHA K. K. (1992). Reasoning about gradual changes of topological relationships. In *Spatio-Temporal Reasoning*, p. 196–219.

FREKSA C. (1992). Temporal reasoning based on semi-intervals. *Artificial Intelligence*, **54**(1), 199–227.

GOODAY J. M. & COHN A. G. (1994). Conceptual neighbourhoods in temporal and spatial reasoning. In *Proceedings of the ECAI Spatial and Temporal Reasoning Workshop*.

HIRSCH R. (1999). A finite relation algebra with undecidable network satisfaction problem. *Logic Journal of the IGPL*, **7**(4), 547–554.

KONIECZNY S., LANG J. & MARQUIS P. (2004). $\mathbf{DA}^2$ merging operators. *Artificial Intelligence*, **157**(1-2), 49–79.

KONIECZNY S., MARQUIS P. & SCHWIND N. (2011). Belief base rationalization for propositional merging. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI'11), to appear*.

KURATA Y. (2009). Semi-automated derivation of conceptual neighborhood graphs of topological relations. In *Proceedings of the 9th Conference on Spatial Information Theory (COSIT'09)*, p. 124–140.

LIGOZAT G. & RENZ J. (2004). What is a qualitative calculus ? a general framework. In *Proceedings of the 8th Pacific Rim International Conferences on Artificial Intelligence (PRICAI'04)*, p. 53–64.

RANDELL D. A., CUI Z. & COHN A. G. (1992). A spatial logic based on regions and connection. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning (KR'92)*, p. 165–176.

Modification de l'ATMS pour une prise en compte efficace de la justification d'hypothèses

Sylvain Dekoker[1,2], Amal El Fallah Seghrouchni[2], Patrick Taillibert[1]

[1] Thales Systèmes Aéroportés
2, Avenue Gay Lussac 78851 Elancourt Cédex
patrick.taillibert@fr.thalesgroup.com

[2] Laboratoire d'Informatique de Paris 6
4 place Jussieu 75005 Paris
sylvain.dekoker@lip6.fr et amal.elfallah@lip6.fr

Résumé : L'ATMS est un système de maintien de la cohérence qui calcule pour chaque fait les ensembles d'hypothèses cohérentes dont il dépend (son label). La justification d'hypothèses dans l'ATMS, bien que possible, pose un problème combinatoire. En effet, la taille du label d'une hypothèse croit linéairement en fonction du nombre d'hypothèse qui la justifie. Or l'ATMS est d'une complexité exponentielle en fonction de la taille des labels. Nous proposons une modification de la définition des labels en considérant des environnements clos pour l'implication. Ainsi le label des hypothèses contient un unique environnement qui croit linéairement en fonction du nombre d'hypothèses qui la justifie. Le calcul de la clôture d'un environnement est d'une complexité polynomiale. Nous montrerons une application de cette technique pour le raisonnement temporel.

Mots-clés : ATMS, maintenance de vérité, hypothèses, raisonnement hypothétique

1 Introduction

Les TMS (Truth Maintenance System) sont des systèmes créés pour faciliter la construction d'applications mettant en œuvre des connaissances pouvant s'avérer contradictoires.

Parmi les TMS on distinguera le Justification based Truth Maintenance System (JTMS) (Doyle, 1979) qui entretient un unique état consistant, et l'Assumption based Truth Maintenance System (ATMS) (de Kleer, 1986; de Kleer & Reiter, 1987; De Kleer, 1992) qui est multi-contexte, et entretient donc tous les états possibles en parallèle. C'est de ce dernier dont parle cet article.

Comme son nom l'indique l'ATMS utilise une approche à base d'hypothèses. Ce sont des sous-ensembles de faits, désignés par le concepteur, qui seront remis en cause lors de l'ajout de contradiction. Le rôle de l'ATMS est de calculer le label de chaque fait, c'est à dire les ensembles d'hypothèses dont il dépend.

Certaines applications nécessitent pour garantir la cohérence de faire se justifier les hypothèses entres elles (raisonnement temporel, diagnostic de cascade de pannes). L'ATMS le permet, mais cela a pour effet de faire croître le label des hypothèses ainsi justifiées. Or l'algorithme de calcul des labels est exponentiel en fonction de la taille des labels. La justification d'hypothèse pose donc rapidement un problème d'efficacité.

Après un rappel sur l'ATMS (section 1.1) nous illustrerons le problème sur un exemple motivant (section 1.2). Dans la partie 2 nous montrerons que la solution du papier d'origine (de Kleer, 1986) ne résout pas le problème (section 2.1). Nous présenterons ensuite un travail sur le raisonnement temporel qui laissait transparaitre une solution (section 2.2). Dans la partie 3 nous présenterons notre solution en commençant par la définition des environnements clos (section 3.1). Puis nous proposerons une modification de l'algorithme pour les calculer (section 3.2). Dans la dernière partie (section 4) nous monterons en quoi la modification des labels peut être utile dans le cadre du raisonnement temporel.

1.1 Assumption based Truth Maintenance System (ATMS)

L'application – aussi appelé résolveur de problème (problem solver) – utilisant un ATMS lui transmet incrémentalement les étapes de son raisonnement et en retour l'ATMS calcule les états cohérents du système.

L'application déclare à l'ATMS des *faits*, des *hypothèses* (qui sont des faits distingués) et les *inférences* qu'elle effectue.

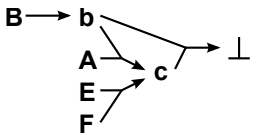
L'ATMS opère en logique des propositions (l'application peut être dans une logique différente), les faits sont des propositions, donc des termes constants (sans variables).

On rappelle que :

- Un ensemble d'hypothèses s'appelle un *environnement*.
- L'ensemble des environnements cohérent dont dépend un fait est son *label*.
- Les environnement minimaux incohérents sont appelés les *nogoods*.

Par exemple (les hypothèses sont notées avec une majuscule, et les faits "simples" avec une minuscule) :

$$\begin{array}{ccc} A \wedge b & \rightarrow & c \\ B & \rightarrow & b \\ E \wedge F & \rightarrow & c \\ b \wedge c & \rightarrow & \bot \end{array}$$



L'ATMS établira que :

- le label de b est $\{B\}$
- le label de c est $\{\{E, F\}\}$
- $\{A, B\}$ et $\{B, E, F\}$ sont incohérent.

Les ensembles de fait représentent des conjonctions, par exemple $\{A, B\}$ signifie $A \wedge B$.

Les inférences transmises à l'ATMS s'appellent des *justifications* et peuvent être de trois formes :

1. une clause de Horn : *liste de faits* $\rightarrow$ *fait*, exprime l'implication habituelle ;
2. une assertion : $\rightarrow$ *fait*, pour désigner un fait toujours vrai ;

3. une contradiction : *liste de faits* $\rightarrow\perp$, tous les faits de la liste ne peuvent être vrais simultanément.

Le rôle de l'ATMS est de calculer le label de chaque fait. Il garantit pour les labels les quatre propriétés qui suivent. Soit f un fait (hypothèse ou non), J l'ensemble des justifications, H l'ensemble des hypothèses. Un label est :

1. Fondé : le fait peut être déduit de chaque environnement du label à l'aide de l'ensemble des justifications. $\forall E \in Lab(f), J \vdash E \rightarrow f$
2. Cohérent : chacun de ses environnements est cohérent (n'est pas un sur-ensemble d'un nogood). $\forall E \in Lab(f), \neg(E, J \vdash \perp)$
3. Complet : tous les environnements susceptibles de produire ce fait sont sur-ensemble d'un environnement du label. $\forall E \subseteq H, \text{ tel que } J \vdash E \rightarrow f, \exists E' \in Lab(f), E' \subseteq E$
4. Minimal : aucun de ses environnements n'est sur-ensemble d'un autre. $\forall E \in Lab(f), \neg(\exists E' \in Lab(f), E' \subset E)$

1.2 Enoncé du problème

La justification d'hypothèse est possible. La cohérence globale est maintenue, et les labels calculés conservent les propriétés exposées. Toutefois, un problème de performance réside. Soit une cascade d'hypothèses :
$H_1 \rightarrow H_2 \rightarrow ... \rightarrow H_n$.

Les labels sont :
$Lab(H_1) = \{\{H_1\}\}$
$Lab(H_2) = \{\{H_1\}, \{H_2\}\}$
...
$Lab(H_n) = \{\{H_1\}, \{H_2\}...\{H_n\}\}$

Chaque nouvelle justification à gauche de la cascade, rajoute un environnement dans les labels des hypothèses de droite. La cardinalité du label d'une hypothèse croit linéairement en fonction du nombre d'hypothèses qui la justifient (y compris transitivement).

Or l'ATMS est exponentiel en fonction de la taille des labels.

2 Etat de l'art

2.1 Assumed node

Dans le papier de référence, De Kleer déconseille de justifier directement des hypothèses et propose les "assumed nodes". Cette technique permet de dissocier la donnée supposée, de l'action de supposer. Un couple (hypothèse, assumed-node) $A \rightarrow a$ représente une donnée rétractable. La donnée portée par a peut être retiré en contredisant A, puis rajoutée à nouveau par une hypothèse A' justifiant a. Cela permet d'éviter de faire se justifier les hypothèses. Sur notre exemple on aurait :
$H_1 \rightarrow h_1$, $H_2 \rightarrow h_2$, ..., $H_n \rightarrow h_n$,
$h_1 \rightarrow h_2 \rightarrow ... \rightarrow h_n$.

Les labels sont :
$Lab(h_1) = \{\{H_1\}\}$
$Lab(h_2) = \{\{H_1\}, \{H_2\}\}$
...
$Lab(h_n) = \{\{H_1\}, \{H_2\}...\{H_n\}\}$

Le problème combinatoire n'est pas résolu puisque les labels sont les mêmes.

2.2 HEART

How to Enhance Assumption based Reasoning with Time (HEART) (Joubel & Raiman, 1990) est une généralisation de l'ATMS pour des données temporelles. Là où l'ATMS classique manipule des faits, HEART manipule des épisodes. Un épisode est un couple (dat, int) qui représente un intervalle de temps Int durant lequel la donnée dat est vraie.

Un environnement est un ensemble d'épisodes. HEART adapte la notion de minimalité en définissant une relation d'ordre permettant de comparer de tels environnements. Les environnements non-minimaux sont redondants et éliminés.

HEART maintient des environnements dont les épisodes sont maximaux en fusionnant les épisodes, portant sur la même donnée, qui se recouvrent, se superposent ou se touchent.

Les nogoods sont des environnements pour l'épisode faux (noté $\perp$), leurs épisodes sont donc maximaux.

Comme Joubel & Raiman (1990) n'évoquent pas la justification directe d'hypothèses, nous utiliserons des assumed nodes. Soit $H_1 = (d, [0, 3[)$ et $H_2 = (d, [1, 2[)$ deux hypothèses, h_1 et h_2 deux épisodes quelconque, avec $H_1 \rightarrow h_1$, $H_2 \rightarrow h_2$, et $h_1 \rightarrow h_2$. En utilisant un ATMS classique on aurait :
$Lab(h_1) = \{\{H_1\}\}$
$Lab(h_2) = \{\{H_1\}, \{H_2\}\}$

Mais dans HEART H_1 porte sur la même donnée que H_2 mais sur un épisode plus grand ($[1, 2[\subseteq [0, 3[$) ; $\{H_1\}$ est donc redondant car non-minimal. HEART simplifie le label de h_2 pour ne conserver que les environnements minimaux :
$Lab(h_2) = \{\{H_2\}\}$

Cette manière de faire résout le problème combinatoire puisque les labels des hypothèses justifiées n'augmentent plus (même si les tests de comparaison d'environnements sont plus coûteux que de simples tests d'inclusion d'ensembles).

Cependant, la notion de redondance qui permet de simplifier les labels, ne s'applique pas forcément à d'autres applications que le raisonnement temporel. De plus les nogoods calculés ne sont pas minimaux par rapport à ceux calculés par l'ATMS classique puisque HEART fusionne les épisodes des environnements de $\perp$ afin de les rendre maximaux. C'est pourquoi nous avons cherché une méthode plus générale pour résoudre notre problème.

3 Nouvelle définition des labels

3.1 Environnements clos pour l'implication

Prenons l'exemple minimal : $H_1 \rightarrow H_2$. Peut-on parler de H_1 indépendamment de H_2 ? L'idée de base est que H_2 est nécessairement vraie si H_1 est vrai. En effet, la seule assignation pour $\{H_1, H_2\}$ où H_1 est vraie est : $\{H_1 \leftarrow true, H_2 \leftarrow true\}$

Nous changeons la définition d'un label pour que chacun de ses environnements contiennent toutes les hypothèses nécessairement vraies en même temps que lui. C'est la clôture d'un environnement pour l'implication : toute hypothèse impliquée par un environnement appartient à cet environnement. Nous parlerons d'environnements clos pour désigner ces nouveaux environnements. Soit H l'ensemble des hypothèses, J l'ensemble des justifications et f un fait :

Définition 1
$\forall E \in Lab(f), E$ est clos si et seulement si $: \forall h \in H$ tel que $E, J \vdash h$ alors $h \in E$

On dira qu'un label est clos si et seulement si tous ses environnements le sont.

Définition 2
On dira que $\widetilde{E}$ est la clôture minimale de E si et seulement si $\widetilde{E}$ est clos et $E \subseteq \widetilde{E}$ et $\neg\exists E'$ clos tel que $E \subseteq E' \subset \widetilde{E}$.

La notation $\widetilde{}$ sera utilisée dans le reste de l'article pour désigner la clôture minimale d'un environnement.

Sur l'exemple minimal on avait :
$Lab(H_1) = \{\{H_1\}\}$
$Lab(H_2) = \{\{H_1\}, \{H_2\}\}$

Avec la nouvelle définition $\{H_1\}$ n'est plus un environnement clos et doit être remplacé par $\{H_1, H_2\}$:
$Lab(H_1) = \{\{H_1, H_2\}\}$
$Lab(H_2) = \{\{H_1, H_2\}, \{H_2\}\}$

Or $\{H_2\} \subseteq \{H_1, H_2\}$ donc pour respecter la propriété de minimalité des labels :
$Lab(H_1) = \{\{H_1, H_2\}\}$
$Lab(H_2) = \{\{H_2\}\}$

Le label d'un fait dont les environnements sont clos est complet au sens ou il contient tous les environnements clos susceptibles de produire ce fait. La troisième propriété est donc modifiée ainsi :

Définition 3
Un label est complet *si et seulement si tous les environnements clos susceptibles de produire ce fait sont sur-ensembles d'un environnement du label ;* $\forall \widetilde{E} \subseteq H$ tel que $J \vdash \widetilde{E} \rightarrow f, \exists E' \in Lab(f), E' \subseteq \widetilde{E}$.

C'est pourquoi dans l'exemple, $\{H_1\}$ est fondé, cohérent et minimal (il n'est pas un sur-ensemble d'un environnement du label) mais il n'est pas clos, $\{H_1, H_2\}$ est fondé,

cohérent et clos mais il n'est pas minimal. Le seul environnement fondé, cohérent, clos et minimal est $\{H_2\}$.

Cette nouvelle définition des labels respecte les quatre propriétés classiques (dont une est légèrement modifiée) auxquelles est ajoutée la propriété de clôture.

Définition 4
Un label est fondé, cohérent, clos, complet (au sens de la définition 3) et minimal.

Deux remarques :
- les labels des hypothèses contiennent au plus un seul environnement,
- les environnements des labels sont toujours les implicants du fait, par contre ils ne sont plus premiers (puisque les implicants premiers par définition sont minimaux).

En appliquant notre nouvelle définition à l'exemple du 1.2 les labels sont les suivants :
$Lab(H_1) = \{\{H_1, H_2, ...H_n\}\}$
$Lab(H_2) = \{\{H_2, ...H_n\}\}$
...
$Lab(H_n) = \{\{H_n\}\}$

Autrement dit lorsqu'une hypothèse en justifie une autre, au lieu d'augmenter la taille des labels on augmentera la taille de l'environnement qu'elle contient. L'ATMS étant d'une complexité exponentielle en fonction de la taille des labels et seulement logarithmique en fonction de la taille des environnements, on comprend bien l'intérêt d'une telle modification.

3.2 Mise en œuvre

Les exemples que nous avons vus jusqu'à présent font justifier une hypothèse que par une seule autre hypothèse. Ce cas simplifié laisse penser que le calcul de la clôture d'un label est immédiat, et ne nécessite qu'une petite modification de l'algorithme. Dans le cas général les hypothèses peuvent aussi être justifiées par des conjonctions. Et alors les environnements non clos peuvent apparaître bien après la prise en compte de la justification qui les rend non clos. Un des piliers de l'ATMS est son fonctionnement incrémental : on doit pouvoir prendre en compte une nouvelle justification sans avoir à analyser l'intégralité des justifications déjà présentes. Nous proposons d'utiliser une table des environnements non-clos qui n'apparaissent pas forcément dès la prise en compte de la justification d'une hypothèse. Prenons l'exemple suivant (figure 1), les H_i sont des hypothèses, les autres lettres des faits :

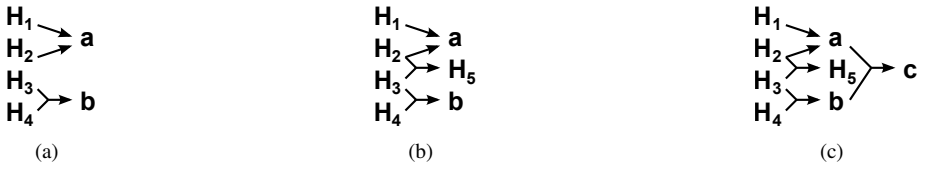


FIGURE 1 – Hypothèse justifiée par une conjonction

On a pour le 1(a) :
$Lab(a) = \{\{H_1\}, \{H_2\}\}$
$Lab(b) = \{\{H_3, H_4\}\}$

Puis (figure 1(b)) rajoutons-y la justification $H_2 \wedge H_3 \rightarrow H_5$, aucun label n'a besoin d'être modifié pour l'instant puisque l'environnement $\{H_2, H_3\}$ n'apparaît nulle part. Par contre il convient de noter que cet environnement n'est pas clos et que lorsqu'il apparaîtra il faudra le remplacer par $\{H_2, H_3, H_5\}$.

Enfin (figure 1(c)) rajoutons maintenant la justification $a \wedge b \rightarrow c$ le label de c devrait être $Lab(c) = \{\{H_1, H_3, H_4\}, \{H_2, H_3, H_4\}\}$, mais comme $\{H_2, H_3\}$ n'est pas clos on a $Lab(c) = \{\{H_1, H_3, H_4\}, \{H_2, H_3, H_4, H_5\}\}$.

3.2.1 Définition et calcul de la table de clôture

Pour pouvoir calculer la clôture de chaque environnement il faut noter les environnement non clos trouvés au fur et à mesure. Nous allons construire une table de clôture C composés de couples d'environnement $(E, \widetilde{E})$ où $\widetilde{E}$ est la clôture minimale de E. C sera complète si elle nous permet de calculer la clôture de tout environnement, et cohérente si elle ne contient aucun environnements incohérent.

Afin de stocker un minimum d'information, C ne contiendra que les éléments minimaux. Intuitivement un élément de C est plus petit qu'un autre s'il est inclu dans ce dernier et permet de calculer les mêmes clôtures. Par exemple : $(\{a\}, \{a, b\}) < (\{a, c\}, \{a, b, c\})$ car $\{a\} \subset \{a, c\}$ et $\{a, b\} \subset \{a, b, c\}$ et $\{a, b\} \cup \{a, c\} = \{a, b, c\}$. Soit la relation d'ordre suivante :

Définition 5
$<$ est une relation d'ordre partielle stricte sur les éléments de C définie par : $(E, \widetilde{E}) < (F, \widetilde{F})$ si et seulement si $E \subset F$ et $\widetilde{E} \cup F = \widetilde{F}$.

$<$ hérite de l'antisymétrie, et de l'anti réflexivité de $\subset$. Pour la transitivité si $(E, \widetilde{E}) < (F, \widetilde{F})$ et $(F, \widetilde{F}) < (G, \widetilde{G})$ alors $\widetilde{E} \cup F = \widetilde{F}$, $\widetilde{F} \cup G = \widetilde{G}$ d'où $\widetilde{E} \cup F \cup G = \widetilde{G}$ or $F \subset G$ donc $\widetilde{E} \cup G = \widetilde{G}$ d'où $(E, \widetilde{E}) < (G, \widetilde{G})$.

Définition 6
Soit C une table de clôture :

1. *C est fondée si : $\forall (E, \widetilde{E}) \in C, \widetilde{E}$ est la clôture minimale de E (définition 2) ;*
2. *C est cohérente si : $\forall (E, \widetilde{E}) \in C, \neg(\widetilde{E} \vdash \bot)$;*
3. *C est complète si :*
   $\forall E \subseteq H, \Big[E \text{ est clos } \Leftrightarrow \Big(\forall (F, \widetilde{F}) \in C, F \subseteq E \Rightarrow \widetilde{F} \subseteq E\Big)\Big]$;
4. *C est minimale si : $\forall c \in C, \neg(\exists c' \in C, c' < c)$.*

Il faut garantir ces quartre propriétés. Pour cela, lorsqu'une justification conclut sur une hypothèse, $f_1 \wedge ... \wedge f_n \rightarrow H$ avec $Lab(H) = \{E\}$ (pour le cas où $Lab(H) = \emptyset$ voir section 3.2.3), il faut mettre à jour la table de clôture, afin de s'assurer qu'elle respecte toujours les quatre propriétés. Soit L l'ensemble des combinaisons des labels des f_i, et soit $C' = \{(F, E \cup F) | F \in L\}$ une table intermédiaire. Il faut :

- s'assurer de la propriété de fondement (c'est à dire vérifier que les éléments de C sont bien des couples de clôture) : $\forall(E, E') \in C$ il faut calculer $\widetilde{E}$ qui vérifie $E' \subseteq \widetilde{E}$ et $\forall(F, F') \in C', F \subseteq \widetilde{E} \Rightarrow F' \subseteq \widetilde{E}$. Si $E' \neq \widetilde{E}$ alors (E, E') est retiré de C et $(E, \widetilde{E})$ y est ajouté.
- verifier la complétude : $\forall(F, F') \in C', \exists(E, \widetilde{E}) \in C, E \subseteq F$, si ce n'est pas le cas alors (F, F') est ajouté dans C,
- vérifier la cohérence des environnements ajoutés, c'est-à-dire retirer les $(E, \widetilde{E})$ de C dont $\widetilde{E}$ est sur-ensemble d'un nogood,
- et retirer de C les éléments non minimaux : $\forall c \in C$ si $\exists c' \in C$ tel que $c' < c$ alors c est retiré de C.

La table de clôture est ainsi obtenue incrémentalement. Elle ne dépend pas de l'ordre d'application des justifications.

3.2.2 Calcul de la clôture d'un environnement

Avec C nous sommes maintenant en mesure de calculer la clôture de chaque environnement. Nous définissons un opérateur de clôture noté $clore$ (algorithme 1) qui calcule la clôture d'un environnement E pour une table de clôture C.

Algorithme 1: $clore(E, C)$ — E un environnement, C une table de clôture

si $\exists F$ *tel que* $(F, \widetilde{F}) \in C$ *et* $F \subseteq E$ **alors retourner** $clore(E \cup \widetilde{F}, C - \{(F, \widetilde{F})\})$
sinon retourner E

Cet opération est au pire cas de $n(n+1)/2$ où n est le cardinal de C. Dans la pratique ce pire cas n'est atteint que sur des instances pathologiques telle que $H_1 \wedge H_2 \rightarrow H_3, H_3 \wedge H_4 \rightarrow H_5, ...H_{n-2} \wedge H_{n-1} \rightarrow H_n$ pour le label de $\{H_1, H_2, H_4, H_6, ..., H_{n-1}\}$.

Cet opérateur est appelé lors du calcul du label des faits "simples" afin de calculer la clôture de chaque environnement de chaque label. Avant de rajouter un environnement E dans un label il faut d'abord le clore en calculant $clore(E, C)$. Si l'environnement trouvé n'est pas sur-ensemble d'un environnement du label, alors on l'y rajoute.

Lorsque C est modifiée (lorsque des éléments y sont rajoutés) il faut vérifier que les environnements des labels sont bien clos en appliquant l'opérateur $clore/2$ avec les éléments nouvellement ajoutés à C pour le deuxième argument.

La complexité rajoutée au calcul d'un fait "simple" est en $O(n^2)$ en fonction de la taille de C dans le pire cas. La taille de C est au maximum le nombre de justifications concluant sur une hypothèse (y compris transitivement).

3.2.3 Calcul des nogoods

La base des nogoods doit toujours contenir les environnements minimaux, par exemple pour utiliser les nogoods pour un calcul de candidats. Or les labels contiennent désormais des environnements clos. C'est pourquoi nous devons retrouver les environnements minimaux qui engendrent ces environnements clos avant de les rajouter aux

nogoods. Pour cela nous allons utiliser la table de clôture "à l'envers" et retirer toutes les hypothèses qui avaient été ajoutées par l'opération de clôture.

Lorsque l'algorithme classique trouve un nogood $\widetilde{N}$ on calculera :

$$N = \widetilde{N} - \bigcup_{(E,\widetilde{E}) \in C, \widetilde{E} \subseteq \widetilde{N}} (\widetilde{E} - E) \tag{1}$$

C'est N qui sera ajouté aux nogoods. De plus pour tous les $(E, \widetilde{E}) \in C$ si $N \subseteq \widetilde{E}$ alors le couple est retiré de C et E est ajouté aux nogoods.

3.3 Expérimentation

Nous avons repris l'exemple initial (section 1.2) en construisant une cascade d'hypothèse. Les justifications sont transmises dans l'ordre : $H_1 \rightarrow H_2$, $H_2 \rightarrow H_3$, ..., $H_{n-1} \rightarrow H_n$. Après implémentation, nous obtenons les résultats suivants, la première ligne est le nombre d'hypothèse de la cascade, les deux suivantes donnent les temps d'exécution en ms pour déclarer les hypothèses, transmettre les justifications, et faire calculer à l'ATMS les labels de chaque hypothèse :

n	200	400	600	800	1000	1200	1400
ATMS classique	461	3575	12549	30945	62639	111732	183616
ATMS "clos"	240	390	881	1563	2663	3756	5066

On voit bien l'explosion combinatoire dans le cas classique, tandis que notre proposition évolue bien plus raisonnablement.

4 Application : les variables estampillées

Dans le cadre d'un travail sur un modèle de plan pour des agents cognitifs, nous avons appliqué cette technique à un type de donnée à évolution discrète dans le temps appelé *variable estampillée*. Le changement de valeur est instantané et ponctuel. L'information est potentiellement incomplète, parvient ponctuellement et éventuellement en retard. Et nous autorisons d'y accéder à tout instant.

Il y a de nombreuses sources d'informations qu'on ne peut pas connaître en temps réel : celles qui doivent transiter par des intermédiaires, qui arrivent en différé ou à des fréquences variables. L'exploitation de ces sources est difficile car une information sur leur changement peut nous parvenir a posteriori (disons à t_4) concernent le passé (à t_2) entre l'instant t_1 de la dernière valeur connue et l'instant t_3 où l'on a utilisé cette donnée ($t_1 < t_2 < t_3 < t_4$), voir la figure 2

Nous utilisons des hypothèses de persistance signifiant l'absence de changement de valeur. Sur des intervalles de temps passé elles permettent de se prémunir contre l'arrivée d'information tardive. Dans l'exemple l'utilisation de l'information à t_3 se fait sous l'hypothèse que la donnée reste constante entre t_1 et t_3. L'arrivée tardive d'une nouvelle valeur concernant t_2 engendrera une contradiction entre les deux valeurs et l'hypothèse de persistance. Si les informations ne nous parviennent pas de la même source alors soit

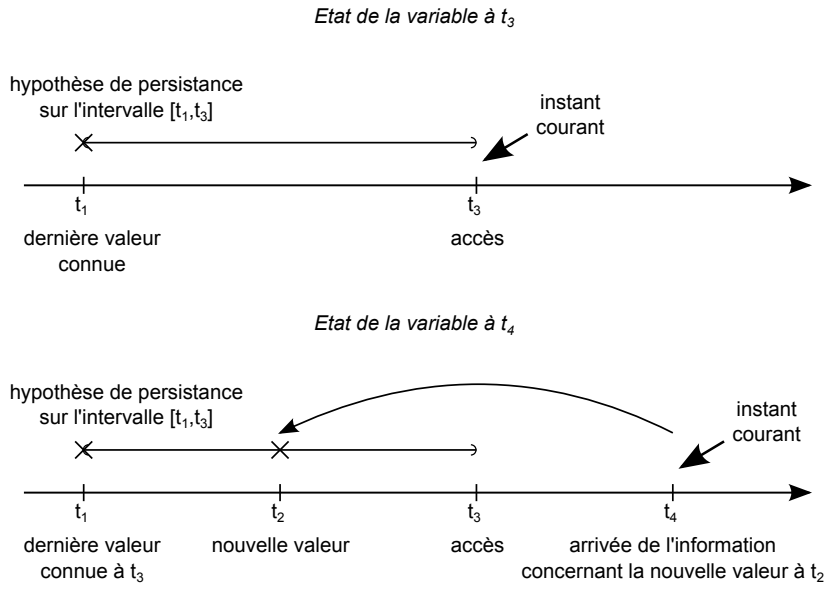


FIGURE 2 – Accès à une variable estampillée

l'une des deux est fausse, soit elles sont toutes deux justes et alors la valeur a effectivement changé. Dans ce dernier cas l'hypothèse de persistance deviendra un nogood, et tout ce qui avait était déduit à partir d'elle sera injustifié (l'agent cessera d'y croire).

Lors d'accès successifs à une variable estampillée on retombe dans le cas d'hypothèses en cascade. Prenons par exemple la figure 3. Soit l'attribut défini par (a, v, t_0), et trois accès à t_1, t_2 et t_3. Il y aura trois hypothèses de persistance $pers(a, [t_0, t_1])$ notée p_1, $pers(a, [t_0, t_2])$ notée p_2 et $pers(a, [t_0, t_3])$ notée p_3. Il y aura les justifications $p_3 \rightarrow p_2, p_3 \rightarrow p_1$ et $p_2 \rightarrow p_1$. Avoir trois hypothèses de persistance différentes permet de d'obtenir des environnements différents pour chacun des accès et tout ce qui en sera déduit. L'utilisation d'environnements clos permet de maîtriser la combinatoire engendrée.

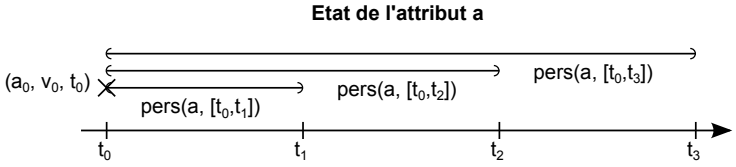


FIGURE 3 – Accès successifs à une variable estampillée

L'utilisation de HEART sur cet exemple ne permet pas d'utiliser les nogoods pour un calcul de candidats puisque les épisodes des nogoods sont fusionnés afin d'être

maximaux. Par ailleurs, notre approche s'applique à des raisonnements symboliques en dehors du raisonnement temporel.

5 Conclusion

Dans cet article nous avons proposé une modification de la définition des labels de l'ATMS. Dans cette nouvelle définition les environnements des labels sont clos pour l'implication, c'est à dire qu'ils contiennent toutes les hypothèses qu'ils impliquent. Cette modification permet de maîtriser la croissance des labels des hypothèses si des inférences concluent à des hypothèses. Leurs labels ne contiennent plus qu'un unique environnement, alors qu'il augmentait linéairement en fonction du nombre d'hypothèses qui la justifiait avec l'ATMS classique – qui est exponentiel en fonction de la taille des labels. Le calcul de la clôture des environnements est polynomial (dans le pire cas) en fonction du nombre de justifications concluant sur une hypothèse (y compris transitivement). Le gain en complexité, pour les instances de problème ayant besoin de justifier les hypothèses entre-elles, est considérable.

Nous avons appliqué l'utilisation de cette modification sur une structure de donnée appliquée au raisonnement temporel.

6 Remerciement

Nous tenons à remercier Philippe Dague pour son avis et ses remarques pendant l'élaboration de l'idée qui a conduit à cet article.

Références

DE KLEER J. (1986). An assumption-based TMS. *Artificial intelligence*, **28**(2), 127–162.

DE KLEER J. (1992). An improved incremental algorithm for generating prime implicates. In *Proceedings of the National Conference on Artificial Intelligence*, p. 780–780 : JOHN WILEY & SONS LTD.

DE KLEER J. & REITER R. (1987). Foundations for assumption-based truth maintenance systems : Preliminary report. In *Proc. American Assoc. for Artificial Intelligence Nat. Conf*, p. 183–188.

DOYLE J. (1979). A truth maintenance system. *Artificial Intelligence*, **12**(3), 231–272.

JOUBEL C. & RAIMAN O. (1990). How time changes assumptions. In *Proc. ECAI-90 : Ninth European Conference on Artificial Intelligence*, p. 378–383.

Utilisation de la logique des hypothèses pour la modélisation des voies de signalisation dans la cellule

Andrei Doncescu [1] - Pierre Siegel [2]
[1] LAAS Toulouse - [2] SNC Toulon

Résumé : La description formelle des interactions dans la cellule est un champ d'investigation très adapté à l'Intelligence Artificielle. On est, en particulier, dans un cadre de représentation de connaissances incomplètes et d'abduction. Les problèmes posés par la complexité algorithmique sont également très présents. Ce papier présente une approche basée sur une logique non-monotone, la logique des hypothèses. Cette logique permet dans un seul formalisme de traiter les questions de raisonnement révisable (défauts) et d'abduction. Des pistes pour la mise en œuvre algorithmique de cette représentation, basées sur la notion de champ de production et d'algorithme de production de cette représentation sont également données.

1 Introduction.

1.1 Biologie, mathématique et informatique.

Aujourd'hui des experts de plus en plus spécialisés ont besoin de la mise à l'échelle de leurs connaissances à partir des données accessibles pour pouvoir découvrir des nouvelles sources de savoir qui se veulent valides et utiles pour apprendre ces systèmes. Dans ce cadre, durant ces dernières décennies, la biologie a connu un développement prolifique dans toutes ses facettes. De nouveaux champs d'applications et d'études comme les biotechnologies, les nanotechnologies ou la bioinformatique ont immergés et prennent une place importante dans le contexte de recherche actuel ayant comme but l'identification, la compréhension et la quantification des phénomènes biologiques vus à l'intérieur du système biologique. La biologie fournit alors de nouveaux défis et champs d'études aux mathématiques, à l'automatique, à l'informatique et aux nouvelles technologies. L'Intelligence Artificielle a ici largement sa place.

Les systèmes biologiques changent nos méthodes d'investigation. Ceci est du à leurs capacités d'avoir une vie autonome par des phénomènes de régulation, de se reproduire, de communiquer. L'analyse de ces systèmes constitue une multitude de données variées par la diversité des modèles biologiques étudiés, des conditions d'analyse, des niveaux d'observation (macroscopique ou microscopique ou moléculaire), des outils d'investigation mis en oeuvre. Cependant, malgré la quantité et la qualité des données générées, de nombreuses questions restent en suspens et l'aspect discret des informations collectées ne permet pas à ce jour d'établir une analyse systémique des phénomènes biologiques.

Dans les tentatives de description du comportement des systèmes vivants, quand la modélisation déductive n'a pas abouti, la démarche par le raisonnement qualitatif basé sur la fonction des molécules a montré ses limites. Si l'on sait, par homologie, attribuer des propriétés à un élément d'un système vivant à partir de sa structure, actuellement, il semble impossible de les déduire ni qualitativement, ni quantitativement. De la même manière, si les propriétés sont connues, nous ne pouvons, clairement, en déduire leur fonction dans la cellule vivante, et à partir des caractéristiques des cellules vivantes, calculer leur comportement dans un environnement déterminé. En général, cette démarche déductive de description comportementale échoue du fait que la fonction d'un élément constitutif du

système vivant dépend du fonctionnement simultané des autres éléments. Le problème récurrent est que, par le calcul ou autrement, les propriétés fonctionnelles de la cellule ne peuvent être déduites des propriétés de ses seuls composants. Dans ce contexte, avec leur capacité à décrire la complexité, les outils mathématiques offrent une perspective d'analyser ces éléments structurels du monde vivant pour proposer des fonctionnalités prenant en compte leurs interactions dynamiques itératives fortement non linéaires.

Diverses approches de cette description du comportement de la cellule sont actuellement développées par la communauté scientifique internationale [5]. La « cellule virtuelle » regroupe un ensemble de logiciels permettant de décrire l'organisation métabolique d'un organisme, d'estimer la distribution de la matière dans son fonctionnement et d'extrapoler ses potentialités de production. L'«E-cell » permet de quantifier l'activité cellulaire à partir de cinétiques parfaitement caractérisées expérimentalement sur le plan catalytique et quantitatif.

La description principalement cinétique qui oblitère l'analyse des mécanismes réels ou possibles pouvant être mis en jeu. Une autre démarche consiste à rechercher une description précise d'une cellule vivante (ou d'une partie) basée sur des mécanismes et des valeurs paramétriques expérimentalement déterminées, à savoir des données excluant toutes valeurs estimées sur des modèles partiels [14].

1.2 La cellule vue du côté de l'Intelligence Artificielle.

Du point de vue de l'Intelligence Artificielle les cellules sont des sources d'informations qui intègrent une myriade de signaux intra et extra cellulaire ayant comme ultime but des sorties optimales. La maladie et le cancer en particulier peuvent être vus comme une pathologie de l'altération dans les réseaux de signalisation de la cellule. L'étude des évènements de signalisation apparaît comme une clef pour les recherches, biologique, pharmacologique et médicale. La propagation de ces types de signaux implique un changement du comportement des protéines sur trois niveaux : régulation de l'activité, interaction et expression. Les trois niveaux sont synchronisés dans une forte dynamique qui conduit aux changements de l'activité des protéines. Depuis une dizaine d'années les réseaux de signalisation ont été étudiés en utilisant des méthodes analytiques basées sur la reconnaissance des protéines par des anticorps spécifiques. En parallèle les puces à ADN (microarrays) sont largement utilisées pour étudier la co-expression de gènes candidats permettant d'expliquer l'étiologie de certaines maladies, dont le cancer. Elles apportent de nombreuses données à la modélisation des graphes d'interactions des gènes entre eux, dont les propriétés mathématiques : connectivité, connexité, présences de boucles positives et négatives sont liées aux propriétés asymptotiques des réseaux de régulation génétique.

En se plaçant du côté logique et informatique, l'étude des réseaux de gênes pose des problèmes identifiés et étudiés en Intelligence Artificielle depuis une trentaine d'année. On est ici est dans le cadre du raisonnement à partir d'informations incomplètes, incertaines, révisables, contradictoires et multi sources. En effet la description logique ou mathématique des voies de signalisation n'est pas complète : les expériences biologiques donnent un certain nombre d'interactions entre protéines mais certainement pas toutes. D'autre part les conditions et la difficulté des expériences font que les données ne sont pas toujours précises. Certaines données peuvent être même erronées et doivent être corrigées ou révisées par la suite. Enfin les informations viennent de sources et d'expériences différentes et peuvent être

contradictoires. A partir de ce constat, général pour la plupart des activités humaines réelles, il faut pourtant raisonner et prendre des décisions. C'est le but de toutes les logiques de l'incertitude et plus particulièrement des logiques non monotones que nous utiliserons dans cette étude.

Une autre problématique de l'IA, fondamentale pour ce cadre, est de compléter in-silico les réseaux de gênes. Le réseau de gêne est incomplet et les expériences biologiques sont faites pour essayer de le compléter. Mais ces expériences sont longues et couteuses. Il faut donc savoir où une molécule (un futur médicament) à une chance d'agir de manière efficace. On est ici dans cadre bien connu du raisonnement abductif. En simplifiant, l'abduction consiste à trouver le/les ensemble "minimaux" d'informations, qu'il faut ajouter à un ensemble F de faits connus, pour déduire un résultat R que l'on veut prouver. Par exemple, ici F peut être le réseau de gênes, R l'inhibition d'une protéine et X des interactions entre gênes non connues. Une telle approche permet donc de cibler les expériences biologiques à faire, et donc de gagner du temps en éliminant les expériences qui ont peu de chance de donner des résultats utilisables

Les deux principales questions qui se posent vont être de représenter formellement ces interactions et d'essayer de les compléter in silico. On est donc bien dans les cadres de la représentation de connaissances incomplètes, de la causalité, du raisonnement incertain et de l'abduction. Et bien entendu il faut d'obtenir des algorithmes d'une complexité algorithmique raisonnable dans la pratique. La cerise sur le gâteau est que tout ceci doit être compréhensible par des biologistes.

Le but est donc d'exprimer les connaissances de biologistes concernant les réseaux de signalisation dans un langage informatique sans ambigüité. Ce langage doit permettre de prendre en compte l'incertain et de donner aussi un sens précis aux formules associées à ce type de connaissances. Ces formules seront utilisées dans un système de démonstration automatique efficace. Ce système permettra, d'une part de déduire des faits à partir des données (déduction), et d'autre part de découvrir d'autres connaissances ou d'expliquer des phénomènes par analogie (raisonnement inductif et abductif), ce qui ne peut pas se faire par des méthodes analytiques du fait de la grande masse de données et de leur complexité. L'objectif étant de développer un système permettant :

1- La modélisation des règles de comportement d'un gène, d'une cellule.
2- Le raisonnement avec des informations incertaines, incomplètes et révisables.
3 - La découverte des connaissances manquantes qui expliquent certains phénomènes biologiques
4- Le choix entre différentes propositions quand plusieurs actions ou solutions sont proposées pour la même situation.
5- L'addition de nouvelles règles de comportement sans remettre en question les règles précédentes.
6- Le raisonnement avec des règles générales sans avoir à compiler toutes les possibilités.

