Traces génériques et contraintes, GenTra4CP revisité

Pierre Deransart

INRIA Domaine de Voluceau, Rocquencourt, BP 105, 78153 Le Chesnay Cedex
Pierre.Deransart@inria.fr

Résumé

Le format de trace générique GenTra4CP a été défini en 2004 avec l'objectif de devenir un format de trace standard pour l'observation des solveurs de contraintes sur domaines finis. Il n'a pratiquement plus été utilisé depuis. Cet article reprend en la généralisant l'idée de trace générique et en donne un cadre formel basé sur des transformations simples de traces. Il analyse alors les limites du format initial proposé, apporte quelques précisions, et montre l'intérêt qu'un traceur générique peut avoir pour développer des applications portables, ou pour des efforts de normalisation, en particulier des contraintes.

Abstract

The generic trace format GenTra4CP has been defined in 2004 with the goal of becoming a standard trace format for the observation of constraint solvers over finite domains. It has not been used afterwards. This paper defines the concept of generic trace formally, based on simple trace transformations. It then analyzes and occasionally corrects shortcomings of the proposed initial format and shows the interest that a generic tracer may bring for developing portable applications or for standardization puposes, in particular in the field of constraints.

1 Introduction

Suite au projet RNTL OADymPPaC [1], un format de trace générique, appelé GenTra4CP (Generic Trace for CP), a été proposé en 2004 pour spécifier des traces de résolution de CSP(FD) dans le but de pouvoir réaliser des outils portables d'analyse de comportement de solveurs. Ce format se présente comme une sorte de norme comportant une syntaxe précise des événements de trace incluant une DTD XML, et une sémantique opérationnelle partielle, dite sémantique observationnelle, qui est en fait une sémantique opérationnelle commune à un ensemble potentiel de solveurs sur les domaines finis.

Des traceurs conformes au standard ainsi défini ont été implantés sur quatre solveurs. Plusieurs outils d'analyse de comportement de la résolution et de stratégies de recherche, ont été mis au point dans quatre environnements différents, à partir de la trace générique GenTra4CP. Ils ont pu être utilisés avec succès après un travail d'adaptation minimal pour chacun d'eux dans plusieurs environnements. Pour autant, à cette époque, aucune caractérisation formelle de la nature "générique" du format de trace n'a pu être donnée. Si la réalisation d'outils à partir d'un format de trace ainsi défini ne posait semble-t-il aucun problème et si l'on avait obtenu un gain considérable en portabilité, il était pratiquement impossible d'apprécier a priori l'effort d'adaptation nécessaire pour que les solveurs puissent utiliser les outils ainsi développés. De plus il n'était pas toujours possible de se rendre compte précisément de ce que certains outils "observaient" effectivement une fois adaptés. Le format GenTra4CP n'a de fait pu être utilisé que dans le projet qui l'a défini.

Cet article tente de combler ces lacunes et reprend en la formalisant l'idée de trace générique. Il analyse formellement la nature générique du format GenTra4CP et les limites du format proposé qui ont pu faire obstacle à son utilisation. Il montre également l'intérêt qu'un traceur générique peut avoir pour des efforts de normalisation, en particulier des contraintes, en proposant une approche de la sémantique reposant sur une sémantique opérationnelle partielle basée sur les traces.

Après une partie préliminaire où l'on adopte une approche de la sémantique opérationnelle d'un processus observé basée sur ses traces, la section 3 introduit quelques relations d'abstraction entre traces

dans le but d'appréhender formellement le concept de trace générique et d'apporter une méthode de preuve de conformité d'un processus à la trace générique. La section 4 expose la mise en œuvre pratique de cette approche et son intérêt en développement de logiciels. La section 5 applique cette approche au cas de GenTra4CP en vérifiant en particulier la conformité d'un solveur. Les descriptions formelles utilisées sont extraites de [1] pour GenTra4CP et [7] pour les solveurs. Cela permet de mieux appréhender la force et les limites de cette approche telle qu'elle a pu être utilisée à l'époque. On peut alors établir un lien entre les efforts de standardisation des contraintes et la méthode de spécification à partir de trace générique (section 6).

2 Préliminaires

Un objet *trace* est constitué d'un état initial s_0 suivi d'une suite finie ou infinie d'*événements de trace* ordonnés, et notée $< s_0, \overline{e} >$. Un ensemble de traces est noté $\mathcal{T}$. Un *préfixe* (fini, de taille t) d'une trace $T = < s_0, \overline{e_n} >$ (finie ou infinie, ici de taille n supérieure ou égale à t) est une trace partielle $U_t =< s_0, \overline{e_t} >$ correspondant aux t premiers événements de T, avec en plus au début l'état initial. Un préfixe réduit à l'état initial est de taille 0. L'ensemble des préfixes de $\mathcal{T}$ sera noté $Pref(\mathcal{T})$, avec $\mathcal{T} \subseteq Pref(\mathcal{T})$.

Toute trace est décomposable en segments ne comportant que des événements de trace, sauf le premier (dans l'ordre chronologique) qui débute par un état. On notera une décomposition avec un opérateur associatif de concaténation dénoté par un point et d'élément neutre ϵ (séquence vide). Ainsi une trace $T = T_1.T_2$ correspond à la trace T_1 (préfixe de T) à laquelle est concaténée la séquence composée des événements T_2. Une trace peut être le reflet de transitions d'états et à chaque événement peut correspondre un nouvel état. Dans ce cas un segment peut également être noté sT_ts_t où s est l'état dans lequel la séquence T_t débute, et s_t l'état atteint (état après le dernier événement de la séquence). Un segment (ou préfixe) de taille nulle est donc réduit soit à la suite vide, soit à un état.

Un *domaine de traces* sur $\mathcal{T}$, $\mathcal{DT}_{\mathcal{T}}$, est un ensemble dont chaque élément est l'ensemble de tous les préfixes d'une ou plusieurs traces de $\mathcal{T}$. Un élément est clos par préfixe. Un tel ensemble est clos pour l'union et l'intersection, et étant donnés deux éléments de $\mathcal{DT}_{\mathcal{T}}$ inclus l'un dans l'autre, le "plus petit" contient l'ensemble de tous les préfixes de quelques traces du "plus grand". Cet ensemble forme un treillis complet $\mathcal{DT}_{\mathcal{T}}(\subseteq, \bot, \top, \cup, \cap)$ dont $\bot$ est l'ensemble vide et $\top = Pref(\mathcal{T})$.

Les traces sont destinées à décrire l'évolution de systèmes décrits par l'évolution de leur état et/ou des suites d'actions. On distinguera deux types de traces :

- les *traces virtuelles* (ensemble $\mathcal{T}^v$) dont les événements sont de la forme $e = (r, s)$ où r est un *type d'action* nommant une transition d'un état à un autre et s, dit *état virtuel*, le nouvel état (état d'arrivée de la transition) décrit par un ensemble de *paramètres*.
- les *traces effectives* (ensemble $\mathcal{T}^w$) dont les événements sont de la forme $e = (a)$ où a est un *état effectif* décrit par un ensemble d'*attributs*. La trace effective est destinée à coder les changements d'états virtuels d'une manière plus synthétique.

On donne ici une description simplifiée de la sémantique des traces, appelée sémantique observationnelle dans [3].

