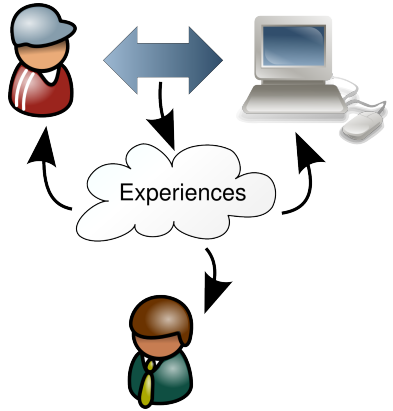
Systèmes à base de traces modélisées

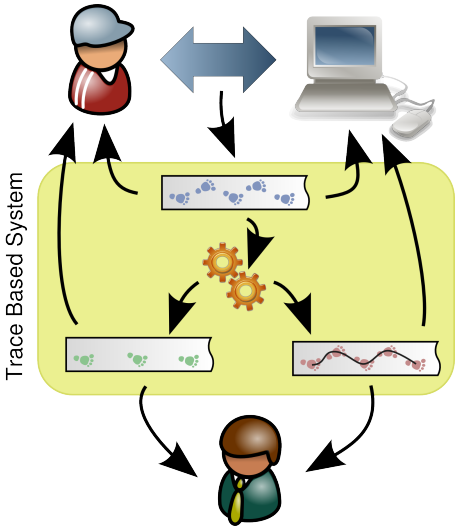
Ingéniere de connaissances d'expérience tracée

Auteur: Pierre-Antoine Champin
Adresse: M2 Intelligence Artificielle — Lyon 1
Date: 2012

1 Introduction



Exploitation de l'expérience tracée



Système à base de traces

Modélisation

Transformation

Collecte

Mise en œuvre

kTBS: A kernel for Trace Based Systems

http://liris.cnrs.fr/sbt-dev/ktbs/

- représente ses données en RDF
  (flexible, extensible)
- communique via HTTP
  (REST, indépendant du langage)

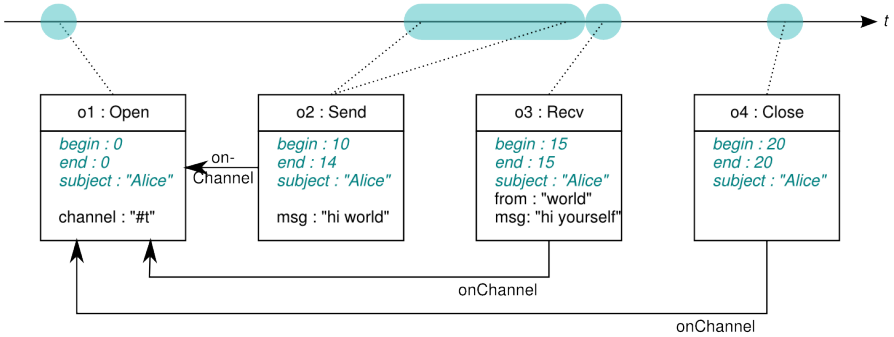
Plan du chapitre

2 Modélisation

Une trace est constituée d'**obsels** (éléments observés).

Chaque obsel est muni:

- d'un type
- d'une date de début et d'une date de fin
- d'un sujet
- d'un ensemble d'attributs
- de relations avec d'autres obsels



Exemple de trace

Modèle de trace

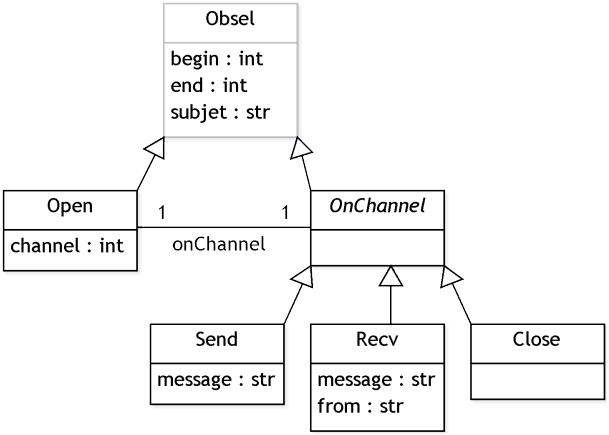
Pour être exploitable, les traces doivent être *explicitement modélisées*.

Le **modèle de trace** définit:

- les types d'obsels,
- les attributs associés à chaque type,
- les relations possibles entre les types.