2. Interactions dans la cellule.

2.2.1 Un petit exemple.

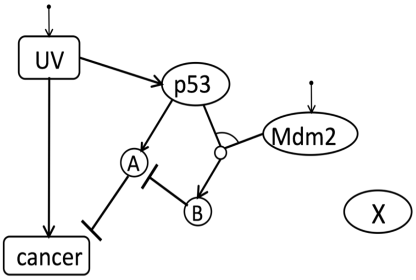


Figure 1

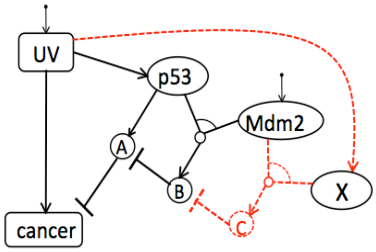


Figure 2

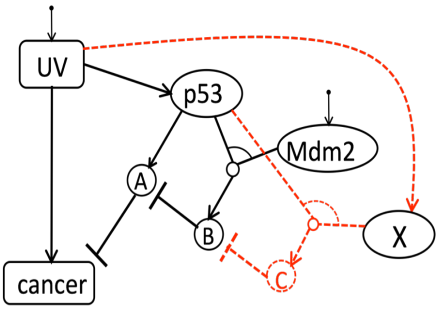


Figure 3

La figure 1 donne un exemple très simplifié d'interactions dans une cellule. Par des mécanismes divers non indiqués ici, les ultraviolets (UV) mettent la cellule en apostase (elle devient de fait immortelle) d'où le cancer. Ceci est représenté par une flèche. D'un autre coté les UV activent la production de la protéine P53. Cette protéine va activer une protéine A qui va bloquer le cancer. Mais P53 lié à la protéine Mdm2 va produire B, qui va bloquer A. Pour un biologiste la question est donc de bloquer le cancer en bloquant B. Les expériences biologiques ont montré que le X pourrait être un candidat pour ce blocage. Les figures 2 et 3 donnent deux types d'interactions possibles avec X pour expliquer ce blocage de B. Ici on a complété le graphe à la main, le problème est d'utiliser l'informatique pour ce faire.

Cet exemple est un jouet bien entendu très élémentaire, dans la pratique cette représentation graphique des voies de signalisation du réseau peut porter sur plusieurs milliers de gênes [20]. Mais le problème est tout de même non trivial car la découverte (l'abduction) va porter d'une part sur la présence éventuelle d'une protéine C et d'autre part sur l'ajout de liens entre protéines.

2.2 Représentation logique.

Les gênes et les protéines sont considérées comme un même objet (les gênes produisent les protéines). On va souvent se restreindre ici à une représentation propositionnelle. Dans la pratique l'étude fine des interactions va demander de représenter des augmentations ou des diminution de concentration de protéines. On sort donc du cadre propositionnel mais les problèmes de base sont les mêmes. Pour représenter une variation de concentration il est par exemple possible d'utiliser des prédicats de type "augmente" ou "diminue" et de limiter l'utilisation de ces prédicats [3].

Pour décrire les interactions entre les gênes dans la cellule, on part d'un langage L de logique classique (propositionnel ou de premier ordre). Dans L, la proposition A (resp $\neg A$) signifie que A est vrai (faux). On pourra dire par exemple *donne(UV)* pour dire que la cellule est soumise à des ultra-violets, ou encore, *écran-verre* –> ¬*donne(UV)* pour dire qu'un écran de verre protège des ultra-violets. On est dans un cadre logique, donc il est possible de représenter à peu près tout ce que l'on veut de manière naturelle. Le prix à payer, si on utilise la totalité du langage peut être l'explosion combinatoire des algorithmes.

Les interactions entres gênes sont une forme très simple de causalité. Pour exprimer ces interactions il est courant d'aller à l'essentiel en donnant deux relations binaires *cause(A,B) et bloque(A,B*) [4]. La première relation veut dire, par exemple, qu'une protéine A déclenche la production d'une protéine B la deuxième est une inhibition. De manière classique ces relations sont représentées dans le réseau de gênes, par A —> B et A —| B. Bien entendu cette causalité est élémentaire et de nombreux travaux très savants ont été écrits pour représenter les causalité [10] [11] [12].

Si l'inférence de logique classique *A —> B* est parfaitement décrite formellement, avec toutes les « bonnes » propriétés mathématiques et informatique (tautologie, non contradiction, transitivité, contraposée, modus ponens, compacité ...), la description des propriétés formelles de la causalité, est moins simple. La causalité ne peut pas être vue comme une relation logique classique. Un exemple élémentaire en est celui de l'expression « S'il pleut la pelouse est mouillée » Cette expression ne peut pas être traduite par la formule *pluie —> pelouse-mouillée,* qui signifierai que dès qu'il pleut la pelouse est automatiquement mouillée. En effet, il peut y avoir des exceptions à cette règle (la pelouse est sous un hangar…). On peut aussi changer l'environnement (on bâche la pelouse). Ces règles révisables et avec exceptions sont bien connues en Intelligence Artificielle. Elles ont donné lieu, en particulier, aux logiques non monotones et aux théories de la révision. D'un autre côté et de manière plus technique, on retrouve ici tous les problèmes classiques qui se posent quand on veut essayer de formaliser et d'utiliser la négation par échec dans des langages de programmation tels que Prolog ou Solar [13].

Pour donner les liens entre nos relations causales *cause* et *bloque*, dans un langage classique (calcul propositionnel ou logique du premier ordre) il faut donc faire deux choses :

- décrire les propriétés internes aux relations cause et bloque

- décrire les liens entre ces relations et la logique classique

Tout ceci en prenant en compte le problème de l'incertain et du révisable. Pour le premier aspect on donnera de manière explicite les liens minimum et nécessaires entre les deux relations causales. Les liens avec la logique classique seront décrits dans un premier temps

en logique des défauts. Ensuite, pour prendre en compte l'aspect de la découverte (abduction, champ de production) on les donnera en utilisant la logique des hypothèses.

Dans notre contexte, pour donner ces liens entre les relations cause et bloque on va au plus simple, en utilisant la logique classique. La solution élémentaire est alors de donner explicitement les deux schémas d'axiomes :

(C1) cause(A,B) ∧ cause(B, C) —> cause(A, C)
(C2) cause(A, B) ∧ bloque (B, C) —> bloque(A, C)

Nous pensons qu'il s'agit là du système axiomatique minimum, nécessaire et sans doute suffisant, pour l'application à la cellule. Pour l'instant il n'y a aucun lien formel entre les deux relations. Il est bien entendu possible d'ajouter d'autres axiomes pour prendre en compte ces liens, comme par exemple [8] [9] mais ceci complique le système formel sans que la pertinence de ces ajouts soit évidente dans ce cadre.

2.3 Causalité et inférence classique.

Dans une première approche, les premières propriétés que l'on a envie de donner peuvent s'exprimer naturellement, par des règles du type :

(1) Si A cause B et si A est vrai, alors B est vrai.
(2) Si A bloque B et si A est vrai, alors B est faux.

Suivant le contexte, le *vrai* peut vouloir dire *connu, certain, cru* ... ou encore, plus techniquement dans un système de démonstration automatique *démontré*.

La première idée est d'exprimer ces lois en logique classique par les axiomes :

cause(A, B) ∧ A —> B
bloque(A, B) ∧ A —> ¬B

On peut aussi les exprimer plus faiblement par des règles d'inférences proches du modus ponens :

cause(A, B) ∧ A / B
bloque(A, B) ∧ A / ¬B

Mais ces deux formulations posent problème dès qu'il y a conflit. Si par exemple on a un ensemble F de quatre formules F = {A, B, cause(A, C), bloque(B, C) }, on va dans les deux approches données ci dessus inférer de F, B et ¬B ce qui est inconsistant. Pour résoudre ce type de conflits, on peut essayer d'utiliser des méthodes inspirées par la programmation par contraintes, par exemple l'utilisation de la négation par échec. Il est aussi possible d'utiliser un raisonnement révisable, en particulier une logique non monotone. La première approche pose beaucoup de problèmes théoriques et techniques si on quitte les cas simples. Ces problèmes sont souvent résolus en ajoutant des propriétés au système formel, propriétés qui posent d'autres problèmes ... et on arrive à une belle usine à gaz. On va étudier ici une approche non monotone. Dans un premier temps, on utilise la logique des défauts.

2.4 Interactions et logique des défauts.

Pour résoudre les conflits vus ci dessus, l'idée intuitive est d'affaiblir la formulation des règles de causalité en :

(1') Si A cause B , et si A est vrai, et s'il *est possible que B*, alors B est vrai.
(2') Si A bloque B, et si A est vrai, et s'il *est possible que B est faux* alors B est faux.

La question est alors de décrire formellement le *possible*. Cette question a commencé à se poser en Intelligence Artificielle il y a une trentaine d'année, quand on a voulu formaliser le raisonnement humain naturel. Dans ce type de raisonnement, on est obligé de raisonner avec des informations incomplètes, incertaines, révisables et quelquefois fausses. D'autre part il faut souvent choisir entre plusieurs conclusions possibles antinomiques. L'exemple de base en est : {Les pingouins sont des oiseaux, Les oiseaux volent, Les pingouins ne volent pas}. Si Titi est un pingouin on arrive à une contradiction, le système est inconsistant. Cette inconsistance peut être levée si on arrive à gérer l'exception en remplaçant "Les oiseaux volent" par "*Généralement* les oiseaux volent". Les logiques non monotones décrivent formellement les modes de raisonnement qui prennent en compte ces phénomènes.

On utilisera ici une des logiques non-monotones les plus connues, la logique des défauts de Reiter. Dans cette logique, les règles (1') et (2') vont s'exprimer intuitivement.
(1") Si A cause B , et si A est vrai, et si *B n'est pas contradictoire*, alors B est vrai.
(2") Si A bloque B", et si A est vrai, et si ¬*B n'est pas contradictoire* alors ¬B est vrai.

Dans la logique des défauts, ces règles vont se représenter par des *défauts normaux* et s'écrire :

d1 : cause(A, B) ∧ A : B / B
d2 : bloque(A, B) ∧ A : ¬B / ¬B

Pour le cas élémentaire qui précède, si A est vrai, on a :
W = {A }
D = {d1, d2}

et on obtient deux extensions :
E1 qui contient B (en appliquant d1)
E2 qui contient ¬B (en appliquant d2)

Le conflit est donc résolu, mais se pose le problème des extensions à préférer ; est ce que B est induit ou bloqué? En fait ceci va vraiment dépendre du contexte. On peut par exemple préférer les interactions positives par rapport aux négatives ; ou bien utiliser des méthodes statistiques ou probabiliste. Une autre approche est de pondérer chaque extension en fonction des propriétés du problème. Du point de vue algorithmique cette préférence peut être évaluée pendant le calcul de l'extension ou sur le résultat final. Nous avons appliqué cette dernière technique à un problème industriel, l'utilisation des logiques des défauts pour simuler le comportement d'un officier de quart à bord d'un sous marin en temps de guerre, avec de bons résultats pratiques [21] [22] [23]. Les références sur le choix de l'extension sont nombreuses dans la littérature. On ne tranchera pas dans cet article pour un type de préférence applicable à la cellule, ça dépendra beaucoup du contexte et les biologistes ne sont pas toujours d'accord entre eux.

3. Une alternative à la logique des défauts : la logique des hypothèses.

La logique des défauts pose nombre de problèmes théoriques et pratiques. Pour résoudre ces problèmes on va utiliser la logique des hypothèses [15] [17] [18] [19]. Un fragment de cette logique est proche des modèles stables définis ultérieurement. Un des avantages de la logique des hypothèses est que les informations sont décrites dans une logique bimodale classique. Il sera alors possible d'utiliser ce formalisme pour décrire l'ensemble de toutes les informations, incertaines ou non. Cette logique va aussi permettre d'utiliser des algorithmes de recherche de conséquence pour traiter l'abduction ; ces mêmes algorithmes étant utilisés pour calculer les extensions.

Dans ce cadre on utilisera la notion de champ de production et les algorithmes liés donnés dans [1] [16]. Ces notions ont permis en particulier de définir le langage Solar [13] développé par l'équipe d'Inoue Katsumi.

3.1 Une très rapide introduction aux logiques modales.

Le calcul propositionnel et la logique du premier ordre donnent une valeur de vérité Vrai ou Faux aux formules. Les logiques modales vont nuancer ces valeurs. Au lieu de pouvoir exprimer uniquement "Il fait (ne fait pas) beau", on peut avoir envie de dire "Je sais qu'il fait beau", "Il est possible qu'il fasse beau", "Je crois qu'il fait beau, "Demain il fera beau" .. Techniquement, pour décrire ce type d'informations on va ajouter *un opérateur modal*, qui sera noté la plupart du temps L ou □ (dans la suite on utilisera L). On associe à cet opérateur son dual M défini par M = ¬L¬ [24]. Le sens qu'on donnera à ces couples {opérateur,dual} varie suivant le contexte. On utilise par exemples les couples {nécessaire, possible} {sait que, croit}, {obligatoire, permis}.. Si f est une formule quelconque, Lf veut dire qu'il est nécessaire que f soit vrai et Mf va signifier que f est possible (Mf = ¬L¬ f dit qu'il n'est pas nécessaire que f soit faux). La définition inductive de l'ensemble des formules autorise la combinaison les modalités dans les formules.
Par exemple L(p —> (M(p —>L¬q))) est une formule de logique modale.

Comme pour toute logique on donne alors des règles d'inférence et des schémas d'axiomes. La base est l'ensemble des règles et axiomes de la logique du calcul propositionnel ou de la logique du premier ordre. Pour les logiques *normales* (celles qui ont une sémantique des mondes possibles de Kripke) on a toujours la règle d'inférence de nécessitation (N) : f / Lf qui signifie que si f est un théorème/tautologie alors Lf est un théorème (ou intuitivement, si f est toujours vrai, alors f est nécessaire). Pour un système normal on va aussi ajouter un axiome de distribution, appelé (K). Cet axiome s'écrit L(A —> B) —> (LA —> L B) et permet de retrouver, pour la modalité, les propriétés de distributivité de la logique de premier ordre. Une logique modale qui contient (N) et (K) est appelée système K. Dans cet article on va aussi utiliser l'axiome (T) : LA —> A (si A est nécessaire/démontré alors A est vrai). En ajoutant au système K cet axiome (T) on obtient le système T.

3.2 Logique des hypothèses.

Pour définir la logique des hypothèses on part d'une logique classique (calcul propositionnel ou logique du premier ordre) à laquelle on ajoute deux opérateurs modaux, **L** et **[H].** L'opérateur **L** a les propriétés du système T et **[H]** celles du système K.

a) Intuitivement, si f est une formule **L**f signifie que f est *certain*, ou encore *démontré* dans un cadre de programmation par contrainte).

b) Pour décrire la notion d'hypothèse on va définir **H** comme le dual de [**H**], c'est à dire que **H**f = ¬[**H**]¬f. Intuitivement **H**f signifie que l'on fait l'hypothèse f.

c) Le seul lien entre les opérateurs **H** est **L** est donné par un axiome de liaison :

Hf —> ¬L(¬f) <=> L(¬f) —> ¬ Hf <=> ¬ (Hf ∧ L(¬f))

Si on fait l'hypothèse f, alors impossible de prouver ¬f .

Il est interdit de faire l'hypothèse f et de prouver ¬f en même temps.

Définitions :

- Une *hypothèse* est une formule fermée **Hf**.

- Une *théorie des hypothèses TH* , est un couple TH = { HY , F} ou F est un ensemble de formules du langage et HY un ensemble d'hypothèses

- Pour définir une extension, dans cette logique, on passe en fait à une approche préférentielle. Une *extension* de TH est un ensemble maximal consistant E = F ∪ H'Y ou H'Y est un sous ensemble de HY (on ajoute à F le plus d'hypothèses possible).

3.2 Défauts et logique des hypothèses :

On peut traduire toute logique des défauts en une théorie des hypothèses. Une théorie des défauts Δ ={W, D} se représentera par une théorie des hypothèses THΔ = { HY , F} ou:

- L'ensemble HY des hypothèses, est l'ensemble de tous les Hb, tels que b est la justification d'un défaut de Δ.

- L'ensemble des formules de F est composé :

a) De toutes les formules de type Lf ou f appartient à W

b) Des formules : Lp ∧ Hq —> Lr, pour tout défaut p :q / r de D.

Pour représenter l'information, on utilise en fait ici seulement un fragment de la logique des hypothèses car il n'y a pas de combinaison de modalités.

On montre alors qu'on représente bien la logique des défauts en supprimant tous les cas pathologiques (en particulier, toute théorie a une extension).

Propriétés :

- Si F est consistant THΔ, a toujours au moins une extension.

- Toute extension E de Δ, est représentable par une extension E' de THΔ. Ces extensions E et E' étant telles que f ∈ E si et seulement si Lf ∈ E'.

- La réciproque n'est pas vraie. Certaines extensions de THΔ, n'ont pas leur dual comme extension de Δ (en particulier dans le cas où Δ n'a pas d'extension). On peut reconnaître ces extensions particulières car ce sont celles qui contiennent au moins une formule ¬Hf sans contenir L¬f. Ceci est intéressant car il n'est pas nécessaire d'éliminer des extensions à cause d'une inconsistance locale, comme en logique des défauts.

Calcul d'extensions et abduction :

Un intérêt de la logique des hypothèses est que les algorithmes de calcul d'extension ne sont pas basés sur le calcul d'un point fixe comme en logique des défauts. En effet, soit F un ensemble de formules, g une formule et E = F ∪ H'Y une extension de F. On a alors que g est vrai dans E si et seulement si F ∪ H'Y |— g. En utilisant le théorème de compacité, il existe donc un sous ensemble fini { Hf1 , .. , Hfn} de HY', tel que :

F ∧ Hf1 ∧... ∧ Hfn |— g.

Donc pour savoir s'il existe une extension E dans laquelle g est vrai, il faudra simplement trouver un ensemble fini d'hypothèses { Hf1, ..., Hfn } tel que :

(a) F ∧ Hf1 ∧... ∧ Hfn est consistant

(b) F ∧ Hf1 ∧... ∧ Hfn |— g

La première condition vient du fait, que par définition, une extension est toujours consistante. Elle se vérifie par un test de satisfaisabilité et peut se réécrire :

(a') F |-/- ¬Hf1 ∨... ∨ ¬Hfn

La seconde condition est une propriété d'abduction, équivalente à :

(b') F ∧ ¬g |— ¬Hf1 ∨... ∨ ¬Hfn

Donc pour trouver si g est vrai dans une extension, il suffit d'ajouter la négation de g à F et de trouver une clause négative, dont tous les littéraux sont des négations d'hypothèses, impliqué par F ∧ ¬g. De plus cette clause négative ne doit pas être impliquée par F seul. L'algorithme va donc être un algorithme d'abduction. Ceci est intéressant car ce même algorithme va permettre à la fois de trouver des extensions et de compléter le réseau de gênes en découvrant de nouvelles interactions.

3.3 Champs de production et algorithme.

Pour formaliser la notion de découverte de nouvelles interactions, on peut utiliser les champs de productions [1] [16]. Si on a un ensemble C de clauses (ou un ensemble F de formules) un champ de production est un ensemble P de formules dont on veut savoir lesquelles sont impliquées par C. Des exemples classiques portent sur la découverte et l'étude des pannes dans un système expert [16].

Une autre utilisation des champs de production porte sur l'abduction dont on vient de voir une application dans 3.2. De manière générale, l'abduction peut se décrire naïvement par : étant donné un ensemble de formules *F* et un résultat cherché *g*, trouver une formule *h* consistante avec F, telle que en ajoutant *h* à *F* on déduit *g* (F∧h |— g). Comme g et F sont connus la découverte de g peut se réécrire en logique par un problème de déduction : trouver h tel que F∧¬g |—¬h et F |-/- ¬h

Dans la pratique h n'est presque jamais une formule quelconque, sa forme syntaxique ou sémantique est précisée par le champ de production. Par exemple dans le cas de la logique des hypothèses le champ de production va être l'ensemble des clauses négatives dont tous les littéraux sont des négations d'hypothèses comme dans 3.2. Ce type de champ de production va se retrouver dans la majeure partie des logiques non-monotones et en particulier les logiques préférentielles [2] , la logiques des défauts et les X-logiques [7].

Une autre application des champs de production va être la recherche de liaisons manquantes dans le graphe des voies de signalisations. On peut chercher à compléter ce graphe avec une seule liaison et le champ de production est alors l'ensemble des clauses négatives unaires du type ¬cause(A,B) ou ¬bloque(A,B). Si on veut compléter avec tous les ensembles de liaisons possibles on pourra prendre comme champ de production l'ensemble des clauses ne contenant que des négations de cause(A,B) et bloque(A,B). Dans ce dernier cas, pour éviter l'explosion combinatoire des résultats il est possible de limiter la longueur des clauses produites. En jouant avec la forme des champs de production et en quittant le strict domaine des clauses bien d'autres utilisations sont possibles.

Pour l'implémentation des algorithmes de production de clauses on été définis dans [1] et [7] avec des bons résultats. La notion de champ de production et les algorithmes liés ont été la base du langage SOLAR développé par l'équipe d'Inoue Katsumi et utilisé dans un cadre de bioinformatique [3] [8] [13].

4. Représentation et utilisation des voies de signalisation.

On va représenter les réseaux de gênes en utilisant la logique des hypothèses et jouer avec cette représentation.

4.1 Représentation.

On se donne donc un réseau qui décrit certaines interactions dans une cellule. On va décrire ce réseau dans une logique des hypothèses TH = { HY, F }.

Pour l'instant on considère que HY contient toutes les formules du type Hf si f est une formule du langage. En fait dans la pratique, cet ensemble va être réduit.

L'ensemble F décrit toutes les informations connues/certaines sur le réseau. Comme les informations sont certaines elles seront traduites naturellement par des formules modales du type **L**f . L'ensemble F va donc contenir en particulier :

- **L**cause(A, B) , si A —>B est un arc du réseau.
- **L**bloque(A, B) , si A —| B est un arc bloquant
- **L**p si une protéine p est activée, et **L**¬p si elle est bloquée.
- On représentera aussi toutes les relations complexes entre protéines. Par exemple si p1 et p2 se lient pour produire une protéine p3, on écrira **L**(p1 ∧p2) —> **L**p3. On pourra aussi représenter les interactions externes avec les UV, avec les médicaments...
- La traduction des défauts normaux d1 et d2 définis dans 2.4 s'écrit naturellement :

d'1 : **L**cause(a, b) ∧ **L**a ∧ **H**b —> **Lb**
d'2 : **L**bloque(a, b) ∧ **L**b ∧ **H**¬b —> **L**¬b

Le langage de représentation est souple (c'est l'intérêt d'une représentation logique). Si par exemple on veut considérer que certaines causalités sont certaines, on ne les traduira pas par des défauts mais par des formules sans hypothèses :

Lcause(a, b) ∧ **L**a —> **Lb**
Lbloque(a, b) ∧ **L**b—> **L**¬b

Bien entendu ce dernier cas peut amener à des contradictions et il faudra alors, avant d'utiliser le réseau, vérifier sa consistance.

4.2 Causalité Abduction et hypothèses

On a maintenant un langage de description pour décrire les réseaux de gênes. Ce langage est assez naturel à condition d'admettre les notions de certitude et d'hypothèse. Il permet de représenter les logiques des défauts et l'incertain. L'intérêt d'un tel langage est qu'il va permettre d'essayer de découvrir de nouvelles voies de signalisation et de constituer des plans d'expériences.

En fait si on a un réseau de gênes, on peut se poser dans un premier temps trois questions :

1) Est ce que le réseau est cohérent?

2) Est ce que dans un certain état du réseau, une protéine p est activée / bloqué?

3) Quelles sont les conditions qui manquent pour activer/ bloquer une protéine?

La première question est simple, c'est de la déduction et plus particulièrement un test de consistance. Le réseau est traduit dans une théorie des hypothèses HH = {HY, W}. Comme aucune hypothèse n'apparait sous forme positive dans la forme normale conjonctive de W, on peut démontrer qu'il suffit simplement de prouver que W est consistant après avoir affecté toutes les hypothèses à faux. On peut aussi sortir du cadre strict de la logique des hypothèses en disant que certaines hypothèses sont vraies, mais les algorithmes sont alors beaucoup moins simples.

La deuxième question est également de la déduction classique. Il faut prouver que p (resps ¬p) est impliqué par l'ensemble W des faits connus : W |— p (resp W |— ¬p). Bien entendu s'il existe deux chaînes de signalisation (de fait deux extensions) l'une donnant p et l'autre ¬p, on ne pourra rien prouver. Et il faudra choisir entre ces extensions.

La troisième question est du domaine de l'abduction. Les conditions manquantes, peuvent être diverses. On peut ajouter au réseau des liens de causalité, des protéines activées ou bloquées, ou encore des hypothèses ; la liste n'est pas exhaustive.

La logique des hypothèses va permette de représenter ces trois possibilités dans le même formalisme. Le champ de production sera composé de toutes les formules qui expriment ces trois données, c'est à dire de toutes les formules qui s'écrivent sur l'ensemble de propositions :

$$PP = \mathbf{H}Y \cup \{\mathbf{L}p\} \cup \{\mathbf{L}\neg p\} \cup \{\mathbf{L}\text{cause}(a, b)\} \cup \{\mathbf{L}\text{bloque}(a, b)\}$$

Bien entendu, le champ de production est ici énorme et il faut le réduire, ou mieux le décrire mieux. C'est l'une des suites du travail à faire.

4.3 Coupure de liaisons.

Une question subsidiaire plus compliquée est de savoir quel(s) arc(s) supprimer dans le réseau de gènes pour obtenir le résultat cherché. Dans ce cas on peut considérer que les liaisons sont également incertaines. Pour exprimer ceci on va remplacer certaines informations certaines du type, **L**cause(a,b) ou **L**bloque(a,b) par des règles avec hypothèses:

H cause(a, b) —> **L**cause(a, b)
H cause(a, b) —> **L**cause(a, b)

On pourrait faire la même chose avec les protéines ou l'absence de protéines en ajoutant des règles du type **H**p —> **L**p et **H**¬p —> **L**¬p. Il s'agit là d'une technique extrême car rien n'est certain, à manier avec précaution car les champs de production sont énormes. Mais elle est intéressante formellement, car elle permet de couper des liaisons sans utiliser des techniques lourdes de suppressions d'information.

5 Conclusion

Nous avons présenté une première approche d'utilisation de la logique des hypothèses et des algorithmes de production de clauses pour représenter, utiliser et compléter les voies de signalisation dans la cellule. Tout ceci se faisant à l'aide d'un même formalisme et d'un même algorithme d'abduction.

Le problème de l'implémentation et de la complexité algorithmique n'a pas été abordé. Il sera sans doute nécessaire de réduire les possibilités d'expression du langage pour arriver à des temps de calcul raisonnables. On retrouve ici la même problématique que pour la résolution pratique des problèmes NP-Complets. Bien entendu le problème est ici plus compliqué, les logiques modales étant dans un cas général PSPACE. Mais dans notre application, on utilise seulement un fragment de ces logiques modales car on ne combine pas les opérateurs modaux. La complexité pratique peut alors rester accessible en contrôlant l'écriture des règles. Par exemple dans un premier temps on peut essayer de se restreindre à quelque chose qui ressemble à des clauses de Horn.

Nous venons de montrer [21] [22] que l'utilisation de la logique des défauts avec une implémentation simple avec Prolog permet de modéliser le comportement humain pour les simulations de combat naval de sous-marins. Les résultats pratiques sont intéressants, une simulation de 4 heures de navigation prend moins d'une seconde sans optimisation du programme. En utilisant dans un premier temps des techniques analogues, ou peut être des ASP, tous les espoirs sont permis.

Une autre question est de donner des bonnes restrictions utile des champs de productions utilisés pour éviter l'explosion combinatoire des résultats. Ici beaucoup reste à faire et l'expérimentation guidera. Mais l'enjeu pratique est important.

Références .

1. Boi J.M, Innocenti E. Rauzy A., P. Siegel P. (92*) Production Fields : a New approach to Deduction Problemss and two Algorithms for propositional Calculus* . Revue d'Intelligence Artificielle, vol 6 - n° 3/1992 , p 235, 255

2. Bossu G., Siegel P. : *Saturation, Nonmonotonic Reasoning and the Closed World Assumption* . Artificial Intelligence 25, Janvier 1985, p. 13-63

3. Doncescu A., Yamamoto Y. and Inoue K. *Biological systems analysis using Inductive Logic Programming*. Proceedings of the 21st International Conference on Advanced Information Networking and Applications (AINA 2007), pages 690-695, IEEE Computer Society, 2007.

4. Doncescu A. , Inoue K. and Yamamoto. K *Knowledge- based discovery in systems biology using CF-induction*. New Trends in Applied Artificial Intelligence: Proceedings of the 20th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2007), Lecture Notes in Artificial Intelligence, volume 4570, pages 395-404, Springer, 2007

5. Demongeot J. *Multi-stationarity and cell differentiation*. J. Biol. Systems., 6, 1-2 (1998).

6. Farmer M., Zakariya M. *: Enabling e-Technology for bio-informatics*. In: UK-Japan High Technology Industry Forum, London, July2004

7. Forget L, Rish V. , Siegel P. *Preferential logics are X-logics*. Journal of Logic Computational, 10, 2000, pp. 1-13. R[[13] L. Forget and V. Risch and P. Siegel. *Preferential logics are X-logics*. Journal of Logic Computational, 10, 2000, pp. 1-13.

8. Inoue K : *Induction as Consequence Finding*. In: Machine Learning, 55(2):109-135, 2004.

9. Inoue K, Saito H.: *Circumscription Policies for Induction*. In: Proceedings of 14th Conf. on Inductive Logic Programming, LNAI 3194, pp.164-179, Springer, September 2004.

10. D. Kayser, F. Levy : *Modélisation symbolique du raisonnement causal*, Intellecta 2004/1, 38, pp 291-232

11. Giunchiglia, E., Lee, J., Lifschitz, V., McCain, N., & Turner, H. (Mars 2004). *Nonmonotonic causal theories*. Artificial Intelligence vol.153 n o 1-2 pp.49-104.

12. Ginsberg, M. L., Smith, D. E. (Juillet 1988). *Reasoning about action II: the qualification problem*. Artificial Intelligence vol.35 n°3 pp.311-342.

13. Nabeshima H. , Iwanuma K., Inoue K. Ray O. : SOLAR: *An automated deduction system for consequence finding*. AI Commun. 23(2-3): 183-203 (2010)

14. Roux-Rouquié M., Hood L., Imbeaud S., Auffray C. *Issues in Computational Methods for Functional Genomics and Systems Biology*. CMSB 2003: 182-186

15. Schwind P. , Siegel P : *Modal logic for Hypothesis Theory,* Fundamentae Informaticae, col 21, n° 1-2 89-101

16. Siegel P. *Représentation et utilisation de la connaissance en calcul propositionnel.* Thèse de Doctorat d'état en Informatique, Université d'Aix-Marseille II, juillet 1987. Sous la direction d'Alain Colmerauer.

17. Siegel P. : *A modal language for Nonmonotonic Reasonning*. Proc. Workshop DRUMS/CEE Marseille 24-27 Février 90.

18. Siegel P. C.Schwind (91) *Hypothesis theory for nonmonotonic reasoning.* First International Proc. Workshop on Nonstandard Queries and Answers, Toulouse, July 1991.

19. Siegel P. , Schwind C. (93) *Modal logic based theory for nonmonotonic reasoning.* Journal of Applied Non classical Logic, vol 3 - n° 1/1993, P 73-92.

20. Tran N. , Baral C. (2007) *Hypothesizing and reasoning about signaling networks*. Journal of Applied Logic 7 (2009) 253-274

21. Toulgoat I. *Modélisation de combat humain dans les simulations de combat naval*. Thèse de doctorat, université de Toulon 31 Janvier 2011

22. Toulgouat I, Siegel, Lacroix Y, Botto J. : *Operator Decision in naval Action's Simulation",* 13th international workshop on Non-Monotonic Reasoning, NMR'10, mai 2010

23. Toulgouat I , Siegel P., Lacroix Y, Botto J : Operator decision modeling : application to a scenario involving two submarines, *in Computer Application and Information Technology in the Maritine Industries,-COMPIT10* Gubbio, Italy, 12-14 April 2010

24. http://fr.wikipedia.org/wiki/Logique_modale

De l'organisation hexagonale des concepts de Blanché à l'analyse formelle de concepts et à la théorie des possibilités

Didier Dubois Henri Prade

Institut de Recherche en Informatique de Toulouse - IRIT
Université de Toulouse, 118, route de Narbonne, 31062 Toulouse cedex 09, France
dubois@irit.fr, prade@irit.fr

Résumé : L'article introduit tout d'abord un cube d'oppositions qui associe le traditionnel carré des oppositions au carré dual obtenu par réciprocation (au sens de Piaget). On montre ensuite que l'extension de Blanché du carré des oppositions en une structure hexagonale conceptuelle repose toujours sur une partition abstraite à 3 éléments. Considérer des partitions à 4 éléments permet d'organiser les 16 connecteurs binaires en un tétraèdre régulier. Enfin, on montre que le cube d'oppositions, une fois interprété en termes modaux, correspond à une généralisation récente de l'analyse formelle de concepts, où des hexagones remarquables apparaissent. Cette généralisation de l'analyse formelle de concepts a été motivée par un parallèle avec la théorie bipolaire des possibilités qui est basée sur 4 fonctions d'ensembles, lesquelles, quoique graduées, peuvent être structurées de manière similaire.

1 Introduction

L'idée d'opposition n'est pas toujours une notion binaire. En effet, tandis que la négation involutive (bi-valuée) classique induit des dichotomies, une partition en plus de deux parties induit implicitement une forme plus riche d'oppositions. Au niveau linguistique, on a remarqué depuis l'antiquité grecque, qu'un énoncé (A) de la forme "tout X est Y" a pour négation l'énoncé (O) "au moins un X n'est pas Y", tandis qu'un énoncé tel que (E) "aucun X n'est Y" correspond clairement à une forme d'opposition encore plus forte au premier énoncé (A). Ces trois énoncés, et la négation de (E), c'est-à-dire (I) "au moins un X est Y", donnent naissance au carré des oppositions bien connu (21), traditionellement associé aux lettres A, I (moitié affirmative) et E, O (moitié négative). Ce carré remonte à Aristote et a été discusté plus avant par Apulée, Boèce et ensuite les logiciens du Moyen-Age.

Notons que d'autres énoncés peuvent être associés à (A) : la négation peut en effet plutôt porter 1) sur X, ou 2) à la fois sur X et Y, c'est-à-dire

1. (e) : "tout non-X est Y", et

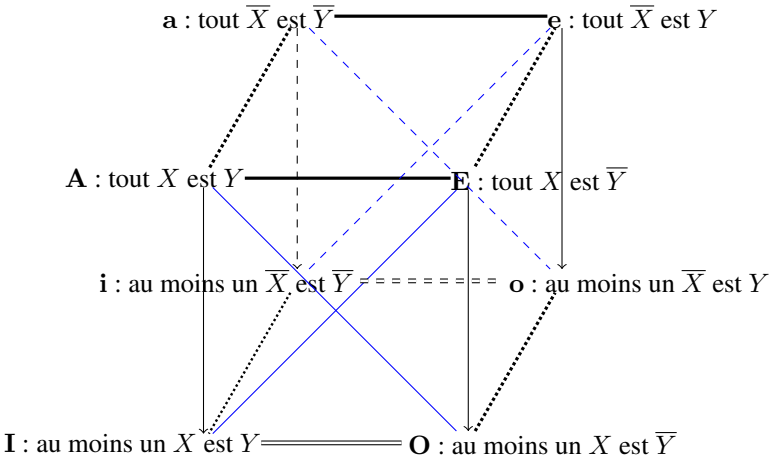


FIGURE 1 – Cube des oppositions

2. (a) : "tout non-X est non-Y",

qui peuvent être eux-même niés, ce qui donne (i) : "au moins un non-X est non-Y" et (o) :"au moins un non-X est Y" respectivement.
Il est intéressant de constater que George Boole (8) et Augustus De Morgan (9) ont considéré ces derniers énoncés en utilisant les transformations correspondant à (e) et (a) respectivement. La dénotation des énoncés par a, i, e, o a été proposée par De Morgan [1] (9).

On est ainsi amené à considérer 8 énoncés, A, I, E, O, a, i, e, o qui peuvent être organisés dans ce qu'on peut appeler un *cube d'opposition* [2] comme celui de la Figure 1, où $\overline{X}$ (resp. $\overline{Y}$) dénote non-X (resp. non-Y). La face de devant du cube est le carré traditionnel des oppositions, A, I, E, O, où les lignes épaisses lient des contraires, les lignes fines doubles sans flèche lient des sous-contraires, les lignes diagonales sans flèche lient des contradictoires, et les flèches verticales pointent sur les subalternes (et expriment des implications si l'ensemble des X's n'est pas vide). La face de derrière constitue un autre carré des oppositions, a, i, e, o, basé sur les "non-X's".

Le carré des oppositions, qui n'a plus été étudié après la fin du Moyen-Age, a perdu progressivement son importance conceptuelle, ainsi que sa traduction géométrique, jusqu'à l'émergence de la logique moderne ; par exemple, le carré est encore dessiné par Scipion Dupleix (16) au début du 17eme siècle, tandis que 150 ans plus tard dans l'Encyclopédie de Diderot et d'Alembert (1), seuls les énoncés logiques associés à ses 4 sommets (avec leurs noms traditionnels A, I, E, O) sont donnés. C'est seulement au début des années 1950 que le carré des oppositions a commencé à connaître un certain re-

1. La notation vient du fait que De Morgan (9) dénote par x les éléments qui sont non-X.

2. Ce cube ne doit pas être confondu avec le "cube logique" introduit par Moretti (20), où toutes les arêtes ont la même interprétation sémantique (correspondant à celle des flèches verticales unidirectionnelles du cube de la Figure 1) et expriment des implications dans une structure de "4-opposition".

gain d'intérêt [3]. Tout d'abord, un physicien, spécialiste de mécanique, Augustin Sesmat (27) et un philosophe de la logique, Robert Blanché (6), ont proposé indépendamment de compléter le carré des oppositions sous la forme d'un hexagone en ajoutant deux sommets, nommés par ce dernier auteur, U et Y respectivement [4] : (Y) est la conjonction de (I) et (O), et correspond ainsi à "au moins un X est Y et au moins un ne l'est pas", tandis que (U) est la disjonction de (A) et (E), et se lit "soit tout X est Y ou aucun X n'est Y", qui est ainsi équivalent à la négation de l'énoncé précédent. De plus, Y implique à la fois (I) et (O), tandis que, (A), tout comme (E), implique (U).

Indépendamment, et à peu près au même moment, le psychologue Jean Piaget (23; 24) dans son traité de logique "opérationnelle" (qu'il place dans la tradition de G. Boole (8)) étudie de manière formelle le groupe de Klein des transformations logiques qu'il a identifié dans ses travaux sur l'acquisition des capacités cognitives qui sont progressivement maîtrisées par les enfants et les pré-adolescents. Il s'agit d'une structure de groupe à 4 éléments constitué par 4 transformations applicables à tout énoncé logique $\phi = f(p, q, r, ...)$ comportant des variables binaires p, q, r, ..., qui sont i) l'identité $I(\phi) = \phi$, la négation $N(\phi) = \neg\phi$, la réciprocation $R(\phi) = f(\neg p, \neg q, \neg r, ...)$, et la corrélation $C(\phi) = \neg f(\neg p, \neg q, \neg r, ...)$. On peut vérifier facilement que $N = RC$, $R = NC$, $C = NR$, et $I = NRC$ (la composition des transformations est l'opération qui définit le groupe). Ce groupe a aussi été identifié au même moment par le mathématicien Walter H. Gottschalk (19) dans sa théorie de la quaternalité ("quaternality") (où il se réfère aussi au carré des oppositions). Comme on peut le voir sur la Figure 1, les lignes en pointillés serrés du cube des oppositions correspondent à la réciprocation, et le groupe de Piaget peut être vu à l'œuvre sur les deux rectangles diagonaux AaOo et EeIi. Par exemple, on a $R(A) = C(N(A)) = a$, $C(O) = N(R(O)) = a$, ou $N(R(C(E))) = N(R(i)) = N(I) = E$.

Le carré des oppositions a une autre lecture classique en termes de modalités, où (A) correspond maintenant à "$\Box p$", (E) à "$\neg\Diamond p$", (I) à "$\Diamond p$", et (O) à "$\neg\Box p$" (ce qui peut être complété en un hexagone). Dans son étude de la négation paraconsistente, Jean-Yves Béziau (4) a récemment introduit deux autres hexagones "modaux" associés respectivement aux carrés (A) : = "p", (E) : = "$\neg\Diamond p$", (I) := "$\Diamond p$", et (O) : = "$\neg p$", et (A) = "$\Box p$", (E) : = "$\neg p$", (I)= "p", et (O) : = "$\neg\Box p$", qui conduit à un dodécaèdre stellaire d'oppositions. Cette découverte a été le point de départ de nouvelles études par en particulier Alessio Moretti (20), Régis Pellissier (22), et Hans Smessaert (28) qui ont développé une théorie géneralisée de n-opposition, où le carré et l'hexagone correspondent respectivement aux cas $n = 2$ et $n = 3$.