Definition 1 (Sémantique Observationnelle (SO)) *Une Sémantique Observationnelle est définie par la donnée du sextuplet* $< S, R, A, T, E_l, I_l, S_0 >$*, où*

- S *:* domaine des états virtuels*,*
- R *: ensemble fini des* types d'actions*, ensemble d'identificateurs étiquetant les transitions.*
- A *:* domaine des états effectifs*,*
- T *:* fonction de transition d'états $T : R \times S \rightarrow S$*, notée* $T(r, s) = s'$ *ou* $T(r, s, s')$ *si c'est une relation,*
- E_l *:* fonction locale d'extraction de trace $E_l : S \times R \times S \rightarrow A$*,*
- I_l *:* fonction locale de reconstruction $I_l : S \times A \rightarrow R \times S$*,*
- $S_0 \subseteq S$*, ensemble des* états initiaux.

Les fonctions locales d'extraction E_l et de reconstruction I_l peuvent être étendues en fonctions E (resp. I) entre traces virtuelles et effectives, et doivent vérifier la relation dite de *fidélité*, $I = E^{-1}$. Les fonctions étendues peuvent être définies à partir des fonctions locales :

$E(s_0e_1...e_i...) = s_0E_l(s_0, r_1, s_1)...E_l(s_{i-1}, r_i, s_i)...$ avec $E_l(s_{i-1}, r_i, s_i) = a_i$, and

$I(s_0a_1...a_i...) = s_0I_l(s_0, a1)...I_l(s_i, a_{i+1})...$ avec $I_l(s_{i-1}, a_i) = (r_i, s_i)$.

Les fonctions locales et de transition seront représentées par des ensembles de règles (figure 1).

La sémantique observationnelle d'un processus observé peut être considérée comme une abstraction d'une sémantique opérationnelle raffinée [2]. Cette relation d'abstraction peut être exprimée comme une relation entre domaines de traces, et aussi bien entre traces effectives que traces virtuelles. Du fait de la propriété de fidélité, la fonction d'abstraction D_w portant sur les traces effectives vérifie avec D_v, la fonction

d'abstraction entre les traces virtuelles, les relations suivantes : $D_v = E_c \circ D_w \circ I_d$ et $D_w = I_c \circ D_v \circ E_d$ ($\circ$ denote la composition de fonctions). Les deux formules se déduisent l'une de l'autre du fait de la propriété de fidélité.

Dans la suite, on supposera la propriété de fidélité vérifiée, quelque soit le niveau de description. Dans ce cas les fonctions d'abstraction et de reconstruction peuvent se déduire l'une de l'autre. Dans la suite, on utilisera donc les seules traces virtuelles pour établir les propriétés recherchées.

$$\mathsf{reduce}\ \frac{< \mathcal{D}(v),\quad S_e,\quad A >}{< \mathcal{D}(v) - \Delta_v^c,\quad S_e \cup \bar{a},\quad A' >} \left\{ \begin{array}{l} \text{supprime } \Delta_v^c), \\ a \text{ réveille } c \\ (c,a) \in A, \\ A' = A - \{(c,a)\} \\ v \in \mathbf{var}(c), \\ \text{génère } \bar{a} \end{array} \right\}$$

$$\mathsf{reduce}\ \frac{\left\{ \begin{array}{l} < \mathcal{D}(v),\quad S_e,\quad A' \cup \{(c,a)\} > \rightarrow \\ < \mathcal{D}'(v),\quad S'_e,\quad A' > \end{array} \right\}}{[\mathsf{reduce},\quad c,\quad v,\quad (S'_e - S_e),\quad (\mathcal{D}(v) - \mathcal{D}'(v)),\quad a]}\ \{\}$$

$$\mathsf{reduce}\ \frac{[\mathsf{reduce},\quad c,\quad v,\quad \bar{a},\quad \Delta_v^c,\quad a]}{\left\{ \begin{array}{l} < \mathcal{D}(v),\quad S_e,\quad A > \rightarrow \\ < \mathcal{D}(v) - \Delta_v^c,\quad S_e \cup \bar{a},\quad A - (c,a) > \end{array} \right\}}\ \{\}$$

FIGURE 1 – Description de la transition reduce dans la SO de GenTra4CP

3 Sous-trace et trace dérivée

On considère ici quelques transformations de traces, relativement simples à identifier et suffisantes pour les besoins de cette approche. Comme on peut se contenter de définir les transformations sur les traces virtuelles, on n'utilise qu'une partie de leur SO, à savoir $< S, R, T, S_0 >$.

Les sous-traces dites "paramétriques" sont obtenues à partir d'un sous-ensemble de paramètres.

Definition 2 (Sous-trace paramétrique) *Etant donnés un ensemble de traces $\mathcal{T}$ spécifié par $< S, R, T, S_0 >$, si S' est un sous-ensemble de S défini par un sous-ensemble des paramètres de S qui ne dépend*[1] *d'aucun paramètre de $S - S'$, R' est un sous-ensemble de R utilisant ou modifiant ces seuls paramètres tels qu'aucun autre type d'action de $R - R'$ ne les modifie, S'_0 est la restriction de S_0 à S', et T' la restriction de T à S' et R', alors l'ensemble de traces $\mathcal{T}'$ spécifié par $< S', R', T', S'_0 >$ est sous-trace (paramétrique) de $\mathcal{T}$, noté $Sub_P(\mathcal{T}, \mathcal{T}')$.*

1. Un paramètre p dépend d'un autre p' si p' intervient dans le calcul d'une valeur de p dans une transition.

Nota : il est possible d'avoir $S' \subseteq S$ et $R' = R$ ($S - S'$ contient des paramètres redondants, c'est à dire qu'ils ne dépendent que des autres paramètres de S et peuvent ainsi être supprimés).

Definition 3 (Champ de dérivation et trace dérivée)

Etant donnés deux ensembles de traces $\mathcal{T}_c$ et $\mathcal{T}_d$, où $\mathcal{T}_c$ et $\mathcal{T}_d$ sont dits respectivement concret *et* dérivé*, l'ensemble de traces $\mathcal{T}_d$ est un champ de dérivation par D de $\mathcal{T}_c$ s'il existe une application $D : Pref(\mathcal{T}_c) \rightarrow Pref(\mathcal{T}_d)$, dite* dérivation *telle que*

pour tout préfixe fini dérivé t_d de taille n et pour tout préfixe concret t_c dont il est l'image, il existe une suite croissante de préfixes concrets $[t_c^0, t_c^1, ..., t_c^i, ..., t_c^{n-1}, t_c]$ (non nécessairement contigus), appartenant tous au domaine de D, et tels que

- $D(t_c^0) \in S_{0,d}$,
- $\forall i > 0$ *si* $D(t_c^i) = t_d^i$ *avec* t_d^i *préfixe de* t_d *constitué des i premiers événements , alors* $D(t_c^{i+1}) = t_d^{i+1}$.

Si D est surjective, l'ensemble $\mathcal{T}_d$ est dit trace dérivée *par D de $\mathcal{T}_c$, que l'on peut noter par la relation binaire $Drv_D(\mathcal{T}_c, \mathcal{T}_d)$.*

Nota : si une trace dérivée t_d est de taille nulle et si elle est image de t_c, alors t_c ne peut être décomposée.

D est une fonction partielle. Il est possible de la rendre totale en considérant que tous les éléments de $S_{0,c}$ ont une image dans $S_{0,d}$ et que tout préfixe entre t_c^i et t_c^{i+1} a pour image $D(t_c^i)$ [2].