(modélisation objet classique)

Exemple de modèle de traces



Modèle de trace (suite)

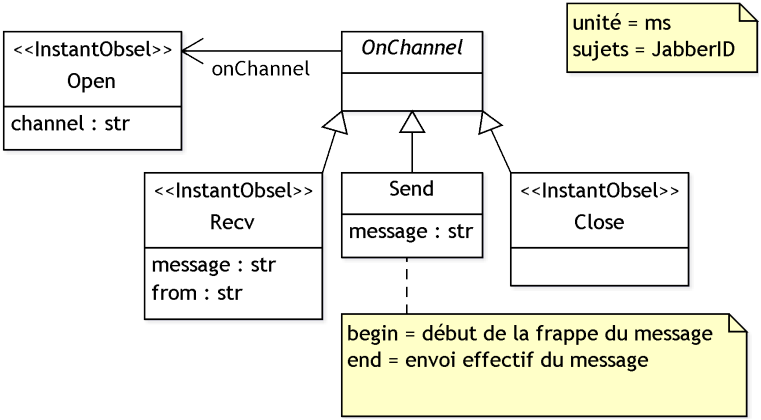
Le modèle de trace peut également définir:

- les contraintes temporelles sur les types d'obsels (durée min ou max),
- les sujets possibles,
- le mode de représentation du temps (cf. dernier chapitre).

(spécifique aux SBT)

Exemple de modèle de traces (suite)

- la granularité temporelle est de 1ms,
- les sujets doivent être des identifiants Jabber,
- les obsels de type `Open`, `Recv` et `Close` doivent être instantanés (`begin = end`),
- la durée d'un obsel `Send` correspond au délais entre le moment ou l'utilisateur commence à taper et le moment ou le message est envoyé.



Expression en UML des contraintes spécifiques SBT

Langage de définition de modèle

Ne pas confondre :

- les langages de **schéma**,
  définissant des *contraintes syntaxiques*
  (e.g. UML, Schéma relationnel, XML-schema)
- les langages de modélisation **logiques**,
  définissant des *inférences sémantiques*
  (*e.g.* ontologie, RDF-S, OWL)

Langage de définition de modèle (suite)

Les deux types de langage sont intéressants pour les modèle de trace, mais pour des raisons différentes :

- langages de schémas utiles pour la collecte
- langages de modélisation logiques utiles pour les transformations

kTBS: retour d'expérience

- langage *inspiré* de RDF-Schema, mais orienté *schéma*
  - exprime des contraintes et non des inférences
- exprimé sous forme d'un graphe
  - moins lisible pour l'humain qu'un diagramme UML
  - simple question de présentation
- ⚠ nommage global des attributs (URI)
  - préfixer les attributs par le nom du type,
    e.g. `send-message` et `recv-message`

Choix de modélisation

L'élaboration d'un modèle de trace est une activité de conception, impliquant des *choix*, guidés par :

- des considérations épistémique

  (de quoi a-t-on besoin pour les usages envisagés ?)
- des considérations techniques

  (quelles informations peut-on obtenir ?)
- des considérations pratiques

  (quelle quantité de données peut-on stocker ?)

Exemple : modèle de l'activité de *chat*

Traitement des traces non-conformes

Lorsqu'il est confronté à une trace non-conforme à son modèle,

- le SBT peut la refuser purement et simplement:
  - plus sûr lorsque les transformations donnent lieu à des actions irrémédiables,
  - mais l'information est perdue (dans le cas de la collecte) ;

- ou il peut accepter cette trace en la marquant comme « non-conforme », en vue d'une intervention d'un utilisateur :
  - soit pour « réparer » la trace, en utilisant les informations conservées,
  - soit pour adapter *le modèle*, mis en défaut par l'expérience réelle ;
  - c'est le choix fait par le kTBS.

3 Transformation

Une trace transformée est définie par:

- une ou plusieurs trace(s) source(s)
- une méthode de transformation
- des paramètres éventuels pour cette méthode

Méthodes de transformation

Filtre temporel

Ne conserve que les obsels situés dans un intervalle temporel donné.

- une seule source
- modèle de sortie = modèle d'entrée
- paramètres :
  - `begin` minimum
  - `end` maximum

NB: selon le modèle, même ce type de transformation peut produire une trace non-conforme à partir d'une trace conforme (par exemple : un X est toujours précédé d'un Y).

Filtre structurel (Sélection)

Ne conserve que les obsels vérifiant certaines contraintes.

- une seule source
- modèle de sortie = modèle d'entrée
- paramètres possibles :
  - types d'obsels
  - attribut + valeur ou plage de valeurs
  - (in)égalités entre attributs
  - ...

Où s'arrête-t-on dans l'expressivité ?

Fusion

Fait l'union des obsels de plusieurs traces.