Dans cet article, nous procédons à un réexamen des travaux de Blanché (7) sur l'organisation structurelle des concepts en termes d'hexagones et présentons une explication simple, basée sur des 3-partitions, de l'"universalité" des hexagones dans l'organisation

3. On consultera avec profit le site d'Alessio Moretti sur de nombreux points d'histoire touchant au carré des oppositions et à ses extensions http ://alessiomoretti.perso.sfr.fr/NOTAncestors.html.

4. R. Blanché (5) introduisit Y tout d'abord en 1952, avant de le compléter avec U dans son article de 1953 (6). Sesmat (27) utilisent déjà les mêmes lettres Y et U pour dénoter les mêmes choses.

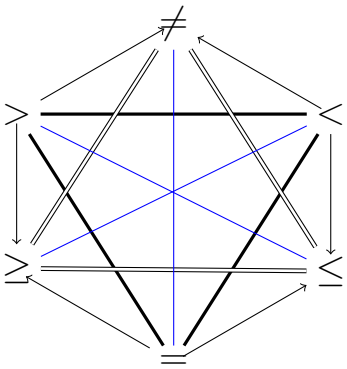


FIGURE 2 – Hexagone de Blanché induit par un pré-ordre complet

des concepts. Nous montrons que le cas des 4-partitions donne naissance à des structures tétraédrales, qu'on trouve en particulier à l'œuvre dans l'organisation logique des 16 connecteurs binaires. Ceci est le sujet de la Section ci-après. La Section 3 revient à une interprétation de type modal pour un nouveau carré d'oppositions (où certaines relations sont cependant modifiées), obtenu directement à partir d'un cadre qui étend la théorie conventionnelle de l'analyse formelle de concepts (18). Ce nouveau carré est aussi associé (de manière standard) à un hexagone qui fait sens, et conduit à une contre-partie modale du cube des oppositions présenté dans l'introduction. Enfin, la Section 4 propose avec la théorie des possibilités bipolaire une contre-partie graduelle de cette structure [5].

2 De l'hexagone au tétraèdre

Dans son livre (7) sur les "structures intellectuelles", Blanché propose différents exemples de situations où des hexagones sont rencontrés, en commençant par les 6 comparateurs existant en mathématiques, qui se trouvent être harmonieusement répartis sur un hexagone : (A) : = " $>$", (E) : = " $<$", (U) : = " $\neq$", (I) : = " $\geq$", (O) : = " $\leq$", et (Y) : = " $=$". Voir la Figure 2. Plusieurs autres exemples sont donnés. Mentionnons, par exemple, parmi d'autres :

- avec des modalités déontiques : (A) ="obligatoire", (E) : ="interdit", (U) : ="régulé", (I) : ="permis", (O) : ="non obligatoire", et (Y) : ="indifférent",
- à propos d'attitudes : (A) ="J'accepte", (E) : ="Je refuse", (U) : ="Je choisis", (I) : ="Je ne dis pas non", (O) : ="Je ne dis pas oui", et (Y) : ="J'hésite",
- à propos de goûts : (A) ="J'aime çà", (E) : ="Je déteste çà", (U) : ="Je n'y suis pas indifférent", (I) : ="Je ne déteste pas çà", (O) : =" Je n'aime pas çà", et (Y) : ="J'y suis indifférent".

Cependant, Blanché (7) n'insiste pas vraiment sur la raison pour laquelle la structure hexagonale est rencontrée dans des domaines très différents. Cela provient du fait que

5. Les deux dernières sections de cet article développent des résultats (15) présentés initialement au 2ème "World Congress on the Square of Opposition".

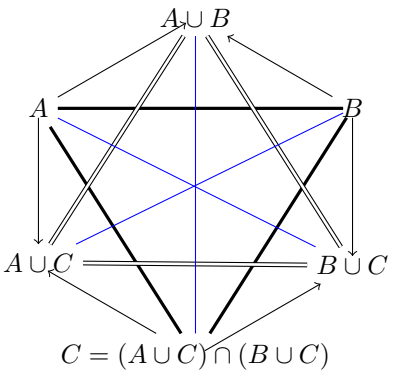


FIGURE 3 – Hexagone induit par une tri-partition (A, B, C)

l'hexagone est sous-tendu par une partition de base en 3 situations mutuellement exclusives, comme nous allons le voir [6].

Soit U un univers du discours partitionné en 3 parties A, B, et C. C'est-à-dire qu'on a $A \cap B = B \cap C = A \cap C = \emptyset$, $A \neq \emptyset$, $B \neq \emptyset$, $C \neq \emptyset$, $A \cup B \cup C = U$. On peut alors aisément construire l'hexagone de la Figure 2. Observons que $(A \cup C) \cap (B \cup C) = C$. Il est clair que chaque flèche correspondant à une arête de l'hexagone traduit une implication. Par exemple, $A \subseteq A \cup B$. Les 3 diagonales expriment l'exclusion mutuelle (e.g., $A \cap (B \cup C) = \emptyset$) ainsi que la complémentarité (puisque $A \cup (B \cup C) = U$). Les 3 lignes épaisses relient des contraires (les deux peuvent être faux en même temps, mais ne peuvent pas être vrais en même temps puisque leur intersection est vide), les 3 doubles lignes relient des sous-contraires (les deux peuvent être vrais en même temps, mais ne peuvent pas être faux en même temps puisque leur union est le référentiel). Clairement, dans l'exemple des relations de comparaison en mathématiques, (A) : = " $>$ ", (E) : = " $<$ ", (Y) : = " $=$ " forment une 3-partition de l'ensemble des paires de valeurs. De plus on a en effet " $\geq$ " = " $>$ " $\cup$ " $=$ ". Une telle partition existe aussi dans les autres cas, e.g., (A) ="J'aime çà", (E) : ="Je déteste çà", et (Y) : ="J'y suis indifférent", et on peut considérer que "Je ne déteste pas çà"= "J'aime çà"ou "Je n'y suis pas indifférent". Notons que A, B, et C jouent des rôles similaires, et qu'il n'est pas nécessaire qu'une des 3 situations de base soit entre ou sépare les deux autres (comme c'est le cas pour "$=$ " par rapport à " $>$ " et " $<$ "), même si ce cas se rencontre souvent.

De plus, dans le cas où $C = \emptyset$, l'hexagone de Blanché (en utilisant la logique propositionelle à la place de propositions quantifiées dans le carré des oppositions de base) se réduit à un losange (puisque les sommets (A) et (I), et (E) et (O), coïncident respectivement). Les sommets correspondent alors à l'affirmation (A = I), à la négation (E = O), à la contradiction (Y) et à l'ignorance (U), qui sont les 4 valeurs de la logique de Belnap (2).

6. Blanché écrit cependant dans son article fondateur (6) p. 130 : "cette triade AYE, système de termes mutuellement exclusifs et collectivement exhaustifs, fournit la molécule première pour la constitution d'un tableau des opposés".

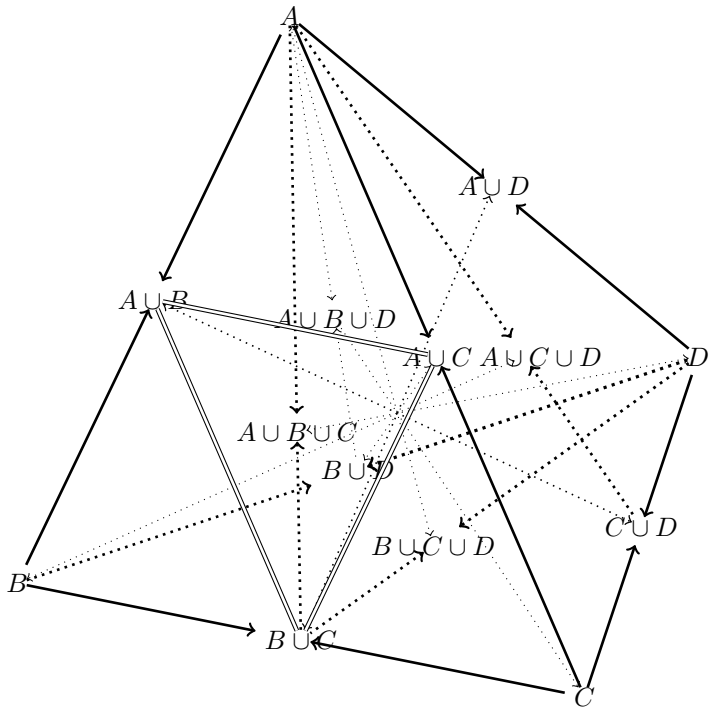


FIGURE 4 – Un tétraèdre induit par une 4-partition (A,B,C,D)

Le processus précédent s'étend au cas des 4-partitions, qui présente aussi un intérêt pratique. Considérons une partition en 4 parties A, B, C, et D, de l'univers du discours. Ainsi, on suppose $A \cap B = B \cap C = A \cap C = A \cap D = B \cap D = C \cap D = \emptyset$, $A \neq \emptyset$, $B \neq \emptyset$, $C \neq \emptyset$, $D \neq \emptyset$, $A \cup B \cup C \cup D = U$. Dans le cas de la 3-partition, l'hexagone est constitué de deux triangles opposés, A, B, C et $A \cup B, A \cup C, B \cup C$. Plutôt que d'utiliser la représentation hexagonale, on peut utiliser une double représentation triangulaire, qui apparaissait déjà dans (27), où le second triangle est maintenant à l'intérieur du premier, et où $A \cup B$ est associé avec le milieu du côté qui joint A et B, et de manière similaire $A \cup C$ et $B \cup C$ sont au milieu respectivement de A et C, et de B et C.

La 4-partition donne naissance à 4 doubles triangles de cette sorte (en considérant à chaque fois 3 parties sur 4), qui constitue les 4 faces d'un tétraèdre régulier ayant pour sommets A, B, C, et D, comme dans la Figure 3. Traçons maintenant toutes les lignes joignant un sommet au milieu des 3 côtés opposés, et associons l'union des sous-ensembles des deux extrémités au point situé au 2/3 de la distance à partir du sommet sur le segment considéré. Par exemple, $A \cup B \cup C$ est associé au point à la distance de 1/3 à partir de $B \cup C$ et de 2/3 à partir de A, i.e., est le centre du triangle A, B, C.

En utilisant des résultats de la géométrie des tétraèdres réguliers, on peut voir que $A \cup B \cup C, A \cup B \cup D, A \cup C \cup D, B \cup C \cup D$ forment un tétraèdre régulier, dual de A, B, C, D. De plus, toutes les lignes qui joignent un sommet à l'un des milieux des

côtés opposés, ou deux milieux de deux arêtes opposées se rencontrent en un même point, qui est le centre du tétraèdre. Ce sont les 7 lignes qui joignent $A \cup B$ et $C \cup D$, $A \cup C$ et $B \cup D$, $A \cup D$ et $B \cup C$, A et $B \cup C \cup D$, B et $A \cup C \cup D$, C et $A \cup B \cup D$, D et $A \cup B \cup C$. Notons que les extrémitiés de chacune de ces lignes sont associées à des sous-ensembles disjoints dont l'union est le référentiel U. Sur chacune des 4 faces du tétraèdre, nous avons la représentation par double triangle, équivalente à la représentation hexagonale, où maintenant le centre est associé à l'union des sous-ensembles correspondant au 3 sommets de la face.

Il y a moins d'exemples naturels de 4-partitions. Un exemple est obtenu quand on compare deux objets s et t au moyen d'une relation réflexive de surclassement pour laquelle "s est non comparable à t" est une option au même titre que "s est préféré à t", "t est préféré à s", et "on est indifférent entre s et t" (25). Par ailleurs, un cas particulier important de 4-partition est induit par deux variables binaires p et q, qui donnent naissance à 4 interprétations mutuellement exclusives, à savoir $p \wedge q$, $p \wedge \neg q$, $\neg p \wedge q$, $\neg p \wedge \neg q$. Le tétraèdre correspondant, qui exhibe tous les connecteurs binaires de la logique classique, est donné en Figure 4.

Comme nous allons le voir, cela offre une alternative à la double structure hexagonale "ouverte" proposée par Blanché (7) pour fournir une vue unifiée de tous les connecteurs binaires. En s'appuyant sur la géométrie des tétraèdres réguliers, les points suivants méritent d'être remarqués :

- tout connecteur qui apparaît entre deux autres sur la même ligne (qui peut être une arête du tétraèdre, ou un segment reliant un sommet au milieu d'une arête opposée) est impliqué par chacun des deux connecteurs figurant à ses extrémités. Par exemple, $p \wedge q$, ou $\neg p \wedge \neg q$ implique $p \equiv q$. De manière similaire, $p \wedge q$, ou $\neg p$, implique $\neg p \vee q = p \rightarrow q$,
- un segment reliant les milieux de deux arêtes opposées, ou un segment reliant un sommet au centre de la face opposée, est associé à une paire de connecteurs (à ses extrémités) dont la conjonction donne la contradiction, et la disjonction la tautologie. Puisque tous ces segments ont un point commun, qui est le centre du tétraèdre, ce centre peut être associé à la fois à la contradiction et à la tautologie ;
- en dehors de la contradiction et de la tautologie, les 14 autres connecteurs binaires apparaissent dans cette construction ;
- les 4 disjonctions $p \vee q$, $\neg p \vee q = p \rightarrow q$, $p \vee \neg q = q \rightarrow p$, et $\neg p \vee \neg q = p \rightarrow \neg q$ sont les sommets du tétraèdre dual à l'intérieur du tétraèdre initial associé au 4 conjonctions $p \wedge q$, $\neg p \wedge q$, $p \wedge \neg q$, et $\neg p \wedge \neg q$;
- p, $\neg p$, q, $\neg q$, $p \equiv q$ et $p \Delta q$ (ou exclusif) sont les sommets d'un octaèdre régulier ;
- le groupe de Piaget peut encore être vu à l'œuvre, quand on considère une arête du tétraèdre avec l'arête parallèle correspondante dans le tétraèdre dual. Par exemple, $p \wedge q$, $\neg p \wedge \neg q$, $p \vee q$, $\neg p \vee \neg q = p \rightarrow \neg q$ sont en effet échangés par les transformations, I, N, R, C du groupe de Piaget.
- la position d'un connecteur sur un segment reflète le nombre de bits à basculer pour passer de lui aux connecteurs qui sont à ses extrémités (et réciproquement). Par exemple, $p \equiv q$ est à mi-chemin entre $p \wedge q$ et $\neg p \wedge \neg q$: pour chacune de

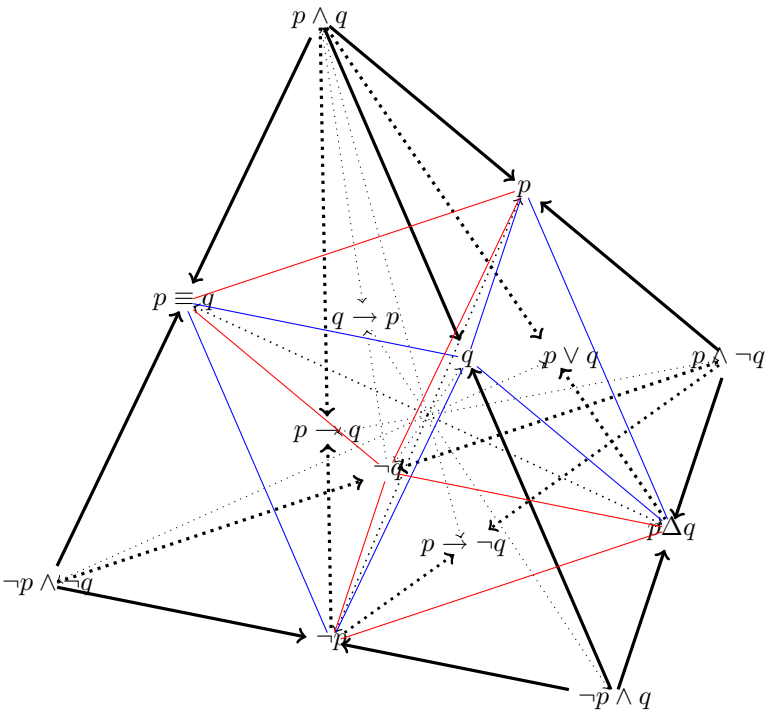


FIGURE 5 – tétraèdre des 16 connecteurs binaires

ces deux dernières formules, une ligne dans leur table de vérité doit passer de 0 à 1 pour obtenir la table de vérité de $p \equiv q$. De manière similaire, $p \rightarrow q$ est à une distance 1/3 de $\neg p$ et à une distance 2/3 de $p \wedge q$: une ligne dans la table de vérité de $\neg p$ doit passer de 0 à 1 pour obtenir la table de $p \rightarrow q$, tandis que 2 lignes dans la table de vérité de $p \wedge q$ doivent passer de 0 à 1 pour obtenir la table de $p \rightarrow q$.

Ainsi, le tétraèdre nous offre une vue très riche des relations existant entre les connecteurs binaires.

Dans la suite de l'article, nous nous tournons vers l'investigation d'une structure qui met en œuvre 8 quantifications ou modalités, les 4 modalités aristotéliciennes, et 4 autres qui en sont le mirroir au travers de la réciprocation de Piaget. Cette structure cubique induit aussi des hexagones remarquables, et est obtenue naturellement à partir des 4 opérateurs qui sont à la base d'une vue généralisée de l'analyse formelle de concepts.

3 Analyse formelle de concepts généralisée

En analyse formelle de concepts (18), une relation R, appelée *contexte formel*, est supposée donnée entre un ensemble d'objets Obj et un ensemble de propriétés $Prop$. Soit $R(x)$ l'ensemble des propriétés possédées par un objet x, et $R^{-1}(y)$ l'ensemble des objets ayant la propriété y. Etant donné un ensemble Y de propriétés, quatre ensembles remarquables d'objets peuvent être définis dans ce cadre (12) (la barre supérieure dénote ici la complémentation ensembliste) :

- $R^{\Pi}(Y) = \{x \in Obj | R(x) \cap Y \neq \emptyset\} = \cup_{y \in Y} R^{-1}(y)$, qui est l'ensemble des objets ayant au moins une propriété dans Y ;
- $R^{N}(Y) = \{x \in Obj | R(x) \subseteq Y\} = \cap_{y \notin Y} \overline{R^{-1}(y)}$, qui est l'ensemble des objets ayant aucune propriété en dehors de Y ;
- $R^{\Delta}(Y) = \{x \in Obj | R(x) \supseteq Y\} = \cap_{y \in Y} R^{-1}(y)$, qui est l'ensemble des objets ayant en commun toutes les propriétés dans Y (et en ayant éventuellement d'autres) ;
- $R^{\nabla}(Y) = \{x \in Obj | R(x) \cup Y \neq Prop\} = \cup_{y \notin Y} \overline{R^{-1}(y)}$, qui est l'ensemble des objets à qui il manque au moins une propriété en dehors de Y.

En fait, l'analyse formelle de concepts (18) standard exploite seulement la 3eme fonction d'ensembles. Cette dernière est suffisante pour définir un *concept formel* comme une paire constituée de son *extension* X et de son *intension* Y telle que $R^{\Delta}(Y) = X$ et $R^{-1\Delta}(X) = Y$, où $(X, Y) \subseteq Obj \times Prop$, et $R^{-1\Delta}(X)$ est défini de la même manière : $\{y \in Prop | R^{-1}(y) \supseteq X\}$. On peut montrer qu'un concept formel est une paire (X, Y) maximale telle que $X \times Y \subseteq R$. Par ailleurs, il a été récemment mis en évidence (10) que les paires (X, Y) telles que $R^{N}(Y) = X$ et $R^{-1N}(X) = Y$ caractérisent des sous-contextes indépendants, au sens où ils ne partagent ni objets ni propriétés. Un tel sous-contexte formel (X, Y) est tel que $(X \times Y) \cup (\overline{X} \times \overline{Y}) \supseteq R$.

On peut montrer (12) que le 4-uplet des valeurs de vérité des 4 énoncés $x \in R^{\Pi}(Y)$, $x \in R^{N}(Y)$, $x \in R^{\Delta}(Y)$, et $x \in R^{\nabla}(Y)$ est nécessaire afin de caractériser complé-

FIGURE 6 – Carré induit par les 4 opérateurs définis dans le cadre de l'analyse formelle de concepts

tement les positions relatives possibles des ensembles $R(x)$ et Y. Il est clair que les 4 expressions ci-dessus sont sous-tendues par les expressions logiques suivantes :
- $R^{\Pi}(Y) : \exists y, y \in R(x) \wedge y \in Y$;
- $R^{N}(Y) : \forall y, y \in R(x) \rightarrow y \in Y$;
- $R^{\Delta}(Y) : \forall y, y \notin R(x) \rightarrow y \notin Y$;
- $R^{\nabla}(Y) : \exists y, y \notin R(x) \wedge y \notin Y$.

Remarquons que les deux premières expressions $\forall y, y \in R(x) \rightarrow y \in Y$, $\exists y, y \in R(x) \wedge y \in Y$, forment un carré des oppositions avec les expressions $\forall y, y \in R(x) \rightarrow y \notin Y$ et $\exists y, y \in R(x) \wedge y \notin Y$, correspondant respectivement à $R^{N}(\overline{Y})$, et à $R^{\Pi}(\overline{Y})$. Il s'agit de la face frontale du cube des oppositions de la Figure 1, c'est-à-dire la face AIEO.

De même, les deux dernières expressions $\forall y, y \notin R(x) \rightarrow y \notin Y$, $\exists y, y \notin R(x) \wedge y \notin Y$ forment un autre carré des oppositions avec les expressions $\forall y, y \notin R(x) \rightarrow y \in Y$, et $\exists y, y \notin R(x) \wedge y \in Y$ correspondant respectivement à $R^{\Delta}(\overline{Y})$, et à $R^{\nabla}(\overline{Y})$. Il s'agit de la face de derrière du cube des oppositions de la Figure 1, c'est-à-dire la face aieo.

Cependant les expressions logiques qui sont à la base respectivement de $R^{\Pi}(Y)$, $R^{N}(Y)$, $R^{\Delta}(Y)$, et $R^{\nabla}(Y)$, à savoir
- $\forall y, y \in \overline{R(x)} \rightarrow y \in \overline{Y} (\Longleftrightarrow \forall y, y \in \overline{Y} \rightarrow y \in \overline{R(x)})$
- $\forall y, y \in \overline{R(x)} \rightarrow y \in \overline{Y} (\Longleftrightarrow \forall y, y \in Y \rightarrow y \in R(x))$
- $\exists y, y \in \overline{R(x)} \wedge y \in \overline{Y} (\Longleftrightarrow \exists y, y \in Y \wedge y \in \overline{R(x)})$
- $\exists y, y \in \overline{R(x)} \wedge y \in \overline{Y} (\Longleftrightarrow \exists y, y \in \overline{Y} \wedge y \in \overline{R(x)})$

où $R(x)$ et Y jouent des rôles symétriques, et peuvent aussi être organisées en un carré des oppositions d'un autre type, comme indiqué sur la Figure 5 (où les lignes épaisses pointillés correspondent à la réciprocation) : Cela correspond à la face du côté gauche du cube des oppositions de la Figure 1, c'est-à-dire à la face AIai.

En pratique, on suppose qu'à la fois $R(x) \neq \emptyset$ et $R(x) \neq Prop$, i.e. tout objet doit avoir au moins une propriété dans $Prop$, mais aucun objet n'a toutes les propriétés de $Prop$. Cette bi-normalisation évite les problèmes d'"import existentiel" puisque ni

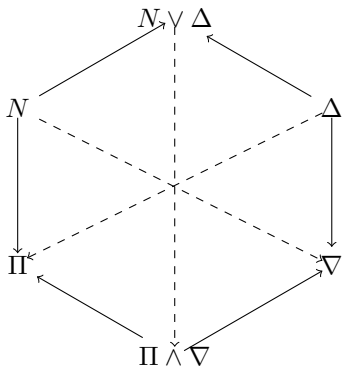


FIGURE 7 – Hexagone induit par les 4 opérateurs à la base de l'analyse formelle de concepts et de la théorie des possibilités

$R(x)$ ni $\overline{R(x)}$ peuvent être vides.

Il est aussi à noter que le principe de construction d'un hexagone à partir d'un carré en ajoutant les deux sommets $Y = I \wedge O$ et $U = A \vee E$ fait encore sens ici. En effet, on peut montrer (12) que sous l'hypothèse de bi-normalisation, la relation d'inclusion suivante est satisfaite :

$$R^N(Y) \cup R^\Delta(Y) \subseteq R^\Pi(Y) \cap R^\nabla(Y).$$

On obtient l'hexagone de la Figure 6. Cet hexagone diffère sémantiquement de celui de la Figure 2, ce qui est normal, puisque l'hexagone de la Fig. 6 n'est pas obtenu à partir d'un carré des oppositions standard. Comme on peut le voir, toutes les lignes sont orientées, y compris les diagonales, et expriment des implications.

L'introduction des opérateurs $R^\Pi(Y)$, $R^N(Y)$, et $R^\nabla(Y)$ (12), au côté de l'opérateur de base $R^\Delta(Y)$ de l'analyse formelle de concepts, a été motivée par un parallèle avec la théorie des possibilités (14), où il existe 4 fonctions d'ensembles de base habituellement dénotées Π, N, Δ, et ∇ qui sont liées directement à des idées d'intersection non-vide ou d'inclusion. Ces fonctions d'ensembles sont graduelles, puisqu'elles prennent leur valeurs dans l'intervalle unité $[0, 1]$. Cependant, restreintes à $\{0, 1\}$, elles correspondent à un système particulier de modalités qui a été étudié dans (13). Dans la prochaine section, nous rappelons brièvement les traits caractéristiques des 4 fonctions d'ensembles de la théorie des possibilités et leur sémantique en tant que modalités, et nous montrons comment elles peuvent être organisées en un cube des oppositions (avec 4 autres fonctions d'ensembles qui leur sont reliées).

4 Théorie des possibilités

La théorie des possibilités a été proposée par Lotfi Zadeh (30) comme une nouvelle théorie pour traiter l'incertain en relation avec des éléments d'information exprimés linguistiquement. Le cadre initial proposé était essentiellement basé sur des mesures de

possibilité max-décomposables. Il est intéressant de remarquer qu'une théorie similaire avait été proposée auparavant par l'économiste anglais G. L. S. Shackle (26) sur la base de motivations complètement différentes ; l'idée de base en est la notion de degré de surprise, qui formellement s'avére être un degré d'impossibilité.

En théorie des possibilités, l'information disponible est représentée au moyen d'une distribution de possibilité. C'est une fonction, en général dénotée π, de l'univers du discours U dans l'intervalle unité $[0, 1]$. La fonction π range les valeurs potentielles d'une variable décrivant les états possibles du monde considéré, selon leur plausibilité. La valeur $\pi(u)$ est interprétée comme la possibilité que cette variable soit égale à u. Une information précise correspond à une situation où $\exists u^*, \pi(u^*) = 1$, et $\forall u \neq u^*, \pi(u) = 0$, tandis que l'ignorance complète est représentée par $\forall u \in U, \pi(u) = 1$. Dans le cas général, π prend des valeurs intermédiaires de degrés de possibilité. Cependant, il est intéressant de considérer aussi le cas particulier binaire où $\forall u \in U, \pi(u) \in \{0, 1\}$. Dans ce cas, π n'est rien d'autre que la fonction caractéristique d'un sous-ensemble de U, qu'on dénotera E (comme "évidence") dans la suite. Cet ensemble E correspond à l'ensemble $R(x)$ (ou $R^{-1}(y)$) en analyse formelle de concepts. L'hypothèse de binormalisation signifie pour E que $E \neq \emptyset$ et que $E \neq U (\Leftrightarrow \overline{E} \neq \emptyset)$.

Soit A un sous-ensemble de l'univers du discours U. Nous rappelons maintenant le système complet des 4 fonctions d'ensembles qui sous-tendent la théorie des possibilités, et leur propriétés caractéristiques :

- i) La *mesure de possibilité (faible)* (ou possibilité potentielle)

$$\Pi(A) = \max_{u \in A} \pi(u)$$

  évalue à quel point il est possible que l'état du monde soit dans A. Quand π se réduit à E, $\Pi(A) = 1$ si $A \cap E \neq \emptyset$, ce qui exprime la cohérence de l'évènement A avec l'évidence E, et $\Pi(A) = 0$ sinon. Les mesures de possibilité sont caracterisées par la propriété suivante de décomposabilité :

$$\Pi(A \cup B) = \max(\Pi(A), \Pi(B))$$

- ii) La *mesure* duale de *nécessité (forte)* (ou nécessité "garantie")

$$N(A) = \min_{u \notin A} 1 - \pi(u) = 1 - \Pi(\overline{A})$$

  évalue à quel point il est certain (nécessairement vrai) que l'état du monde est dans A. Quand π se réduit à E, $N(A) = 1$ si $E \subseteq A$, ce qui exprime que l'évidence E implique A, et $N(A) = 0$ sinon. La dualité de N par rapport à Π exprime que A est d'autant plus certain que l'évènement opposé $\overline{A}$ est impossible. Les mesures de nécessité sont caractérisées par la propriété suivante de décomposabilité :

$$N(A \cap B) = \min(N(A), N(B))$$

- iii) La mesure de possibilité *(forte)* (ou *possibilité "garantie"*)

$$\Delta(A) = \min_{u \in A} \pi(u)$$

évalue à quel point *toutes* les valeurs dans A sont possibles pour l'état du monde, étant donné l'évidence E. Quand π se réduit à E, $\Delta(A) = 1$ si $A \subseteq E$, et $\Delta(A) = 0$ sinon. Les mesures de possibilité forte sont caractérisées par la propriété suivante :

$$\Delta(A \cup B) = \min(\Delta(A), \Delta(B))$$

.

– iv) La mesure duale de nécessité *(faible)* (ou nécessité potentielle)

$$\nabla(A) = \max_{u \notin A} 1 - \pi(u) = 1 - \Delta(\overline{A})$$

évalue à quel point il y a une valeur en dehors de A qui est impossible [7]. Quand π se réduit à E, $\nabla(A) = 1$ si $A \cup E \neq U$, et $\nabla(A) = 0$ sinon. Les mesures de nécessité faible sont caractérisées par la propriété suivante :

$$\nabla(A \cap B) = \max(\nabla(A), \nabla(B))$$

La possibilité faible et la nécessité forte sont des fonctions croissantes par rapport à l'inclusion ensembliste (comme le sont les mesures de probabilité), tandis que les deux autres fonctions d'ensembles sont décroissantes. Les deux premières mesures sont naturellement liées à une expression de l'information négative où on exprime ce qui est (non)-impossible, tandis que la mesure de possibilité forte (tout comme sa duale) est associée à une expression de l'information positive ; voir (3) pour une discussion détaillée de cette bipolarité. Sous l'hypothèse que la distribution de possibilité est bi-normalisée, i.e. ici $\exists u^*, \pi(u^*) = 1$, et $\exists u^{**}, \pi(u^{**}) = 0$, une contrepartie de l'inclusion existant en analyse formelle de concepts tient :

$$\forall A, \max(\Delta(A), N(A)) \leq \min(\Pi(A), \nabla(A)).$$

Ces 4 fonctions d'ensembles qui expriment des modalités Π, N, Δ, et ∇ dans le cadre de la théorie des possibilités, ou dans celui des opérateurs correspondants en analyse formelle de concepts, peuvent être organisées en un cube d'oppositions, de concert avec les modalitées $\neg\Pi$, $\neg N$, $\neg\Delta$, et $\neg\nabla$ qui sont associées aux précédentes par négation et complémentation respectivement ; voir la Figure 7.

Les expressions logiques à la base des opérateurs introduits en analyse formelle de concepts, ou de l'évaluation des différentes modalités en théorie des possibilités binaire, correspondent clairement aux sommets du cube des oppositions construit au début de cet article (voir Figure 1). Notons cependant que le cube de la Figure 7 n'est pas disposé de la même façon que dans la Figure 1 afin de rendre la face AIai plus visible. De plus, il est dupliqué afin de mieux montrer différentes relations entre les sommets, que nous allons commenter maintenant :

– Les lignes horizontales en pointillés serrés de la face arrière correspondent à une relation de réciprocation. En théorie des possibilités, cela revient à changer π en

7. Comme dans le cas du sommet "sans nom" (O) (4) dans le carré des oppositions classique, il est notable que cette 4eme fonction d'ensembles est moins facile à interpréter que les trois autres.

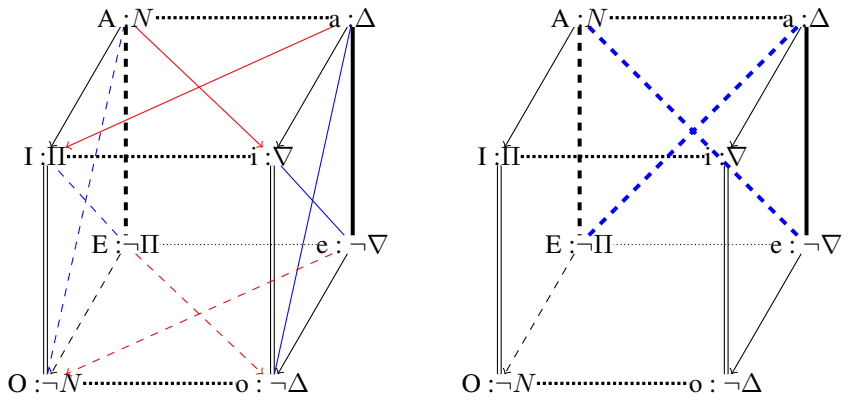


FIGURE 8 – Cube des opérateurs de l'analyse formelle de concepts / de la théorie des possibilités

$1 - \pi$, et A en $\overline{A}$. Cela correspond à renverser le sens de l'inclusion entre les ensembles A et E, et à interpréter les liens entre N et Δ comme l'opposition entre nécessité et suffisance. Il est clair que le groupe de transformations de Piaget s'applique encore ici, comme dans la Figure 1.

– Les lignes verticales épaisses de la face arrière relient des contraires : la nécessité N et l'impossibilité $\neg\Pi$, la possibilité garantie Δ et le manque de certitude potentielle $\neg\nabla$. Elles sont complétées par deux diagonales épaisses (seulement visibles sur la face arrière du cube de droite) qui relient aussi des contraires : la possibilité forte Δ et l'impossibilité $\neg\Pi$ d'une part, et la nécessité N et le manque de certitude faible $\neg\nabla$ d'autre part.
– On peut voir les contradictoires aux extrémités des lignes diagonales non-orientées sur les deux faces latérales du cube de gauche : Π et $\neg\Pi$, N et $\neg N$, Δ et $\neg\Delta$, ∇ et $\neg\nabla$.
– Les 4 arêtes du cube dirigées de l'arrière à l'avant expriment des implications : ce qui est certain est possible ($N \leq \Pi$), ce qui est garanti possible est potentiellement certain ($\Delta \leq \nabla$). De manière équivalente, ce qui est impossible ne peut être certain ($\neg\Pi \leq \neg N$), et ce qui n'est pas potentiellement certain n'est pas garanti possible ($\neg\nabla \leq \neg\ \Delta$). de plus, les diagonales des faces du dessus et du dessous du cube de gauche dirigées de l'arrière à l'avant expriment des implications : ce qui est garanti possible est potentiellement possible ($\Delta \leq \Pi$), ce qui est vraiment certain est faiblement certain ($N \leq \nabla$), ce qui n'est pas potentiellement possible n'est pas garanti possible ($\neg\Pi \leq \neg\Delta$), ce qui n'est pas faiblement certain n'est pas vraiment certain $\neg\nabla \leq \neg\ N$).
– Les lignes verticales doubles de la face de devant relient des sous-contraires : possibilité Π et manque de certitude garantie $\neg N$, nécessité faible ∇ et manque de possibilité garantie $\neg\Delta$.

Ainsi, les interrelations entre les 8 modalités de la théorie des possibilités (ou entre les opérateurs correspondants en analyse formelle de concepts), peuvent être complètement

décrites sur le cube des oppositions qui a été introduit, et donnent par ailleurs naissance au nouvel hexagone de la Figure 6.

5 Conclusion

Depuis très longtemps, le carré des oppositions a été considéré comme le point de départ de l'étude des syllogismes. En distinguant les contraires des contradictoires (et des sous-contraires), et en se basant sur deux manières différentes d'introduire la négation dans une assertion, le carré des oppositions est au cœur de l'expression des capacités humaines en matière de raisonnement, comme le confirme sa relation avec le groupe de transformations de Piaget, et joue un rôle important dans la catégorisation conceptuelle des situations comme l'a souligné Blanché. L'article a mis en évidence que deux théories modernes développées depuis trente ans pour analyser les relations entre objets et propriétés, et pour modéliser l'incertitude épistemique respectivement, avaient aussi leurs racines dans le carré des oppositions.

Plus précisément, l'article a introduit un cube des oppositions (déjà en puissance dans les écrits de De Morgan) dont les faces montrent différents types de carrés des oppositions et qui est fermé par les transformations du groupe de Piaget. On peut disposer sur ce cube toutes les modalités à l'œuvre en théorie des possibilités bipolaire, ou tous les opérateurs de l'analyse formelle de concepts. De plus ses faces peuvent être étendues à des hexagones de différents types.

Par ailleurs, l'article a aussi souligné que l'hexagone de Sesmat-Blanché est obtenu dès qu'on considère une tri-partition de l'univers du discours, tandis qu'une quadri-partition nous permet de décrire les interrelations entre les16 connecteurs binaires au sein d'une structure tétraédrale. Ceci suggère d'autres développements. Mentionnons-en quelques-uns. Dans la construction de l'hexagone, si A, B, et C forment une partition ordonnée (supposons B entre A et C) d'un univers ordonné du discours U, alors A et C représentent des antonymes, et deux négations involutives sont clairement à l'œuvre : l'une échangeant A et C, et laissant B inchangé, et l'autre reflétant la complémentation d'ensembles. Une telle idée de travailler avec deux types de négation a fait aussi l'objet d'une discussion d'un point de vue géometrique (29), sans référence au carré des oppositions, mais en considérant la négation cyclique de la logique multi-valuée de Post ainsi que la négation dans la logique à 3 valeurs de Kleene et dans la logique à 4 valeurs de Belnap. Notons que U peut être n'importe quoi, un domaine d'attribut par exemple, aussi bien qu'un ensemble de sous-ensembles de valeurs de vérité. Ce dernier cas quand $U = \{\{T\}, \{F\}, \{T, F\}\}$ est d'un intérêt particulier pour décrire les états compatibles avec un état d'information incomplète (le sous-ensemble $\{T, F\}$ corrrespond à la situation où on ne peut pas prouver à partir de l'information disponible si un énoncé est vrai (T) ou s'il est faux (F), le singleton $\{T\}$ (resp. $\{F\}$) correspond au cas où on peut prouver que l'énoncé est vrai (resp. faux) (11).

Une autre direction de recherche, serait de considérer l'extension de la représentation tétraédrale à des logiques multivalentes. Pour ce faire, on peut s'inspirer de travaux

existants (17) sur la version graduelle du système des quatre relations strictes à la base de la comparaison des objets. La logique de Lukasiewicz semblerait de ce point de vue un candidat intéressant puisque les lois du tiers-exclu et de non-contradiction y sont préservées. Par exemple, si on considère l'arête $(p \wedge q) - q - (\neg p \wedge q)$ du tétraèdre de la Figure 4, la vérité de p diminuerait en allant de $p \wedge q$ à q, et alors sa fausseté augmenterait en passant de q à $\neg p \wedge q$.

Au delà d'un intérêt historique évident, la perspective proposée ici permet une vision unifiée de différentes théories, et peut aussi aider à un enrichissement conceptuel mutuel de ces théories. Plus généralement, il est important de mieux comprendre les structures d'opposition qui ne sont pas basées sur de simples dichotomies, et qu'on peut rencontrer dans de nombreux domaines de l'intelligence artificielle relatifs au raisonnement, à la décision, et à l'argumentation, notamment.

Références

[1] Anonyme. Syllogisme. In : Encyclopédie ou Dictionnaire Raisonné des Sciences, des Arts et des Métiers, de D. Diderot et J. (le Rond) d'Alembert. 1751-1772. http ://portail.atilf.fr/encyclopedie/Formulaire-de-recherche.htm

[2] N. D. Belnap. A useful four-valued logic. In J.M. Dunn et G. Epstein, editors, Modern UsesMultiple-Valued Logic. D.Reidel, Dordrecht, The Netherlands, 837(1977).

[3] S. Benferhat, D. Dubois, S. Kaci, H. Prade. Modeling positive et negative information in possibility theory. Int. J. of Intelligent Systems, 23,1094–1118, 2008.

[4] J.-Y. Béziau. New light on the square of oppositions et its nameless corner. Logical Investigations, 10, 218–233, 2003.

[5] R. Blanché. Quantity, modality, et other kindred systems of categories. Mind, LXI(243), 369–375, 1952.

[6] R. Blanché. Sur l'opposition des concepts. Theoria, 19, 89–130,1953.

[7] R. Blanché. Structures Intellectuelles. Essai sur l'Organisation Systématique des Concepts. Vrin, Paris. 1966.

[8] G. Boole. An Investigation of *The Laws of Thought* on which are founded The Mathematical Theories of Logic et Probabilities. Chap. XV : The Aristotelian logic et its modern extensions, examined by the method of its treatise. Macmillan, 1854. Reprinted by Dover, NewYork, 1958.