Cette définition est illustrée par le diagramme 2.

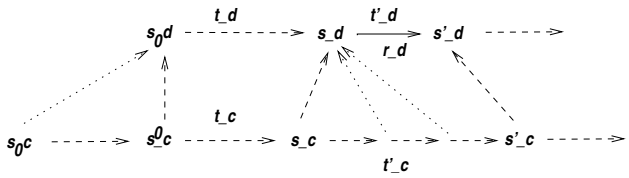

FIGURE 2 – Dérivation entre traces concrètes et dérivée (les flèches pointillées correspondent à la dérivation totale), t_x dénote un préfixe.

L'approche adoptée ici privilégie l'existence a priori d'un ensemble de traces, sans préjugé de la manière dont il a été produit ou spécifié. Elle favorise l'idée que la trace dérivée est le résultat de calculs sur les traces dites concrètes et formalisés par l'application D. A noter également que cette définition implique que la trace concrète qui a une image par D a une décomposition unique en préfixes du domaine de D.

2. Ainsi une trace, dont aucun préfixe de taille non nulle n'est dans le domaine de D, a pour image un état de $S_{0,d}$.

Si les traces dérivées peuvent être vues comme calculées à partir de traces concrètes, ces dernières peuvent être considérées comme des "implantations" ou des raffinements des premières dans un système où les états sont plus précis et où il y a des actions plus détaillées. Si une sous-trace peut être vue comme une dérivation, celle-ci s'en distingue par le fait qu'elle n'est pas nécessairement une sous-trace.

Propriétés 1

Etant données deux dérivations D_1 et D_2, si D_1 est surjective ou si D_2 est totale, $D_1 \circ D_2$ est une dérivation.

Une sous-trace au sens de la définition 2 est une trace dérivée.

Dans ce qui suit, on donne une méthode de preuve pour établir que deux ensembles de traces spécifiés par leur SO sont liés par une dérivation.

Definition 4 (Trace simulable)

Etant donnés deux ensembles de traces $\mathcal{T}_c$ (concret) et $\mathcal{T}_d$ (dérivé), définis respectivement avec $< S_c, R_c, T_c, S_{0,c} >$ et $< S_d, R_d, T_d, S_{0,d} >$, $\mathcal{T}_c$ est simulable *dans $\mathcal{T}_d$ si R_c et R_d sont en correspondance biunivoque par h, et s'il existe une application $d : S_c \to S_d$ qui vérifie :*

- $\forall s_0 \in S_{0,c}, d(s_0) \in S_{0,d}$.
- $\forall r_c \in R_c, s_c, s'_c \in S_c,\ T_c(s_c, r_c, s'_c) \Rightarrow \exists s_d, s'_d \in S_d, d(s_c) = s_d \wedge d(s'_c) = s'_d \wedge T_d(s_d, h(r_c), s'_d)$.

Théorème 1

Etant donnés deux ensembles de traces $\mathcal{T}_c$ (concret) et $\mathcal{T}_d$ (dérivé), tels que $\mathcal{T}_c$ est simulable dans $\mathcal{T}_d$, alors $\mathcal{T}_d$ est également un champ de dérivation pour $\mathcal{T}_c$ et la dérivation associée est totale.

Corollaire 1

Etant donnés deux ensembles de traces $\mathcal{T}$ et $\mathcal{T}'$ tels qu'il existe une sous-trace paramétrique de $\mathcal{T}$ simulable dans $\mathcal{T}'$, alors $\mathcal{T}'$ est un champ de dérivation pour $\mathcal{T}$.

4 Traces génériques

L'idée de trace générique répond à des besoins de spécification et de portabilité. On décrit un processus ou un algorithme par son comportement observable, c'est à dire par la trace d'opérations abstraites qu'il est censé implémenter. De plus on souhaite que cette description soit assez générale pour servir à la description d'une famille de processus, et enfin, on veut pouvoir l'utiliser dans certaines applications pour lesquelles cette information est suffisante. Ce peut être le cas d'applications comme le monitorage, le débogage, la construction d'outils de visualisation du comportement, ou même d'une application dont la trace a été définie a priori.

Definition 5 (Trace générique (TG)) *Etant donnée une famille de processus $p \in P$, à chacun desquels correspond un ensemble de traces $\mathcal{T}_p$, un ensemble de traces $\mathcal{T}_g$ est dit générique pour P, si, pour chaque processus p de la famille, il existe une dérivation de ses traces $\mathcal{T}_p$ par D_p, qui est une sous-trace de $\mathcal{T}_g$, soit*
$\forall p \in P,\ \exists\ \mathcal{T}$ *telle que* $Drv_{D_p}(\mathcal{T}_p, \mathcal{T}) \wedge Sub_P(\mathcal{T}_g, \mathcal{T})$.

Les principales questions qui se posent alors sont les suivantes.

- Comment s'assurer que l'on a correctement implanté la TG dans un processus particulier, ou, si on considère que la TG est une sorte de norme, en quoi une trace produite par ce processus est "conforme" à la TG ?
- La TG peut-elle être utilisée directement pour développer des applications dont on pourra garantir qu'elles fonctionneront avec tous les processus conformes ?
- La TG peut-elle être étendue à d'autres processus sans remettre en cause les implantations déjà réalisées et les applications existantes ?

Les questions précédentes ont alors les réponses possibles suivantes.

Conformité à la trace générique

On dira que la trace est générique pour un processus si elle satisfait la définition 5, c'est à dire qu'il existe une sous-trace de la trace générique qui est une dérivation de la trace (prédéfinie ou non) du processus. On dira dans ce cas que la trace du processus est "conforme" à la trace générique. Il est alors possible, soit d'implanter exactement la trace générique et donc que le traceur produise $Sub_P(T_p, T_g)$, soit d'implanter un traceur dans le processus p et prouver que $\exists \mathcal{T}', Drv_{D_p}(\mathcal{T}_p, \mathcal{T}') \wedge Sub_p(\mathcal{T}_g, \mathcal{T}')$.

Mise au point d'outils à partir de la trace générique

L'intérêt d'une trace générique est qu'elle facilite la mise au point d'outils susceptibles d'être utilisables sur tous les processus conformes. La mise au point s'effectue en considérant que l'outil utilise au moins des sous-traces de la TG correspondant à des processus décrits, voire toute la trace. Cela est possible à partir de la connaissance de la sémantique de la TG, c'est à dire de la connaissance de sa SO. L'utilisation pourra alors se faire à partir de chaque trace conforme des processus de la famille. L'effort d'adaptation de l'outil est alors réduit au minimum possible : celui de l'implantation de la dérivation D au niveau du processus (modification du traceur du processus pour produire la TG), ou au niveau de l'outil (adaptation de la trace du

processus pour reproduire la TG). La figure 3 illustre les deux possibilités.

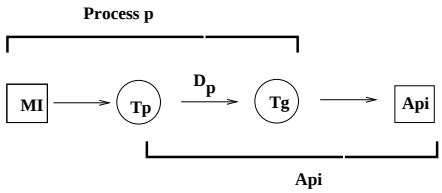

FIGURE 3 – Utilisations d'une trace générique : adaptation du processus ou de l'application

L'existence d'une spécification formelle de la trace générique permet également d'en réaliser un prototype qui s'ajoutera aux processus potentiellement décrits et devra répondre aux mêmes critères de conformité. Pour tester les outils en cours de mise au point on peut donc le faire, soit à partir d'un processus conforme, soit à partir d'un prototype conforme également. Tout ce que la démarche garantit est que si un outil est mis au point à partir d'un processus conforme dont une abstraction est la trace générique complète, il pourra fonctionner sur toutes les autres implantations conformes.

Extensions de la trace générique

Tant que les extensions de la TG préservent le fait que les processus conformes le sont par rapport à une sous-trace de la TG étendue, leur conformité par rapport à la TG étendue est préservée. Toute extension est telle que la trace originale reste une sous-trace générique. De cette manière les outils continuent de fonctionner sur les processus pour lesquels ils ont été réalisés.

5 La trace générique GenTra4CP

Dans le document final [1], la trace générique GenTra4CP est définie par une SO partielle[3] et par de nombreux attributs de la trace effective, très détaillé mais informellement décrits. Une syntaxe partielle sous forme d'une DTD XML décrivant chaque type d'événement de la trace effective est donnée.

On reprend ici le la description originale des états virtuels de GenTra4CP [1] (section 3.3.1)[4]

< < Début de Citation...

Definition 6 *(Solver State)*
A solver state is a 8-tuple : $\mathbb{S} = (\mathcal{V}, \mathcal{C}, \mathcal{D}, A, E, R, S_c, S_e)$
where : $\mathcal{V}$ *is the set of declared variables ;* $\mathcal{C}$ *is the set of declared constraints ;* $\mathcal{D}$ *is the function that assigns to each variable in* $\mathcal{V}$ *its domain (a set of values in the finite set* D*) ;* A *is the set of active couples of the form (constraint, solver event) ;* E *is the set of solved constraints ;* R *is the set of unsatisfiable (or rejected) constraints.* S_c *is the set of sleeping constraints ;* S_e *is the set of solver events to propagate ("sleeping events").*

A, S_c, E and R are used to describe four specific states of a constraint during the propagation stage : active, sleeping, solved or rejected.

The *store of constraints* constraints taken into account. The store is called σ in the following and defined as the partition $\sigma = \{c \mid \exists (c, a) \in A\} \cup S_c \cup E \cup R$. All the constraints in σ are defined, thus $\sigma \subseteq \mathcal{C}$. The set of variables involved in the c constraint is denoted by $\mathbf{var}(c)$. The predicate $false(c, \mathcal{D})$ (resp. $solved(c, \mathcal{D})$) holds when the constraint c is considered as unsatisfiable (resp. solved : it is universally true and does not influence further reductions any more) by the domains in $\mathcal{D}$.

The search is often described as the construction of a search-tree.

Definition 7 *(Search-Tree State)*
The search-tree is formalized by a set of ordered labeled nodes $\mathcal{N}$ *representing a tree, and a function* Σ *which assigns to each node a solver state. The nodes in* $\mathcal{N}$ *are ordered by the construction. Three kinds of nodes are defined and characterized by three predicates : failure leave (*$failed(\mathbb{S})$*), solution leave (*$solution(\mathbb{S})$*), and choice-point node (*$choice\text{-}point(\mathbb{S})$*). The last visited node is called* current node *and is denoted* n*. The usual notion of* depth *is associated to the search-tree : the depth is increased by one between a node and its children. The function* δ *assigns to a node* n *its depth* $\delta(n)$*. Therefore, the state of the search-tree is a quadruple :* $\mathbb{T} = (\mathcal{N}, \Sigma, \delta, n)$.

In the initial solver state, n_0 denotes the root of the search-tree and all the sets that are part of $\mathbb{S}$ are empty.

... Fin de Citation > >

Le reste de la description consiste à décrire chaque type d'événement de trace effective (appelé dans [7] "generic trace schema") en y introduisant d'autres attributs plus ou moins redondants (par exemple les identifiants interne et externes d'une contrainte).

On illustre l'approche formelle de la trace générique en analysant formellement une des "implantations" de GenTra4CP proposées dans [7]. Dans cet article, trois "spécialisations" de la trace génériques sont proposées pour 3 solveurs (GNU-Prolog, Choco et PaLM). Elles

3. SO partielle dans la mesure où seuls quelques paramètres sont considérés pour le modèle sémantique.

4. On utilise ici n au lieu de ν pour désigner le nœud courant de l'arbre de choix.

$$\text{new variable}\ \frac{\mathcal{V},\ \ \mathcal{D}}{\mathcal{V}\cup\{v\},\ \ \mathcal{D}\cup\{(v,\mathcal{D}_{v,i})\}}\ \left\{\begin{array}{l} v\notin\mathcal{V},\\ \mathcal{D}(v)=D_{v,i}\end{array}\right\}$$

$$\text{new constraint}\ \frac{\mathcal{C}}{\mathcal{C}\cup\{c\}}\ \left\{\begin{array}{l} c\notin\mathcal{C},\\ \mathbf{var}(c)\subseteq\mathcal{V}\end{array}\right\}$$

$$\text{post}\ \frac{A}{A\cup\{(c,\bot)\}]}\ \left\{\begin{array}{l} c\in\mathcal{C},\\ c\notin\sigma\end{array}\right\}$$

$$\text{choice point}\ \frac{\mathcal{N},\ \ \Sigma,\ \ \mathbb{S}}{\mathcal{N}\cup\{n\},\ \ \Sigma\cup\{(n,\mathbb{S})\},\ \ n}\ \left\{\begin{array}{l} \textit{ch-pt}(\mathbb{S}),\\ n\notin\mathcal{N}\end{array}\right\}$$

$$\text{back to}\ \frac{\mathbb{S},\ \ \nu}{\Sigma(n),\ \ n}\ \left\{\begin{array}{l} n\neq\nu,\\ n\in\mathcal{N},\\ \textit{ch-pt}(\mathbb{S})\end{array}\right\}$$

$$\text{solution}\ \frac{\mathcal{N},\ \ \Sigma,\ \ \mathbb{S}}{\mathcal{N}\cup\{n\},\ \ \Sigma\cup\{(n,\mathbb{S})\},\ \ n}\ \left\{\begin{array}{l} \textit{sol}(\mathbb{S}),\\ n\notin\mathcal{N}\end{array}\right\}$$

$$\text{failure}\ \frac{\mathcal{N},\ \ \Sigma,\ \ \mathbb{S}}{\mathcal{N}\cup\{n\},\ \ \Sigma\cup\{(n,\mathbb{S})\},\ \ n}\ \left\{\begin{array}{l} \textit{flr}(\mathbb{S}),\\ n\notin\mathcal{N}\end{array}\right\}$$

$$\text{remove}\ \frac{\sigma}{\sigma-\{c\}}\ \left\{\ c\in\sigma\ \right\}$$

$$\text{restore}\ \frac{\mathcal{D}(v)}{\mathcal{D}(v)\cup\Delta_v}\ \left\{\begin{array}{l} v\in\mathcal{V},\\ \Delta_v\cap\mathcal{D}(v)=\emptyset,\\ \Delta_v\subseteq\mathcal{D}_{v,i}\end{array}\right\}$$

FIGURE 4 – SO de GenTra4CP (contrôle, sans le paramètre δ)

Contrôle		**Propagation)**	
new variable	$v,\ \ D_{v,i}$	reduce	$c,\ \ v,\ \ \bar{a},\ \ \Delta_v^c,\ \ \mathrm{a}$
new constraint	c	suspend, solved	c
post, remove	c	reject	$c,\ \ a$
restore	$v,\ \ \Delta_v$	awake	$c,\ \ a$
choice point	n	schedule	$c,\ \ a$
back to	$n,\ \ n'$		
solution, failure	n		

TABLE 1 – Attributs de la trace effective décrits dans la SO de GenTra4CP

consistent en une description de la sémantique opérationnelle de chaque solveur par une SO. On montre ici la conformité de la sémantique donnée pour PaLM [6]. Ce choix vient de ce que PaLM est, parmi les trois solveurs testés, le plus éloigné des principes qui sous-tendent le format GenTra4CP. La SO de GenTra4CP est rappelé dans les figures 4 et 5.