[9] A. De Morgan. On the structure of the syllogism. Trans. Camb. Phil. Soc., VIII, 379–408, 1846. Reprinted in : On the Syllogism et Other Logical Writings, (P. Heath, ed.), Routledge et Kegan Paul, London, 1966.

[10] Y. Djouadi, D. Dubois, H. Prade. Possibility theory et formal concept analysis : Context decomposition et uncertainty handling. In : Computational Intelligence for Knowledge-Based Systems Design, Proc.13th Inter. Conf. on Information Processing et Management of Uncertainty (IPMU 2010), (E. Hüllermeier, R. Kruse et F. Hoffmann, eds.), Dortmund, June 28 – July 2, Springer, LNCS 6178, 260–269, 2010.

[11] D. Dubois. On ignorance and contradiction considered as truth-values. Logic Journal of the IGPL, 16(2) : 195–216, 2008.
[12] D. Dubois, F. Dupin de Saint-Cyr, H. Prade. A possibility-theoretic view of formal concept analysis. Fundamentae Informaticae, 75, 195–213, 2007.
[13] D. Dubois, P. Hajek, H. Prade. Knowledge-driven versus data-driven logics. J. of Logic, Language, et Information, 9, 65–89, 2000.
[14] D. Dubois, H. Prade. Possibility theory : qualitative et quantitative aspects. In : Quantified Representation of Uncertainty et Imprecision, (D. Gabbay, P. Smets, eds.), Handbook of Defeasible Reasoning et Uncertainty Management Systems, Kluwer Acad. Publ., Vol.1, 169–226, 1998.
[15] D. Dubois, H. Prade. Squares of opposition in generalized formal concept analysis et possibility theory. Hand Book of the 2nd World Congress on the Square of Opposition, (J.-Y. Béziau, K. Gan-Krzywoszyńska, eds.), Corte, Corsica, June 17-20, 2010, Università di Corsica - Pasquale Paoli, p. 42.
[16] S. Dupleix. De l'opposition des énonciations. Book IV, chap 10, in : La Logique ou Art de Discourir et de Raisonner. 1607. Reprinted by Fayard, Paris, 1984.
[17] J. Fodor, M. Roubens, Fuzzy Preference Modelling et Multicriteria Decision Support. Kluwer Acad. Publ., Dordrecht, 1994.
[18] B. Ganter et R. Wille. Formal Concept Analysis. Mathematical Foundations. Springer Verlag, 1999.
[19] W. H. Gottschalk. The theory of quaternality. J. Symbolic Logic. 18, 193–196, 1953.
[20] A. Moretti. The Geometry of Logical Opposition. PhD dissertation, University of Neuchâtel, Switzerland, March 2009. http ://alessiomoretti.perso.sfr.fr/NOTMorettiPhD2009GeometryLogicalOpposition.pdf
[21] T. Parsons. The traditional square of opposition. The Stanford Encyclopedia of Philosophy (Fall 2008 Edition), E. N. Zalta (ed.), URL = http ://plato.stanford.edu/archives/fall2008/entries/square/.
[22] R. Pellissier. 2-opposition et the topological hexagon. In : New Perspectives on the Square of Opposition, (Béziau J.-Y. et Payette G., eds.), Peter Lang, Bern, 2010.
[23] J. Piaget. Traité de Logique. Essai de logistique opératoire. Armand Colin, Paris, 1949.
[24] J. Piaget. Essai sur les Transformations des Opérations Logiques : Les 256 opérations ternaires de la logique bivalente des propositions. Presses Universitaires de France, Paris, 1952.
[25] M. Roubens, Ph. Vincke. Preference Modeling. LNEMS 250, Springer Verlag, Berlin, 1985.
[26] G. L. S. Shackle. Decision, Order, et Time in Human Affairs. Cambridge University Press, UK, 1961.
[27] A. Sesmat. Logique II : Les Raisonnements, la Logistique. Hermann, Paris. 1951.
[28] H. Smessaert. On the 3D visualisation of logical relations", Logica Universalis, 3, 303–332, 2009.
[29] A. C. Varzi, M. Warglien. The geometry of negation. J. of Applied Non-Classical Logics, 13, 9-19, 2003.
[30] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets et Systems, 1, 3–28, 1978.

Revisiting Numerical Pattern Mining with Formal Concept Analysis

Mehdi Kaytoue[1], Sergei O. Kuznetsov[2], and Amedeo Napoli[1]

[1] Laboratoire lorrain de recherche en informatique et ses applications (LORIA)
Campus Scientifique, B.P. 70239 – Vandœuvre-lès-Nancy – France

[2] Higher school of economics, State University (HSE)
Pokrovskiy Bd. 11 – 109028 Moscow – Russia

Résumé : In this paper, we investigate the problem of mining numerical data in the framework of Formal Concept Analysis. The usual way is to use a scaling procedure –transforming numerical attributes into binary ones– leading either to a loss of information or of efficiency, in particular w.r.t. the volume of extracted patterns. By contrast, we propose to directly work on numerical data in a more precise and efficient way, and we prove it. For that, the notions of closed patterns, generators and equivalent classes are revisited in the numerical context. Moreover, two original algorithms are proposed and used in an evaluation involving real-world data, showing the predominance of the present approach.

1 Introduction

In this paper, we investigate the problem of mining numerical data. This problem arises in many practical situations, e.g. analysis of gene and transcriptomic data in biology, soil characteristics and land occupation in agronomy, demographic data in economics, temperatures in climate analysis, etc. We introduce an original framework for mining numerical data based on advances in itemset mining and in Formal Concept Analysis (FCA, [1]), respectively condensed representations of itemsets and pattern structures in FCA [2]. The mining of frequent itemsets in binary data, considering a set of objects and a set of associated attributes or items, is studied for a long time and usually involves the so-called "pattern flooding" problem [3]. A way of dealing with pattern flooding is to search for equivalence classes of itemsets, i.e. itemsets shared by the same set of objects (or having the same image). For an equivalence class, there is one maximal itemset, which corresponds to a "closed set", and possibly several minimal elements corresponding to "generators" (or "key itemsets"). From these elements, families of association rules can be extracted [3]. These particular itemsets are also related to FCA , where a concept lattice is built from a binary context and where formal concepts are closed sets of objects and attributes.

The present work is rooted both in FCA and pattern mining with the objective of extracting interval patterns from numerical data. Our approach is based on "pattern

structures" where complex descriptions can be associated with objects. In [4], in the context of gene expression data mining, we introduced pattern structures for numerical data, and showed how to extract closed interval patterns : an interval pattern is a vector of intervals, each dimension corresponding to a range of values of a given attribute ; it is closed when composed of the smallest intervals characterizing a same set of objects.

We here complete and extend this first attempt. Considering numerical data, some general characteristics of equivalence classes remain, e.g. one maximal element which is a closed pattern and possibly several generators which are minimal patterns w.r.t. a subsumption relation defined on patterns. We show that directly extracting patterns from numerical is more efficient using pattern structures than working with binary data and associated scaling procedures. We also provide a semantic to interval patterns in the Euclidean space, and design and experiment algorithms to extract frequent closed interval patterns and their generators.

The problem of mining patterns in numerical data is usually referred as quantitative itemset/association rule mining [5]. Generally, an appropriate discretization splits attribute ranges into intervals maximizing some interest functions, e.g. support, confidence. However, none of these works discusses the notion of equivalence classes, closed patterns, and generators, and this is one of the originality of the present paper.

The plan of the paper is as follows. In the next section, we introduce the problem of mining numerical data and interval patterns. Then, we recall the basics of FCA and interordinal scaling. We pose a number of questions that we propose to answer using our framework of interval pattern structures. We then detail two original algorithms for extracting closed interval patterns and their generators that we experiment in the last section. Finally, we end the paper in discussing related work and giving perspectives to the present research work. Note also that an extended version of this paper can be found in [6, 7], along with algorithms pseudo-code and a deeper discussion on the usefulness of interval patterns in classification problems and privacy preserving data-mining.

2 Problem definition

We propose a definition of interval patterns for numerical data. Intuitively, each object of a numerical dataset corresponds to a vector of numbers, where each dimension stands for an attribute. Accordingly, an interval pattern is a vector of intervals, where each dimension describes the range of possible values for a given numerical attribute associated with some objects. We only consider finite intervals and that the set of attributes/dimensions is ordered.

Numerical dataset, interval pattern and support. A numerical dataset is given by a set of objects G, a set of numerical attributes M, where attribute $m \in M$ has for range a set of real numbers W_m. $m(g) = w$ means that w is the value of attribute m for object g. An interval pattern is a vector of intervals $d = \langle [a_i, b_i] \rangle_{i \in \{1,...,|M|\}}$ where $a_i, b_i \in W_{m_i}$, and each component corresponds to an attribute following a canonical order on vector dimensions, and $|M|$ denotes the number of attributes. An object g is in the image of an interval pattern $\langle [a_i, b_i] \rangle_{i \in \{1,...,|M|\}}$ when $m_i(g) \in [a_i, b_i]$, $\forall i \in \{1, ..., |M|\}$. The support of d, denoted by $sup(d)$, is the cardinality of the image of d.

	$m_1 \leq 4$	$m_1 \leq 5$	$m_1 \leq 6$	$m_1 \geq 4$	$m_1 \geq 5$	$m_1 \geq 6$	$m_2 \leq 7$	$m_2 \leq 8$	$m_2 \leq 9$	$m_2 \geq 7$	$m_2 \geq 8$	$m_2 \geq 9$	$m_3 \leq 4$	$m_3 \leq 5$	$m_3 \leq 6$	$m_3 \leq 8$	$m_3 \geq 4$	$m_3 \geq 5$	$m_3 \geq 6$	$m_3 \geq 8$
g_1		×	×	×	×		×	×	×	×					×	×	×	×	×	
g_2			×	×	×	×		×	×	×	×		×	×	×	×	×			
g_3	×	×	×	×				×	×	×	×			×	×	×	×	×		
g_4	×	×	×	×					×	×	×	×				×	×	×	×	×
g_5		×	×	×	×			×	×	×	×			×	×	×	×	×		

TABLE 2 – Interordinally scaled context encoding the numerical dataset from Table 1.

Running example. Table 1 is a numerical dataset with objects in $G = \{g_1, ..., g_5\}$, attributes in $M = \{m_1, m_2, m_3\}$. The range of m_1 is $W_{m_1} = \{4, 5, 6\}$, and we have $m_1(g_1) = 5$. Here, we do not consider either missing values or multiple values for an attribute. $\langle [5, 6], [7, 8], [4, 6] \rangle$ is an interval pattern in Table 1, where a vector dimension i corresponds to an attribute $m_i \in M$, with image $\{g_1, g_2, g_5\}$ and support 3. The fact that $a_i, b_i \in W_{m_i}$ in the definition of an interval pattern makes the search space finite.

	m_1	m_2	m_3
g_1	5	7	6
g_2	6	8	4
g_3	4	8	5
g_4	4	9	8
g_5	5	8	5

TABLE 1 – A numerical dataset.

Interval pattern search space. Given a set of attributes $M = \{m_i\}_{i \in \{1, |M|\}}$, the search space of interval patterns is the set D of all interval vectors $\langle [a_i, b_i] \rangle_{i \in \{1, ..., |M|\}}$, with $a_i, b_i \in W_{m_i}$. The size of the search space is given by

$$|D| = \prod_{i \in \{1, ..., |M|\}} (|W_{m_i}| \times (|W_{m_i}| + 1))/2$$

In our example, all possible intervals for m_1 are in $\{[4, 4], [5, 5], [6, 6], [4, 5], [5, 6], [4, 6]\}$. Considering also attributes m_2 and m_3, we have $6 \times 6 \times 10 = 360$ patterns.

The classical problem of "pattern flooding" in data-mining is even worst for numerical data. Indeed, with three attributes, there is only $2^3 = 8$ possible itemsets, compared to the 360 interval patterns in the example with same amount of attributes. A solution widely investigated in itemset-mining relies on condensed representations with so called closed itemsets and generators, e.g. in [3]. Moreover, the analysis of numerical datasets can be considered within the formal concept analysis framework (FCA) [1], which is closely related to itemset-mining [8]. Accordingly, we are interested in adapting the notions of (frequent) closed itemsets and their generators to interval patterns within the FCA framework, and in providing them with appropriate semantics.

3 Interval patterns in FCA

In their seminal book on FCA, [1] define a discretization procedure turning numerical data into binary data called *interordinal scaling*. It allows to encode in binary data any interval of values from a numerical dataset.

Formal concept analysis. FCA starts with a formal context (G, N, I) where G denotes a set of objects, N a set of attributes, or items, and $I \subseteq G \times N$ a binary relation between G and N. The statement $(g, n) \in I$, or gIn, means : "the object g has attribute n". The

two operators $(\cdot)'$ define a Galois connection between the powersets $(\mathfrak{P}(G), \subseteq)$ and $(\mathfrak{P}(N), \subseteq)$, with $A \subseteq G$ and $B \subseteq N : A' = \{n \in N \mid \forall g \in A : gIn\}$ and $B' = \{g \in G \mid \forall n \in B : gIn\}$. A pair (A, B), such that $A' = B$ and $B' = A$, is called a *(formal) concept*, while A is called the $extent$ and B the $intent$.

From an itemset-mining point of view, concept intents correspond to closed itemsets, since $(.)''$ is a closure operator. For any subset $B \subseteq N$, B'' is the largest itemset having the same image. An equivalence class is a set of itemsets with same closure (and same image). A subset $B \subseteq N$ is a generator iff $\nexists C \subset B$ with $C' = B'$. Generators are the smallest elements w.r.t. set inclusion in an equivalence class.

Interordinal scaling. Given a numerical attribute m with range W_m, interordinal scaling builds a binary table with $2 \times |W_m|$ binary attributes. They are denoted by "$m \leq w$" and "$m \geq w$", $\forall w \in W_m$, and called IS-items. An object g has an IS-item "$m \leq w$" (resp. "$m \geq w$") iff $m(g) \leq w$ (resp. $m(g) \geq w$). Applying this scaling to our example gives Table 2. It is possible to apply classical mining algorithms to process this table for extracting itemsets composed of IS-items. These itemsets are called IS-itemsets in the following.

IS-itemsets can be turned into interval patterns, since an IS-item gives a constraint for an attribute m on its range W_m. For example, the IS-itemset $\{m_1 \leq 5, m_1 \leq 6, m_1 \geq 4, m_2 \leq 9, m_2 \geq 7\}$ corresponds to the interval pattern $\langle [4, 5], [7, 9], [4, 8] \rangle$. We have here the interval $[4, 8]$ for attribute m_3, i.e. covering its whole range, since no constraint in the itemset involves m_3. Therefore, mining interval patterns can be considered after discretizing numerical data. However, this scaling builds binary data with a very important amount of binary attributes compared to the original ones, even with our small example. Hence, when original data are very large, the size of the resulting formal context involves hard computations. Accordingly, this raises the following questions : (i) Can we avoid scaling and directly work on numerical data instead of searching for IS-itemsets ? (ii) Can we adapt the notions of condensed representations such as closed patterns and generators for numerical data, and efficiently compute those patterns ? (iii) What would be the semantics that could be provided to closed patterns and generators ?

4 Revisiting numerical pattern mining

In this section, we answer those questions. First, we show that a closure operator can be defined for interval patterns based upon their image. Then, we provide interval patterns with appropriate semantics for defining the notion of equivalence classes of patterns, closed and generator patterns. After discussing why working with interordinal scaling is not acceptable thanks to the semantics of interval patterns, we propose two efficient algorithms for mining closed interval patterns and generators.

4.1 A closure operator for interval patterns

We use the formalism of pattern structures [2], an extension of formal contexts to complex data in FCA. It defines a closure operator from any partially ordered set of object descriptions called patterns.

Formally, let G be a set of objects, $(D, \sqcap)$ be a semi-lattice of object descriptions, and $\delta : G \rightarrow D$ be a mapping : $(G, (D, \sqcap), \delta)$ is called a *pattern structure*. Elements of D are called *patterns*, and ordered as follows $c \sqcap d = c \iff c \sqsubseteq d$. Intuitively, objects in G have descriptions in $(D, \sqcap)$. For example, g_1 in Table 1 has description $\langle [5,5], [7,7], [6,6] \rangle$ where D is the set of all possible interval patterns ordered with $\sqcap$ which is made precise below. Let the two operators $(.)^{\square}$ defined as follows, with $A \subseteq G$ and $d \in (D, \sqcap)$

$$d^{\square} = \{g \in G | d \sqsubseteq \delta(g)\} \qquad A^{\square} = \bigsqcap_{g \in A} \delta(g)$$

These operators form a Galois connection between $(\mathfrak{P}(G), \subseteq)$ and $(D, \sqsubseteq)$. $(.)^{\square\square}$ is a closure operator, meaning that any pattern d such as $d = d^{\square\square}$ is closed.

Interval pattern structures. Interval patterns can be ordered within a meet-semi-lattice when the infimum is defined as follows [4]. Let $c = \langle [a_i, b_i] \rangle_{i \in \{1,...,|M|\}}$, and $d = \langle [e_i, f_i] \rangle_{i \in \{1,...,|M|\}}$ two intervals patterns. Infimum is given by $c \sqcap d = \langle [min(a_i, e_i), max(b_i, f_i)] \rangle_{i \in \{1,...,|M|\}}$. The ordering relation induced by this definition is : $c \sqsubseteq d \iff [e_i, f_i] \subseteq [a_i, b_i], \ \forall i \in \{1, ..., |M|\}$.

Consider now a numerical dataset, e.g. Table 1. $(D, \sqsubseteq)$ is the finite ordered set of all interval patterns. $\delta(g) \in D$ is the pattern associated to an object $g \in G$. Then :

$$\langle [5,6], [7,8], [4,8] \rangle^{\square} = \{g_1, g_2, g_5\}$$
$$\{g_1, g_2, g_5\}^{\square} = \langle [5,6], [7,8], [4,6] \rangle$$

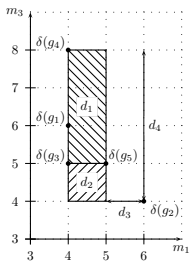
Hence $\langle [5,6], [7,8], [4,8] \rangle$ is closed, its closure being $\langle [5,6], [7,8], [4,6] \rangle$.

4.2 Semantics

An interval pattern d is a $|M|$-dimensional vector of intervals and can be represented by a hyper-rectangle (or rectangle for short) in Euclidean space $\mathbb{R}^{|M|}$, whose sides are parallel to the coordinate axes. This geometrical representation provides a semantics for interval patterns. In formal terms, this interpretation is given by $\mathcal{I} = (\mathbb{R}^{|M|}, (.)^{\mathcal{I}})$ where $\mathbb{R}^{|M|}$ is the interpretation domain, and $(.)^{\mathcal{I}} : D \rightarrow \mathbb{R}^{|M|}$ the interpretation function. Figure 1 gives four interval pattern representations in $\mathbb{R}^2$, with only attributes m_1 and m_3 of our example. The image of d_1 is given by all objects g whose description $\delta(g)$ is included in the rectangle associated with d_1, i.e. the set $\{g_1, g_3, g_4, g_5\}$. We can interpret the closure operator $(.)^{\square\square}$ according to this semantics. The first operator $(.)^{\square}$ applies to a rectangle and returns the set of objects whose description is included in this rectangle. The second operator $(.)^{\square}$ applies to a set of objects and returns the smallest rectangle that contains their descriptions, i.e. the convex hull of their corresponding descriptions.

4.3 Closed interval patterns and generators

Now, we can revisit the notion of equivalence classes of itemsets [3] : an equivalence class of interval patterns is a set of rectangles containing the same object descriptions (based on all rectangles in the searchspace given in Section 2). This enables to define the notions of (frequent) closed interval patterns ((F)CIP) and (frequent) interval pattern generators ((F)IPG), adapted from classical case.



$d_1 = \langle [4,5],[5,8] \rangle$
$d_1^{\square} = \{g_1, g_3, g_4, g_5\}$
$d_2 = \langle [4,5],[4,5] \rangle$
$d_2^{\square} = \{g_3, g_5\}$
$d_3 = \langle [5,6],[4,4] \rangle$
$d_3^{\square} = \{g_2\}$
$d_4 = \langle [6,6],[4,8] \rangle$
$d_4^{\square} = \{g_2\}$

FIGURE 1 – Interval patterns in the Euclidean space.

Equivalence class, (frequent) closed interval pattern and generator. Two interval patterns c and d with same image are equivalent, i.e. $c^{\square} = d^{\square}$: we write $c \cong d$. $\cong$ is an *equivalence relation*, i.e. reflexive, transitive and symmetric. The set of patterns equivalent to a pattern d is denoted by $[d] = \{c | c \cong d\}$ and called the equivalence class of d. A pattern d is *closed*, and noted CIP, if there does not exist any pattern e such as $d \sqsubseteq e$ with $d \cong e$. Dually, a pattern d is a *generator*, and noted GIP, if there does not exist a pattern e such as $e \sqsubseteq d$ with $d \cong e$. Finally, a pattern d is *frequent* if its image has a higher cardinality than a given minimal support threshold $minSup$.

We illustrate these definitions with two dimensional interval patterns and their representation in Figure 1, considering attributes m_1 and m_3. $\langle [4,5],[6,8] \rangle \cong \langle [4,6],[6,8] \rangle$ with image $\{g_1, g_4\}$. $\langle [4,6],[6,8] \rangle$ is not closed as $\langle [4,6],[6,8] \rangle \sqsubseteq \langle [4,5],[6,8] \rangle$, and both patterns have same image, i.e. $\{g_1, g_4\}$. $\langle [4,5],[5,8] \rangle$ is closed. $\langle [4,6],[5,8] \rangle$ and $\langle [4,5],[4,8] \rangle$ are the generators of the closed interval pattern $d_1 = \langle [4,5],[5,8] \rangle$ with image $\{g_1, g_3, g_4, g_5\}$. Among the four patterns in Figure 1, d_1 is the only frequent interval pattern with $minSup = 3$.

Based on the above semantics, an equivalence class is a set of rectangles containing the same set of object descriptions, with a (unique) closed pattern corresponding to the smallest rectangle, and one or several generator(s) that are largest rectangle(s).

4.4 IS-itemsets versus interval patterns

Recall that interordinal scaling allows to build binary data encoding all interval of values from a numerical dataset. Therefore, one may attempt to mine classical closed itemsets and generators in these data with existing data-mining algorithms. Here we show why this should be avoided.

Local redundancy of IS-itemsets. Extracting all IS-itemsets in our example (from Table 2) gives $31,487$ IS-itemsets. This is surprising since there are 360 possible interval patterns. In fact, many IS-itemsets are locally redundant. For example, $\{m_1 \leq 5\}$ and $\{m_1 \leq 5, m_1 \leq 6\}$ both correspond to interval pattern $\langle [4,5],[7,9],[4,8] \rangle$: the constraint $m_1 \leq 6$ is redundant w.r.t. $m_1 \leq 5$ on the set of values W_{m_1}. Hence there is no 1-1-correspondence between IS-itemsets and interval patterns. However, there is a 1-1-correspondence between closed IS-itemsets and CIP only [4]. Later we show that local redundancy of IS-itemsets makes the computation of closed sets very hard.

Global redundancy of IS-itemset generators. Since IS-itemset generators are the smallest itemsets, they do not suffer of local redundancy. However, we remark another kind of redundancy, called global redundancy : it happens that two different and incomparable IS-itemset generators correspond to two different interval pattern generators, but one subsuming the other. In Table 2, both IS-itemsets $N_1 = \{m_1 \leq 4, m_3 \leq 5\}$ and $N_2 = \{m_1 \leq 4, m_3 \leq 6\}$, with same image $\{g_3\}$ are generators, i.e. there does not exist a subset of these itemsets with same image. However, their corresponding interval pattern are respectively $c = \langle [4,4], [7,9], [4,5] \rangle$ and $d = \langle [4,4], [7,9], [4,6] \rangle$ and we have $d \sqsubseteq c$, while $c^{\Box} = d^{\Box}$, hence c is not an interval pattern generator.

4.5 Algorithms

We detail a depth-first enumeration of interval patterns, starting with the most frequent one. Based on this enumeration, we design the algorithms *MinIntChange* and *MinIntChangeG* for extracting respectively frequent closed interval patterns (FCIP) and frequent interval pattern generators (FIPG).

Interval pattern enumeration. The basic idea of pattern generation lies in *minimal changes* for generating the direct subsumers of a given pattern. For example, two minimal changes can be applied to $[4,6]$. The first consists in replacing the right border with the value of W_{m_1} immediately lower that 6, i.e. 5, for generating the interval $[4,5]$. The second consists in repeating the same operation for the left border, generating the interval $[5,6]$. Repeating these two operations allows to enumerate all elements of $(D_{m_1}, \sqcap)$. A right minimal change is defined formally as, given $a, b, v \in W_m$, $a \neq b$, $mcr([a,b]) = [a,v]$ with $v < b$ and $\nexists w \in W_m$ s.t. $v < w < b$ while a left minimal change $mcl([a,b])$ is formally defined similarly. Minimal changes give direct next subsumers and implies a monotonicity property of frequency, i.e. support of $[a,v]$ is less than or equal to support of $[a,b]$. To avoid generating several times the same pattern, a lectic order on changes, or equivalently on patterns, is defined. After a right change, one can apply either a right or left change ; after a left change one can apply only a left change. Figure 2 shows the depth-first traversal (numbered arrows) of diagram of $(D_{m_1}, \sqcap)$. Backtrack occurs when an interval of the form $[w,w]$ is reached ($w \in W_{m_1}$), or no more change can be applied. Each minimal change can be interpreted in term of an IS-item. For example, if $[a,b]$ corresponds to the IS-itemsets $\{m \geq a, m \leq b\}$ then $mcr([a,b]) = [a,v]$ corresponds to $\{m \geq a, m \leq b, m \leq v\}$, i.e. adding $m \leq v$ to the original IS-itemset. The same applies dually to left minimal changes. These IS-items characterizing minimal changes are drawn on Figure 2. This figure accordingly represents a prefix-tree, factoring out the effort to process common prefixes or minimal changes, and avoiding redundancy problems inherent in interordinal scaling. The generalization to several attributes is straightforward. A lectic order is classically defined on numerical attributes as a lexicographic order, e.g. $m_1 < m_2 < m_3$. Then changes are applied as explained above for all attributes respecting this order, e.g. after applying

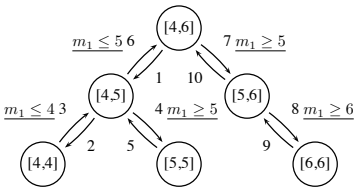


FIGURE 2 – Depth-first traversal of $(D_{m_1}, \sqcap)$.

a change to attribute m_2, one cannot apply a change to attribute m_1.

Extracting FCIP with MintIntChange. The pattern enumeration starts with the minimal pattern w.r.t $\sqsubseteq$ and generates its direct subsumers with lower or equal support. The next problem now is that minimal changes do not necessarily generate patterns with strictly smaller support. Therefore, we should apply changes until a pattern with different support is generated to identify a closed interval pattern (FCIP) but this would not be efficient. We adopt the idea of the algorithm *CloseByOne* [9] : before applying a minimal change, the closure operator $(.)^{\square\square}$ is applied to the current pattern, allowing to skip all equivalent patterns. Indeed, the minimal pattern d w.r.t. $\sqsubseteq$ is closed as it is given by $d = G^{\square}$. Applying a minimal change returns a pattern c with strictly smaller support, since $d \sqsubseteq c$ and d is closed. If c is frequent, we can continue, apply the closure operator and next changes in lectic order, allowing to completely enumerate all FCIP. Since a FCIP may have several different associated generators, it can be generated several times. Still following the idea of *CloseByOne*, a canonicity test can be defined according to lectic order minimal changes. Consider a pattern d generated by a change at attribute $m_j \in M$. Its closure is given by $d^{\square\square}$. If $d^{\square\square}$ differs from d for some attributes $m_h \in M$ such as $m_h < m_j$, then $d^{\square\square}$ has already been generated : it is not canonically generated, hence the algorithms backtracks.

Extracting FIPG with MintIntChangeG. We now adapt *MinIntChange* to extract FIPG, following a well-known principle in itemset-mining algorithms [10]. For any FCIP d, a minimal change implies that the support of the resulting pattern c is strictly smaller than the support of d. Therefore, c is a good generator candidate of the next FCIP. Accordingly, at each step of the depth-first enumeration a FGIP candidate c is generated from the previous one b, by applying a minimal change characterized by $b^{\square\square}$. Then, each candidate c has to be tested to be a true generator or not. We know that the candidate has no subsumers in its branch with same support. However, it could exist a branch with another FGIP e with same image and resulting from less changes. Considering the lectic order on minimal changes, we use a reverse traversal of the tree (see Figure 2 : 7,8,9,10,1,4,5,2,3,6), as already suggested in the binary case in [10]. Since generators correspond to largest rectangles, i.e. on which the fewest minimal changes have been applied, if c is not a generator, a generator e associated to its equivalence class has already been generated, and c is discarded. To check the existence of e, we look up in an auxiliary data-structure storing already extracted FGIP. Precisely, if the data structure contains a FGIP e with same support than candidate c, such that $e \sqsubseteq c$, c is discarded, and the algorithm backtracks. Otherwise c is declared as a FIPG and stored. We have experimented the *MinIntChangeG* algorithm with two well-known and adapted data structures, a trie and a hashtable.

5 Experiments

We evaluate the performances of the algorithms designed in Java : *MinIntChange*, *MinIntChangeG-h* with auxiliary hashtable and *MinIntChangeG-t* with auxiliary trie. Recalling that closed IS-itemsets and CIP are in 1-1-correspondence, we compare the performance for mining interordinal scaled data with the closed-itemset-mining algo-

rithm *LCMv2* [1]. For studying the global redundancy effect of IS-itemset generators, we use the generator-mining-algorithm *GrGrowth* [2]. Both implementations in C++ are available from the authors. All experiments are conducted on a 2.50Ghz machine with 16GB RAM running under Linux 2.6.18-92.e15. We choose dataset from the Bilkent repository [3], namely Bolts (BL), Basketball (BK) and Airport (AP), AP being worst case where each attribute value is different.

First experiments compare *MinIntChange* for extracting FCIP and *LCMv2* for extracting equivalent frequent closed IS-itemsets in Table 3. Second experiments consist in extracting frequent interval pattern generators (FIPG) with algorithms *MinIntChange-h* and *MinIntChange-t*. We also extract frequent itemset generators (FISG) in corresponding binary data after interordinal scaling with *GrGrowth* for studying the global redundancy effect in Table 3.

Dataset	minSupp	MinIntChange	LCMv2	$\|FCIP\|$
BL	80%	< 50	< 50	1,130
	50%	252	**100**	32,107
	25%	1,215	**1,060**	171,192
	10%	**1,821**	1,950	268975
	1	**1,905**	2,090	272,223
AP	80%	4,595	**1,470**	346,741
	50%	**143,939**	149,580	16,214,345
	25%	**413,805**	899,180	58,373,631
	10%	**506,985**	6,810,125	80,504,566
	1	**517,548**	6,813,591	82,467,124

FIGURE 3 – Execution time for extracting FCIP

In both cases, using binary data is better when the minimal support is high (e.g. 90%). For low supports, a critical issue, our algorithms deliver better execution times. Most importantly, the global redundancy effect discards the use of binary data, e.g. only 1.6% of all FISG are actually FIPG in dataset BL. Finally, the algorithm *MinIntChangeG-t* outperforms *MinIntchangeG-h*. *MinIntChangeG-t* however needs more memory : it stores each closed set of objects as a word in the trie with associated associated FIPGs.

It is very interesting to analyse the compression ability of closed interval patterns and generators. For that, we compare in each dataset the number of those patterns w.r.t. to all possible interval patterns. It gives the ratio of closed (generators) in the whole search space. In both cases, ratio varies between 10^{-7} and 10^{-9}. This means that the volume of useful interval patterns, either closed or generators, is very low w.r.t. the set of all possible interval patterns, justifying our interest in equivalence classes.

1. http://research.nii.ac.jp/~uno/codes.htm
2. http://www.comp.nus.edu.sg/~wongls/projects/pattern-spaces/grgrowth-v1/
3. http://funapp.cs.bilkent.edu.tr/

Dataset	minSupp	GrGrowth	MinIntChangeG-h	MinIntChangeG-t	$\|FIPG\|$	$\|FISG\|$	$\frac{\|FIPG\|}{\|FISG\|}$
BL	90%	< 50	< 50	< 50	176	194	90%
	80%	< 50	< 50	< 50	1,952	2,823	69%
	50%	**150**	1,212	529	66,350	222,088	29%
	25%	**3,432**	27,988	3,893	411,442	3,559,419	11%
	1	123,564	438,214	**24,141**	1,165,824	69,646,301	**1.6%**
BK	90%	< 50	1,268	1,207	67,737	75,058	84%
	85%	**4,565**	26,154	12,139	554,956	799,574	69%
	80%	Untractable	512,126	**107,700**	2,730,812	NA	NA

TABLE 3 – Execution time for extracting FIPG and global redundancy evaluation.

6 Concluding remarks

We discussed the important problem of pattern discovery in numerical data with an original formalization of interval patterns. The classical FCA/itemset-mining settings are adapted accordingly : from a closure operator naturally rise the notions of equivalence classes, closed and generator patterns, and we designed corresponding algorithms. An appropriate semantics of interval patterns shows from a theoretical (redundancy) and practical (computation times) points of view that mining equivalent binary data (encoding all possible intervals) is not acceptable. This is due to the fact the interval patterns are provided with a stronger partial ordering than IS-itemsets (classical set inclusion), hence pattern structures yield significantly less generators w.r.t. their semantics.

In data-mining, closed patterns and their generators are crucial for extracting valid and informative association rules [3], while generators can be preferable to closed patterns following the minimum descriptions length principle. How these notions can be shifted to interval patterns is an original perspective of research rising questions concerning missing values, fault-tolerant patterns, and interestingness measures that are critical issues even in classical itemset mining : although the compression ability of closed interval patterns and generators is spectacular, the number of patterns remains too high for large datasets. However, bringing the problem of numerical pattern mining into well known settings may enable these perspectives of research.

References

[1] B. Ganter and R. Wille. *Formal Concept Analysis*. Springer, 1999.

[2] B. Ganter and S. O. Kuznetsov. Pattern structures and their projections. In *9th Int. Conf. on Conceptual Structures*, LNCS (2120), pages 129–142. Springer, 2001.

[3] Y. Bastide, R. Taouil, N. Pasquier, G. Stumme, and L. Lakhal. Mining frequent patterns with counting inference. *SIGKDD Expl.*, 2(2) :66–75, 2000.

[4] M. Kaytoue, S. O. Kuznetsov, A. Napoli, and S. Duplessis. Mining gene expression data with pattern structures in formal concept analysis. *Inf. Sci.*, 181(10) :1989–2001, 2011.

[5] R. Srikant and R. Agrawal. Mining quantitative association rules in large relational tables. In *ACM SIGMOD Int. Conf. on Management of Data*. ACM, 1996.

[6] M. Kaytoue, S. O. Kuznetsov, and A. Napoli. Pattern Mining in Numerical Data : Extracting Closed Patterns and their Generators. Research Report RR-7416, INRIA, 2010.

[7] M. Kaytoue, S. O. Kuznetsov, and A. Napoli. On pattern discovery in numerical data with formal concept analysis. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2011. To appear.

[8] G. Stumme, R. Taouil, Y. Bastide, N. Pasquier, and L. Lakhal. Computing iceberg concept lattices with titanic. *Data Knowl. Eng.*, 42(2), 2002.

[9] S. O. Kuznetsov and S. A. Obiedkov. Comparing performance of algorithms for generating concept lattices. *J. Exp. Theor. Artif. Intell.*, 14(2-3) :189–216, 2002.

[10] T. Calders and B. Goethals. Depth-first non-derivable itemset mining. In *SIAM International Conference on Data Mining*, 2005.

Rationalisation de bases de croyances pour la fusion en logique propositionnelle

Sébastien Konieczny, Pierre Marquis, Nicolas Schwind

CRIL - CNRS
Université d'Artois
Lens, France
{konieczny,marquis,schwind}@cril.fr

Résumé :
Les opérateurs de fusion de bases de croyances prennent typiquement en compte l'ensemble des modèles de chacune des bases, y compris ceux qui contredisent les contraintes d'intégrité. Dans cet article, nous montrons qu'un tel choix n'est pas adapté à toutes les applications. Nous introduisons la notion de « rationalisation » des bases par rapport aux contraintes d'intégrité. Nous définissons formellement une condition d'indépendance des opérateurs de fusion à différentes formes de rationalisation et nous montrons comment ces conditions interagissent avec les postulats IC standard pour la fusion de croyances. En particulier, nous donnons une caractérisation axiomatique d'un opérateur de fusion satisfaisant un critère d'indépendance à la rationalisation des bases et qui satisfait aussi l'ensemble des postulats IC.

1 Introduction

Les opérateurs de fusion de croyances (Konieczny & Pino Pérez, 2011; Revesz, 1997; Lin, 1996; Liberatore & Schaerf, 1998; Konieczny & Pino Pérez, 2002; Konieczny *et al.*, 2004) ont pour but de définir les croyances d'un groupe d'agents, étant données les croyances de chacun. Des contraintes d'intégrité, représentant des lois physiques ou des normes, sont souvent exploitées lors du processus de fusion. Fusionner des bases de croyances sous contraintes d'intégrité n'est pas trivial. Il existe en effet plusieurs manières de procéder, plus ou moins « rationnelles ». Une caractérisation logique des opérateurs de fusion « raisonnables », exprimée sous la forme d'un ensemble de postulats de rationalité, a été proposée dans la littérature (Konieczny & Pino Pérez, 2002) : les postulats IC. Ces postulats décrivent un ensemble de propriétés logiques que les opérateurs de fusion devraient satisfaire. De tels opérateurs sont appelés opérateurs de fusion IC.

Les opérateurs de fusion de bases de croyances existants prennent typiquement en compte l'ensemble des modèles de chacune des bases, y compris ceux qui contredisent les contraintes d'intégrité. Un tel choix n'est cependant pas adapté à toutes les applications. En particulier, lorsque les contraintes d'intégrité représentent une connaissance

certaine et indéniable à propos du monde (par exemple, des lois physiques), l'utilisation dans le processus de fusion de modèles « incorrects » (i.e., ne satisfaisant pas les contraintes d'intégrité) doit être remise en question. Par exemple, Condotta *et al.* (2009) ont récemment proposé un cadre de fusion d'informations spatiales ou temporelles qualitatives exprimées en logique propositionnelle, dans lequel les contraintes d'intégrité sont utilisées pour encoder les lois spatiales ou temporelles. Ainsi, les modèles « non réalisables » (par exemple, un modèle représentant un ensemble de trois instants t_1, t_2, t_3 de la droite et tel que $t_1 < t_2 < t_3 < t_1$) ne doivent pas être considérés dans le processus de fusion, au sens où ils ne doivent avoir aucun impact sur la base spatiale ou temporelle résultante.

Illustrons le problème à travers un exemple simple. Considérons la situation suivante, où une pièce est munie d'une ampoule électrique et de deux interrupteurs. L'ampoule électrique est allumée uniquement lorsque que les deux interrupteurs sont dans la même position (soit « on », soit « off »). L'ampoule est actuellement allumée. La position de chacun des interrupteurs est inconnue, et l'information disponible à leur sujet est contradictoire : une source d'information prétend que le premier interrupteur est en position « off », alors qu'une autre source d'information prétend que les deux interrupteurs sont en position « on ». Que peut-on déduire de ces informations ?

La réponse à cette question dépend de l'opérateur de fusion utilisé. Considérons par exemple l'opérateur de fusion IC basé sur la distance de Hamming et la somme comme fonction d'agrégation ($\Delta^{d_H,\Sigma}$) ; selon cet opérateur, les modèles de la base fusionnée représentant les croyances du groupe correspondent aux modèles des contraintes d'intégrité les plus proches du profil formé des deux sources d'information, où la distance entre deux modèles est égale au nombre de propositions atomiques sur lesquels ils diffèrent. Deux modèles sont compatibles avec les contraintes d'intégrité, 00 (les deux interrupteurs sont en position « off ») et 11 (les deux interrupteurs sont en position « on »). 00 est à une distance de 0 de la première source (c'est un modèle de la base de croyances correspondante) et à une distance de 2 de la deuxième source. 11 est à une distance de 1 de la première source (puisque pour ce modèle, seul le fait que le deuxième interrupteur est en position « on » contredit l'information fournie par la première source) et à une distance de 0 de la deuxième source. En utilisant la somme comme fonction d'agrégation, seul le deuxième modèle (11) résulte du processus de fusion. Ainsi, les croyances du groupe correspondent au fait que les deux interrupteurs sont en position « on ».

Cependant, le fait que le modèle 11 est à une distance de 1 de la première source provient seulement du fait qu'il est à une distance 1 du modèle 01 de cette source, alors que 01 ne correspond pas à un monde réalisable étant donné les contraintes d'intégrité. Il est alors raisonnable de disqualifier un tel modèle. Si l'on considère à présent que la première source d'information a prétendu que les deux interrupteurs sont en position « off » (l'unique modèle compatible avec ses croyances et les contraintes d'intégrité), alors en utilisant le même opérateur de fusion, il résulte de la base fusionnée le fait que les deux interrupteurs sont soit tous les deux en position « on », soit tous les deux en position « off ». Ce résultat est plus satisfaisant puisqu'il n'y a pas de raison de donner plus de crédit au modèle 11 qu'au modèle 00 lorsqu'une source établit finalement 00 tandis que l'autre établit 11.