Afin de montrer que la SO de PaLM est bien conforme, on doit prendre en compte quelques propriétés de la trace générique, en particulier :

(G1) $\textit{sol}(\mathbb{S})\Rightarrow R=\emptyset$
(G2) $\textit{flr}(\mathbb{S})\Leftrightarrow R\neq\emptyset$
(G3) $(\textit{evtype}=\text{reduce})\Rightarrow R=\emptyset$
(G4) $(\textit{evtype}=\text{awake})\Rightarrow(R=\emptyset\wedge A=\emptyset)$
(G5) $(\textit{evtype}=\text{schedule})\Rightarrow(R=\emptyset\wedge A=\emptyset)$

On admet :

$$\text{reduce}\ \frac{<\mathcal{D}(v),\ \ S_e,\ \ A>}{<\mathcal{D}'(v),\ \ S'_e,\ \ A'>}\ \left\{\begin{array}{l} \mathcal{D}'(v)=\mathcal{D}'(v)-\Delta_v^c\\ \text{supprime}\ \Delta_v^c\\ (c,a)\in A\\ v\in\mathbf{var}(c)\\ \text{Red. gén. }\bar{a}\\ A'=A-(c,a)\\ S'_e=S_e\cup\bar{a}\end{array}\right\}$$

$$\text{suspend}\ \frac{<A,\ \ S_c>}{<A-\{(c,a)\},\ \ S_c\cup\{c\}>}\ \{(c,a)\in A\}$$

$$\text{solved}\ \frac{<A,\ \ E>}{<A-\{(c,a)\},\ \ E\cup\{c\}>}\ \left\{\begin{array}{l} (c,a)\in A,\\ \textit{solved}(c,\mathcal{D})\end{array}\right\}$$

$$\text{reject}\ \frac{<A,\ \ R>}{<A-\{(c,a)\},\ \ R\cup\{c\}>}\ \left\{\begin{array}{l} (c,a)\in A,\\ \textit{false}(c,\mathcal{D})\end{array}\right\}$$

$$\text{awake}\ \frac{<A,\ \ S_c>}{<A\cup\{(c,a)\},\ \ S_c-\{c\}>}\ \left\{\begin{array}{l} c\in S_c\\ a\in S_e\cup\{\bot\}\\ \textit{awcond}(c,a)\end{array}\right\}$$

$$\text{schedule}\ \frac{<S_c,\ \ S_e>}{<S'_c,\ \ S'_e>}\ \left\{\begin{array}{l} c\in S_c,\\ e\in S_e,\\ \textit{action}(c,a)\end{array}\right\}$$

FIGURE 5 – SO de GenTra4CP (propagation)

(P1) $\textit{dependence}(c,a)\Leftrightarrow\textit{awcond}(c,a)$
(P2) $\textit{select}(a)\Rightarrow\exists c\in\mathcal{C}\ \textit{action}(c,a)$
(P3) $\exists v\in\mathbf{var}(c),\mathcal{D}(v)=\emptyset\Rightarrow\textit{false}(c,\mathcal{D})$

Théorème 2

La trace générique GenTra4CP, restreinte aux événements décrits aux figures 4 et 5 sauf back to *et* solved*, est une sous-trace de GenTra4CP, dérivée de la trace spécifiée pour PaLM (figures 6 et 7).*

La preuve est en annexe 1.

6 Trace générique et spécification de contraintes

Comme on l'a vu, une trace générique définie par uns SO est susceptible de constituer une forme de spécification d'une sémantique opérationnelle pour un ensemble de processus capables de produire une trace conforme. Cette approche peut être appliquée aux contraintes. La question est alors de pouvoir spécifier une trace générique qui couvrirait l'ensemble des contraintes que l'on souhaite décrire, c'est à dire qui couvrirait aussi bien différents types de contraintes (simples, globales, ...) , différents domaines (FD, intervalle, ...), différentes formes de solveurs (CSP, SAT ou règles, comme CHR), différents niveaux (algorithmes, modules, modélisation) que différents aspects (langages, interactions, interfaces, ...).

On se limite ici au cas CSP. Chaque contrainte a une sémantique déclarative définie par la relation qu'elle représente sur son domaine. La trace générique peut

$$\text{new variable} \quad \texttt{idem GenTra4CP}$$

$$\text{new constraint} \quad \texttt{idem GenTra4CP}$$

$$\text{post} \quad \texttt{idem GenTra4CP}$$

$$\text{choice point} \quad \texttt{idem GenTra4CP}$$

$$\text{solution} \ \frac{\mathcal{N},\ \ \Sigma,\ \ \mathbb{S}}{\mathcal{N} \cup \{n\},\ \ \Sigma \cup \{(n, \mathbb{S})\},\ \ n} \left\{ \begin{array}{l} sol(\mathbb{S}), \\ n \notin \mathcal{N} \end{array} \right\}$$

$$\text{failure} \ \frac{\mathcal{N},\ \ \Sigma}{\mathcal{N} \cup \{n\},\ \ \Sigma \cup \{(n, \mathbb{S})\},\ \ n} \left\{ \begin{array}{l} n \notin \mathcal{N}, \\ R \neq \emptyset \end{array} \right\}$$

$$\text{remove} \quad \texttt{idem GenTra4CP}$$

$$\text{restore} \ \frac{\mathcal{D}(v),\ \ Q_t,\ \ \mathcal{E}}{\mathcal{D}(v) \cup R_v,\ \ Q_t \cup \bar{a},\ \ \mathcal{E} - E}$$

$$\left\{ \begin{array}{l} v \in \mathcal{V}, \\ R_v \subseteq \{d \in \mathbb{D} | \mathcal{E}(v,d) \cap \sigma \neq \emptyset\}, \\ E = \{\mathcal{E}(v,d) | d \in R_v\}, \\ \bar{a} \text{ actions de restauration de } \mathcal{D}(v) \end{array} \right\}$$

FIGURE 6 – SO de la trace de PaLM [7] (contrôle)

alors constituer une description des effets possibles de chaque contrainte prise isolément et dans un ensemble, indépendamment de l'algorithme particulier qui l'implante. Dans ce sens une telle sémantique joue le rôle d'une description à minima de ce que l'on doit pouvoir observer du comportement d'un ensemble de contraintes. Elle peut alors être utilisée pour définir a priori toutes sortes d'interfaces, en particulier pour la modélisation de problèmes.

En pratique, et c'est ce qui a été fait pour GenTra4CP, on va partir de la définition d'une trace effective dont la signification pourra être donnée par une fonction de reconstruction. On la complétera avec une SO éventuellement partielle, spécifiant un ensemble de traces acceptables à partir desquels le comportement d'un ensemble de contraintes pourra être simulé et et qui pourra servir à la mise au point d'applications.

Nous illustrons cette démarche de sémantique générique, avec un exemple simple, en montrant deux traces génériques du même problème obtenues avec GNU-Prolog et PaLM. Les deux solveurs ont été instrumentés pour produire la trace générique pour CSP(FD) et celle-ci peut être "comprise" conformément à la reconstruction décrite à la figure 8.

Les deux traces (figure 9) correspondent à la résolution de (syntaxe GNU-Prolog) `fd_element_var(I,[2,5,7],A), (A#=I ; A#=2)` qui admet une seule solution[5]. La sémantique dé-

5. En fait PaLM produit des "raccourcis" tels qu'une séquence de reduce suspend schedule awake ne comporte que

$$\text{reduce} \ \frac{\mathcal{D}(v),\ \ Q_t,\ \ \mathcal{E}}{\left\{ \begin{array}{l} \mathcal{D}(v) - \Delta_v^{c_a},\ \ Q_t \cup \{\bar{a}\}, \\ \mathcal{E} \cup \{(v,d,C) \mid d \in \Delta_v^{c_a}\} \end{array} \right\}}$$

$$\left\{ \begin{array}{l} v \in \mathbf{var}(c), \\ A = \{(c,a)\}, \\ \Delta_v^{c_a} \neq \emptyset \text{ set of inconsistent values for } v, \\ R = \emptyset, \\ C \subseteq \sigma \text{ explains the removal of } \Delta_v^{c_a} \text{ from } \mathcal{D}(v), \\ \text{The reduction generates } \bar{a} \end{array} \right\}$$

$$\text{suspend} \ \frac{< A,\ \ S_c >}{< \emptyset,\ \ S_c \cup \{c\} >} \ \{A = \{(c,a)\}\}$$

$$\text{reject} \ \frac{< A,\ \ R >}{< \emptyset,\ \ R \cup \{c\} >} \left\{ \begin{array}{l} A = \{(c,a)\}, \\ v \in \mathbf{var}(c), \\ \mathcal{D}(v) = \emptyset \end{array} \right\}$$

$$\text{awake} \ \frac{S_c,\ \ A}{S_c - \{c\},\ \ \{(c,a)\}} \left\{ \begin{array}{l} A = \emptyset, \\ c \in S_c, \\ R = \emptyset, \\ a \in Q_h \cup \{\perp\}, \\ dependence(c,a) \end{array} \right\}$$

$$\text{schedule} \ \frac{Q_h,\ \ Q_t}{\{a\},\ \ Q_t - \{a\}} \left\{ \begin{array}{l} select(a), \\ A = \emptyset, \\ a \in Q_t, \\ R = \emptyset \\ S_c \neq \emptyset \end{array} \right\}$$

FIGURE 7 – SO de la trace de PaLM [7] (propagation)

clarative de cette contrainte (toutes les variables, y compris celles de la liste, portent sur des domaines finis) peut se formuler : `fd_element_var(I, L, V)` (`L` liste) contraint `V` à être égale au `I`ème élément de `L`. En d'autres termes tous les triplets tels que $i \in$ `I`, $u \in$ `L`(i), $v \in$ `V` et $u = v$ sont valides. L'intervalle `[`*a*-*b*`]` dénote *de a à b* et `[`*a*,*b*`]`, *a et b*.

On peut observer[6] que les traces produites présentent des différences (entre autres) :

- le domaine de `I` qui n'est pas le même pour GNU (`[1-3]`) et pour PaLM (`[0-2]`) ;
- l'ordre et l'importance des retraits de valeurs et l'ordre du choix des variables dont on modifie le domaine ;
- les arbres de choix qui sont différents ;
- une variable spécifique qui apparaît dans la trace de PaLM (`v-1`).

Certaines variations ne sont pas significatives sur le

reduce awake. Un tel raccourci n'a pas de sémantique dans GenTra4CP (qui pourrait cependant être adaptée) et indique simplement que la sémantique de PaLM décrite dans [7] n'était pas tout à fait conforme à la trace générique.

6. GenTra4CP produit les traces dans un format XML, lisible mais trop verbeux. La présentation utilisée ici est une traduction plus concise.

$$\text{new variable}\ \frac{[\text{new variable},\quad v,\quad D_{v,i}]}{\left\{\begin{array}{l} <\mathcal{V},\ \ \mathcal{D}>\rightarrow \\ <\mathcal{V}\cup\{v\},\ \ \mathcal{D}\cup\{(v,D_{v,i})\}> \end{array}\right\}}\ \{\}$$

$$\text{new constraint}\ \frac{[\text{new constraint},\quad c]}{\left\{\begin{array}{l} <\mathcal{C}>\rightarrow \\ <\mathcal{C}\cup\{c\}> \end{array}\right\}}\ \{\}$$

$$\text{post}\ \frac{[\text{post},\quad c]}{\left\{\begin{array}{l} <A>\rightarrow \\ <A\cup\{(c,\bot)\}> \end{array}\right\}}\ \{\}$$

$$\text{choice point}\ \frac{[\text{choice point},\quad n]}{\left\{\begin{array}{l} <\mathcal{N},\ \ \Sigma,\ \ \mathbb{S}>\rightarrow \\ <\mathcal{N}\cup\{n\},\ \ \Sigma\cup\{(n,\mathbb{S})\},\ \ n> \end{array}\right\}}\ \{\}$$

$$\text{reduce}\ \frac{[\text{reduce},\quad c,\quad v,\quad \bar{a},\quad \Delta_v^c,\quad a]}{\left\{\begin{array}{l} <\mathcal{D}(v),\ \ S_e,\ \ A>\rightarrow \\ <\mathcal{D}(v)-\Delta_v^c,\ \ S_e\cup\bar{a},\ \ A-(c,a)> \end{array}\right\}}\ \{\}$$

$$\text{suspend}\ \frac{[\text{suspend},\quad c,\quad a]}{\left\{\begin{array}{l} <A,\ \ S_c>\rightarrow \\ <A-\{(c,a)\},\ \ S_c\cup\{c\}> \end{array}\right\}}\ \{\}$$

$$\text{awake}\ \frac{[\text{awake},\quad c,\quad a]}{\left\{\begin{array}{l} <A,\ \ S_c>\rightarrow \\ <A\cup\{(c,a)\},\ \ S_c-\{c\}> \end{array}\right\}}\ \{\}$$

FIGURE 8 – Quelques règles de la SO de GenTra4CP (reconstruction)

```
 1[0]choice point node(0)
 2[1]newVariable v1 [0-mx]
 3[1]newVariable v2 [0-mx]
 4[1]newConstraint c1
fd_element([v1,[2,5,7],v2])
 5[1]post c1
 6[1]reduce c1 v1 [0,4-mx]
 7[1]reduce c1 v2
   [0-1,3-4,6,8-mx]
 8[1]suspend c1
 9[1]choice point node(1)
10[2]newConstraint c4
    x_eq_y([v2,v1])
11[2]post c4
12[2]reduce c4 v2 [5,7]
13[2]reduce c4 v1 [1,3]
14[2]suspend c4
15[2]schedule v2 dom
16[2]awake c1
17[2]reject c1
18[2]failure node(2)
...
```

```
 0[0]newVariable v0 I [0-mx]
 1[0]newVariable v1 A [0-mx]
 2[0]newConstraint c0
     element(I,[2,5,7],A)
 3[0]post c0
 4[0]suspend c0
 5[0]awake c0
 6[0]reduce c0 v0 [3-mx] max
 7[0]reduce c0 v1 [0,1] min
 8[0]reduce c0 v1 [8-mx] max
 9[0]suspend c0
10[0]newConstraint c1 eq(I,A)
11[0]post c1
12[0]suspend c1
13[0]awake c0 (v0,max)
14[0]reduce c0 v1 [2-7] empty
15[0]reject c0 empty
16[0]failure
17[0]newVariable v-1 I [0-1]
18[0]reduce c2 v-1 [0,1] empty
...
```