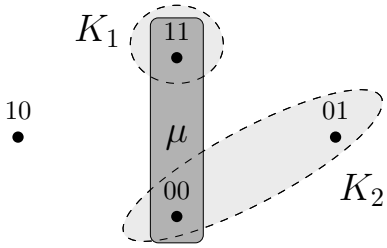


FIGURE 1 – Représentation graphique des contraintes d'intégrité μ et des deux sources (K_1 et K_2).

Cet exemple simple montre que certains opérateurs de fusion IC permettent aux mondes « impossibles » (i.e., ne satisfaisant pas les contraintes d'intégrité) de jouer un rôle dans le processus de fusion, autrement dit d'avoir un impact sur la base fusionnée résultante. Il convient donc de définir une nouvelle propriété pour les opérateurs de fusion, ayant pour but d'imposer à l'opérateur de retourner une base fusionnée équivalente à la base obtenue si chacune des bases initiales est préalablement « rationalisée » par rapport aux contraintes d'intégrité.

Dans la suite, nous proposons trois postulats d'indépendance correspondant à différents principes de rationalisation, et nous étudions la manière dont ces postulats interagissent avec les postulats IC standard pour la fusion de croyances.

Le reste de l'article est organisé comme suit. Dans la section 2, nous introduisons quelques notations et rappelons quelques définitions et résultats concernant les opérateurs de fusion IC. Dans la section 3, nous proposons une définition formelle des propriétés d'indépendance et donnons quelques résultats concernant ces propriétés. Dans la section 4, nous donnons une caractérisation axiomatique de l'opérateur de fusion basé sur la distance drastique et la fonction d'agrégation « somme » ($\Delta^{d_D,\Sigma}$), en termes d'indépendance à la rationalisation des bases. Enfin, nous concluons dans la section 5. Les preuves des propositions sont fournies en annexe.

2 Préliminaires

On considère un langage propositionnel $\mathcal{L}$ défini à partir d'un ensemble fini de variables propositionnelles $\mathcal{P}$ et des connecteurs logiques usuels. $\bot$ (respectivement, $\top$) est la constante booléenne faux (respectivement, vrai).

Une interprétation (ou monde) est une application de $\mathcal{P}$ dans $\{0, 1\}$. L'ensemble des interprétations est noté $\mathcal{W}$. Une interprétation I est un modèle d'une formule $\phi \in \mathcal{L}$, noté $I \models \phi$, si et seulement si elle rend la formule vraie. $mod(\phi)$ dénote l'ensemble des modèles d'une formule ϕ, i.e., $mod(\phi) = \{I \in \mathcal{W} \mid I \models \phi\}$. Soit M un ensemble d'interprétations ; φ_M dénote une formule de $\mathcal{L}$ dont l'ensemble des modèles est M.

Les *contraintes d'intégrité* μ sont représentées par une formule. Une *base* K dénote l'ensemble des croyances d'un agent, c'est un ensemble fini de formules propositionnelles interprétées conjonctivement, ainsi K est identifié à la conjonction de ses éléments.

Un *profil* $\overline{\mathcal{K}} = \langle 1, \ldots, n \rangle$ est un vecteur d'agents impliqués dans un processus de fusion. Un *profil de croyances* $\mathcal{K} = \langle K_1, \ldots, K_n \rangle$ est un vecteur de bases, la base K_i représentant les croyances de l'agent i. Lorsque cela n'est pas nécessaire, on ne fera pas de distinction entre les notions de profil et de profil de croyances, i.e., chaque base est identifiée comme l'agent fournissant cette base, et le terme « profil » sera préféré à « profil de croyances ».

Dans le cadre de ce papier, les agents jouent des rôles équivalents dans le processus de fusion, ainsi un profil $\mathcal{K} = \langle K_1, \ldots, K_n \rangle$ est vu de manière équivalente comme un multi-ensemble $\{K_1, \ldots, K_n\}$. Un profil est dit p-cohérent si toutes les bases du profil sont cohérentes (cette hypothèse est standard mais elle n'est pas admise de manière systématique dans cet article). $\sqcup$ dénote l'union multi-ensembliste et $\equiv$ l'équivalence entre les profils (deux profils de croyances sont équivalents lorsqu'il existe une bijection entre ces deux profils de telle sorte que chaque base du premier profil soit équivalente à son image dans le second profil). $\mathcal{K}^n$ dénote le multi-ensemble où $\mathcal{K}$ apparaît n fois, i.e., $\mathcal{K}^n = \underbrace{\mathcal{K} \sqcup \ldots \sqcup \mathcal{K}}_{n}$. $\bigwedge \mathcal{K}$ dénote la conjonction des bases de croyances de $\mathcal{K}$, i.e., $\bigwedge \mathcal{K} = \bigwedge \{K_i \mid K_i \in \mathcal{K}\}$. Enfin, on note $I \models \mathcal{K}$ lorsque $I \models \bigwedge \mathcal{K}$.

Exemple 1
L'exemple décrit dans l'introduction peut être formalisé de la manière suivante : on a $\mathcal{P} = \{on_s_1, on_s_2\}$ *(où* on_s_1 *(respectivement,* on_s_2*) représente le fait que « le premier interrupteur (respectivement, le second) est en position on ») ;* $\mathcal{K} = \langle K_1, K_2 \rangle$*, avec* $K_1 = \{\neg on_s_1\}$*,* $K_2 = \{on_s_1 \wedge on_s_2\}$ *;* $\mu = on_s_1 \leftrightarrow on_s_2$*.*

Un opérateur de fusion Δ est une application qui associe à une formule μ et un profil $\mathcal{K}$ une nouvelle base $\Delta_\mu(\mathcal{K})$. Nous rappelons à présent les propriétés logiques standard qui décrivent le comportement rationnel d'un opérateur de fusion (Konieczny & Pino Pérez, 2002) :

Définition 1 (Opérateur de fusion IC)
Un opérateur de fusion Δ *est un* opérateur de fusion IC *ssi pour toute formule* μ, μ_1, μ_2*, pour tout profil p-cohérent* $\mathcal{K}, \mathcal{K}_1, \mathcal{K}_2$ *et pour toute base de croyances cohérente* K_1*,* K_2*, il satisfait les postulats suivants :*

(IC0) $\Delta_\mu(\mathcal{K}) \models \mu$ *;*

(IC1) *Si* μ *est cohérente, alors* $\Delta_\mu(\mathcal{K})$ *est cohérente ;*

(IC2) *Si* $\bigwedge \mathcal{K} \wedge \mu$ *est cohérente, alors* $\Delta_\mu(\mathcal{K}) \equiv \bigwedge \mathcal{K} \wedge \mu$ *;*

(IC3) *Si* $\mathcal{K}_1 \equiv \mathcal{K}_2$ *et* $\mu_1 \equiv \mu_2$*, alors* $\Delta_{\mu_1}(\mathcal{K}_1) \equiv \Delta_{\mu_2}(\mathcal{K}_2)$ *;*

(IC4) *Si* $K_1 \models \mu$*,* $K_2 \models \mu$ *et* $\Delta_\mu(\{K_1, K_2\}) \wedge K_1$ *est cohérente, alors* $\Delta_\mu(\{K_1, K_2\}) \wedge K_2$ *est cohérente ;*

(IC5) $\Delta_\mu(\mathcal{K}_1) \wedge \Delta_\mu(\mathcal{K}_2) \models \Delta_\mu(\mathcal{K}_1 \sqcup \mathcal{K}_2)$ *;*

(IC6) *Si* $\Delta_\mu(\mathcal{K}_1) \wedge \Delta_\mu(\mathcal{K}_2)$ *est cohérente, alors* $\Delta_\mu(\mathcal{K}_1 \sqcup \mathcal{K}_2) \models \Delta_\mu(\mathcal{K}_1) \wedge \Delta_\mu(\mathcal{K}_2)$ *;*

(IC7) $\Delta_{\mu_1}(\mathcal{K}) \wedge \mu_2 \models \Delta_{\mu_1 \wedge \mu_2}(\mathcal{K})$ *;*

(IC8) *Si* $\Delta_{\mu_1}(\mathcal{K}) \wedge \mu_2$ *est cohérente, alors* $\Delta_{\mu_1 \wedge \mu_2}(\mathcal{K}) \models \Delta_{\mu_1}(\mathcal{K}) \wedge \mu_2$*.*

Remarquons que les opérateurs de fusion IC considèrent qu'une base incohérente ne fournit aucune information dans le processus de fusion (Konieczny & Pino Pérez, 2002), c'est pourquoi les profils considérés en entrée de tels opérateurs sont supposés p-cohérents.

Un opérateur de fusion IC correspond à un assignement syncrétique (Konieczny & Pino Pérez, 2002) :

Définition 2 (Assignement syncrétique)
Un assignement syncrétique *est une application qui associe à tout profil p-cohérent $\mathcal{K}$ un préordre $\leq_{\mathcal{K}}$ sur les interprétations*[1] *et telle que pour tout profil p-cohérent $\mathcal{K}, \mathcal{K}_1, \mathcal{K}_2$ et pour toute base de croyances cohérente $K_1, K_2, \leq_{\mathcal{K}}$ satisfait les conditions suivantes :*

(1) *Si $I \models \mathcal{K}$ et $J \models \mathcal{K}$, alors $I \simeq_{\mathcal{K}} J$;*

(2) *Si $I \models \mathcal{K}$ et $J \not\models \mathcal{K}$, alors $I <_{\mathcal{K}} J$;*

(3) *Si $\mathcal{K}_1 \equiv \mathcal{K}_2$, alors $\leq_{\mathcal{K}_1} = \leq_{\mathcal{K}_2}$;*

(4) *Si $\forall I \models K_1$, alors $\exists J \models K_2$ $J \leq_{\langle K_1, K_2 \rangle} I$;*

(5) *Si $I \leq_{\mathcal{K}_1} J$ et $I \leq_{\mathcal{K}_2} J$, alors $I \leq_{\mathcal{K}_1 \sqcup \mathcal{K}_2} J$;*

(6) *Si $I <_{\mathcal{K}_1} J$ et $I \leq_{\mathcal{K}_2} J$, alors $I <_{\mathcal{K}_1 \sqcup \mathcal{K}_2} J$.*

Théorème 1 (Konieczny & Pino Pérez (2002))
Un opérateur de fusion Δ est un opérateur de fusion IC ssi il existe un assignement syncrétique qui associe à tout profil p-cohérent $\mathcal{K}$ un préordre total $\leq_{\mathcal{K}}$ tel que pour toute formule μ, $mod(\Delta_\mu(\mathcal{K})) = \min(mod(\mu), \leq_{\mathcal{K}})$.

Plusieurs familles d'opérateurs de fusion peuvent être définies, en particuliers des opérateurs de fusion à base de distances, i.e., des opérateurs caractérisés par une distance entre interprétations et par une fonction d'agrégation f (une application qui associe à un vecteur de nombres réels positifs un nombre réel positif) (Konieczny *et al.*, 2004) :

Définition 3 (Opérateur de fusion à base de distances)
Soit d une distance entre interprétations et f une fonction d'agrégation. L'opérateur de fusion $\Delta^{d,f}$ est défini pour tout profil $\mathcal{K}$ et toute formule μ par $mod(\Delta^{d,f}_\mu(\mathcal{K})) = \min(mod(\mu), \leq_{\mathcal{K}})$ où le préordre $\leq_{\mathcal{K}}$ est défini par :

- *$I \leq_{\mathcal{K}} J$ si et seulement si $d(I, \mathcal{K}) \leq d(J, \mathcal{K})$,*
- *$d(I, \mathcal{K}) = f_{K \in \mathcal{K}}(d(I, K))$,*
- $d(I, K) = \begin{cases} min_{J \models K} d(I, J) & \text{Si } K \text{ est cohérente}^2, \\ 0 & \text{sinon.} \end{cases}$

1. Pour tout préordre $\leq_{\mathcal{K}}, \simeq_{\mathcal{K}}$ dénote la relation d'indifférence correspondante et $<_{\mathcal{K}}$ l'ordre strict correspondant. Lorsqu'un profil $\mathcal{K}$ n'est formé que d'une seule base K, on écrit $\leq_K$ plutôt que $\leq_{\langle K \rangle}$ pour des raisons de clarté de notation.

2. Typiquement, les opérateurs de fusion à base de distances sont appliqués à des profils p-cohérents, ainsi K est supposée cohérente. Ici, lorsque K est incohérente, on définit $d(I, K) = 0$.

Parmi les distances usuelles, on trouve d_D, la distance drastique ($d_D(I, J) = 0$ si $I = J$ et 1 sinon), et d_H la distance de Hamming ($d_D(I, J) = n$ si I et J diffèrent sur n variables). Notons que certains opérateurs à base de distances ne sont pas des opérateurs de fusion IC (pour assurer cela, certaines conditions doivent être vérifiées par f, voir (Konieczny *et al.*, 2004)). Néanmoins, le choix d'une fonction d'agrégation usuelle telle que Σ, $Gmax$ $(leximax)$, etc. induit un opérateur de fusion IC.

Les opérateurs de fusion de croyances sont étroitement liés aux opérateurs de révision de croyances (Alchourrón *et al.*, 1985). Si Δ est un opérateur de fusion IC, alors on peut l'associer à un opérateur de révision AGM $\circ_\Delta$:

Définition 4 ($\circ_\Delta$)
Soit Δ un opérateur de fusion. L'opérateur de révision qui lui correspond, noté $\circ_\Delta$, est défini par

$$K \circ_\Delta \mu = \Delta_\mu(\langle K \rangle).$$

Théorème 2 ((Konieczny & Pino Pérez, 2002))
Si Δ est un opérateur de fusion IC (i.e., il satisfait (IC0-IC8)), alors $\circ_\Delta$ est un opérateur de révision (i.e., il satisfait (R1-R6) (Katsuno & Mendelzon, 1991b)).

3 Indépendance à la rationalisation

Rationaliser une base de croyances par rapport à des contraintes d'intégrité consiste à modifier cette base de telle sorte à ce que ses modèles correspondent à des mondes « réalisables », i.e., à des modèles des contraintes d'intégrité. La première version de rationalisation d'une base que nous introduisons est la rationalisation par expansion. Elle induit le postulat d'indépendance suivant :

Définition 5 ((Ind))
Un opérateur de fusion Δ satisfait (Ind) ssi pour toute formule μ et pour tout profil $\langle K_1, \ldots, K_n \rangle$, $\Delta_\mu(\langle K_1, \ldots, K_n \rangle) \equiv \Delta_\mu(\langle K_1 \wedge \mu, \ldots, K_n \wedge \mu \rangle)$.

Ce postulat impose que la base résultant de la fusion d'un profil doit être équivalente à la base résultant de la fusion du profil obtenu en ayant retiré de chacune de ses bases tout modèle ne satisfaisant pas les contraintes d'intégrité (i.e., chaque base est interprétée en conjonction avec les contraintes d'intégrité).

L'exemple 1 montre que certains opérateurs de fusion IC ne satisfont pas ce postulat :

Exemple 1 (suite)
$\Delta^{d_H,\Sigma}$ ne satisfait pas (Ind). En effet, $\Delta_\mu^{d_H,\Sigma}(\langle K_1, K_2 \rangle) \not\equiv \Delta_\mu^{d_H,\Sigma}(\langle K_1 \wedge \mu, K_2 \wedge \mu \rangle)$:
- $\Delta_\mu^{d_H,\Sigma}(\langle K_1, K_2 \rangle) \equiv on_s_1 \wedge on_s_2$;
- $\Delta_\mu^{d_H,\Sigma}(\langle K_1 \wedge \mu, K_2 \wedge \mu \rangle) \equiv on_s_1 \leftrightarrow on_s_2$.

(Ind) est proche de la condition d'indépendance aux alternatives non pertinentes (IIA) en théorie du choix social (Arrow, 1963; Arrow *et al.*, 2002) pour l'agrégation de relations de préférences. (IIA) impose que le choix (global) entre deux alternatives ne doit

dépendre que de la préférence de chacun des agents à propos de ces deux alternatives uniquement, indépendamment des autres alternatives. En termes de vote, (IIA) peut être exprimé comme le fait que deux profils de préférences qui coïncident lorsqu'ils sont projetés sur un agenda donné doivent conduire à l'élection du même candidat (Kelly, 1978). Dans notre cadre de fusion de croyances, l'ensemble des modèles des contraintes d'intégrité joue le rôle de l'agenda dans le cadre du vote. Par conséquent, la condition (IIA) peut être exprimée dans le cadre de la fusion de croyances comme suit :

Définition 6 (IIA)
Un opérateur de fusion Δ satisfait (IIA) ssi pour toute formule μ et pour tout profil $\langle K_1, \ldots, K_n \rangle, \langle K'_1, \ldots, K'_n \rangle$: si $\langle K_1 \wedge \mu, \ldots, K_n \wedge \mu \rangle \equiv \langle K'_1 \wedge \mu, \ldots, K'_n \wedge \mu \rangle$, alors $\Delta_\mu(\langle K_1, \ldots, K_n \rangle) \equiv \Delta_\mu(\langle K'_1, \ldots, K'_n \rangle)$.

Intuitivement, la condition (IIA) est proche de la condition (Ind). Formellement on montre qu'elles sont « presque équivalentes » :

Proposition 1
Δ satisfait (IIA) ssi Δ satisfait (Ind) et (IC3).

Lorsqu'une base de croyances est incohérente avec les contraintes d'intégrité, sa rationalisation par expansion produit une base incohérente, ce qui est problématique puisque les opérateurs de fusion IC ne considèrent que des profils p-cohérents en entrée. Pour pallier le problème, nous proposons d'étendre l'ensemble des postulats IC à un postulat supplémentaire :

Définition 7 (Opérateur de fusion EIC)
Un opérateur de fusion Δ est un opérateur de fusion EIC (pour opérateur de fusion IC Étendu) ssi pour toute formule μ, pour tout profil[3] $\mathcal{K}$, $\mathcal{K}_1$, $\mathcal{K}_2$ et pour toute base de croyances cohérente K_1, K_2, pour tout $n > 0$, les postulats (IC0)-(IC8) ainsi que le postulat suivant sont satisfaits :

(Inc) $\Delta_\mu(\langle \bot \rangle^n) \equiv \mu$.

Selon (Inc), de la fusion de profils « triviaux » formés de bases incohérentes doit résulter une base équivalente aux contraintes d'intégrité elles-mêmes. On peut facilement montrer que (Inc) est satisfait par tout opérateur de fusion à base de distances (cf. définition 3).

La condition (Inc) est donnée dans la définition 7 sous une forme canonique, au sens où elle décrit le comportement d'un opérateur de fusion pour les profils triviaux, mais elle ne dit rien explicitement lorsque certaines bases du profil considéré (et pas forcément toutes) sont incohérentes. Néanmoins, on peut montrer que les opérateurs de fusion vérifiant (Inc) et certains postulats (IC) (en particulier, les opérateurs de fusion (EIC) appliqués à tout profil) renvoient une base fusionnée équivalente à celle obtenue en retirant du profil les bases incohérentes. Formellement :

3. Notons que contrairement à ce qui est requis dans la définition des opérateurs de fusion IC usuels (définition 1), la p-cohérence des profils n'est pas exigée ici.

Proposition 2
Soit Δ un opérateur de fusion vérifiant (Inc), (IC0), (IC1), (IC5) et (IC6). Pour tout profil p-cohérent $\mathcal{K}$, pour toute formule μ et pour tout $m > 0$,

$$\Delta_\mu(\mathcal{K} \sqcup \langle\bot\rangle^m) \equiv \Delta_\mu(\mathcal{K}).$$

Afin de dériver un théorème de représentation pour les opérateurs EIC, nous définissons maintenant un nouveau type d'assignement :

Définition 8 (Assignement syncrétique étendu)

Un assignement syncrétique étendu *est une application qui associe à tout profil $\mathcal{K}$ un préordre $\leq_\mathcal{K}$ sur les interprétations qui satisfait les conditions (1) - (6) (cf. définition 2) et la condition supplémentaire suivante, pour tout $n > 0$:*

(0) $I \simeq_{\langle\bot\rangle^n} J$.

Ainsi, le théorème de représentation standard pour les opérateurs de fusion IC peut être étendu aux opérateurs de fusion EIC :

Proposition 3
Un opérateur de fusion Δ est un opérateur de fusion EIC ssi il existe un assignement syncrétique étendu qui associe à tout profil $\mathcal{K}$ un préordre total $\leq_\mathcal{K}$ tel que pour toute formule μ, $mod(\Delta_\mu(\mathcal{K})) = \min(mod(\mu), \leq_\mathcal{K})$.

On vérifie facilement que tout opérateur de fusion à base de distances qui est un opérateur de fusion IC est également un opérateur de fusion EIC.

Revenons maintenant au problème d'indépendance. Imposer (Ind) pour un opérateur de fusion peut être considéré comme trop restrictif, étant donné l'aspect relativement drastique du processus de rationalisation correspondant. En effet, d'après la proposition 2, fusionner un profil contenant une base de croyances dont aucun modèle ne satisfait les contraintes d'intégrité peut être réalisé en supprimant la base du profil en amont du processus de fusion. De manière plus parcimonieuse, plutôt que de retirer de telles bases du profil, on pourrait les « réparer ». Dans cette optique, certains opérateurs de changement de croyances peuvent être exploités dans le but de calculer, pour chaque base incohérente avec μ, la base la plus « proche » pleinement compatible avec μ.

Un choix naturel est d'utiliser un opérateur de révision (Alchourrón *et al.*, 1985; Katsuno & Mendelzon, 1991b) lorsque l'on veut « réparer » les bases de manière globale et un opérateur de mise à jour (Katsuno & Mendelzon, 1991a) lorsque l'on veut « réparer » les bases de manière locale, modèle par modèle. Les propriétés d'indépendance correspondantes sont données comme suit [4] :

Définition 9
(Ind-$\circ$) *Un opérateur de fusion Δ satisfait (Ind-$\circ$) ssi l'on peut associer à chaque agent i un opérateur de révision AGM $\circ_i$ (Katsuno & Mendelzon, 1991b) tel que pour toute*

4. Rappelons que chacune des bases K_i du profil correspond aux croyances de l'agent i.

formule μ et pour tout profil $\langle K_1, \ldots, K_n \rangle$,

$$\Delta_\mu(\langle K_1, \ldots, K_n \rangle) \equiv \Delta_\mu(\langle K_1 \circ_1 \mu, \ldots, K_n \circ_n \mu \rangle).$$

Définition 10
(Ind-$\diamond$) *Un opérateur de fusion Δ satisfait (Ind-$\diamond$) ssi l'on peut associer à chaque agent i un opérateur de mise à jour KM $\diamond_i$ (Katsuno & Mendelzon, 1991a) tel que pour toute formule μ et pour tout profil $\langle K_1, \ldots, K_n \rangle$,*

$$\Delta_\mu(\langle K_1, \ldots, K_n \rangle) \equiv \Delta_\mu(\langle K_1 \diamond_1 \mu, \ldots, K_n \diamond_n \mu \rangle).$$

Nous pouvons remarquer que dans les définitions 9 et 10, aucun lien n'est imposé entre les opérateurs « rationalisants » $\circ_i$ ou $\diamond_i$ et l'opérateur de fusion Δ en question. En outre, aucune condition d'homogénéïté n'est imposée, ainsi les agents fournissant les bases peuvent avoir des politiques de révision ou de mise à jour différentes. Ainsi, ces propriétés d'indépendance sont assez permissives.

Pourtant, il s'avère qu'il n'existe pas d'opérateur de fusion EIC qui satisfait (Ind-$\diamond$). En effet, la proposition suivante montre que pour les opérateurs de fusion EIC, la propriété d'indépendance à la rationalisation par mise à jour n'est pas compatible avec les postulats IC les plus fondamentaux.

Proposition 4
Il n'existe aucun opérateur de fusion vérifiant (IC0), (IC1), (IC2) et (Ind-$\diamond$).

D'autre part, les postulats (Ind) et (Ind-$\circ$) sont compatibles avec l'ensemble des postulats IC :

Proposition 5
Pour toute fonction d'agrégation f, $\Delta^{d_D,f}$ satisfait (Ind) et (Ind-$\circ$).

En particulier, puisque $\Delta^{d_D,\Sigma}$ est un opérateur de fusion IC (Konieczny & Pino Pérez, 2002) et puisqu'il vérifie (Inc), on obtient le corollaire suivant :

Corollaire 1
$\Delta^{d_D,\Sigma}$ est un opérateur de fusion EIC vérifiant (Ind) et (Ind-$\circ$).

Une problématique intéressante consiste alors à déterminer l'ensemble des opérateurs de révision AGM « rationalisants » pouvant être choisis de sorte à ce que le postulat (Ind-$\circ$) soit satisfait. Il s'avère que ce choix se restreint à un seul opérateur de révision AGM :

Proposition 6
Soit Δ un opérateur de fusion EIC vérifiant (Ind-$\circ$). Tout opérateur de révision « rationalisant » $\circ_i$ considéré dans (Ind-$\circ$) est l'opérateur de révision $\circ_\Delta$ correspondant à Δ

au sens de la définition 4. De plus, $\circ_\Delta$ est $\circ^D$, l'opérateur de révision drastique, défini comme suit :

$$K \circ^D \mu = \begin{cases} K \wedge \mu & \text{si } K \wedge \mu \text{ est cohérent,} \\ \mu & \text{sinon.} \end{cases}$$

Une conséquence importante de cette proposition est le corollaire suivant :

Corollaire 2
Soit Δ un opérateur de fusion EIC. Δ satisfait (Ind) si et seulement si Δ satisfait (Ind-$\circ$).

Ce dernier résultat montre que pour les opérateurs de fusion EIC, les deux notions d'indépendance (Ind) et (Ind-$\circ$) coïncident. C'est pour cette raison que nous nous focalisons uniquement sur (Ind), dans le reste de l'article.

4 Une caractérisation de l'opérateur de fusion $\Delta^{d_D,\Sigma}$

Le corollaire 1 indique que l'ensemble des opérateurs EIC n'est pas vide puisqu'il contient l'opérateur $\Delta^{d_D,\Sigma}$.

La question qui se pose alors naturellement est la suivante : quels sont exactement les opérateurs EIC (pas forcément à base de distances) qui satisfont (Ind) ? Dans la suite, on donne un théorème de représentation qui répond à cette question.

Définition 11 (Assignement filtrant)
Un assignement filtrant *est un assignement syncrétique étendu vérifiant la condition suivante, pour toute base de croyances K_1, K_2 :*

(F) *Si $I <_{K_1} J$ et $J <_{K_2} I$, alors $I \simeq_{\langle K_1,K_2\rangle} J$.*

La condition (F) indique que si un monde est vu comme strictement plus plausible qu'un autre monde pour un profil singleton, et si l'ordre de plausibilité est inversé pour un autre profil singleton, alors ces deux mondes doivent avoir la même plausibilité pour l'union des deux profils. Autrement dit, lorsque la condition (F) est vérifiée en plus des conditions (1) et (2) (cf. définition 2), il suffit de comparer la plausibilité de deux mondes distincts I, J pour deux profils singletons $\langle K_1\rangle, \langle K_2\rangle$ indépendamment dans le but de déterminer la plausibilité relative de I et J pour le profil doubleton $\langle K_1, K_2\rangle$.

Remarquons que la condition (F) peut être vue comme une version plus restrictive de la condition (4) (cf. définition 2) en présence des conditions (1) et (2) :

Proposition 7
Tout assignement satisfaisant les conditions (1), (2) et (F) satisfait la condition (4).

Réciproquement, la contrainte exprimée par la condition (F) n'est pas une conséquence des conditions (1), (2) et (4), ce qui peut être illustré par l'exemple suivant. Considérons trois mondes distincts I, J, L et deux bases de croyances $K_1 \equiv \varphi_{\{I,J\}}$ et $K_2 \equiv \varphi_{\{I,L\}}$. En présence des conditions (1) et (2), on a $I \simeq_{K_1} J <_{K_1} L$ et $I \simeq_{K_2} L <_{K_2} J$. Bien que la condition (4) ait pour but de capturer un comportement d'équité, à elle

seule elle n'impose pas aux deux mondes J et L d'avoir la même plausibilité pour le profil $\langle K_1, K_2\rangle$: on pourrait avoir par exemple $J <_{\langle K_1,K_2\rangle} L$. La condition (F), elle, implique $J \simeq_{\langle K_1,K_2\rangle} L$.

La proposition suivante montre qu'à travers un assignement filtrant, tous les mondes sont ordonnés sur deux niveaux de plausibilité au plus, pour tout profil singleton $\langle K\rangle$:

Proposition 8
Tout assignement vérifiant les conditions (1), (2), (6) et (F) associe à tout profil singleton $\langle K\rangle$ un préordre total unique $\leq_K$ sur les interprétations tel que pour toute base de croyances K et pour toute interprétation I, J, $I <_K J$ ssi $I \models K$ et $J \not\models K$.

La proposition 8 est un résultat clé pour prouver la proposition suivante sur les assignements syncrétiques filtrants. Étant donnés une interprétation I et un profil $\mathcal{K}$, on note $|I(\mathcal{K})| = |\{K_i \in \mathcal{K} \mid I \models K_i\}|$, i.e., le nombre des bases de croyances dans $\mathcal{K}$ pour lesquelles I est un modèle. On a la proposition suivante :

Proposition 9
Soit $\leq_{\mathcal{K}}$ un préordre sur les interprétations qui associe à un profil $\mathcal{K}$ un assignement syncrétique filtrant. On a

$$I <_{\mathcal{K}} J \text{ ssi } |I(\mathcal{K})| > |J(\mathcal{K})|.$$

Une conséquence importante de la proposition 9 est le théorème de représentation suivant pour les opérateurs de fusion EIC vérifiant (Ind) :

Proposition 10
Un opérateur de fusion EIC Δ satisfait (Ind) ssi il existe un assignement syncrétique filtrant qui associe à tout profil $\mathcal{K}$ un préordre total $\leq_{\mathcal{K}}$ tel que pour toute formule μ, $mod(\Delta_\mu(\mathcal{K})) = \min(mod(\mu), \leq_{\mathcal{K}})$.

Une autre conséquence de la proposition 9 est donnée par le corollaire suivant :

Corollaire 3
Il existe un seul assignement syncrétique filtrant.

Une conséquence directe des corollaires 3 et 1 et de la proposition 10 est que :

Corollaire 4
$\Delta^{d_D,\Sigma}$ est l'unique opérateur de fusion EIC vérifiant (Ind).

Ce résultat fournit une caractérisation axiomatique complète de l'opérateur de fusion EIC à base de distances $\Delta^{d_D,\Sigma}$ en terme d'indépendance à la rationalisation des bases.

5 Conclusion

Dans cet article, nous avons étudié le cas où les bases de croyances sont « rationalisées » par rapport aux contraintes d'intégrité au cours d'un processus de fusion. Une telle « rationalisation » est souhaitée pour les applications dans lesquelles certains opérateurs de fusion IC pourraient renvoyer un résultat contre-intuitif parce qu'ils prennent en compte les mondes non « réalisables ». C'est le cas en particulier des applications pour lesquelles les contraintes d'intégrité encodent des contraintes très fortes comme des lois physiques. Par exemple, lorsque les formules propositionnelles sont obtenues via une traduction de représentations provenant d'un cadre plus expressif (comme les réseaux de contraintes qualitatives spatiales ou temporelles), les contraintes d'intégrité sont utilisées pour exclure les mondes non représentatifs d'une situation spatiale ou temporelle cohérente (de tels mondes sont créés durant le processus de traduction) (Condotta *et al.*, 2009).

Nous avons défini formellement plusieurs conditions d'indépendance pour les opérateurs de fusion et nous avons étudié l'interaction de ces conditions avec les postulats IC qui constituent un standard pour la fusion de croyances. En particulier :

- Puisque la rationalisation par expansion peut conduire à des bases incohérentes et donc à des profils non p-cohérents, nous avons étendu les postulats IC à un nouvel axiome (Inc) qui caractérise un comportement pour les opérateurs de fusion appliqués aux profils formés de bases incohérentes ; nous avons fourni un théorème de représentation pour le nouvel ensemble de postulats, appelés EIC, où la condition de p-cohérence sur les profils est relâchée.
- Nous avons montré que l'indépendance à la rationalisation par mise à jour est impossible pour les opérateurs de fusion EIC, puisque cette propriété d'indépendance est en conflit avec certains postulats IC fondamentaux. Nous avons également montré que l'indépendance à la rationalisation par révision est équivalente à l'indépendance à la rationalisation par expansion pour les opérateurs de fusion EIC.
- Enfin, nous avons prouvé qu'il existe un unique opérateur de fusion EIC qui satisfait la propriété d'indépendance à la rationalisation par expansion (ou, de manière équivalente, par révision), l'opérateur à base de distances $\Delta^{d_D,\Sigma}$, qui utilise la distance drastique d_D et la somme comme fonction d'agrégation. Á notre connaissance, il s'agit du premier opérateur de fusion IC pour lequel une caractérisation axiomatique complète est donnée.

Dans cet article nous avons montré qu'il existe un unique opérateur EIC vérifiant (Ind). Il est possible d'associer à n'importe quel opérateur de fusion EIC Δ et à n'importe quelle méthode de rationalisation R un opérateur de fusion « rationalisé » Δ^R, en effectuant tout d'abord l'opération de rationalisation de manière explicite en utilisant R, puis l'opération de fusion en utilisant Δ. Les résultats de cet article montrent que les opérateurs Δ et Δ^R sont distincts (sauf pour l'opérateur $\Delta^{d_D,\Sigma}$). Une perspective de recherche est d'étudier le comportement et les propriétés vérifiées par de tels opérateurs Δ^R. Étudier les liens logiques entre les propriétés d'indépendance (Ind), (Ind-$\circ$) et (Ind-$\diamond$) pour les opérateurs qui ne satisfont pas tous les postulats EIC est également un sujet pour des recherches futures.

Références

ALCHOURRÓN C. E., GÄRDENFORS P. & MAKINSON D. (1985). On the logic of theory change : Partial meet contraction and revision functions. *Journal of Symbolic Logic*, **50**, 510–530.

ARROW K. J. (1963). *Social choice and individual values*. New York : Wiley, second edition.

K. J. ARROW, A. K. SEN & K. SUZUMURA, Eds. (2002). *Handbook of Social Choice and Welfare*, volume 1. North-Holland.

CONDOTTA J.-F., KACI S., MARQUIS P. & SCHWIND N. (2009). Merging qualitative constraints networks using propositional logic. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU'09)*, p. 347–358.

KATSUNO H. & MENDELZON A. O. (1991a). On the difference between updating a knowledge base and revising it. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning(KR'91)*, p. 387–394.

KATSUNO H. & MENDELZON A. O. (1991b). Propositional knowledge base revision and minimal change. *Artificial Intelligence*, **52**(3), 263–294.

KELLY J. S. (1978). *Arrow impossibility theorems*. Series in economic theory and mathematical economics. New York : Academic Press.

KONIECZNY S., LANG J. & MARQUIS P. (2004). DA^2 merging operators. *Artificial Intelligence*, **157**(1-2), 49–79.

KONIECZNY S. & PINO PÉREZ R. (2002). Merging information under constraints : a logical framework. *Journal of Logic and Computation*, **12**(5), 773–808.

KONIECZNY S. & PINO PÉREZ R. (2011). Logic based merging. *Journal of Philosophical Logic*, **40**, 239–270.

LIBERATORE P. & SCHAERF M. (1998). Arbitration (or how to merge knowledge bases). *IEEE Transactions on Knowledge and Data Engineering*, **10**, 76–90.

LIN J. (1996). Integration of weighted knowledge bases. *Artificial Intelligence*, **83**, 363–378.

REVESZ P. Z. (1997). On the semantics of arbitration. *International Journal of Algebra and Computation*, **7**, 133–160.

Annexe : preuves

Proposition 1 : *(Seulement si)* Supposons que Δ vérifie (IIA). On a $\langle K_1 \wedge \mu, \ldots, K_n \wedge \mu\rangle \equiv \langle K_1 \wedge \mu \wedge \mu, \ldots, K_n \wedge \mu \wedge \mu\rangle$. Par (IIA) on obtient $\Delta_\mu(\langle K_1, \ldots, K_n\rangle) \equiv \Delta_\mu(\langle K_1 \wedge \mu, \ldots, K_n \wedge \mu\rangle)$. Par conséquent, Δ satisfait (Ind). (IC3) est évident. *(Si)* Supposons que Δ vérifie (Ind) et (IC3) et montrons que Δ satisfait (IIA). Soit $\langle K_1 \wedge \mu, \ldots, K_n \wedge \mu\rangle \equiv \langle K'_1 \wedge \mu, \ldots, K'_n \wedge \mu\rangle$. Par (IC3) on a $\Delta_\mu(\langle K_1 \wedge \mu, \ldots, K_n \wedge \mu\rangle) \equiv \Delta_\mu(\langle K'_1 \wedge \mu, \ldots, K'_n \wedge \mu\rangle)$. Or par (Ind) on a $\Delta_\mu(\langle K_1 \wedge \mu, \ldots, K_n \wedge \mu\rangle) \equiv \Delta_\mu(\langle K_1, \ldots, K_n\rangle)$ et $\Delta_\mu(\langle K'_1 \wedge \mu, \ldots, K'_n \wedge \mu\rangle) \equiv \Delta_\mu(\langle K'_1, \ldots, K'_n\rangle)$. Par conséquent,

$\Delta_\mu(\langle K_1, \ldots, K_n\rangle) \equiv \Delta_\mu(\langle K'_1, \ldots, K'_n\rangle)$. □

Proposition 2 : Soit Δ un opérateur de fusion vérifiant (Inc), (IC0), (IC1), (IC5) et (IC6). Supposons $\mu \not\models \bot$ (si $\mu \models \bot$, alors la conclusion s'obtient trivialement de (IC0)). Par (Inc) on a $\Delta_\mu(\langle\bot\rangle^m) \equiv \mu$. Or, par (IC0) on a $\Delta_\mu(\mathcal{K}) \models \mu$. Ainsi, $\Delta_\mu(\mathcal{K}) \wedge \Delta_\mu(\langle\bot\rangle^m) \equiv \Delta_\mu(\mathcal{K})$, qui est cohérente par (IC1). Donc par (IC5) et (IC6), on a $\Delta_\mu(\mathcal{K}) \wedge \Delta_\mu(\langle\bot\rangle^m) \equiv \Delta_\mu(\mathcal{K} \sqcup \langle\bot\rangle^m)$. Donc $\Delta_\mu(\mathcal{K} \sqcup \langle\bot\rangle^m) \equiv \Delta_\mu(\mathcal{K})$. □

Proposition 3 : *(Seulement si)* Soit Δ un opérateur de fusion EIC. On définit un assignement comme suit : $\forall I, J, I \leq_{\mathcal{K}} J$ ssi $I \models \Delta_{\varphi_{\{I,J\}}}(\mathcal{K})$. Cet assignement satisfait les conditions (1) à (6) (voir (Konieczny & Pino Pérez, 2002), théorème 11). Et par (Inc), $\forall I, J, \Delta_{\varphi_{\{I,J\}}}(\langle\bot\rangle^n) \equiv \varphi_{\{I,J\}}$, donc $I \simeq_{\langle\bot\rangle^n} J$. Ainsi, la condition (0) (cf. définition 8) est vérifiée.
(Si) Considérons un assignement syncrétique étendu. Définissons Δ par $mod(\Delta_\mu(\mathcal{K}))= \min(mod(\mu), \leq_{\mathcal{K}})$. Δ satisfait (IC0) - (IC8) (voir (Konieczny & Pino Pérez, 2002), théorème 11). Or par (IC0) on a $\Delta_\mu(\langle\bot\rangle^n) \models \mu$, et par la condition (0) et la définition de Δ, on a $\forall I \models \mu, I \models \Delta_\mu(\langle\bot\rangle^n)$. Par conséquent, Δ satisfait (Inc). □

Proposition 4 : On montre tout d'abord que tout opérateur de mise à jour rationalisant $\diamond_i$ est l'opérateur de mise à jour drastique $\diamond^D$, défini par $K \diamond^D \mu \equiv K$ si $K \models \mu$, μ sinon. $\diamond^D$ correspond à l'assignement fidèle (voir (Katsuno & Mendelzon, 1991b)) tel que $\forall I, J, L, I \simeq_{\varphi_{\{L\}}} J$. Par l'absurde, supposons qu'il existe $\diamond_j \neq \diamond^D$. Par (U1) et (U3), $\varphi_{\{L\}} \diamond_j \varphi_{\{I,J\}} \models \varphi_{\{I,J\}}$, donc $\forall I, J, L$ tels que $J <^{\diamond_j}_{\varphi_{\{L\}}} I, \varphi_{\{L\}} \diamond_j \varphi_{\{I,J\}} \equiv \varphi_{\{J\}}$. Donc par (U2) et (U8) on a (*) $\varphi_{\{I,L\}} \diamond_j \varphi_{\{I,J\}} \equiv \varphi_{\{I,J\}}$. Par ailleurs, de (Ind-$\diamond$) et (*) on obtient $\Delta_{\varphi_{\{I,J\}}}(\langle\varphi_{\{I,L\}}\rangle) \equiv \Delta_{\varphi_{\{I,J\}}}(\langle\varphi_{\{I,L\}} \diamond_j \varphi_{\{I,J\}}\rangle) \equiv \varphi_{\{I,J\}}$. Enfin, par (IC0), (IC1) et (IC2) on a $\Delta_{\varphi_{\{I,J\}}}(\langle\varphi_{\{I,L\}}\rangle) \equiv \varphi_{\{I\}}$. Contradiction. Maintenant, soit $K \not\models \mu, K \wedge \mu \not\models \bot$. Par définition de $\diamond^D$ et par (Ind-$\diamond$), $\Delta_\mu(\langle K\rangle) \equiv \mu$. Ceci contredit (IC2). □

Proposition 5 : Définissons tout opérateur de révision rationalisant $\circ_i$ comme étant l'opérateur de révision drastique $\circ^D$, défini par $K \circ^D \mu \equiv K \wedge \mu$ si $K \wedge \mu \not\models \bot$, μ sinon. Remarquons que $\forall K, \forall I \models \mu, d_D(I, K) = d_D(I, K \wedge \mu) = d_D(I, K \circ^D \mu)$, ce qui conclut la preuve. □

Proposition 6 :
• Soit K une base. Pour tout opérateur de révision rationalisant $\circ_i$, par (R1) et (R3) on a $(K \circ_i \mu) \models \mu$ et $(K \circ_i \mu) \not\models \bot$. Donc par (Ind-$\circ$) et (IC2), on a $\Delta_\mu(\langle K\rangle) \equiv \Delta_\mu(\langle K \circ_i \mu\rangle) \equiv K \circ_i \mu$. Donc pour tout $\circ_i$, $\circ_i = \circ_\Delta$.
• Par conséquent, l'assignement associé à $\circ_\Delta$ coïncide avec la restriction aux profils singletons de l'assignement associé à Δ. Puisque Δ est un opérateur de fusion IC, par le théorème 1 il correspond à un assignement vérifiant les conditions (1) à (6) (cf. définition 2). Par l'absurde, supposons maintenant que $\circ_\Delta \neq \circ^D$ avec $\circ^D$ l'opérateur de révision drastique (voir la preuve de la proposition 5). Alors $\exists I, J, L$ tels que (*) $I <_{\varphi_{\{I\}}} J <_{\varphi_{\{I\}}} L$, i.e., il existe un préordre total $<_{\varphi_{\{I\}}}$ à 3 niveaux distincts. Soit $K \equiv \varphi_{\{J,L\}}$. Par les conditions (1) et (2), on a (**) $J \simeq_K L <_K I$. Afin d'éviter une écriture trop lourde, notons plus simplement $<$ le préordre total $<_{\langle\varphi_{\{I\}},K\rangle}$. Maintenant,
  – Par (*), (**) et par la condition (6), on a $J < L$ *(i)* ;
  – Par (R2), $K \circ_\Delta \varphi_{\{I,J\}} \equiv \varphi_{\{J\}}$, et $\varphi_{\{I\}} \circ_\Delta \varphi_{\{I,J\}} \equiv \varphi_{\{J\}}$. Alors par (Ind-

$\circ$), $\Delta_{\varphi_{\{I,J\}}}(\langle K,\varphi_{\{I\}}\rangle) \equiv \Delta_{\varphi_{\{I,J\}}}(\langle \varphi_{\{J\}},\varphi_{\{I\}}\rangle)$. Pa conséquent, par (IC0), (IC1) et (IC4), on obtient $\Delta_{\varphi_{\{I,J\}}}(\langle K,\varphi_{\{I\}}\rangle) \equiv \varphi_{\{I,J\}}$, ce qui signifie que $I \simeq J$ *(ii)*;

– De manière similaire, on a $I \simeq L$ *(iii)*.

(i), (ii) et (iii) pris conjointement contredisent $\circ_\Delta \neq \circ^D$. □

Proposition 7 : Soient K, K' deux bases cohérentes, et soit $I \models K$. La condition (4) est trivialement vérifiée si $I \models K'$. Supposons donc $I \not\models K'$. Soit $J \models K'$. If $J \models K$, alors les conditions (1) et (2) impliquent que $J <_{\langle K,K'\rangle} I$, et la condition (4) est vérifiée. Supposons maintenant $J \not\models K$. Les conditions (1) et (2) impliquent $I <_K J$ et $J <_{K'} I$. Par la condition (F), on obtient $I \simeq_{\langle K,K'\rangle} J$ et la condition (4) est vérifiée. □

Proposition 8 : *(Si)* Trivial par la condition (2).
(Seulement si) Soit K une base.

– On montre tout d'abord que (*) si $I \not\models K$ et $J \not\models K$, alors $I \simeq_K J$: soit $I \not\models K$, $J \not\models K$ et supposons par l'absurde que $I <_K J$ (le cas où $J <_K I$ est symétrique et la preuve est similaire). Soit $L \models K$. En utilisant les conditions (2) et (F), on obtient $I \simeq_{\langle K,\varphi_{\{J\}}\rangle} L$. Alors par la condition (6), on a *(i)* $I <_{\varphi_{\{J\}}} L$. Or $I \simeq_{\varphi_{\{I,L\}}} L <_{\varphi_{\{I,L\}}} J$. Donc par la condition (F), on a *(ii)* $I \simeq_{\langle \varphi_{\{J\}},\varphi_{\{I,L\}}\rangle} L$. *(i)* et *(ii)* ensemble avec la condition (6) impliquent que $I <_{\langle \varphi_{\{J\}},\varphi_{\{I,L\}}\rangle} L$, ce qui contredit *(ii)*.

– Maintenant, supposons $I <_K J$. Par l'absurde, supposons $I \not\models K$ ou $J \models K$. Pour conclure la preuve, il suffit de considérer séparément les trois cas induits, chacun conduisant à une contradiction, en utilisant (*) ainsi que les conditions (1) et (2). □

Proposition 9 : – En premier lieu, on peut montrer que (*) si $I <_{\mathcal{K}} J$ alors $I \leq_{\mathcal{K}\sqcup\{K\}} J$, par induction sur le nombre de bases dans $\mathcal{K}$, et en utilisant les conditions (5) et (F).

– Ensuite, on montre la proposition par induction sur le nombre de bases dans $\mathcal{K}$. Le cas de base $|\mathcal{K}| = 1$ provient directement de la proposition 8. Le cas de base $|\mathcal{K}| = 2$, quant à lui, provient directement de la propriété suivante : $I <_{\langle K,K'\rangle} J$ ssi ($I \leq_K J$ et $I <_{K'} J$) ou ($I <_K J$ et $I \leq_{K'} J$), ce qui peut être montré en utilisant les conditions (5), (6) et (F). En utilisant l'hypothèse de récurrence, i.e., $\forall k \leq n$ avec $n \geq 2$, $I <_{\mathcal{K}} J$ ssi $|I(\mathcal{K})| > |J(\mathcal{K})|$, on peut prouver la propriété pour tout profil $\mathcal{K} \sqcup \{K'\}$ (c'est-à-dire, de taille $k+1$) en utilisant les conditions (5) et (F) pour la partie (Seulement si) de la propriété, la condition (6) pour la partie (Si) de la propriété. □

Proposition 10 : *(Seulement si)* Soit Δ un opérateur de fusion EIC vérifiant (Ind). Par le corollaire 2, Δ satisfait (Ind-$\circ$). Définissons un assignement syncrétique étendu comme dans la preuve de la proposition 3 et montrons que cet assignement satisfait la condition (F). Par la proposition 6, $\forall \circ_i$ considéré dans (Ind), $\circ_i = \circ^D$ (voir la preuve de la proposition 5) et $\forall K$, $<_K^{\circ^D} = <_K$. Soient $I <_K J$ et $J <_{K'} I$. Soit $K'' \equiv \varphi_{\{I,J\}}$. Alors $K \circ^D K'' \equiv \varphi_{\{I\}}$ et $K' \circ^D K'' \equiv \varphi_{\{J\}}$. Ainsi, par (Ind-$\circ$), (IC0), (IC1) et (IC4), $\Delta_{K''}(\langle K, K'\rangle) \equiv \Delta_{K''}(\langle \varphi_{\{I\}}, \varphi_{\{J\}}\rangle) \equiv K''$. Donc, $I \simeq_{\langle K,K'\rangle} J$.
(Si) Considérons un assignement syncrétique filtrant. Définissons Δ par $mod(\Delta_\mu(\mathcal{K}))= \min(mod(\mu), \leq_{\mathcal{K}})$. De la proposition 3, Δ satisfait (Inc), (IC0) - (IC8). Soit $\mathcal{K} = \langle K_1, \ldots, K_n\rangle$ un profil et $\mathcal{K}' \equiv \langle K_1 \wedge \varphi_{\{I,J\}}, \ldots, K_m \wedge \varphi_{\{I,J\}}\rangle$. En utilisant la proposition 9, on montre que $I <_{\mathcal{K}} J$ ssi $I <_{\mathcal{K}'} J$. Par conséquent, Δ satisfait (Ind). □

Utilisation de la programmation par ensembles réponses (Answer Set Programming) sur de “petits” problèmes

Yves Moinard[1]

INRIA-Bretagne Atlantique
IRISA, campus de Beaulieu, 35042 RENNES yves.moinard@inria.fr

Résumé : En traduisant en programmation par ensembles réponses deux petites devinettes classiques, nous illustrons la puissance et certaines limites des systèmes actuels. Le problème n’est pas de traduire le problème de la manière la plus efficace. Il s’agit de le traduire d’une façon pas trop ad-hoc, qui tire pleinement profit de l’aspect déclaratif de ce type de programmation. Ces exemples montrent qu’on n’est pas loin de cet objectif affiché de la programmation par ensembles-réponses sur de tels exemples, même s’il reste des progrès à faire pour les systèmes existant. Nous évoquons deux pistes : une meilleure intégration entre l’instantiateur et le solveur, permettant au moins de détecter les cas où le premier suffit, et une recherche d’indexation automatique permettant de diminuer la mémoire nécessaire.

Mots-clés : Programmation logique, ensembles réponses, planification

1 Introduction

On examine ici la facilité avec laquelle on peut traduire de petits problèmes en un programme exécutable en programmation par ensembles réponses (que nous abrégerons par l’acronyme anglais ASP pour “answer set programming”) (Baral, 2003). Depuis gringo3 (juillet 2010) la lignée clingo (gringo + clasp)[1], (Gebser *et al.*, 2009) semble la plus efficace en général. Auparavant, il était souvent plus commode de traduire ces problèmes grâce à DLV (Leone *et al.*, 2006)[2] et ses extensions comme DLV-Complex[3] (Calimeri *et al.*, 2009), qui admettent des structures de données plus riches (suites, ensemble), avec parfois la possibilité d’intégrer de petits scripts de calcul. L’usage de clingo exigeait que chaque variable du corps d’une règle soit introduite par un prédicat dont le domaine est donné au départ. Cette lourde contrainte a disparu depuis gringo3, l’intégration de scripts a aussi été ajoutée (non utilisée ici). D’autre part les symboles de fonction permettent de simuler (plus ou moins élégamment) des suites. Ainsi, les

[1] http ://sourceforge.net/projects/potassco/
[2] http ://www.dbai.tuwien.ac.at/proj/dlv/
[3] https ://www.mat.unical.it/dlv-complex

principaux avantages qui existaient en faveur de l'utilisation de DLV se sont estompés. L'aspect ouvert de clingo, contre un système DLV moins transparent, en plus de ses performances en général meilleures, apporte des arguments en faveur de clingo. D'autres sytèmes[4], comme Asperix[5] (Lefèvre *et al.*, 2010) ou Cmodels[6] demeurent en version préliminaire ou (parfois inclusif) sont d'une utilisation pratique plus complexe.

La programmation par ensemble réponse constitue sans doute le système le plus proche de l'idéal de la programmation déclarative : *le problème est le programme*, inutile de se préoccuper de la façon de calculer la solution. La *solution* est un ensemble de modèles particuliers appelés *ensembles réponses* (que l'on abrégera encore par l'acronyme anglais *AS*). En réalité bien sûr il faut se préoccuper un peu de la façon dont le système va calculer les réponses. Selon le problème posé, cette préoccupation est plus ou moins complexe. La programmation par ensembles réponses est bien adaptée par exemple à des problèmes de parcours dans des graphes, et elle permet d'écrire des programmes élégants et naturels. Ce formalisme proche la logique classique traite de façon très naturelle de petits problèmes comme le fameux problème de *l'eau et du zèbre*. `http ://www.hakank.org/answer_set_programming/zebra.lp` de Hakan Kjellerstrand fournit l'énoncé, et surtout deux transcriptions en ASP élégantes, dont une seule est efficace. Cela illustre le fait (évident, et reconnu par les créateurs ou utilisateurs de systèmes ASP) qu'il est illusoire de s'affranchir de toute considération d'efficacité. Il reste que les critères permettant de juger du caractère "naturel" d'une solution (un programme ASP qui résout un problème posé) sont flous. La compétition ASP tend naturellement à privilégier les systèmes qui permettent, pour de fins connaisseurs de ces systèmes, de résoudre un problème le plus rapidement possible et avec le moins de place mémoire. Pour tout langage informatique, ce seul critère est insuffisant, mais c'est encore plus insuffisant pour un langage déclaratif. Une citation extraite des résultats de la seconde compétition ASP évoque l'importance cet aspect, mais il demeure délicat de mesurer objectivement les performances en ce domaine :

> The Potassco team is the clear winner [semble encore vrai pour la Third competition] [...]. Given that the goal of declarative problem solving is to minimize the effort of programmers, it is of equal interest to investigate the performance of teams that used a single collection of systems with uniform parameter settings. (Denecker *et al.*, 2009).

Voici deux petits exemples avec des traductions ASP qui semblent "naturelles". Le premier est simple. Le second n'est pas trop complexe, mais il semble qu'aucun programme "simple et naturel" ne donne directement le résultat. Les systèmes actuels (comme clingo) permettent de le résoudre, mais il ne suffit plus de transcrire l'énoncé.

2 Un exemple de rêve : le pont miné

> Quatre soldats doivent franchir de nuit un pont qui sautera une heure après que le premier soldat aura mis le pied sur le pont. Les soldats sont plus

[4]Liste de systèmes : `http ://www.cs.uni-potsdam.de/~torsten/asp/`.

[5]`http ://www.info.univ-angers.fr/pub/claire/asperix/`

[6]`http ://www.cs.utexas.edu/users/tag/cmodels.html`

ou moins blessés ce qui donne leur durée respective de franchissement du pont en minutes : 25, 20, 10 et 5. Le pont ne supporte que deux soldats à la fois et l'unique lampe est indispensable pour traverser. Comment font-ils ?

Le programme qui suit est une copie quasi littérale de l'énoncé. C'est une suite de règles "`tête :- corps.`" : si chaque élément du corps est satisfait, la tête l'est. Pour les données on utilise des faits, ou règles sans corps (`corps`$\equiv \top$) :

```
soldat(0..3). temps(0,25). temps(1,20). temps(2,10).
temps(3,5). lieu(depart). lieu(arriv). #const tmax= 60.
```

(La notation `N1..N2` signifie que toute valeur entre N1 et N2 est possible.)

Voici la signification des prédicats introduits :

`at(M,L,S)` : À l'étape M, le soldat (ou la lampe) S est en L.
`move(M,L,S)` : En M, S part vers L, qu'il atteindra en M+1.
`duree(M,T)` : En M, les soldats ont mis un temps T pour effectuer leur trajet.

Les noms de variables (apparaissant comme paramètre de prédicat ou de fonction) commencent par une majuscule, contrairement aux noms de constantes, fixées une fois pour toutes à chaque exécution d'un programme.

Initialisation : 4 soldats et lampe au départ, durée du trajet actuel nulle :

```
at(1,depart,S):-soldat(S). at(1,depart,lampe). duree(0,0).
```

Génération des déplacements : Un soldat ne peut traverser que s'il part du côté de la lampe. On essaie de traverser tant que le but n'est pas atteint [`not goal(M-1)`, facultatif, mais accélère sensiblement]. `not` est la *négation par défaut* : `not goal(M-1)` est vrai quand `goal(M-1)` ne figure pas dans l'ensemble réponse considéré. Ne plus traverser après "tmax" (explosion). Tout cela est aisément traduit par une règle *de génération* indéterministe (si le corps est satisfait, `move(M,L,S)` peut être pris ou pas) :

```
{move(M,L,S)} :- at(M,L0,S), soldat(S), at(M,L0,lampe),
    lieu(L),L0 != L,duree(M-1,T), T<tmax, not goal(M-1).
```

Description du résultat du déplacement représenté par les `move(M,L,_)` :

(1) Ceux qui arrivent en L : `at(M+1,L,S) :- move(M,L,S).`

(2) *Axiomes du cadre* Ceux qui ne se sont pas déplacés restent en `L0` (2a) ou en `L` (2b) :

(2a) `at(M+1,L0,S) :- move(M,L,S1),`
`at(M,L0,S), L0 != L, not move(M,L,S).`

(2b) `at(M+1,L,S) :- move(M,L,S1), at(M,L,S).`

(Ces deux axiomes sont un peu particuliers au problème, pour accélérer, mais un seul axiome du cadre, encore plus simple, et généralisable, suffirait.)

Moins de 3 soldats traversent. On utilise une *contrainte* : règle sans tête, ce qui équivaut à `tête`$\equiv \bot$, qui élimine les ensembles réponses satisfaisant le corps :

```
:- move(M,L,S1;S2;S3), soldat(S1;S2;S3), S1 <S2, S2 <S3.
```

Utiliser la notation clingo `move(M,L,S1;S2...)` qui remplace `move(M,L,S1), move(M,L,S2),...` allège l'écriture (et accélère).

Utiliser $<$ au lieu de `!=` (qui représente $\neq$) diminue la taille de l'instantiation.

Un soldat suffit à porter la lampe, qu'il faut emporter à chaque traversée :

```
move(M,L,lampe) :- move(M,L,S).
```

Le calcul du temps de traversée est peu élégant, car la version actuelle de clingo n'autorise pas ici les "méta-prédicats" #max et #sum (cela devrait évoluer ?) :

```
duree22(M,T0+T2) :- move(M,L,S1;S2), duree(M-1,T0),
    temps(S1,T1), temps(S2,T2), T1 < T2.        (cas de 2 soldats)
duree21(M) :- duree22(M,T). duree(M,T) :- duree22(M,T).
duree(M,T0+T) :- move(M,L,S), duree(M-1,T0), temps(S,T),
    not duree21(M).                             (cas où un seul soldat traverse)
```

Comme souvent en ASP dans un problème de planification, on décrit le but, puis une contrainte élimine les modèles ne satisfaisant pas ce but :

```
goal(M-1) :- at(M,arriv,0;1;2;3).     (le but est atteint en M)
goal0 :- goal(M).                     (le but est atteint)
:- not goal0.                         (élimine les AS ne satisfaisant pas le but)
```

On en déduit la durée totale : `duree1(T) :- goal(M), duree(M,T).`

Deux terminaisons. Un méta prédicat d'*optimisation* détecte les modèles minimisant la durée (1) ; ou une contrainte élimine les modèles qui dépassent le temps permis (2) :

1. `#minimize [ duree1(T)= T].`
2. `:- duree1(T), T > tmax.`

Les deux fournissent le même résultat avec cet énoncé car 60 est le temps minimum possible. L'optimisation est assez bien implémentée car elle accélère un peu (0.140s. avec et 0.170 s. sans) et elle accepte un énoncé moins coopératif. Il y a deux solutions :

```
move(1,arriv,2) move(1,arriv,3) move(1,arriv,lampe)
move(2,depart,2) move(2,depart,lampe)
move(3,arriv,0) move(3,arriv,1) move(3,arriv,lampe)
move(4,depart,3) move(4,depart,lampe)
move(5,arriv,2) move(5,arriv,3) move(5,arriv,lampe)
duree1(60) Optimization : 60          L'autre solution est similaire.
```

Le programme ASP est immédiat et naturel, et fournit la solution demandée sans effort de programmation. Chaque règle correspond de façon directe à une partie de l'énoncé. Cette facilité est due au très petit espace de recherche nécessaire : il est inutile de réfléchir au problème, traduire l'énoncé suffit. Remarquons, par rapport à un simple datalog, que l'utilisations de la négation par défaut `not` dans l'axiome du cadre facilite sensiblement l'écriture.

On va maintenant étudier un exemple où écrire le programme n'est pas si simple.

3 Un exemple récalcitrant : l'éléphant et les bananes

3.1 Présentation du vieil éléphant mangeur de bananes

> Un planteur a produit 3 000 bananes. Il ne dispose que d'un vieil éléphant qui consomme une banane au kilomètre et ne peut porter que 1000 bananes au plus. Le marché se trouve à 1000 km de la plantation. Combien de bananes le planteur pourra-t-il porter au maximum au marché ?

L'espace des solutions envisageables est grand. Il faut réfléchir avant de programmer sous peine de crash (`bad_alloc`). Il faut élaguer l'espace de recherche. Un gros avantage d'ASP apparaît ici : il est facile d'introduire le fruit de telles réflexions de façon

naturelle dans le programme. Mais des limites de l'aspect déclaratif apparaissent. Les matheux résolvent ce problème sans programmer. Au contraire, le but ici est de trouver des procédés les plus généraux possibles qui facilitent la programmation efficace.

Une méthode applicable à tout problème de ce type change les unités de mesure. On introduit un *pas* par exemple de 10 bananes (ou km), et on lance le programme. Cela fournit un résultat, optimal à cette granularité-là, qui permet de diminuer le pas, mais avec un encadrement issu du résultat grossier déjà obtenu. Il semble naturel de faire porter l'encadrement sur les distances des dépôts intermédiaires (indispensables ici). Les distances fournies par les solutions à 10 près permettront de lancer une seconde recherche, encadrée, jusqu'à obtenir le résultat à l'unité près. Malgré cela, il reste très difficile d'écrire un programme satisfaisant. Il faut donc élaguer l'espace de recherche par l'ajout d'hypothèses qui nécessitent une réflexion sur ce problème précis.

Appelons un ensemble d'hypothèses *pertinent* si, à partir de tout itinéraire satisfaisant l'énoncé, on peut construire un itinéraire satisfaisant ces hypothèses qui apporte au moins autant de bananes au marché. Un ensemble est *faiblement pertinent* s'il est pertinent dans tous les cas où existe au moins un itinéraire satisfaisant ces hypothèse : cela permet de résoudre le problème si les données numériques sont légèrement modifiées. On essaie d'abord les hypothèses fortes, et s'il n'y a aucune solution, on essaiera l'hypothèse plus faible, laquelle produira un espace de recherche plus grand. Démontrer la pertinence des hypothèses considérées ici dépasse le cadre de cet article :

1. Imposer le nombre de dépôts (deux intermédiaires) et d'étapes (9),
2. Se limiter aux trajets qui s'arrêtent dès le marché atteint,
3. et aux distances entre dépôts croissantes.
4. Imposer la charge maximale aux parcours aller, avec deux variantes :
   (a) charge maximale toujours, ou
   (b) charge maximale si disponible, sinon tout ce qui est disponible.
5. Revenir sans bananes pour les parcours retour (facile à démontrer).

3.2 Un premier programme, semblable au précédent

Voici un programme, facile à écrire, mais à piètres performances.

`#const ban = 3000.` `#const load = 1000.` `#const dist = 1000.`
Ex. d'"unité de calcul" : `#const step =100.` Constantes en "unité de calcul" :
#const bu = ban/step. #const leu = load/step. #const du = dist/step.

`d(0..3).` (Dépôts utilisés, y compris départ 0 et arrivée 3)
`#const stagemax = 9.` (Nombre maxi d'étapes)
`stage(0..stagemax).` ("Étapes" – ou "temps"– possibles)

Valeurs possible, en "unités" (quantité de bananes, charge, distance) :
`ldn(0..bu). len(1..leu). dn1(1..leu).`

`dd01(DD0,DD1,DD2)` : `DDi`, distance possible entre dépôts `i` et `i + 1`, tenant compte d'éventuelles restrictions imposées (grâce à `dd2valkm`), et de l'hypothèse 3 :

```
encadr(D) :- dd2valkm(D,Infkm,Supkm).
dd2val(D,1..leu/2-1) :- d(D), D+1 < 3, not encadr(D).
```

```
dd2val(D,Infkm/step..Supkm/step) :- D < 3,
     dd2valkm(D,Infkm,Supkm).
dd012(DD0,DD1,DD2) :- dd2val(0,DD0), dd2val(1,DD1),
     DD0 <= DD1, DD2 = du - DD0 - DD1, DD1 <= DD2.
dd01(DD0,DD1) :- dd012(DD0,DD1,DD2).
```

Première séparation des "ensembles réponse" (AS) : une seule configuration (atome en `descr`) de dépôts par AS. On génère un seul `ddescr(dd01(DD0,DD1))` par ensemble réponse grâce à la règle suivante (" `:` " signifie "tel que" pour clingo) :

```
1 { ddescr(dd01(DD0,DD1)) : dd012(DD0,DD1,DD2)} 1.
```

`dd(D1,D2,DD)` : Les dépôts D1 et D2 sont voisins et distants de DD, fournit les distances, et les dépôts D2 voisins de D1 (l'hypothèse 2 évite de définir `dd(3,2,_)`). Le symbole fonctionnel facultatif `ddO1` facilite l'écriture de certaines règles :

```
dd(0,1,DD0) :- dd012(DD0,DD1,DD2), ddescr(dd01(DD0,DD1)).
dd(1,0,DD0) :- dd012(DD0,DD1,DD2), ddescr(dd01(DD0,DD1)).
dd(1,2,DD1) :- dd012(DD0,DD1,DD2), ddescr(dd01(DD0,DD1)).
dd(2,1,DD1) :- dd012(DD0,DD1,DD2), ddescr(dd01(DD0,DD1)).
dd(2,3,DD2) :- dd012(DD0,DD1,DD2), ddescr(dd01(DD0,DD1)).
```

`move(S,D,LE)` : À l'étape S l'éléphant quitte D0 [move(S-1,D0,LE0)] pour aller vers un voisin D, avec la charge LE.
`loadD(S,D,LD)` : En S, la charge de D après le départ de l'éléphant de D0 est LD.
`loadDS(S,D,LD)` est un `loadD(S,D,LD)` en un S où LD est modifiée.
`loadE(S,LE)` : En S, la charge emportée par l'éléphant est LE.

Initialisations, mouvement fictif (S=0) :

```
move(0,0,0):-d(0). loadDS(0,0,bu). loadDS(0,D,0):-d(D),D>0.
```

Premier vrai mouvement (S=1) : quitte 0 pour 1 avec LE.

```
move(1,1,leu). loadDS(1,0,B-LE):-move(1,1,LE),loadD(0,0,B).
loadD(1,D,0) :- loadD(0,D,0), d(D), D>0.
```

Mouvement général : 1. "aller", de D à D+1 (hypothèse 4a, une règle supplémentaire serait nécessaire pour l'hypothèse faible 4b) :

```
{ move(S+1,D+1,leu)} :- move(S,D,LE), move(S-1,D0,LE0),
     dd(D0,D,DD0), dd(D,D+1,DD), loadD(S,D,LD0),
     LDA = LD0 + LE - DD0, LDA >= leu, S < stagemax.
```

2. "retour", de D à D-1 (hypothèse 5) :

```
{ move(S+1,D-1,DD)} :- move(S,D,LE), move(S-1,D0,LE0),
     dd(D0,D,DD0), dd(D,D-1,DD), loadD(S,D,LD0),
     LDA = LD0 + LE - DD0, LDA >= DD, S < stagemax.
 :- move(S,D,LE),move(S,D1,LE1),D<D1.
```

(Un seul `move` par étape `S`.)

Calcul de loadD (loadDS est utilisé pour le seul dépôt modifié à chaque étape) :

```
loadDS(S+1,D,LD1) :- move(S+1,D1,LE1), move(S,D,LE),
     move(S-1,D0,LE0), S>0, dd(D0,D,DD), loadD(S,D,LD),
     LD1 = LD+LE-DD-LE1, LD1 >= 0.
loadD(S,D,LD) :- loadDS(S,D,LD).
```

```
loadD(S+1,D,LD) :- move(S+1,D1,LE1), move(S,D0,LE0),
   loadD(S,D,LD), D != D0.
```
(Axiome du cadre pour les dépôts non modifiés.)

```
loadES(S) :- move(S,D,LE).
```
(S est une étape non finale.)

Élimine les trajets qui n'arrivent pas au marché (3) :

```
:- move(S,D,LE), D < 3, not loadES(S+1).
```

`LE` dans `move(stagemax,3,LE)` est la charge apportée au marché, en unité `step`, et `#maximize` ne garde que les AS avec le plus grand nombre possible de bananes :

```
loadAu(stagemax+1,LE-DD2) :- move(stagemax,ndep,LE),
     dd(ndep-1,ndep,DD2).
loadA(S,LU * step) :- loadAu(S,LU).
```
(en bananes)

```
#maximize [ loadA(S,LD) = LD].
```

`ddkm(I,J,DD * step) :- dd(I,J,DD), I < J.` (Distances entre les dépôts en km, utile pour l'utilisateur qui diminuera la valeur de l'unité interne.)

Ce programme assez facile à écrire (et donc, et c'est plus important, à modifier, par exemple pour améliorer ses performances) trouve la solution, mais il faut plusieurs tentatives car lancer avec `step=1` sans encadrement produit `bad_alloc`.

On peut obtenir laborieusement le résultat ainsi :
Premier lancement `step=100.` donne (en 1/3s) deux AS optimaux (similaires), avec en particulier `loadA(10,500) ddkm(1,2,300) ddkm(0,1,200)`
dont on s'inspire pour relancer le programme avec les données suivantes :
`step=33.dd2intervalkm(0,150,250).dd2intervalkm(1,250,350).`
On trouve encore 2 AS optimaux, en moins de 2s :
`loadA(10,528) ddkm(2,3,462) ddkm(1,2,330) ddkm(0,1,198)`
On relance, avec ces données :
`#const step =10. dd2valkm(0,190,210). dd2valkm(1,320,350).`
On trouve encore deux optimaux en 1 s., avec un peu plus de bananes à vendre (530) :
`loadA(10,530) ddkm(2,3,470) ddkm(1,2,330) ddkm(0,1,200)`

À ce stade, il est possible mais long d'obtenir le résultat, car
`#const step=1. dd2valkm(0,194,206). dd2valkm(1,324,336).`
produit `bad_alloc`, bien que les intervalles exigés soit assez petits.
L'encadrement très fin suivant donne le résultat, mais c'est peu satisfaisant :
`#const step=1. dd2valkm(0,197,203). dd2valkm(1,327,333).`

Résultat (un des 2 optimaux fournis) en plus de 6mn :
move(0,0,0) move(1,1,1000) move(2,0,200) move(3,1,1000) move(4,2,1000) move(5,1,333) move(6,0,200) move(7,1,1000) move(8,2,1000) move(9,3,1000) loadA(10,533) ddkm(2,3,467) ddkm(1,2,333) ddkm(0,1,200)

De nombreuses tentatives d'amélioration du programme ont échouées. Il en existe sûrement, mais l'aspect "déclaratif" affiché s'éloigne.

3.3 Un autre type de programme

Au lieu de faire, comme c'est naturel en ASP, une solution par AS, on place la solution dans un seul AS. Les règles qui génèrent des solutions sont remplacées par des règles qui calculent toutes les solutions dans un seul AS. La règle suivante est supprimée :

```
1 { ddescr(dd01(DD0,DD1)) : dd012(DD0,DD1,DD2)} 1.
```

Les deux règles générant les "mouvements généraux (allers et retours)" sont remplacées par des règles qui calculent tous les trajets possibles dans le même AS. Chaque trajet nécessite donc un indice complexe qui contient le couple (DD0,DD1) concerné et la liste des dépôts déjà visités, ainsi que des charges de l'éléphant. On utilise des symbole de fonction pour construire les nouveaux termes, remplaçant le prédicat `move` par une fonction `m`. Le programme complet est un peu long pour figurer ici, mais voici quelques règles. Le début, jusqu'à la règle de génération suivante, est inchangé.

```
1 { ddescr(dd01(DD0,DD1)) : dd012(DD0,DD1,DD2)} 1.
```

Cette règle est remplacée par un nouveau prédicat `dd` qui "indexe" chaque ancien `dd` par (DD0,DD1) (ici encore le symbole de fonction `dd01` est facultatif) :

```
dd01(DD0,DD1) :- dd012(DD0,DD1,DD2).
dd(dd01(DD0,DD1),0,1,DD0) :- dd01(DD0,DD1).
dd(dd01(DD0,DD1),1,0,DD0) :- dd01(DD0,DD1).
dd(dd01(DD0,DD1),1,2,DD1) :- dd01(DD0,DD1).
dd(dd01(DD0,DD1),2,1,DD1) :- dd01(DD0,DD1).
dd(dd01(DD0,DD1),2,3,DD2) :- dd012(DD0,DD1,DD2).
```

Voici comment sont décrits les trajets, en utilisant des fonctions `m` ("mouvement") et `i` ("itinéraire") au lieu du prédicat `move`. Un prédicat `itiopt` décrit les trajets `i` optimaux trouvés (voir un exemple ci-dessous dans le résultat). On utilise aussi le symbole de fonction `l` (pour "load"), `l(D,LD)` signifie que la charge du dépôt D est LD. Le premier paramètre de `i` (terme en `dd01`) contient l'information sur l'espacement des dépôts. Le second `S` indique le numéro de l'étape (redondant ici, contenu dans le terme en `m`, mais simplifie des règles). Le troisième est le terme en `m` qui représente la suite des dépôts dans leur ordre de visite. Le quatrième est le terme en `l` décrit ci-dessus.

Ainsi, un mouvement est maintenant représenté grâce à un symbole de fonction `m`, qui permet de le décrire en un seul terme, et non plus par un prédicat `move` qui ne décrit qu'une étape (mais était plus aisé à manipuler).

`m(M,D,LE)` : le mouvement obtenu à la suite du mouvement précédent `M` ,
`D` est le dépôt où l'éléphant arrive maintenant, `LE` étant la charge avec laquelle il est parti du dépôt `D0` précédent [donc `M` est `m(Ma,D0,LE0)`, `D0` étant le dépôt quitté, avec la charge `LE0`, `Ma` étant le mouvement d'avant].

L'écriture des règles n'est pas facilitée : l'aspect déclaratif est moins net !
Un mouvement est décrit à l'aide de la fonction `m` au lieu du prédicat `move` ci-dessus.

Initialisations : `#const mi = m(m0,0,0).`

```
i(dd01(DD0,DD1),0,m(m0,0,0),l(0,bu)) :- dd01(DD0,DD1).
i(dd01(DD0,DD1),0,m(m0,0,0),l(D,0)) :- dd01(DD0,DD1),
    d(D), D > 0.
i(dd01(DD0,DD1),1,m(m(m0,0,0),1,leu),l(0,bu-leu)) :-
    dd01(DD0,DD1).
i(dd01(DD0,DD1),1,m(m(m0,0,0),1,leu),l(D,0)) :-
    dd01(DD0,DD1), d(D), D>0.
```

Poursuite du trajet : cas d'un parcours "retour", D = D0 -1 :

```
i(dd01(DD0,DD1),S+1,m(m(m(Ma,Da,LEa),D0,LE0),D0-1,DP),l(D0,LD))
:- i(dd01(DD0,DD1),S,m(m(Ma,Da,LEa),D0,LE0),l(D0,LD0)),
   dd(dd01(DD0,DD1),D0,D0-1,DP),dd(dd01(DD0,DD1),Da,D0,DP0),
   S < stagemax, LD = LD0 + LE0 - DP0 - DP, LD >= 0.
```

La règle pour les parcours “aller” est similaire (là encore, deux règles avec l’hypothèse faible 4b). Voici l’axiome du cadre pour les dépôts autres que D0 (D = D0 +/- 1) :

```
i(dd01(DD0,DD1),S+1,m(m(m(Ma,Da,LEa),D0,LE0),D,LE),l(Di,LDi))
:-i(dd01(DD0,DD1),S+1,m(m(m(Ma,Da,LEa),D0,LE0),D,LE),l(D0,LD)),
   S < stagemax, Di != D0,
   i(dd01(DD0,DD1),S,m(m(Ma,Da,LEa),D0,LE0),l(Di,LDi)).
```

Le reste du programme est immédiat. Les performances sont très supérieures, et on peut considérer le problème résolu. Il est dommage que la méthode préconisée en général pour les problèmes de planification (un AS par tentative, ou presque) rencontre un problème de taille mémoire. L’idéal serait que le système soit capable de traiter le programme en `move` de façon aussi économique que ce programme en `m`. En effet, dans le programme en `m`, l’utilisateur guide les calculs. Remarquons qu’ici la structure du programme est très simple (au détriment de la facilité d’écriture). Il faudrait un système capable de constituer quelque chose de ce genre à partir des données en `move` et `at`. Est-ce faisable ?

Voici les performances de ce programme : `step=1` sans encadrement ne passe pas mais la suite (1) `step =10` (seul) (2) `step=1, dd2valkm(0,170,230)` `dd2valkm(1,300,360)` passe sans problème en environ 5s, et donne ceci :

```
itiopt(dd01(200,333),loadAU(533),loadA(533),
  i(dd01(200,333),9,m(m(m(m(m(m(m(m(m(m(m0,0,0),1,1000),
    0,200),1,1000),0,200),1,1000),2,1000),1,333),2,1000),
    3,1000), l(2,1)))
```

et un autre trajet similaire.

3.4 La surprise gringo (ne résout pas tout)

Afin de voir comment limiter l’explosion en taille, lançons gringo (“instantiateur” sans le “solveur” clasp), avec `step(1)` seul (sans intervalle). Cela donne le résultat (en 96 s sur l’ordinateur utilisé, mais l’essentiel du temps est sans doute utilisé pour écrire le fichier résultat car le filtrage ne fonctionne pas avec gringo). Cela donne donc le résultat cherché directement, sans utiliser d’unité interne ni d’encadrement a priori (mais en utilisant les hypothèses 1 – 5 ci-dessus). L’ennui est que clingo (gringo + clasp) ne sait pas que gringo seul suffirait ici à donner le résultat. Et il semble (d’après Torsten Schaub) qu’il ne soit pas si facile que cela de faire que clingo s’en aperçoive assez tôt pour éviter d’utiliser de la mémoire de façon inutile ici, et catastrophique (gringo pourrait-il s’en apercevoir ?). On se situe donc ici dans une marge où gringo seul passe et pas clingo, mais des tests avec l’hypothèse faible suggèrent que cette marge n’est pas si étroite que cela, et que donc détecter ces cas augmenterait les possibilités de clingo.

4 Conclusion

L'état actuel des systèmes ASP permet d'envisager à court terme une réelle utilisation pour des problèmes de planification de difficulté moyenne. Il s'agit d'un formalisme facilitant l'écriture et la modification des programmes, alors qu'un formalisme spécifique demande une phase de familiarisation. On a donné deux exemples : un cas simple où la traduction ASP est quasi littérale et très naturelle, un plus complexe nécessitant des optimisations qui sont elles aussi aisément traduisibles en ASP. Pour une utilisation plus aisée, les systèmes actuel devraient encore être un peu améliorés, ce qui est naturel pour un formalisme général et récent. L'exemple de l'éléphant, plus aisé à traiter à l'aide d'un seul ensemble réponse (ce qui contredit la plupart des préconisations usuelles) semble indiquer qu'il serait bon que l'utilisateur puisse indiquer au système comment traiter certains prédicats. Ainsi, dans la version "usuelle" avec un essai par ensemble réponse, le fait pour l'utilisateur de préciser que `loadD` doit satisfaire l'axiome du cadre est susceptible de faciliter l'écriture du programme, mais surtout l'efficacité du calcul : des méthodes spécifiques permettraient d'épargner de la mémoire. Un méta prédicat du genre `#cadre` où `#cadre loadD/1/1/1.` signifierait que `loadD` doit être traité comme satisfaisant l'axiome du cadre : le premier paramètre étant le temporel, le second l'indice et le troisième la valeur soumise à l'axiome du cadre, devant rester identique, pour chaque indice, sauf si une règle demande de la modifier. On ne peut pas aller trop loin dans cette direction car on reviendrait à un système spécifique. Mais le gain en facilité d'expression et de calcul, ainsi que le caractère général de ce comportement, semble justifier ce genre d'ajout. On peut aussi envisager soit un méta prédicat `#mvt` signalant ici que `move` doit être traité de façon particulière, ou mieux (si possible)une détection automatique de ce genre de situation. Il semble que de telles petites évolutions des systèmes actuels, orientées confort de l'utilisateur, amélioreraient sensiblement leurs performances effectives (et non pas théoriques).

Références

BARAL C. (2003). *Knowledge representation, reasoning and declarative problem solving*. Cambridge University Press.

CALIMERI F., COZZA S., IANNI G. & LEONE N. (2009). An ASP System with Functions, Lists, and Sets. In LPNMR'09, p. 483–489.

DENECKER M., VENNEKENS J., BOND S., GEBSER M. & TRUSZCZYŃSKI M. (2009). The 2nd Answer Set Programming Competition. LPNMR'09 p. 637–654.

GEBSER M., KAUFMANN B. & SCHAUB T. (2009). The conflict-driven answer set solver *clasp* : Progress report. LPNMR'09, p. 509–514.

LEFÈVRE C., NGOMA S. & NICOLAS P. (2010). ASPeRiX : un solveur ASP du premier ordre. *IAF 2010*
`://gdri3iaf.info.univ-angers.fr/spip.php?article121`

LEONE N., PFEIFER G., FABER W., EITER T., GOTTLOB G., PERRI S. & SCARCELLO F. (2006). The DLV System for Knowledge Representation and Reasoning. *ACM Transactions on Computational Logic (TOCL)*, **7**(3), 499–562.