FIGURE 9 – Traces effectives partielles de GNU-Prolog et PaLM pour l'exemple donné. Le deuxième attribut indique la profondeur de l'arbre de choix (paramètre δ ignoré ici)

plan sémantique (renommages, extra-variables, ...), et les traces restent compréhensibles et donc utilisables avec le même modèle générique. Certaines variations (domaine de `I`, attributs particuliers, ...) nécessiteraient probablement d'être fixées, ou au moins précisées pour arriver à une forme de standard.

7 Discussion

Les sémantiques des traces présentées ici (MI) se situent dans le cadre de la "sémantique observable" de Lucas [9] ou la sémantique des traces partielles de Cousot [2]. Les paramètres des états virtuels sont, comme l'exprime Lucas, des "objets syntaxiques utilisés pour représenter le déroulement de mécanismes opérationnels". Les traces sont des représentations abstraites qui permettent de ne tenir compte que des aspects que l'on souhaite ici communs à un ensemble de processus dotés de leur propre sémantique. Le choix d'articuler différentes formes de traces correspond au besoin de rapprocher différentes approches pragmatiques : spécification formelle de sémantiques plus ou moins abstraites et manipulations empiriques de traces dans des systèmes à base de traces, comme dans [10]. Nous avons établi ici une méthode particulière pour prouver la conformité de la trace d'un processus à une trace générique, mais cette approche permet également d'établir des liens formels avec la théorie des traces [4].

Nous avons montré ici que la définition de la trace GenTra4CP relève bien d'un tel cadre théorique et nous avons caractérisé par des opérations relativement simples (sous-traces paramétriques et similarité) les liens formels existant entre les processus observés et la trace générique. Cette analyse nous a révélé des insuffisances dans la définition même de GenTra4CP comme l'absence de test formel de conformité des traces des solveurs. Simonis & al [11] définissent une trace "générique" pour une famille d'outils bien définie. Ils notent que la trace générique GenTra4CP contient trop de détails pour une spécification trop complexe. Dans le type d'applications concernées, les besoins d'information sont en effet limités et il peut paraître plus facile de réaliser une instrumentation ad-hoc d'un solveur, plutôt que faire l'effort d'implanter a priori une trace générique avec tous ses détails. Mais cela est probablement faux si l'on souhaite réaliser un véritable interface générique entre plusieurs sortes de solveurs et de nombreuses applications potentielles.

Cette étude montre que c'est plutôt la trop grande laxité du modèle adopté qui pourrait être en cause, bien des paramètres essentiels à la description sémantique restant optionnels, ce qui lui fait perdre une partie de son utilité. Une approche plus exigeante mais plus utile serait sans doute d'avoir plus d'événements de trace (éventuellement plus spécifiques) portant sur des paramètres communs non optionnels.

Par ailleurs, comme il a été observé dans la section 4, il appartient aux développeurs de chaque processus ou de chaque outil, de produire une (sous-)trace

générique ou d'adapter des outils, développés sur la base de cette trace, au processus particulier. L'investissement à réaliser se mesure par l'écart formel qu'il y a entre la trace propre du processus et la trace générique (formellement la dérivation, cf. figure 3). Il peut paraître plus facile d'implanter une trace ad-hoc plutôt que d'adapter un traceur ou un outil à une trace prédéfinie. Langevine et Ducassé ont bien montré [8] qu'une approche générique pouvait avoir plus d'avantages que d'inconvénients, mais elle s'apparente à un effort de standardisation.

Un tel effort ne peut résulter que de l'action d'une communauté importante, et non d'un petit groupe comme dans le cas de GenTra4CP. Le projet de standard [5] se concentre principalement sur la définition d'un interface Java englobant en particulier les types principaux de variables, contraintes unaires, binaires et globales et certaines capacités à trouver des solutions. La question sémantique ne peut cependant être évitée. Si la sémantique déclarative de contraintes simples ne pose guère de problème de spécification, il n'en est pas de même de la sémantique opérationnelle, sa précision éventuelle dépendant des usages que l'on souhaite en faire. L'approche présentée ici, fondée sur une sémantique de trace générique, peut constituer une voie dans la mesure où elle offre un cadre pour spécifier des effets directs et collatéraux, liés par exemple aux interactions de contraintes, indépendamment d'implantations particulières.

8 Conclusion

GenTra4CP a constitué une démarche innovante en utilisant une sémantique de traces partielle pour répondre à un double objectif de spécification de résolution de CSP (domaines finis) et de portabilité d'outils d'analyses. Un tel effort s'apparente à une démarche de standardisation, mais difficile à utiliser, en raison de ses limites (limites du groupe qui l'a réalisée, insuffisances techniques et limites du domaine des contraintes traitées).

Nous avons introduit un cadre formel simple basé sur la théorie des traces et l'interprétation abstraite pour expliquer la méthode de construction de traces génériques et montrer l'intérêt potentiel de cette approche pour rendre compte de sémantiques partielles de contraintes.

Obtenir une sémantique sous-forme de trace générique pour un ensemble significatif de contraintes simples ou globales représente certainement un effort considérable. Il semble cependant qu'une telle approche permettrait non seulement d'assurer une certaine portabilité d'applications potentielles, mais aussi d'aller vers des descriptions de traitements entre différentes approches de systèmes de gestion de connaissances mêlant contraintes et règles de déduction.

Références

[1] A. Agoun, T. Baudel, P. Deransart, M. Ducassé, J.-D. Fekete, and N. Jussien. Tools for Dynamic Analysis of Constraints Resolution (OADymPPaC). Technical report, Inria Rocquencourt and École des Mines de Nantes and INSA de Rennes and Université d'Orléans and Cosytec and ILOG, May 2004. Projet RNTL. `http ://contraintes.inria.fr/OADymPPaC`.

[2] P. Cousot and R. Cousot. Systematic design of program transformation frameworks by abstract interpretation. In *Proc. of POPL 2002*, pages 178–190, 2002.

[3] P. Deransart. Conception de Traces et Applications (vers une méta-théorie des traces), June 2011. Working document `http ://hal.inria.fr/`.

[4] V. Diekert and G. Rozenberg. *The Book of Traces*. World Scientific Publishing, Singapoore, 1995.

[5] Jacob Feldman. JSR-331, Java Constraint Programming API. Tr, Java Community Process, Cork Constraint Computation Centre, 2011.

[6] Narendra Jussien and Vincent Barichard. The PaLM system : explanation-based constraint programming. In *Proceedings of TRICS : Techniques foR Implementing Constraint programming Systems, a post-conference workshop of CP 2000*, pages 118–133, Singapore, September 2000.