Reconstruction de la sémantique d'admissibilité en défauts et programmation logique

Farid Nouioua, Vincent Risch

LSIS - UMR CNRS 6168,
Faculté des Sciences de Saint–Jérôme,
avenue Escadrille Normandie–Niemen,
13397 Marseille cedex 20, France
{farid.nouioua,vincent.risch}@lsis.org

Résumé : Nous proposons de spécialiser la notion d'extension justifiée de façon à représenter les extensions admissibles (et donc préférées et stables) au sein d'une théorie de défauts. En nous appuyant sur la correspondance entre extensions justifiées et ι-answer sets (iota answer sets), nous montrons alors que tout ensemble admissible d'arguments d'un cadre argumentatif abstrait peut être directement calculé à partir des ι-answer sets du programme logique correspondant. En particulier, il est possible d'étendre un tel programme par un ensemble de contraintes *ad hoc* permettant d'en filtrer directement les ensembles admissibles.

Mots-clés : Argumentation abstraite, défauts, programmation logique

1 Introduction

Depuis l'introduction de la notion de cadre argumentatif abstrait par Phan Minh Dung (cf. Dung (1993a), Dung (1993b), Dung (1995)), plusieurs auteurs ont considéré les liens associant cette théorie avec les défauts ou la programmation logique, (Dung (1995), Bondarenko *et al*. (1997), Nieves *et al*. (2008), Egly *et al*. (2010)). L'ensemble de ces travaux procède d'une vision commune qui souligne avec succès le rôle fondamental que joue au cœur de ces formalismes la notion de conflit entre deux informations. Le cadre argumentatif apparaît donc comme une approche particulièrement éclairante du mécanisme non-monotone mis en oeuvre au sein d'une théorie de défaut ou d'un programme logique. Sans aucunement remettre en cause l'intérêt profond de cette démarche, nous proposons toutefois de revenir à la question inverse, à notre sens insuffisament explorée, à savoir : qu'est-ce qu'une théorie de défauts ou un programme logique peuvent nous apprendre d'un cadre argumentatif abstrait ? En particulier, parce que tout cadre argumentatif apparaît formellement comme un fragment d'une théorie de défauts, il nous semble judicieux d'explorer de façon plus précise comment l'un des concepts de base de l'approche argumentative – l'extension admissible – est apparenté

au même concept dans les approches non-monotones citées. Nos motivations sont multiples : nous espérons élucider de façon plus précise les liens entre ces différents formalismes et, à un moment où se pose avec acuité la question des approches logiques de la notion d'argument (par exemple dans Amgoud & Besnard (2009)), précisément être en mesure de proposer les bases d'une réévaluation de l'argumentation abstraite à l'aune de la logique. De ce point de vue, nous pensons aussi qu'il serait intéressant de connaître les conditions formelles permettant d'appliquer aux cadres argumentatifs les très nombreux travaux menés du côté de la logique sur les préférences, sur la notion d'inférence cumulative ou sur d'autres propriétés. Enfin, en replaçant les cadres argumentatifs dans la perspective de la programmation logique, nous espérons pouvoir mieux bénéficier du remarquable pouvoir calculatoire des systèmes qui sont actuellement proposés. Nous commençons donc par montrer que tout ensemble d'arguments, maximal, sans conflit, correspond très exactement à la notion d'extension justifiée d'une théorie de défauts, initiée par Lukaszewicz (1988), et par contrecoup aux ι-answer sets (ou ι-as) d'un programme logique (cf. Gebser *et al*. (2009)). A partir de contraintes spécifiques sur les extensions justifiées d'une théorie de défauts, nous proposons ensuite une caractérisation de la notion d'ensemble admissible d'arguments d'une théorie argumentative abstraite obtenue à partir de cette théorie de défauts. Etant donnée la correspondance entre extensions justifiées et ι-as, nous montrons alors que tout ensemble admissible d'un cadre argumentatif abstrait peut être caractérisé à partir des ι-as du programme logique équivalent. Il devient notamment possible d'adjoindre à ce programme logique un ensemble de contraintes permettant d'exfiltrer directement à partir des ι-as les ensembles admissibles d'arguments. Il est à noter que les ι-as héritent de la propriété de semi-monotonie des extensions justifiées, ce qui souligne leur rôle central dans le cadre d'un traitement incrémental des arguments à partir d'un programme logique. Dans le second paragraphe, nous rappelons les concepts de base liés aux cadres argumentatifs abstraits Dung (1995) et des théories de défauts Reiter (1980). Dans le troisième paragraphe, nous établissons une correspondance stricte entre ensembles d'arguments maximal sans conflit et extensions justifiées, ce qui permet ensuite la caractérisation de tout ensemble admissible d'arguments comme un cas particulier d'extension justifiée. Le quatrième paragraphe étend cette caractérisation aux ι-as et au calcul d'extensions admissibles à l'aide de contraintes d'intégrité.

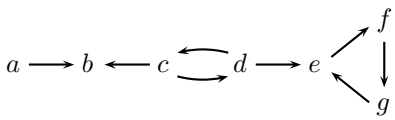
2 Préliminaires

Nous rappelons ici brièvement quelques définitions de base concernant les cadres argumentatifs abstraits d'abord, les théories de défauts ensuite. La notion de programme logique est abordée plus loin.

Un *cadre argumentatif* est une paire $\langle AR, attaque\rangle$ telle que AR est un ensemble et $attaque$ est une relation entre les éléments AR i.e. $attaque \subseteq AR \times AR$. Tout élément de AR est appelé *argument* et $a\ attaque\ b$ signifie qu'il y a une attaque de a à b. Dans ce cas a est appelé *contre-argument* de b. Par extension, un ensemble $S \subseteq AR$ *attaque* un argument $a \in AR$ ssi un argument de S attaque a. Au contraire, S *défend* a ssi pour tout $b \in AR$, si $b\ attaque\ a$ alors $S\ attaque\ b$. Dans ce cas, a est aussi dit être *acceptable au regard de* S. La relation d'attaque induit différents degrés de cohérence

entre arguments. En premier lieu $S \subseteq AR$ est dit *sans conflit* ssi il n'y a aucun a et b de S tel que a *attaque* b. S est *admissible* ssi S est sans conflit et défend tous ses éléments. Une *extension préferée* est un sous-ensemble $\subseteq$-maximal admissible de AR. Enfin S est une *extension stable* ssi S est sans conflit et attaque tout argument qui n'est pas dans S.

Exemple 1 Soit le cadre argumentatif *AF* suivant, les flèches représentant la relation d'attaque entre les arguments a, b, c, d, e, f, g :



Les ensembles admissibles sont : $\emptyset$, $\{a\}$, $\{c\}$, $\{d\}$, $\{a, c\}$, $\{a, d\}$, $\{d, f\}$, $\{a, d, f\}$. Les extensions préferées sont $\{a, c\}, \{a, d, f\}$. L'unique extension stable est $\{a, d, f\}$. Quelque soit le type d'extension considérée (admissible, préferée, ou stable), celle-ci est un sous-ensemble d'un ensemble $\subseteq$-maximal sans conflit parmi $\{a, c, e\}, \{a, c, f\}$, $\{a, c, g\}, \{a, d, f\}, \{a, d, g\}, \{b, d, f\}, \{b, d, g\}, \{b, e\}$.

Un défaut est une expression de la forme $\frac{\alpha:\beta}{\gamma}$, α, β, γ y étant des formules closes du premier ordre. α est appelé *prérequis*, β *justification*, et γ *conclusion*. Etant donné un ensemble de défauts D, $PREREQ(D)$, $JUST(D)$, et $CONS(D)$ font respectivement référence aux ensembles des prérequis, des justifications et des conséquences des défaults de D. Une *théorie de défauts* Δ est la paire (W, D), W y désignant un ensemble de formules du premier ordre et D un ensemble de défauts. Intuitivement, une conséquence de défaut est valide si le prérequis de ce défaut est valide, et si rien ne contredit la justification. La conséquence essentielle de cette approche est capturée par la notion d'extension. Les caractérisations que nous utilisons des R- et J- extensions (notions respectivement dues à Reiter (1980) et Lukaszewicz (1988)) sont données dans Risch (1996). Soit $\Delta = (W, D)$ une théorie de défauts. Un sous-ensemble D' de D est dit *enraciné* ssi pour tout $d \in D'$, il existe une séquence $d_0, d_1, \ldots, d_k$ d'éléments de D' telle que (1) $PREREQ(\{d_0\}) \in Th(W)$, (2) Pour $1 \leq i \leq k-1, PREREQ(\{d_{i+1}\}) \in Th(W) \cup CONS(\{d_0, d_1, \ldots d_i\})$, et $d_k = d$. Soit alors D' un sous-ensemble quelconque de D ; $E = Th(W \cup CONS(D'))$ est : (1) une *J-extension* de Δ ssi D' est un sous ensemble enraciné $\subseteq$-maximal de D tel que pour tout $\beta \in JUST(D')$, $\neg\beta \notin E$; (2) une *R-extension* de Δ ssi elle est une J-extension, et pour tout défaut $d \in D \setminus D', d = \frac{\alpha:\beta}{\gamma}$, $\alpha \notin E$ ou $\neg\beta \in E$. Lorsque $E = Th(W \cup CONS(D'))$ est une extension (J- ou R-), l'ensemble D' est appelé ensemble de *défauts générateurs* de E, et noté $GD(E, \Delta)$. Il est à remarquer que les J-extensions disposent de propriétés intéressantes pour le calcul : elles existent toujours, et les R-extensions en représentent un cas particulier aisément caractérisable Risch (1996). De plus, elles sont semi-monotones : ajouter de nouveaux défauts à une théorie ne remet pas en cause les J-extensions déjà produites. Dans la suite, nous n'aurons pas en général l'utilité du pouvoir expressif de la logique du premier ordre dans

son ensemble, et nous nous restreindrons à des théories propositionelles de défauts.

Exemple 2 Soit la théorie de défauts $\Delta = (\emptyset, \{\frac{c:\neg a \wedge \neg b}{d}, \frac{\top:\neg a}{a \vee b}, \frac{\top:a}{a}\})$. Δ a deux J-extensions $E^1 = Th(a \vee b)$, $E^2 = Th(a)$ parmi lesquelles seule E^2 est une R-extension ; $GD(E^1, \Delta) = \{\frac{\top:\neg a}{a \vee b}\}$, $GD(E^2, \Delta) = \{\frac{\top:a}{a}\}$.

Nous en venons à présent aux théories de défauts en tant que cadres argumentatifs. Soit $\Delta = (W, D)$ une théorie de défauts et $\Lambda = \{\beta_1, \ldots, \beta_n\} \subseteq JUST(D)$. Une formule λ est dite *conséquence défaisable de* Δ *et* Λ (Dung (1995)) s'il existe une séquence $(e_0, e_1, \ldots e_n)$ avec $e_n = \lambda$ telle que, pour tout $e_i, 0 \leq i \leq n$, (1) $e_i \in W$, ou (2) e_i est consequence logique de membres précédents de la séquence, ou (3) e_i est la conclusion γ d'un défaut $\frac{\alpha:\beta}{\gamma}$ dont le prérequis α est un membre précédent dans la séquence et dont la justification β appartient à Λ. Λ est appelé support de λ au regard de Δ. La théorie Δ est alors interprétée comme cadre argumentatif de la manière suivante : (1) $AR_\Delta = \{(\Lambda, \lambda) \mid \Lambda \subseteq JUST(D), \Lambda$ est support de λ au regard de $\Delta\}$; (2) $(\Lambda, \lambda)\ attaque_\Delta\ (\Lambda', \lambda')$ ssi $\neg\lambda \in \Lambda'$. Inversement, tout cadre argumentatif $AF = \langle AR, attaque\rangle$ peut être traduit modulairement par une théorie de défauts $\Delta_{AF} = (\emptyset, D_{AF})$ de manière suivante : (1) $D_{AF} = \{\frac{:\top}{a} \mid a \in AR\}$; (2) pour tout argument a, b de AR tel que $b\ attaque\ a$, si d_a dénote l'unique defaut de D_{AF} associé à a, ajouter $\neg b$ à $JUST(\{d_a\})$. En d'autres termes, à chaque argument de $\langle AR, attaque\rangle$ est associé un unique défaut sans prérequis dont la conséquence est l'argument lui-même, et dont l'ensemble de justifications est le plus petit ensemble contenant $\top$ et les négations des arguments qui attaquent cet argument.

Exemple 1 (suite) Soit le cadre argumentatif *AF* donné plus haut. La théorie correspondante de défauts est donnée par $\Delta_{AF} = (\emptyset, \{\frac{:\top}{a}, \frac{:\neg a, \neg c}{b}, \frac{:\neg d}{c}, \frac{:\neg c}{d}, \frac{:\neg d, \neg g}{e}, \frac{:\neg e}{f}, \frac{:\neg f}{g}\})$

Il est à noter que si la traduction d'un cadre argumentatif en une théorie de défauts génère un nombre de défauts égal au nombre d'arguments de départ, la traduction en sens opposé (d'une théorie de défauts vers un cadre argumentatif) génère généralement un nombre beaucoup plus important (potentiellement infini) d'arguments. Cette complexification est essentiellement due au fait qu'on passe du langage propositionnel pris dans son ensemble (pour les défauts) à un fragment réduit aux atomes propositionnels, avec ou sans négation (pour les arguments). Un autre point remarquable est qu'à l'image des approches logiques de l'argumentation (cf. Besnard & Hunter (2008)), les traductions proposées ici conduisent à définir une notion d'argument avec structure, sous la forme $(support, conclusion)$. Enfin, remarquons encore que la notion de *conséquence défaisable* permet précisément d'exprimer tout argument sous forme structurée en faisant en sorte d'éliminer les prérequis de la théorie initiale de défauts.

Comme cela est montré dans Dung (1995), il y a une correspondance exacte entre R-extensions d'une théorie de défauts et extensions stables d'un cadre argumentatif abstrait. Soit A une théorie du premier ordre, et $A' \subseteq AR_\Delta$ un ensemble d'arguments obtenus à partir d'une théorie de défauts Δ. On définit (1) $arg(A) = \{(\Lambda, \lambda) \in AR_\Delta \mid \forall\beta \in \Lambda, \beta \cup A \not\vdash\}$; (2) $flat(A') = \{\lambda \mid \exists(\Lambda, \lambda) \in A'\}$. Soit Δ une théorie de

défauts. Alors (lemme 42 et théorème 43 de Dung (1995)) : (1) E est une R-extension de Δ ssi $E = \mathit{flat}(arg(E))$; (2) Etant donnée E une R-extension de Δ, $arg(E)$ est une extension stable de $\langle AR_\Delta, attaque_\Delta \rangle$; (3) Etant donnée E' une extension stable de $\langle AR_\Delta, attaque_\Delta \rangle$, $\mathit{flat}(E')$ est une R-extension de Δ. Nous établissons le résultat préliminaire suivant :

Théorème 1
Soit $\Delta = (W, D)$ une théorie de défauts et $\langle AR_\Delta, attaque_\Delta \rangle$ le cadre argumentatif correspondant. Alors :

(1) $(\Lambda, \lambda) \in AR_\Delta$ ssi il existe $D' \subseteq D$, D' enraciné dans W tel que $\Lambda = JUST(D')$ et $\lambda \in Th(W \cup CONS(D'))$;

(2) $\mathit{flat}(AR_\Delta) = \bigcup_{\substack{D' \in 2^D \\ D' \text{grounded}}} Th(W \cup CONS(D'))$.

A partir de ce résultat, il est immédiat que dans AR_Δ, un argument attaque un argument (Λ, λ) ssi il existe $\beta \in \Lambda$ tel que $\neg\beta \in \mathit{flat}(AR_\Delta)$.

3 Admissibilité à partir des J-extensions

Envisageons de façon plus approfondie la question de l'obtention d'arguments de AR_Δ à partir d'une théorie de défauts Δ. Notre idée est d'établir une correspondance aussi fine et générale que possible entre tout ensemble de défauts applicables et un sous-ensemble d'arguments, de façon à mettre en relation les notions respectives d'extensions dans AR_Δ et Δ. Le théorème 1 souligne le rôle crucial joué par les sous-ensembles de D dans la constitution d'un argument (Λ, λ) de AR_Δ, puisque pour tout argument de ce type, il existe $D' \subseteq D$ tel que $\Lambda = JUST(D')$ et $\lambda \in Th(W \cup CONS(D'))$. Ce dernier ensemble est précisément de la forme prise par les différentes sortes d'extensions de Δ (à partir de différents types de contraintes sur sa génération). En d'autres termes, nous souhaitons être capable de relier certains types d'ensembles de AR_Δ avec certains types d'extensions de Δ *via* $JUST(D')$ (pour les supports d'arguments) et $Th(W \cup CONS(D'))$ (pour les supports des *mêmes* arguments). Dans ce but, nous commençons par introduire les définitions suivantes :

Definition 1
Etant donnée une théorie de défauts $\Delta = (W, D)$, et $D' \subseteq D$, soit $AR_\Delta(D') = \{(\Lambda, \lambda) \in AR_\Delta \mid \lambda \in Th(W \cup CONS(D'))\}$

Clairement, $AR_\Delta(D) = AR_\Delta$. Soit $E = Th(W \cup CONS(D'))$; clairement encore, par construction, $E = \mathit{flat}(AR_\Delta(D'))$. Le théorème suivant explicite le lien existant entre les ensembles d'arguments issus de $AR_\Delta(D')$ et ceux issus de $arg(Th(W \cup CONS(D')))$:

Théorème 2
Etant donnée une théorie de défauts $\Delta = (W, D)$ et $D' \subseteq D$, soit $E = Th(W \cup CONS(D'))$. Alors : (1) $AR_\Delta(D') \subseteq arg(E)$ ssi $\forall \beta \in JUST(D'), \beta \cup E \nvdash$; (2) $AR_\Delta(D') = arg(E)$ ssi E est une R-extension de Δ.

Exemple 2 (suite) Soit la théorie de défauts $\Delta = (\emptyset, D)$ considérée plus haut, on rappelle que Δ a deux J-extensions E^1 et E^2 parmi lesquelles E^2 seule est une R-extension. Il est aisé de vérifier par exemple que $(\{\neg a\}, a \vee b)$ ou $(\{a\}, a)$ appartiennent à $arg(E^1)$ mais que $(\{a\}, a) \notin AR_\Delta(GD(E^1, \Delta))$, ce qui illustre le fait qu'en général, pour $D' \subseteq D$ quelconque, $arg(Th(W \cup CONS(D'))) \not\subseteq AR_\Delta(D')$. En d'autres termes $AR_\Delta(D')$ correspond précisément aux arguments de AR_Δ obtenus à partir des défauts de D', ni plus ni moins. Par ailleurs, $AR_\Delta(GD(E^2, \Delta)) = arg(E^2)$, puisque E^2 est une R-extension.

A partir de ces résultats, il devient aisé de caractériser un ensemble sans conflit d'arguments à partir d'un sous-ensemble D' de défauts :

Théorème 3
Etant donnée une théorie de défauts $\Delta = (W, D)$, soit $D' \subseteq D$, D' enraciné dans W et $E = Th(W \cup CONS(D'))$. $AR_\Delta(D')$ est sans conflit ssi $\forall \beta \in JUST(D'), \neg\beta \notin E$.

Corollaire 1
Soit $\Delta = (W, D)$ une théorie de défauts et $\langle AR_\Delta, attaque_\Delta \rangle$ le cadre argumentatif correspondant. (1) Soit E_Δ un sous-ensemble sans conflit $\subseteq$-maximal quelconque de AR_Δ. Alors $flat(E_\Delta)$ est une J-extension de Δ. (2) Soit E une J-extension quelconque de Δ. Alors $AR_\Delta(GD(E, \Delta))$ est un sous-ensemble sans conflit $\subseteq$-maximal de AR_Δ.

La question est donc d'arriver à filtrer à l'intérieur des J-extensions en vue d'obtenir une représentation des extensions admissibles au sein d'une théorie de défauts. C'est ce que permet le théorème de caractérisation suivant :

Théorème 4
Soit $\Delta = (W, D)$ une théorie de défauts et $\langle AR_\Delta, attaque_\Delta \rangle$ le cadre argumentatif correspondant. Soit $\{D_{i, i \in \mathbb{N}}\}$ une énumération de tous les sous-ensembles D qui sont enracinés dans W, avec $E(D_i) = Th(W \cup CONS(D_i))$ pour tout $i \in \mathbb{N}$. Pour tout $D' \subseteq D$, $AR_\Delta(D')$ est un ensemble admissible de $\langle AR_\Delta, attaque_\Delta \rangle$ ssi (1) il existe $i \in \mathbb{N}$ tel que $D' = D_i$ (2) il existe $j \in \mathbb{N}$ tel qu $D' \subseteq D_j$ et $E(D_j)$ est une J-extension (3) pour tout $k \in \mathbb{N}$, $(\forall \beta \in JUST(D'), \neg\beta \in E(D_k) \Rightarrow (\exists \gamma \in E(D'), \neg\gamma \in JUST(D_k))$

Ce que dit ce théorème, c'est que pour qu'un sous-ensemble d'une J-extension puisse correspondre à un ensemble admissible, il est nécessaire de vérifier que si la négation d'une justification utilisée dans la construction de ce sous-ensemble est trouvée dans un ensemble enraciné (i.e. un argument de l'ensemble est attaqué) alors la négation d'une formule de cet ensemble enraciné doit elle-même être trouvée parmi les justifications du sous-ensemble de départ (i.e. l'argument attaqué est défendu). En d'autres termes, pour calculer les ensembles admissibles au sein d'une théorie de défauts (et donc les extensions préférées pour les sous-ensembles $\subseteq$-maximaux), il suffit de filtrer à l'intérieur des J-extensions de cette théorie de défauts. On notera que le cas pour lequel la consequence $(\exists \gamma \in E(D'), \neg\gamma \in JUST(D_k))$ de l'implication (3) est toujours vraie permet

de caractériser les extensions stables, puisque celles-ci correspondent directement aux R-extensions telles que caractérisées plus haut.

4 Liens avec les ι-answer sets

Dans le contexte de la programmation logique, il a été montré par Delgrande *et al.* (2003) que la notion de J-extension correspond à un relâchement sur les contraintes de génération d'un answer set. Cette idée est complètement formalisée avec la notion d'ι-as introduite dans Gebser *et al.* (2009) comme exacte contrepartie de celle de J-extension au sein de toute théorie de défauts équivalente à un programme logique. A la suite de Gebser *et al.* (2009), rappelons qu'un programme logique normal est un ensemble fini de règles de la forme

$$p_0 \leftarrow p_1, \ldots, p_m, not\ p_{m+1}, \ldots, not\ p_n$$

chaque p_i y désigant un *atome*. Pour une règle r, on pose $head(r) = p_0$ et $body(r) = \{p_1, \ldots, p_m, not\ p_{m+1}, \ldots, not\ p_n\}$ De plus, $body^+(r) = \{p_1, \ldots, p_m\}$ et $body^-(r) = \{not\ p_{m+1}, \ldots, not\ p_n\}$. Etant donné un programme logique Π, $head(\Pi) = \{head(r) \mid r \in \Pi\}$, $body(\Pi) = \{body(r) \mid r \in \Pi\}$, $body^+(\Pi) = \bigcup_{r\in\Pi} body^+(r)$, $body^-(\Pi) = \bigcup_{r\in\Pi} body^-(r)$. Enfin, on notera qu'une tête de règle qui est vide est assimilée à $\bot$, alors qu'un corps vide est assimilé à $\top$. Un programme Π est dit *basique* si $body^-(\Pi) = \emptyset$. Tout programme basique Π a un unique modèle $\subseteq$-minimal, denoté par $Cn(\Pi)$, et qui est le plus petit ensemble d'atomes clos pour les règles de Π. Le *reduct* d'un programme logique Π relativement à un ensemble X d'atomes est le programme basique $\Pi^X = \{head(r) \leftarrow body^+(r) \mid r \in \Pi, body^-(r) \cap X = \emptyset\}$. Si $X = Cn(\Pi^X)$, alors X est un *answer set* de Π. On pose $Cn^+(\Pi) = Cn(\Pi^\emptyset) = Cn(head(r) \leftarrow body^+(r) \mid r \in \Pi)$. Soit Π un programme logique et X un ensemble d'atomes, X est un ι-as de Π si $X = Cn^+(\Pi')$ pour un certain $\Pi' \subseteq \Pi$ maximal, tel que (1) $body^+(\Pi') \subseteq Cn^+(\Pi')$ et (2) $body^-(\Pi') \cap Cn^+(\Pi') = \emptyset$. Les ι-as d'un programme Π correspondent aux extensions justifiées d'une théorie de défauts donnée par la traduction modulaire suivante : chaque règle r de Π produit un défaut $\frac{\bigwedge body^+(r) : \neg|body^-(r)|}{head(r)}$ (avec $| S |= \{a \mid not\ a \in S\}$), et $W = \emptyset$. Etant donné $\Pi' \subseteq \Pi$, soient $AR_\Pi(\Pi') = \{(body^-(r), head(r)) \mid r \in \Pi'\}$, et $flat_\Pi(\Pi') = \{head(r) \mid r \in \Pi'\}$. Soit $\Delta_\Pi = (\emptyset, D_\Pi)$, une théorie de défauts obtenue par la traduction donnée ci-dessus, alors clairement $AR_\Delta(D') = AR_\Pi(\Pi')$ et $flat(AR_\Delta(D')) = flat_\Pi(\Pi')$ pour tout $D' \subseteq D_\Pi$ obtenu à partir de $\Pi' \subseteq \Pi$ *via* cette traduction. On remarquera qu'à partir de cette traduction et du théorème 1 toutes les règles ont un corps positif vide. A partir du corollaire 1 on obtient immédiatement :

Corollaire 2
Soit Π un programme dont le corps positif est vide et $AR_\Pi(\Pi)$ le cadre argumentatif correspondant. Pour tout $\Pi' \subseteq \Pi$, $AR_\Pi(\Pi')$ est un sous-ensemble sans conflit $\subseteq$-maximal de $AR_\Pi(\Pi)$ ssi $head(\Pi')$ est un ι-as de Π.

Du théorème 4 on obtient directement :

Corollaire 3
Soit Π un programme dont le corps positif est vide et $AR_\Pi(\Pi)$le cadre argumentatif correspondant. Soient $X_1, \ldots, X_k$ une collection quelconque de tous les ι-as de Π et $AR_\Pi(\Pi_1), \ldots, AR_\Pi(\Pi_k)$ les sous-ensembles sans conflit $\subseteq$-maximaux de $AR_\Pi(\Pi)$ correspondants. Pour tout $\Pi' \subseteq \Pi$, $AR_\Pi(\Pi')$ est un ensemble admissible de $AR_\Pi(\Pi)$ ssi il existe i, $1 \leq i \leq k$, tel que $\Pi' \subseteq \Pi_i$ et $(\forall r' \in \Pi')(\forall r \in \Pi \setminus \Pi')(body^-(r') \cap head(r) \neq \emptyset) \Rightarrow (head(\Pi') \cap body^-(r) \neq \emptyset)$.

Exemple 1 (suite) A partir du cadre argumentatif *AF* proposé plus haut, on obtient le programme Π_{AF} suivant :

$$\begin{array}{llll} r^1: & a & \leftarrow & \\ r^2: & b & \leftarrow & not\ a, not\ c \\ r^3: & c & \leftarrow & not\ d \\ r^4: & d & \leftarrow & not\ c \end{array} \qquad \begin{array}{llll} r^5: & e & \leftarrow & not\ d, not\ g \\ r^6: & f & \leftarrow & not\ e \\ r^7: & g & \leftarrow & not\ f \end{array}$$

Huit ι-as sont générés : $X_1 = \{a, c, e\}$, $X_2 = \{a, c, f\}$, $X_3 = \{a, c, g\}$, $X_4 = \{a, d, f\}$, $X_5 = \{a, d, g\}$, $X_6 = \{b, d, f\}$, $X_7 = \{b, d, g\}$, $X_8 = \{b, e\}$. Par exemple, et en particulier, X_1 et X_5 correspondent respectivement aux sous-ensembles sans conflit $\subseteq$-maximaux de $AR_\Pi(\Pi_{AF})$ suivants : $AR_\Pi(\Pi_1) = \{(\{\}, a), (\{not\ d\}, c), (\{not\ d, not\ g\}, e)\}$, $AR_\Pi(\Pi_5) = \{(\{\}, a), (\{not\ c\}, d), (\{not\ f\}, g)\}$ avec $\Pi_1 = \{r^1, r^3, r^5\}, \Pi_4 = \{r^1, r^4, r^6\}, \Pi_5 = \{r^1, r^4, r^7\}$. En appliquant le corollaire 3, il est aisé de vérifier qu'alors que $(\{not\ f\}, g)$ de $AR_\Pi(\Pi_5)$ *attaque* $(\{not\ d, not\ g\}, e)$ de $AR_\Pi(\Pi_1)$ (i.e. $body^-(r^5) \cap head(r^7) \neq \emptyset$), $AR_\Pi(\Pi_1)$ ne se défend pas contre cette attaque (i.e. $head(\Pi_1) \cap body^-(r^7) = \emptyset$). On en déduit que $AR_\Pi(\Pi_1)$ n'est pas admissible et que l'argument $(\{not\ d, not\ g\}, e)$ doit être retiré de $AR_\Pi(\Pi_1)$ de façon à obtenir $\{(\{\}, a), (\{not\ d\}, c)\}$ comme un sous-ensemble admissible de $AR_\Pi(\Pi_{AF})$. Des vérifications similaires s'appliquent à chacun des ι-as trouvés.

La définition suivante introduit directement la contrepartie d'un ensemble d'arguments dans un programme logique :

Definition 2
Soit Π un programme logique et X un ensemble d'atomes. X est appelé answer set admissible *de Π ssi il existe $\Pi' \subseteq \Pi$ tel que $X = flat_\Pi(\Pi')$ et $AR_\Pi(\Pi')$ est un ensemble admissible de $AR_\Pi(\Pi)$.*

A la suite des propositions faites par Gebser *et al.* (2009) pour obtenir les ensembles de réponses à partir des ι-as, nous étendons nos programmes logiques avec des contraintes d'intégrité dont le but est ici de filtrer directement à l'intérieur des ι-ensembles de réponses les ensembles admissibles. Rappelons qu'une contrainte d'intégrité est une règle c dont la tête est vide, c'est à dire :

$$\leftarrow p_1, \ldots, p_m, not\ p_{m+1}, \ldots, not\ p_n$$

Toujours à la suite de Gebser *et al.* (2009), une contrainte c est dite *satisfaite au regard* d'un ensemble X d'atomes si $body^+(c) \not\subseteq X$ ou $body^-(c) \cap X \neq \emptyset$. Afin d'éliminer

dans les ι-as les sous-ensembles qui ne correspondent pas à des ensembles admissibles de $AR_\Pi(\Pi)$ pour un programme donné Π obtenu à partir d'un cadre argumentatif, on pose

$$\mathcal{C}_\Pi^{Ad} = \{\leftarrow head(r'), body^-(r) \mid r, r' \in \Pi, head(r) \in body^-(r')\}.$$

Definition 3
Soit Π un programme, $\mathcal{C}_\Pi^{Ad}$ un ensemble de contraintes d'intégrité, et X un ensemble d'atomes. X est dit admissible au regard de $\mathcal{C}_\Pi^{Ad}$ *ssi X est un sous-ensemble d'un ι-as de Π tel que toute contrainte $c \in \mathcal{C}_\Pi^{Ad}$ est satisfaite au regard de X.*

Théorème 5
Soit Π un programme logique et X un ensemble d'atomes. Alors X est un answer set admissible de Π ssi X est admissible au regard de $\mathcal{C}_\Pi^{Ad}$.

Exemple 1 (suite) En revenant au cadre argumentatif initial *AF* donné plus haut, et au programme logique correspondant Π_{AF}, on y ajoute l'ensemble de contraintes $\mathcal{C}_{\Pi_{AF}}^{Ad}$:

$$\begin{array}{llll} c^{12}: & \leftarrow & b & \\ c^{32}: & \leftarrow & b, not\ d & \\ c^{45}: & \leftarrow & e, not\ c & \end{array} \qquad \begin{array}{llll} c^{56}: & \leftarrow & f, not\ d, not\ g & \\ c^{67}: & \leftarrow & g, not\ e & \\ c^{75}: & \leftarrow & e, not\ f & \end{array}$$

Par commodité chaque contrainte est étiquetée à partir des deux règles de Π_{AF} qui la produisent, i.e. c^{ij} dénote $\leftarrow head(r^i), body^-(r^j)$ quand $head(r^i) \in body^-(r^j)$. Remarquons que les constraintes $c^{34} : \leftarrow d, not\ d$ et $c^{43} : \leftarrow c, not\ c$ qui devraient appartenir à $\mathcal{C}_{\Pi_{AF}}^{Ad}$ peuvent être retirées car elles sont toujours satisfaites. $X_1 = \{a, c, e\}$ est éliminé par c^{75} puisque $body^+(c^{75}) \subseteq X_1$ alors que $body^-(c^{75}) \cap X_1 = \emptyset$, et par conséquent c^{75} n'est pas satisfaite au regard de X_1. Inversement, $X_1 \setminus \{e\}$ est bien un ensemble admissible de Π_{AF} au regard de $\mathcal{C}_{\Pi_{AF}}^{Ad}$. On notera aussi que tout ensemble contenant b est éliminé d'emblée par c^{12}.

5 Conclusion

Nous avons présenté une caractérisation de la sémantique d'admissibilité de Dung dans le cadre des défauts et de la programmation logique. A notre avis, les résultats obtenus, quoique naturels, dénotent un lien plus étroit entre argumentation abstraite, défauts, et programmation logique que celui initialement décrit par Dung. En particulier, Dung (1995), Nieves *et al.* (2008) ou Egly *et al.* (2010) adoptent une approche qui consiste à coder le graphe des attaques comme une méta-connaissance livrée à un programme chargé de le traiter. Hors, il apparaît par exemple superflu de passer par une interprétation prédicative d'un cadre argumentatif abstrait pour pouvoir le représenter en programmation logique. Si, faute d'une étude qui dépasserait le cadre de cet article, nous ne prétendons pas en l'état disposer d'une approche calculatoire plus performante que Nieves *et al.* (2008) ou Egly *et al.* (2010), notre méthode relève toutefois d'un point de vue qui va au-delà de la seule ingénierie visant le traitement d'un cadre argumentatif

par un programme. En montrant que la sémantique d'admissibilité correspond clairement à un fragment des défauts et des programmes logiques, et en élucidant comment, nous espérons montrer que ces trois formalismes contribuent simplement aux éclairages différents d'un même objet mathématique, dont la représentation et le traitement bénéficieront donc des acquis obtenus dans chacune des approches considérées. Dans une perpective future, nous envisageons d'une part de donner une interprétation argumentative à la notion de prérequis d'un défaut (ou de façon équivalente, au corps positif d'une règle de programme logique), d'autre part d'étendre notre approche à d'autres sémantiques argumentatives connues, notamment les extensions complètes, CF2 et la sémantique semi-stable.

Références

AMGOUD L. & BESNARD P. (2009). Bridging the gap between abstract argumentation systems and logic. In L. GODO & A. PUGLIESE, Eds., *Scalable Uncertainty Management : Proceedings of SUM 2009*, volume 5785 of *Lecture Notes in Computer Science*, p. 12–27 : Springer.

BESNARD P. & HUNTER A. (2008). *Elements of Argumentation*. The MIT Press.

BONDARENKO A., DUNG P. M., KOWALSKI R. A. & TONI F. (1997). An abstract, argumentation-theoric approach to default reasoning. *Artificial Intelligence*, **93**(1-2), 63–101.

DELGRANDE J. P., GHARIB M., MERCER R. E., RISCH V. & SCHAUB T. (2003). Lukaszewicz-style answer set programming : A preliminary report. In M. D. VOS & A. PROVETTI, Eds., *Answer Set Programming, Advances in Theory and Implementation, Proceedings of the 2nd Intl. ASP'03 Workshop*, volume 78 of *CEUR Workshop Proceedings* : CEUR-WS.org.

DUNG P. M. (1993a). An argumentation semantics for logic programming with explicit negation. In *ICLP : 10th International Conference on Logic Programming*, p. 616–630.

DUNG P. M. (1993b). On the acceptability of arguments and its fundamental role in nonmonotonic reasoningand logic programming. In *IJCAI : 13th International Joint Conference on Artificial Intelligence*, p. 852–859.

DUNG P. M. (1995). On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, **77**(2), 321–358.

EGLY U., GAGGL S. A. & WOLTRAN S. (2010). Answer-set programming encodings for argumentation frameworks. *Argument and Computation*, **1**(2), 147–177.

GEBSER M., GHARIB M., MERCER R. E. & SCHAUB T. (2009). Monotonic answer set programming. *Journal of Logic and Computation*, **19**(4), 539–564.

LUKASZEWICZ W. (1988). Considerations on default logic – an alternative approach. *Computational Intelligence*, **4**, 1–16.

NIEVES J. C., CORTÉS U. & OSORIO M. (2008). Preferred extensions as stable models. *Theory and Practice of Logic Programming*, **8**(4), 527–543.

REITER R. (1980). A logic for default reasoning. *Artificial Intelligence*, **13**, 81–132.

RISCH V. (1996). Analytic tableaux for default logics. *Journal of Applied Non-Classical Logics*, **6**, 71–88.

Vers une version possibiliste de la théorie des fonctions de croyance

Henri Prade[1], Agnès Rico[2]

[1] Institut de Recherche en Informatique de Toulouse - IRIT
Université de Toulouse, 118, route de Narbonne, 31062 Toulouse cedex 09, France
prade@irit.fr

[2] Equipe de Recherche en Ingénierie des Connaissances - ERIC
Université Claude Bernard Lyon 1, 43 bld du 11 novembre 69100 Villeurbanne, France
agnes.rico@univ-lyon1.fr

Résumé : Cet article propose une contrepartie qualitative à la théorie des fonctions de croyance de Shafer. La fonction de masse est remplacée par une affectation possibiliste. La somme des poids ne fait plus 1, mais c'est le maximum des poids qui fait 1. Des fonctions d'ensemble, contreparties de celles existant en théorie des fonctions de croyance, sont définies, où l'addition est remplacée par le maximum. Ce nouveau cadre permet de représenter des possibilités imprécises. Nous étudions en particulier le lien entre cette construction et la transformée de Möbius qualitative. Les affectations possibilistes peuvent être vues comme des ensembles généralisés, entre lesquels une relation d'inclusion peut être définie. Les règles de combinaison de l'information dans ce cadre sont également définies.

Mots-clés : théorie des fonctions de croyance, théorie des possibilités, information imprécise, fusion d'informations

1 Introduction

De nombreux travaux concernant la représentation de l'incertitude ou la modélisation des préférences utilisent des données quantitatives et des structures additives. Elles sont à la base des approches les plus classiques et de leurs généralisations comme la théorie des probabilités et la théorie des fonctions de croyance (Shafer, 1976) pour l'incertain, ou l'agrégation multicritère où on trouve des sommes pondérées et plus généralement des intégrales de Choquet dans le cas des préférences. Les informations à traiter sont cependant souvent d'origine qualitative. Ainsi, en théorie des possibilités (Zadeh, 1978), ou dans l'intégrale de Sugeno (Sugeno, 1977), le maximum remplace l'addition, ce qui les rend compatibles avec de telles données.

La théorie des fonctions de croyance, qui inclut la théorie des probabilités et la théorie des possibilités comme des cas particuliers, fournit un cadre approprié pour représenter des informations imprécises et incertaines au moyen d'une densité de probabilité sur les

sous-ensembles du référentiel. Cette théorie constitue aussi un système remarquable de probabilités imprécises. Nous nous intéressons dans cet article à une contrepartie qualitative de la théorie des fonctions de croyance où la structure additive est remplacée par une structure maxitive.[1] Plus précisément, nous partons d'une affectation possibiliste qui remplace la fonction de masse, et l'addition est remplacée par le maximum. Dans ce cas, la somme des poids attribués par l'affectation possibiliste ne vaut plus 1, mais c'est le maximum des poids attribués qui est égal à 1. Quoique l'idée d'affectation possibiliste soit déjà ancienne (Dubois & Prade, 1984, 1985), et qu'elle ait retenu assez tôt l'attention de quelques auteurs (Tsiporkova & Baets, 1998), et qu'elle ait été etudiée à nouveau plus récemment au travers de la transformée de Möbius qualitative (Grabisch, 2004; Dubois & Fargier, 2009), ce cadre qualitatif de représentation de l'incertitude n'a jamais fait l'objet d'études systématiques.

Cet article, qui constitue une version révisée de (Prade & Rico, 2011), est organisé de la manière suivante : La section 2 présente les définitions et les notations. La section 3 montre l'évidence possibiliste comme une possibilité imprécise. La section 4 présente une différence majeure entre le cas classique et le cas qualitatif : pour une fonction de croyance possibiliste donnée, l'affectation possibiliste n'est pas unique. La section 5 concerne l'implication entre affectations possibilistes et les relations avec le comportement des éléments focaux. La section 6 est dédiée aux règles de combinaison, tandis que la section 7 conclut sur quelques perpectives.