[7] Ludovic Langevine, Pierre Deransart, and Mireille Ducassé. A generic trace schema for the portability of cp(fd) debugging tools. In K.R. Apt, F. Fages, F. Rossi, P. Szeredi, and Jozsef Vancza, editors, *Recent Advances in Constraints*, number 3010 in LNAI. Springer Verlag, May 2004.

[8] Ludovic Langevine and Mireille Ducassé. Design and implementation of a tracer driver : Easy and efficient dynamic analyses of constraint logic programs. *Theory and Practice of Logic Programming, Cambridge University Press*, 8(5-6), Sep-Nov 2008.

[9] Salvador Lucas. Observable Semantics and Dynamic Analysis of Computational Processes. Technical Report LIX/RR/00/02, Laboratoire d'Informatique LIX, 2000.

[10] Lotfi Sofiane Settouti. *Système à base de trace modélisées : Modèles et Langages pour l'exploitation de traces d'interactions*. PhD thesis, Université Claude Bernard - Lyon I, January 2011.

[11] Helmut Simonis, Paul Davern, Jacob Feldman, Deepak Mehta, Luis Quesada, and Mats Carlsson. A Generic Visualization Platform for CP. In Karen Petrie, editor, *Proceedings of the 16th International Conference on Principles and Practice of Constraint Programming*, St Andrews, Scotland, September 2010.

ANNEXE : Preuve du théoreme 2

Preuve 1

On montre qu'une sous-trace de PaLM (issue de la SO de PaLM) est simulable dans une sous-trace "PaLM" de GenTra4CP.

On considère la trace générique GenTra4CP, restreinte à tous ses événements décrits aux figures 4 et 5 sauf back to *et* solved. *La suppression de l'événement* back to *n'affecte pas la construction de l'arbre de recherche, mais seulement son parcours et celle de* solved *correspond à la suppression du paramètre E de l'état du solveur. De plus, on "oublie" le paramètre concernant les explications, ce paramètre n'étant pas formalisé de manière générique dans la sémantique de GenTra4CP.*

Selon la définition 2, la restriction au sous-ensemble d'événements considérés est bien une sous-trace de GenTra4CP.

La sous-trace de PaLM considérée ici consiste à simplement ignorer les explications. Cela n'apporte aucune restriction sur les ensembles de types d'action (définition 2).

Dans le format GenTra4CP et la sous-trace considérée, la partie contrôle $\mathbb{T}_g$ utilise 4 paramètres : $\mathcal{N}, \Sigma, \delta, \nu$, et la partie solveur $\mathbb{S}_g$ 8 paramètres : $\mathcal{V}, \mathcal{C}, \mathcal{D}, A, E, R, S_c, S_e$. Au total 12 paramètres.

Dans PaLM, la partie contrôle $\mathbb{T}_p$ utilise 5 paramètres : $\mathcal{N}, \Sigma, \delta, \nu, Q_t$, et la partie solveur $\mathbb{S}_p$ 9 paramètres : $\mathcal{V}, \mathcal{C}, \mathcal{D}, A, R, S_c, Q_h, Q_t, \mathcal{E}$. Au total 13 paramètres (Q_t est commun). On note les différences suivantes :

- *E, le sous-ensemble du "store" contenant les contraintes universellement satisfaites disparaît dans PaLM qui ne réalise pas de test de satisfaction ("entailment").*
- *l'ensemble S_e des événements à traiter du solveur PaLM est une queue ($S_e = Q_h \cup Q_t$) dont la tête Q_h contient l'événement courant pris en compte.*
- *L'état du solveur de PaLM contient un paramètre additionnel $\mathcal{E}$, fonction d'explication qui sert à mémoriser les "explications". E est une fonction partielle : $\mathcal{E} : \mathcal{V} \times \mathbb{D} \longrightarrow \mathcal{P}(\sigma)$ [7] qui affecte à chaque retrait de valeur (v, d) ($v \in \mathcal{V}, d \in \mathcal{D}(v)$) un ensemble de contraintes non relaxées qui "explique" le retrait. Cette fonction partielle est mise à jour au cours des événements* reduce *et* restore.
- *A, dans PaLM, a au plus un élément.*

On montre alors que la sous-trace ainsi restreinte de PaLM est simulable dans la sous-trace "PaLM" GenTra4CP.

On utilise le théorème 1. On définit l'application d entre les états modifiés $\mathbb{T}_p \times \mathbb{S}_p$ et $\mathbb{T}_g \times \mathbb{S}_g$, soit : (on omet δ qui se déduit directement de $\mathcal{N}$)

$\mathcal{N}, \Sigma, \nu, \mathcal{V}, \mathcal{C}, \mathcal{D}, A, R, S_c, Q_h, Q_t$ *et*

$\mathcal{N}, \Sigma, \nu, \mathcal{V}, \mathcal{C}, \mathcal{D}, A, R, S_c, S_e$

de la manière suivante : identité pour les 9 premiers paramètres de PaLM $\mathcal{N}, \Sigma, \nu, \mathcal{V}, \mathcal{C}, \mathcal{D}, A, R, S_c$, puis $Q_h \cup Q_t = S_e$.

Les types d'actions ont les mêmes noms et leur ensemble est restreint à ceux indiqués sur les figures 6 et 7.

Les états initiaux $\mathbb{T}_{0,p} \times \mathbb{S}_{0,p}$ et $\mathbb{T}_{0,g} \times \mathbb{S}_{0,g}$ à considérer sont :

$\{rc_p\}, (rc_p, \mathbb{S}_{0,p}), rc_p, \emptyset_p, \emptyset_p, \emptyset_p, \emptyset_p, \emptyset_p, \emptyset_p, \emptyset_p, \emptyset_p$ *et*

$\{rc_g\}, (rc_g, \mathbb{S}_{0,p}), rc_g, \emptyset_g, \emptyset_g, \emptyset_g, \emptyset_g, \emptyset_g, \emptyset_g, \emptyset_g, \emptyset_g$

new variable, new constraint, post, choice point *et* remove *sont en correspondance puisque les règles sont identiques et les paramètres concernés sont les mêmes.*

Pour solution *et* failure, *il en est de même à condition d'utiliser les propriétés (G1) et (G2).*

Le cas de restore *est plus complexe. Néanmoins, si on ignore les explications et si on prend pour Δ_v, R_v ($\Delta_v = R_v$) pour la même variable v, les conditions associées à l'événement de Gentra4CP se déduisent des propriétés des explications (restitution de valeurs retirées donc inexistante dans l'état courant du domaine de v). Par contre il faut alors expliciter dans la transition GenTra4CP la mise à jour de S_e.*

reduce. *A $\Delta_v^{c_a}$ correspond Δ_v^c (ensemble des valaurs inconsistantes) de la trace générique. Par (G3) les propriétés $R = \emptyset$ se correspondent. Enfin comme $d(Q_h \cup Q_t) = S_e$, $d(Q_h \cup Q_t \cup \bar{a}) = S_e \cup \bar{a}$.*

suspend. *Dans les états initiaux en correspondance $(c, a) \in A$, et $A' = A - \{(c, a)\}$ dans les états finaux.*

reject. *Utilise (P3) pour les états initiaux, et les états finaux se correspondent.*

awake. *Utilise (P1) et (G4).*

schedule. *Utilise (P2) et (G5). S_c et S_e sont invariants dans la trace générique.*

7. $\mathcal{P}(\sigma)$: ensemble des parties du store σ (instance des contraintes qui sont dans A, S_c et R pour PaLM).