2 Définitions et notations

Dans la théorie des fonctions de croyance, le référentiel $\Omega = \{1, \cdots, N\}$ est constitué par un ensemble d'éléments représentant des états mutuellement exclusifs du monde. Un état de connaissance est représenté par une fonction de masse m, qui associe à tout sous-ensemble $A \subseteq \Omega$, un poids $m(A)$ qui constitue la *part* de croyance associée à A exactement. Notons que si A n'est pas un singleton, l'énoncé "l'état réel du monde est dans A" est imprécis. Plus précisément une fonction de masse est une fonction $m : 2^{\Omega} \to [0, 1]$ telle que $\sum_{A \subseteq \Omega} m(A) = 1$. Tout $A \subseteq \Omega$ tel que $m(A) > 0$ est appelé élément focal de m. Si $m(\emptyset) = 0$ l'information est dite cohérente.

A une fonction de masse m sont associées les fonctions d'ensemble suivantes :

$$\forall A \subseteq \Omega, \quad Bel(A) = \sum_{\emptyset \neq B \subseteq A} m(B)\,; \quad \forall A \subseteq \Omega, \quad Pl(A) = \sum_{B \cap A \neq \emptyset} m(B)\,;$$
$$\forall A \subseteq \Omega, \quad Q(A) = \sum_{A \subseteq B} m(B)\,; \qquad \forall A \subseteq \Omega, \quad \eth(A) = \sum_{A \cup B \neq \Omega} m(B).$$

Bel est la fonction de croyance, Pl est la plausibilité, et Q est la communalité. $\eth(A) = 1 - Q(\overline{A})$ où $\bar{A}$ est le complémentraire de A est rarement utilisée. Commençons par rappeler les principaux résultats présentés dans (Prade & Rico, 2011).

[1] Le cadre qualitatif, maxitif, décrit dans cet article ne doit pas être confondu avec quelques propositions faites par certains auteurs pour rendre plus qualitatif le cadre des fonctions de croyance (Parsons & Mamdani, 1993), ou des cadres apparentés de fusion d'informations (Dezert & Smarandache, 2006), tout en conservant une structure "additive" appliquée respectivement à $\{0, +\}$ (ou à un partitionnement plus général de l'intervalle $[0, 1]$) et à une échelle finie de niveaux symboliques.

2.1 Les fonctions d'ensemble de la contrepartie qualitative

Dans ce qu'on peut appeler la théorie possibiliste de l'évidence l'addition est remplacée par le maximum et la fonction de masse m est remplacée par une fonction d'ensemble $\sigma : 2^\Omega \to [0,1]$ telle que $\max_{A\subseteq\Omega} \sigma(A) = 1$. Cette fonction s'appellera affectation possibiliste. Comme dans la théorie des fonctions de croyance, $\mathcal{F}_\sigma = \{A \subseteq \Omega | \, \sigma(A) > 0\}$ est l'ensemble des éléments focaux de σ. σ peut être considérée comme une distribution de possibilité sur 2^Ω tout comme m peut être considérée comme une distribution de probabilité sur 2^Ω. Si $\sigma(\emptyset) = 0$ alors σ est dite normale. Dans la suite de l'article nous ne considérons que des affectations possibilistes normales. Comme dans le cas quantitatif une affectation possibiliste σ définit les fonctions d'ensemble suivantes :

Définition 1
$\forall A \subseteq \Omega,\ Bel^{pos}(A) = \max_{B\subseteq A} \sigma(B)$, $Bel^{pos}(\emptyset) = 0$, $Bel^{pos}(\Omega) = 1$;
$\forall A \subseteq \Omega,\ Pl^{pos}(A) = \max_{\emptyset\neq A\cap B} \sigma(B)$, $Pl^{pos}(\emptyset) = 0$, $Pl^{pos}(\Omega) = 1$;
$\forall A \subseteq \Omega,\ Q^{pos}(A) = \max_{A\subseteq B} \sigma(B)$, $Q^{pos}(\emptyset) = 1$, $Q^{pos}(\Omega) = \sigma(\Omega)$;
$\forall A \subseteq \Omega,\ \eth^{pos}(A) = \max_{A\cup B\neq\Omega} \sigma(B)$, $\eth^{pos}(\emptyset) = \max_{B\neq\Omega} \sigma(B)$, $\eth^{pos}(\Omega) = 0$.

Dans le cadre additif[2] la fonction de croyance Bel et la plausibilité Pl vérifie $\forall A \subseteq \Omega,\ Bel(A) \leq Pl(A)$. De plus elles sont reliées par la relation de dualité : $\forall A \subseteq \Omega,\ Pl(A) + Bel(\bar{A}) = 1 - m(\emptyset)$.

Les fonctions Bel^{pos} et Pl^{pos} sont croissantes et vérifient $\forall A$, $Bel^{pos}(A) \leq Pl^{pos}(A)$. Les fonctions Q^{pos} et $\eth^{pos}$ sont quand à elles décroissantes. La relation $\forall A$, $Q^{pos}(A) \leq \eth^{pos}(A)$ est vraie si $\sigma(\Omega) = 0$. Par ailleurs, il existe une forme de relation de dualité entre Bel^{pos} et Pl^{pos} et entre Q^{pos} et $\eth^{pos}$:

Proposition 1
$\forall A \subseteq \Omega$, $\max(Pl^{pos}(A), Bel^{pos}(\bar{A})) = 1$ *et* $\max(Q^{pos}(A), \eth^{pos}(\bar{A})) = 1$.

2.2 Quelques cas particuliers

Regardons de plus près quelques cas particuliers.

L'imprécision totale : $\sigma(\Omega) = 1$ et σ vaut 0 ailleurs. Dans ce cas, $\forall A \neq \Omega$, $Bel^{pos}(A) = 0$, $\forall A \neq \emptyset$, $Pl^{pos}(A) = 1$. De plus on a $\forall A$, $Q^{pos}(A) = 1$; $\forall A$, $\eth^{pos}(A) = 0$.

L'imprécision partielle : il existe un unique ensemble $A \subseteq \Omega$ tel que $\sigma(A) = 1$ et cet ensemble A est tel que $|A| > 1$, où $||$ est la cardinalité. Dans ce cas on a
$Bel^{pos}(B) = 1$ si $A \subseteq B$ et 0 si $A \not\subseteq B$;
$Pl^{pos}(B) = 1$ si $A \cap B \neq \emptyset$ et 0 si $A \cap B = \emptyset$;
$Q^{pos}(B) = 1$ $B \subseteq A$ et 0 si $B \not\subseteq A$;
$\eth^{pos}(B) = 1$ si $B \cup A \neq \Omega$.

[2] Il faut comprendre ici, et dans la suite du papier, que ce "cadre additif" réfère à la théorie des fonctions de croyance de Shafer, sans pour autant suggérer que ces fonctions seraient additives !

Connaissance totale : Il existe un unique $\omega \in \Omega$ tel que $\sigma(\{\omega\}) = 1$ et σ vaut 0 ailleurs. Dans ce cas on a $\forall A$, $Bel^{pos}(A) = Pl^{pos}(A) = 1$ si $\omega \in A$ et 0 si $\omega \notin A$. $Q^{pos}(A) = 1$ si $A = \{\omega\}$ et 0 $\forall A \neq \{\omega\}$; $Q^{pos}(A) = 1$ si $A \neq \{\Omega\}, \neq \overline{\{\omega\}}$.

L'incertain précis : σ est une affectation possibiliste incertaine si $\sigma(A) > 0$ implique que A est un singleton. Dans ce cas $Bel^{pos}(A) = Pl^{pos}(A) = \max_{\omega \in A} \sigma(\omega)$ est une mesure de possibilité au sens classique. $Q^{pos}(A) = 0$ pour tout A tel que $|A| > 1$ et $Q^{pos}(\{\omega\})) = \sigma(\{\omega\})$.

L'incertain possibiliste : Dans ce cas σ est non nulle seulement pour certains ensembles A_i qui sont inclus les uns dans les autres $A_1 \subseteq \cdots \subseteq A_n$. Dans cette situation, dans le cas de la théorie des fonctions de croyance, Pl est une possibilité et Bel est une nécessité. Dans la contrepartie qualitative cette propriété reste vraie. Plus précisément nous avons le résultat suivant :

Proposition 2
Si σ est non nulle seulement pour certains ensembles A_i qui sont inclus les uns dans les autres ($A_1 \subseteq \cdots \subseteq A_n$), alors
Bel^{pos} *est une nécessité, i.e.* $Bel^{pos}(A \cap B) = \min(Bel^{pos}(A), Bel^{pos}(B))$,
Pl^{pos} *est une possibilité, i.e.* $Pl^{pos}(A \cup B) = \max(Pl^{pos}(A), Pl^{pos}(B))$,
$Q^{pos}(A \cup B) = \min(Q^{pos}(A), Q^{pos}(B))$,
$\eth^{pos}(A \cap B) = \max(\eth^{pos}(A), \eth^{pos}(B))$.

3 L'évidence possibiliste vue comme une possibilité imprécise

En supposant Ω fini, une fonction de masse m, en théorie des fonctions de croyance de Shafer, définit une probabilité imprécise car m est compatible avec toute distribution de probabilité p obtenue en partageant la masse $m(A)$ entre tous les éléménts ω appartenant à A :
$\forall A \in 2^{\Omega}, \exists k_A \in [0,1]^A$ tel que $m(A) = \sum_{\omega \in A} k_A(\omega)$ et $p(\omega) = \sum_{A \ni \omega} k_A(\omega)$.

$\sum_{A \neq \emptyset} m(A) = 1$ assure que $\sum_{\omega \in \Omega} p(\omega) = 1$. Alors $m(A)$ vue comme une masse peut être répartie entre les éléments de A. m est alors associée à une famille de distributions de probabilités dites compatibles. Dans ce cas m représente un ensemble de probabilités imprécises particulières. En posant $P(A) = \sum_{\omega \in A} p(\omega)$, on montre que

$$Bel(A) = \sum_{\emptyset \neq B \subseteq A} m(B) \leq P(A) \leq Pl(A) = \sum_{B \cap A \neq \emptyset} m(B).$$

De même dans notre cas on peut définir une structure possibiliste imprécise, mais la situation est un peu plus compliquée : Considérons une affectation possibiliste σ et $\mathcal{U}_\sigma = \bigcup_{i|\sigma(A_i)>0} A_i$. Si nous voulons interpréter les valeurs $\sigma(A)$ comme provenant d'une possibilité imprécise il existerait une distribution de possibilités π et des applications

$\kappa_A \in [0,1]^A$ telles que $\sigma(A) = \max_{\omega \in A} \kappa_A(\omega)$ et $\pi(\omega) = \max_{B \ni \omega} \kappa_B(\omega)$.

Remarquons que $\Pi(A) = \max_{\omega \in A} \pi(\omega) = \max_{\omega \in A} \max_{B \ni \omega} \kappa_B(\omega) \geq \sigma(A)$; et que $\forall \omega \in \Omega - \mathcal{U}_\sigma, \pi(\omega) = 0$. On a $\max_{A \neq \emptyset} \sigma(A) = 1$ qui entraine que $\max_{\omega \in \Omega} \pi(\omega) = 1$. Alors, $\sigma(A)$ peut être interprété comme une contribution à la possibilité de A. Nous pouvons de plus montrer que

$$Bel^{pos}(A) \leq \Pi(A) \leq Pl^{pos}(A).$$

Exemple. $\mathcal{F}_\sigma = \{A, \Omega\}$. Supposons que $\sigma(A) \geq \sigma(\Omega)$. Alors $\sigma(A) = \max_{\omega \in A} \kappa_A(\omega)$, $\sigma(\Omega) = \max_{\omega \in \Omega} \kappa_\Omega(\omega)$ et $\pi(\omega) = \max(\kappa_A(\omega), \kappa_\Omega(\omega))$. Par convention, $\kappa_A(\omega) = 0$ si $\omega \notin A$. Alors $\Pi(A) = max(\sigma(A), \max_{\omega \in A} \kappa_\Omega(\omega)) = \sigma(A)$. Mais, $\Pi(\Omega) = max(\max_{\omega \in \Omega} \kappa_A(\omega), \sigma(\Omega)) = \sigma(A)$. Si $\sigma(A) < \sigma(\Omega)$, $\sigma(A) \leq \Pi(A) \leq \sigma(\Omega) = \Pi(\Omega) = 1$.

Une mesure de possibilité Π permet de définir une mesure de nécessité associée en posant $N(A) = 1 - \Pi(\overline{A})$. Donc par dualité, les inégalités $Bel^{pos}(A) = \max_{B \subseteq A} \sigma(B) \leq \Pi(A) \leq Pl^{pos}(A) = \max_{B \cap A \neq \emptyset} \sigma(B)$ entrainent les inégalités

$$\min_{B \subseteq \overline{A}} 1 - \sigma(B) \geq N(A) \geq \min_{B \cap \overline{A} \neq \emptyset} 1 - \sigma(B),$$

faisant ainsi apparaitre une structure minitive associée.

On retrouve la logique possibiliste (Dubois & Prade, 2004) quand la borne supérieure de $N(A)$ est 1, i.e., quand $\min_{B \subseteq \overline{A}} 1 - \sigma(B) = 1$, ce qui est équivalent à $\forall B \subseteq \overline{A}, \sigma(B) = 0$. Cela exprime que rien de ce qui est dans $\overline{A}$ n'est possible de manière sûre, ce qui est intuitivement satisfaisant (dans le cas contraire, $N(A)$ ne pourrait pas atteindre 1). Cela montre que comme la logique possibiliste correspond à une famille de mesures de possibilité ayant un unique élément maximal (correspondant à la distribution de possibilité la moins spécifique compatible avec des contraintes de la forme $N(A) \geq \alpha$), l'évidence possibiliste nous permet de représenter des familles plus générales de mesures de possibilité, correspondant à des valeurs de possibilités et de nécessités imprécises. Ce qui rejoint des préoccupations récentes en logique possibiliste (Benferhat *et al.*, 2011).

4 Non unicité de l'affectation possibiliste

Dans le cas classique, la fonction de croyance Bel (ou sa duale Pl) est associée à une unique fonction de masse m. Dans le cadre qualitatif cette propriété est fausse

	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	$\{1,2,3\}$
σ_1	1	0	1	0	0	0	1
σ_2	1	0	1	1	1	1	0
$Bel_1^{pos} = Bel_2^{pos}$	1	0	1	1	1	1	1

De plus l'équivalence $\forall A \subseteq \Omega$, $Bel_1(A) \leq Bel_2(A) \Leftrightarrow Pl_2(A) \leq Pl_1(A)$ n'est pas satisfaite par les fonctions Bel^{pos} et Pl^{pos}. Par exemple si on considère

	$\{1\}$	$\{2\}$	$\{1,2\}$
σ_1	0	1	0
σ_2	1	1	1

on a $Bel_1^{pos} \leq Bel_2^{pos}$ et $Pl_1^{pos} \leq Pl_2^{pos}$.

On peut seulement affirmer que si les fonctions Bel_1^{pos} et Bel_2^{pos} sont associées à σ_1 et σ_2, on a que $Bel_1^{pos} = Bel_2^{pos}$ entraine que σ_1 et σ_2 sont égales sur les singletons.

L'information d'une fonction de croyance possibiliste ne représente donc pas toute l'information contenue dans l'affectation possibiliste qui la définit. En effet, deux affectations possibilistes peuvent ne pas contenir la même information et définir la même fonction de croyance possibiliste.

Dans la théorie des fonctions de croyance, la fonction de masse associée à une fonction de croyance est calculée à l'aide de la transformée de Möbius. Dans le cas qualitatif, la transformée de Möbius (Grabisch, 2003, 2004; Dubois & Fargier, 2009) associe à une fonction de croyance possibiliste Bel^{pos} un intervalle d'affectations possibilistes :

Définition 2
La transformée de Möbius qualitative d'une fonction d'ensemble croissante v est
$\{\sigma | \forall A \subseteq \Omega \ v(A) = \max_{B \subseteq A} \sigma(B)\} = \{\sigma | \sigma \in [\sigma_*, \sigma^*]\}$
où σ_* et σ^* sont les affectations possibilistes définies par :
$\forall A \subseteq \Omega$, $\sigma^*(A) = v(A)$,
$\forall A \subseteq \Omega$, $\sigma_*(A) = 0$ *si* $\exists B \subset A$ *tel que* $v(B) = v(A)$ *et* $\sigma_*(A) = v(A)$ *sinon.*

L'application de cette définition à la fonction de croyance possibiliste Bel^{pos} donne un intervalle $[\sigma_*^B, \sigma^{B*}]$.

Exemple

	$\{1\}$	$\{2\}$	$\{3\}$	$\{1,2\}$	$\{1,3\}$	$\{2,3\}$	$\{1,2,3\}$
Bel^{pos}	1	0	1	1	1	1	1
σ^*	1	0	1	1	1	1	1
σ_*	1	0	1	0	0	0	0

De la même façon la fonction Pl^{pos} correspond à l'intervalle $[\sigma_*^P, \sigma^{P*}]$.

On a l'inégalité $\sigma_*^B \leq \sigma_*^P \leq \sigma^{B*} \leq \sigma^{P*}$. Donc si les fonctions Bel^{pos} et Pl^{pos} sont connues partout, σ est déterminée de manière unique si et seulement si $\sigma_*^P = \sigma^{B*}$.

La fonction Q^{pos} est décroissante, mais nous pouvons définir aussi un intervalle d'affectations possibilistes comme cela a été fait avec la fonction d'ensemble $\min_{A \subseteq B} \sigma(B)$ dans (Dubois & Fargier, 2009).

Définition 3
Soit $v : 2^\Omega \to [0,1]$ une fonction d'ensemble décroissante alors
$\{\sigma | \forall A \subseteq \Omega \ v(A) = \max_{A \subseteq B} \sigma(B)\} = \{\sigma | \sigma \in [\sigma_*, \sigma^*]\}$
où σ_* et σ^* sont les affectations possibilistes définies par :
$\forall A \subseteq \Omega$, $\sigma^*(A) = v(A)$,
$\forall A \subseteq \Omega$, $\sigma_*(A) = 0$ *si* $\exists B$ *tel que* $A \subset B$ *et* $v(B) = v(A)$ *;* $\sigma_*(A) = v(A)$ *sinon.*

Les fonctions Q^{pos} et $\mathfrak{d}^{pos}$ définissent donc les intervalles $[\sigma_*^Q, \sigma^{Q*}]$ et $[\sigma_*^{\mathfrak{d}}, \sigma^{\mathfrak{d}*}]$.

Quand les quatre fonctions Bel^{pos}, Pl^{pos}, Q^{pos} et $\mathfrak{d}^{pos}$ sont connues partout, σ est définie de manière unique si et seulement si l'intersection des quatre intervalles construits par la transformée de Möbius qualitative est réduit à un point. Ce n'est pas toujours le cas. Par exemple si nous considérons $\Omega = \{1, 2, 3\}$ et les affectations possibilistes σ_1 et σ_2 définies par :

σ_1 vaut 1 sur $\{1\}$, $\{3\}$ $\{1, 2, 3\}$ et 0 sur les autres ensembles ;

σ_2 vaut 0 sur $\{2\}$ et 1 sur les autres ensembles ;

mais nous avons $Bel_1^{pos} = Bel_2^{pos}$, $Pl_1^{pos} = Pl_2^{pos}$, $Q_1^{pos} = Q_2^{pos}$ et $\mathfrak{d}_1^{pos} = \mathfrak{d}_2^{pos}$.

5 Implication entre affectations possibilistes

D'après la section précédente, même si nous connaissons les quatre fonctions d'ensemble Bel^{pos}, Pl^{pos}, Q^{pos} et $\mathfrak{d}^{pos}$, σ peut ne pas être connue de manière unique. Il est donc difficile de faire jouer à σ le même rôle que la fonction de masse m dans le cas classique. Par contre dans (Dubois, 2011) il est montré que ce rôle peut être joué par la borne inférieure de la transformée de Möbius de la fonction de croyance possibiliste Bel^{pos} associée à σ. Les éléments focaux pris en compte sont alors ceux de cette borne inférieure. Il est alors possible de généraliser au cas qualitatif certains résultats présentés dans (Dubois & Prade, 1986) pour le cadre additif.

Soient σ une affectation possibiliste qui définit une fonction de croyance possibiliste Bel^{pos} et σ_* la borne inférieure de la transformée de Möbius de Bel^{pos}. Dans cette partie on note A_i les éléments focaux de σ_* : $\sigma_*(A_i) \neq 0$. Leur ensemble est $\mathcal{F}_{\sigma_*} = \{A_i\}_{i \in I}$ où I est un ensemble fini d'indices. Il est alors possible de définir le core et le support d'une affectation possibiliste comme dans le cas des fonctions de croyance de Shafer (Dubois & Prade, 1986).

5.1 Noyau et Support d'une affectation possibiliste

Définition 4

Le noyau de σ est $\mathcal{C}(\mathcal{F}_{\sigma_*}) = \cap_{i \in I} A_i$,

Le support de σ est $\mathcal{S}(\mathcal{F}_{\sigma_*}) = \cup_{i \in I} A_i$.

Soient deux affectations possibilistes σ et σ' qui définissent deux fonctions de croyance possibilistes Bel^{pos} et $Bel^{pos\prime}$ qui permettent de définir deux affectations possibilistes σ_* et σ'_*. On a alors deux ensembles $\mathcal{F}_{\sigma_*}$ et $\mathcal{F}_{\sigma'_*}$. On note $\{A_i\}_{i \in I}$ l'ensemble des éléments focaux de $\mathcal{F}_{\sigma_*}$ et $\{A'_j\}_{j \in J}$ l'ensemble des éléments focaux de $\mathcal{F}_{\sigma'_*}$, avec I et J deux ensembles finis d'indices. Avec σ_* (resp. σ'_*) on définit Q^{pos} (resp. $Q^{pos\prime}$).

L'inclusion de $\mathcal{F}_{\sigma_*}$ dans $\mathcal{F}_{\sigma'_*}$ peut être vue de deux manières différentes :

- La première correspond à : tout élement A' de $\mathcal{F}_{\sigma'_*}$ contient un élément A de $\mathcal{F}_{\sigma_*}$ tel que $\sigma'_*(A') \leq \sigma(A)$.
- La seconde correspond à : tout élément A de $\mathcal{F}_{\sigma_*}$ est inclus dans un élément A' de $\mathcal{F}_{\sigma'_*}$ tel que $\sigma(A) \leq \sigma'(A')$.

On va donc définir deux inclusions qui reprennent les notations du cas classique.

Définition 5
$\mathcal{F}_{\sigma_*} \subseteq \mathcal{F}_{\sigma'_*}$ si et seulement si $Bel^{pos\prime} \leq Bel^{pos}$.

Cette première définition est équivalente à $\forall A,\ \max_{A'_j \subseteq A} \sigma'_*(A'_j) \leq \max_{A_i \subseteq A} \sigma_*(A_i)$.

Proposition 3
Si $\mathcal{F}_{\sigma_*} \subseteq \mathcal{F}_{\sigma'_*}$ alors $\mathcal{C}(\mathcal{F}_{\sigma_*}) \subseteq \mathcal{C}(\mathcal{F}_{\sigma'_*})$.

Preuve: $\forall A'_j$, $\sigma'_*(A'_j) \neq 0 \leq \max_{A_i \subseteq A'_j} \sigma_*(A_i)$ donc $\exists A_i$ tel que $A_i \subseteq A'_j$. Donc $\forall A'_j$, $\cap_i A_i \subseteq A'_j$ ce qui implique $\cap_i A_i \subseteq \cap_j A'_j$. ■

Remarque : Dans le cas additif on a la définition $\mathcal{F}_{\sigma_*} \subseteq \mathcal{F}_{\sigma'_*}$ si et seulement si $[Bel^{pos}, Pl^{pos}] \subseteq [Bel^{pos\prime}, Pl^{pos\prime}]$. Dans le cas qualitatif la situation est différente car nous n'avons pas l'équivalence entre $Bel^{pos\prime} \leq Bel^{pos}$ et $Pl^{pos} \leq Pl^{pos\prime}$.

Pour conserver la définition avec inclusion d'intervalle comme dans le cas additif on devrait rajouter la condition $\forall A,\ \max_{A \cap A_i \neq \emptyset} \sigma_*(A_i) \leq \max_{A \cap A'_j \neq \emptyset} \sigma'_*(A'_j)$. On aurait alors que l'inclusion implique $\mathcal{S}(\mathcal{F}_{\sigma_*}) \cap \mathcal{S}(\mathcal{F}_{\sigma'_*}) \neq \emptyset$. En effet on aurait $\forall A_i$, $\sigma_*(A_i) \neq 0 \leq \max_{A \cap A'_i \neq \emptyset} \sigma'_*(A'_i)$ donc $\exists A'_j$ tel que $A_i \cap A'_j \neq \emptyset$ donc $\cup_i A_i \cap \cup_j A'_j \neq \emptyset$.

La deuxième inclusion est :

Définition 6
$\mathcal{F}_{\sigma_*} \bar{\subseteq} \mathcal{F}_{\sigma'_*}$ si et seulement si $Q^{pos} \leq Q^{pos\prime}$.

Cette deuxième inclusion est équivalente à $\forall A,\ \max_{A \subseteq A_i} \sigma_*(A_i) \leq \max_{A \subseteq A'_j} \sigma'_*(A'_j)$.

Proposition 4
Si $\mathcal{F}_{\sigma_*} \bar{\subseteq} \mathcal{F}_{\sigma'_*}$ alors $\mathcal{S}(\mathcal{F}_{\sigma_*}) \subseteq \mathcal{S}(\mathcal{F}_{\sigma'_*})$.

Preuve: $\forall A_i$, $\sigma_*(A_i) \neq 0 \leq \max_{A_i \subseteq A'_j} \sigma'_*(A'_j)$ donc $\exists A'_j$ tel que $A_i \subseteq A'_j \subseteq \cup_j A'_j$. Donc $\cup_i A_i \subseteq \cup_j A'_j$. ■

Toujours en nous inspirant de (Dubois & Prade, 1986) il semble naturel de définir la relation entre deux affectations possibilistes de la manière suivante.

Définition 7
$\sigma \vDash \sigma'$ si et seulement si
$\forall A,\ \max_{A \subseteq A_i} \sigma_*(A_i) \leq \max_{A \subseteq A'_j} \sigma'_*(A'_j)$ *et* $\max_{A'_j \subseteq A} \sigma'_*(A'_j) \leq \max_{A_i \subseteq A} \sigma_*(A_i)$.

A l'aide des propositions précédentes concernant les noyaux et les supports on peut facilement démontrer le résultat suivant :

Proposition 5
Si $\sigma \vDash \sigma'$ alors $\mathcal{S}(\mathcal{F}_{\sigma_*}) \subseteq \mathcal{S}(\mathcal{F}_{\sigma'_*})$ et $\mathcal{C}(\mathcal{F}_{\sigma_*}) \subseteq \mathcal{C}(\mathcal{F}_{\sigma'_*})$.

5.2 Lien entre implication et comportement des éléments focaux minimum et maximum

Nous allons maintenant nous intéresser aux éléments focaux minimum et maximum au sens de l'inclusion. Plus précisément nous allons étudier le lien entre l'implication de deux affectations possibilistes et le comportement de ces ensembles.

Définition 8
$\mathcal{F}_{\sigma_*^{min}} = \{A \subseteq \Omega \,|\sigma_*(A) \neq 0$ *et* $\forall B$ *tel que* $B \subset A, \sigma_*(B) = 0\}$.
$\mathcal{F}_{\sigma_*^{max}} = \{A \subseteq \Omega \,|\sigma_*(A) \neq 0$ *et* $\forall B \subseteq \Omega$ *tel que* $A \subset B, \sigma_*(B) = 0\}$.

Proposition 6
Soient σ et σ' deux affectations possibilistes.

$$\textit{Si } \sigma \vDash \sigma' \textit{ alors } \begin{cases} \forall A \in \mathcal{F}_{\sigma'^{min}_*}, \exists B \textit{ tel que } B \subseteq A \textit{ et } B \in \mathcal{F}_{\sigma_*^{min}} \\ \forall A \in \mathcal{F}_{\sigma_*^{max}}, \exists B \textit{ tel que } A \subseteq B \textit{ et } B \in \mathcal{F}_{\sigma'^{max}_*}. \end{cases}$$

Preuve: Soient σ et σ' tels que $\sigma \vDash \sigma'$.
Si $A \in \mathcal{F}_{\sigma'^{min}_*}$ alors $\sigma'_*(A) \neq 0 \leq \max_{B \subseteq A} \sigma_*(B)$, donc il existe $B \subseteq A$ tel que $\sigma_*(B) \neq 0$, donc A contient un ensemble appartenant à $\mathcal{F}_{\sigma_*^{min}}$.
Si $A \in \mathcal{F}_{\sigma_*^{max}}$ alors $\sigma_*(A) \neq 0 \leq \max_{A \subseteq B} \sigma'_*(B)$, donc il existe $A \subseteq B$ tel que $\sigma'_*(B) \neq 0$, donc A est inclus dans un élément de $\mathcal{F}_{\sigma'^{max}_*}$. ■

Il est possible d'avoir $\sigma \vDash \sigma'$ avec $\mathcal{F}_{\sigma'^{min}_*} \not\subseteq \mathcal{F}_{\sigma_*^{min}}$ ou $\mathcal{F}_{\sigma_*^{max}} \not\subseteq \mathcal{F}_{\sigma'^{max}_*}$. Pour les exemples suivants remarquons que si σ est telle que, si $A \subset B$ avec $\sigma(A) \neq 0$ et $\sigma(B) \neq 0$ alors $\sigma(A) < \sigma(B)$, σ est égale à σ_*.
Exemple : Soit $\Omega = \{1, 2\}$ et σ_*, σ'_* définies par :

	$\{1\}$	$\{2\}$	$\{1,2\}$
σ_*	0.5	0	1
σ'_*	0	0	1

nous avons $\{1, 2\} \in \mathcal{F}_{\sigma'^{min}_*}$ et $\{1, 2\} \notin \mathcal{F}_{\sigma_*^{min}}$.

Exemple : $\Omega = \{1, 2\}$, σ_* et σ'_* sont définies par :

	$\{1\}$	$\{2\}$	$\{1,2\}$
σ_*	1	1	0
σ'_*	0	0.5	1

nous avons$\{2\} \in \mathcal{F}_{\sigma_*^{max}}$ et $\{2\} \notin \mathcal{F}_{\sigma'^{max}_*}$.

Proposition 7
Soient σ, σ' deux affectations possibilistes telles que $\sigma \vDash \sigma'$ et $\sigma' \vDash \sigma$, alors $\mathcal{F}_{\sigma_*^{min}} = \mathcal{F}_{\sigma'^{min}_*}$ et $\mathcal{F}_{\sigma_*^{max}} = \mathcal{F}_{\sigma'^{max}_*}$.

Preuve: Soit $A \in \mathcal{F}_{\sigma_*^{min}}$. D'après la proposition précédente $\sigma' \vDash \sigma$ implique $\exists B$ $B \subseteq A$ et $B \in \mathcal{F}_{\sigma'^{min}_*}$. Si $B \neq A$, alors $\sigma \vDash \sigma'$ implique $\exists C$ $C \subseteq B$ et $C \in \mathcal{F}_{\sigma_*^{min}}$. $C \subset A$ ce qui contredit $A \in \mathcal{F}_{\sigma_*^{min}}$ donc $B = A$ ce qui entraine $\mathcal{F}_{\sigma_*^{min}} \subseteq \mathcal{F}_{\sigma'^{min}_*}$. Par symétrie de la proposition on a aussi $\mathcal{F}_{\sigma'^{min}_*} \subseteq \mathcal{F}_{\sigma_*^{min}}$.

Soit $A \in \mathcal{F}_{\sigma_*^{max}}$. D'après la proposition précédente, $\sigma \models \sigma'$ implique $\exists B\ A \subseteq B$ et $B \in \mathcal{F}_{\sigma'^{max}_*}$. Si $B \neq A$, alors $\sigma' \models \sigma$ implique $\exists C\ B \subseteq C$ et $C \in \mathcal{F}_{\sigma_*^{max}}$. $A \subset C$ ce qui contredit $A \in \mathcal{F}_{\sigma_*^{max}}$ donc $B = A$ ce qui entraine $\mathcal{F}_{\sigma_*^{max}} \subseteq \mathcal{F}_{\sigma'^{max}_*}$. De la même façon on montre que $\mathcal{F}_{\sigma'^{max}_*} \subseteq \mathcal{F}_{\sigma_*^{max}}$. ■

Proposition 8
Dans le cas binaire si σ, σ' sont deux affectations possibilistes telles que $\mathcal{F}_{\sigma_*^{min}} = \mathcal{F}_{\sigma'^{min}_*}$ et $\mathcal{F}_{\sigma_*^{max}} = \mathcal{F}_{\sigma'^{max}_*}$ alors $\sigma \models \sigma'$ et $\sigma' \models \sigma$.

Preuve: Dans le cas binaire si on considère deux ensembles A et B tels que $A \subset B$ on a soit $\sigma_*(A) = \sigma_*(B) = 0$ soit $\sigma_*(A) = 1$ ou $\sigma_*(B) = 1$ mais pas les deux. Les ensembles des éléments focaux minimum et celui des éléments focaux maximum sont donc identiques. Si les ensembles sont égaux pour deux affectations possibilistes σ et σ', alors $\sigma = \sigma'$ ce qui permet de conclure. ■

6 Les règles de combinaison

Dans la théorie des fonctions de croyance, à côté de la règle bien connue de la combinaison de Dempster (Shafer, 1976) d'autres règles ont été proposées pour la fusion des fonctions de masse. En particulier, la règle de combinaison disjonctive est une autre règle de base (Dubois & Prade, 1986). Ces règles et leurs propriétés se retrouvent dans le cadre qualitatif (Prade & Rico, 2011). Plus précisément on a les résultats suivants :

Définition 9
Soient σ et σ' deux affectations possibilistes.

La règle de combinaison conjonctive est :

$(\sigma \otimes \sigma')(\emptyset) = 0$ *et*

$\forall A \subseteq \Omega$, $(\sigma \otimes \sigma')(A) = \max_{B \cap C = A \neq \emptyset} \min(\sigma(B), \sigma'(C))$.

La règle de combinaison disjonctive est :

$(\sigma \oplus \sigma')(\emptyset) = 0$ *et*

$\forall A \subseteq \Omega$, $(\sigma \oplus \sigma')(A) = \max_{B \cup C = A \neq \emptyset} \min(\sigma(B), \sigma'(C))$.

La règle de combinaison conjonctive a déjà été suggérée dans (Dubois & Prade, 1985).

Remarque : La fonction $\sigma \otimes \sigma'$ est une affectation possibiliste si et seulement si il existe B et C tels que $B \cap C \neq \emptyset$ et $\sigma(B) = \sigma'(C) = 1$. Si les affectations possibilistes sont normales alors $\sigma \oplus \sigma'$ est une affectation possibiliste. De plus la règle de combinaison $\otimes$ conserve les propriétés suivantes :

Proposition 9
$\otimes$ est commutative : $\sigma \otimes \sigma' = \sigma' \otimes \sigma$ pour tout σ,σ'.

$\otimes$ est associative : $(\sigma \otimes \sigma') \otimes \sigma'' = \sigma \otimes (\sigma' \otimes \sigma'')$.

$\otimes$ possède une élément neutre σ_0 : $\sigma_0 \otimes \sigma = \sigma$

avec σ_0 l'affectation possibiliste qui vaut 0 partout sauf pour Ω.

L'opérateur $\oplus$ est aussi commutatif et associatif. Si on note $Bel^{pos} \oplus Bel^{pos\prime}$ la fonction de croyance possibiliste associée à l'affectation possibiliste $\sigma \oplus \sigma'$ et $Q^{pos} \otimes Q^{pos\prime}$ la communalité possibiliste associée à l'affectation possibiliste $\sigma \otimes \sigma'$, alors les propriétés suivantes, vérifiées dans la théorie des fonctions de croyance, restent vraies dans le cas qualitatif.

Proposition 10
$Bel_1^{pos} \oplus Bel_2^{pos} = \min(Bel_1^{pos}, Bel_2^{pos})$ *et* $Q_1^{pos} \otimes Q_2^{pos} = \min(Q_1^{pos}, Q_2^{pos})$.

7 Conclusion

Dans cet article, nous avons esquissé un début de contrepartie qualitative à la théorie des fonctions de croyance. Il est assez remarquable qu'en remplaçant la somme par le maximum, nous préservons des propriétés du cas classique.

Il est clair que si nous avions remplacé la somme par le minimum, au lieu du maximum, on obtiendrait une structure minitive avec des propriétés similaires, mais avec une interprétation différente. Par exemple, $\min_{B \subseteq A} \sigma(B)$ suggère que σ doit maintenant être comprise comme une affectation de nécessité.

Au-delà des résultats présentés dans cet article, de nombreuses questions demeurent. L'une d'entre elles concerne les fonctions de croyance possibilistes. En effet dans le cas classique les fonctions de croyance séparables sont décomposables en fonctions à support simple et cette décomposition est unique. Il semble aussi naturel de définir les fonctions à support simple dans notre contexte (Prade & Rico, 2011), mais cette décomposition sera t-elle toujours possible et unique (Chemin *et al.*, 2011) ?

Par ailleurs, nous avons récemment proposé une méthode pour décrire des objets acceptables par les intégrales de Sugeno (Prade & Rico, 2010). Plus précisément pour des objets décrits par des combinaisons de propriétés, nous avons étudié l'utilisation des intégrales de Sugeno comme un outil de représentation de ces objets. Dans cette méthode l'intégrale de Sugeno a été définie en utilisant la mesure qui est nommée Bel^{pos} dans ce papier. Nous espérons que les implications entre les affectations possibilistes nous aiderons à mieux comprendre les implications entre les intégrales de Sugeno, et comment les objets sont sélectionnés avec les intégrales de Sugeno.

Une autre voie de recherche digne d'intérêt, considérant que plusieurs affectations possibilistes distinctes conduisent aux mêmes fonctions de croyance si elles sont associées à la même borne inférieure de la transformée de Möbius qualitative, serait l'étude d'une extension du cadre présenté ici basée non plus sur $\max$ (et $\min$), mais sur leur raffinement leximin et leximax de façon à éviter des effets de noyade de certains des éléments focaux de l'affectation possibiliste (et sa non-unicité).

Références

BENFERHAT S., HUÉ J., LAGRUE S. & ROSSIT J. (2011). Interval-based possibilistic logic. In *Proc. 22nd Inter. Joint Conf. on Artif. Intellig. (IJCAI'11)*, p. 16–22, Barcelona.

CHEMIN M., RICO A. & PRADE H. (2011). Decomposition of possibilistic belief functions into simple support functions. In *submited*.

DEZERT J. & SMARANDACHE F. (2006). Introduction to the fusion of quantitative and qualitative beliefs. *int. Journal of Information and Security*, **20**, 9–49.

DUBOIS D. (2011). Fuzzy measures on finite scales as families of possibility meas. In *in conf. proc of European Society For Fuzzy Logic and Technology (EUSFLAT-LFA)*, Aix-Les-Bains, France.

DUBOIS D. & FARGIER H. (2009). Capacity refinements and their application to qualitative decision evaluation. In C. SOSSAI & G. CHEMELLO, Eds., *Proc. 10th Europ. Conf. on Symboloic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, LNCS 5590, p. 311–322, Verona.

DUBOIS D. & PRADE H. (1984). Upper and lower possibilities induced by a multivalued mapping. In E. SANCHEZ, Ed., *Proc. IFAC Symp. on Fuzzy Information, Knowledge Representation and Decision Analysis*, Pergamon Press, p. 152–174, Marseille.

DUBOIS D. & PRADE H. (1985). Evidence measures based on fuzzy information. *Automatica*, **21**, 547–562.

DUBOIS D. & PRADE H. (1986). A set-theoretic view of belief functions. logical operations and approximations by fuzzy sets. *Int. J. General Systems*, **12**, 193–226.

DUBOIS D. & PRADE H. (2004). Possibilistic logic : a retrospective and prospective view. *Fuzzy Sets and Systems*, **144**, 3–23.

GRABISCH M. (2003). The symmetric Sugeno integral. *Fuzzy Sets and Systems*, **139**(3), 473–490.

GRABISCH M. (2004). The Möbius transform on symmetric ordered structures and its application to capacities on finite sets. *Discrete Mathematics*, **287**(1-3), 17–34.

PARSONS S. & MAMDANI E. (1993). Qualitative Dempster-Shafer theory. In *proc. III Imacs International Workshop on Qualitative Reasoning and Decision Technologies*.

PRADE H. & RICO A. (2010). Describing acceptable objects by means of Sugeno integrals. In *Proc. 2nd IEEE International Conference of Soft Computing and Pattern Recognition*, Cergy-Pontoise.

PRADE H. & RICO A. (2011). Possibilistic evidence. In *Conf. on Symbolic and Quantitative Approaches to Reasoning with Uncertainty (ECSQARU)*, LNCS 5590, p. 713–722, Belfast.

SHAFER G. (1976). *A Mathematical Theory of Evidence*. Princeton Univ. Press. Princeton, NJ.

SUGENO M. (1977). Fuzzy measures and fuzzy integrals : a survey. *Fuzzy Automata and Decision Processes (M.M. Gupta, G.N. Saridis, and B.R. Gaines, eds.)*, p. 89–102.

TSIPORKOVA E. & BAETS B. D. (1998). A general framework for upper and lower possibilities and necessities. *Inter. J. of Uncert., Fuzz. and Knowledge-Based Syst.*, **6**, 1–34.

ZADEH L. A. (1978). Fuzzy sets as a basis for a theory of possibility. *Fuzzy Sets and Systems*, **1**, 3–28.

Avec le soutien de