- plusieurs sources
- modèle de sortie : doit englober les modèles des sources
- pas de paramètres

NB: même si toutes les sources sont du même type et conformes, la trace fusionnée peut être non-conforme (par exemple : deux X ne peuvent pas se chevaucher temporellement).

Exemples :

- fusion de la trace d'aujourd'hui et celle d'hier
- fusion de la trace de *chat* et de la trace de *mail*

Enrichissement

Conserve tous les obsels de la source, et y ajoute des informations (attributs, relations, nouveaux obsels).

- une seule source
- modèle de sortie : doit englober le modèle de la source
- paramètres : règle(s) d'enrichissement

Nécessite un langage pour décrire les enrichissements.

Exemples :

- calculs sur les valeurs des attributs
- comptage d'obsels d'un certains types
- traitements plus complexes (inférence, TAL...)

Ré-écriture de motifs

Crée de nouveaux obsels sur la base d'un motif recherché dans la source.

- une seule source
- modèle de sortie : dépend de la ré-écriture
- paramètres :
  - motif recherché
  - motif produit

Exemples :

- observés `Monologue/Dialogue` à partir de la trace du *chat*

Composites

Composer plusieurs méthodes (séquentiellement ou parallèlement).

- une seule source
- modèle de sortie : dépend des méthodes composées
- paramètres : dépendent des méthodes composées

Exemple :

- enrichissement pour déterminer l'humeur d'un message, suivi d'une ré-écriture en `Dispute/Neutre/Badinage`

Discussion

Les méthodes décrites ci-avant couvrent en grande partie les besoins des SBT.

On peut cependant évoquer la nécessité :

- de calcul d'indicateurs hors du SBT,
- d'avoir des méthodes plus spécifiques,
- de méthodes calculées directement à partir des modèles,
- d'utiliser les techniques de traitement de flux.

Indicateurs

- Un **indicateur** est une mesure globale effectuée sur une trace; exemples :
  - nombre total de messages échangés
  - temps moyen d'une session de *chat*
- Souvent considérés comme distincts des traces,
- ils peuvent (doivent?) être considérés comme des obsels particuliers, daté par les obsels ayant servi à leur calcul, exemples :
  - nombre total de messages échangés hier
  - temps moyen des sessions de *chat* du mois dernier

Méthodes dérivées

Une méthode dérivée est constituées :

- d'une méthode parente
- de valeurs pour certains paramètres de la méthode parente
- éventuellement de paramètres qui lui sont propres

Permet de faciliter l'utilisation de méthodes très génériques (enrichissement, ré-écriture, composite) en crystalisant une manière particulière de les utiliser.

Exemples de méthodes dérivées

Détection d'humeur

- parente: enrichissement
- paramètre fournis: algorithme de TAL
- paramètre requis: degré de certitude de la reconnaissance

Types de conversation (Dispute/Neutre/Badinage)

- parente: composite (séquentielle)
- paramètres fournis:
  - méthode "Détection d'humeur"
  - méthode ré-écriture de motif

Transformations automatiques

- Selon la manière dont sont exprimés les modèles, il peut être possible de générer automatiquement la transformation d'un modèle à l'autre.
- C'est par exemple l'objet de certains travaux sur l'alignement d'ontologies.
- Cela se ramène en général à une forme d'enrichissement (inférence) ou de ré-écriture de motif.

Traitement de flux

Le traitement de flux (*stream processing*) peut sembler proche des tranformations de traces. Cependant, les deux diffèrent par certains hypothèses sous-jacentes :

- les SBT ambitionnent de *conserver* les traces, alors que les systèmes de traitement de flux supposent que le flux produit plus de données qu'on ne peut en conserver ;
- une trace contient des informations temporelles, alors que le temps est une *méta-donnée* du flux.

En revanche, le traitement de flux peut s'avérer intéressant dans l'étape de collecte.

Calcul incrémental

- Chaque fois qu'une trace évolue, il peut devenir nécessaire de re-caculer ses traces transformées,
- et ainsi de suite avec leurs propres traces transformées.
- Se pose alors la question de calculer les transformations de manière *incrémentale*.

Évolution d'une trace

Quelconque:

des obsels sont ajoutés ou retirés de la trace.

Monotone (logiquement):

des obsels sont uniquement ajoutés à la trace.

Monotone temporellement:

des obsels sont uniquement ajoutés *après* les obsels existants (en considérant la date de *fin*).

NB: on pourrait définir la monotonie temporelle par rapport à la date de début, mais la date de fin semble plus pertinente (collecte).

Propagation de la monotonie

- Certaines méthodes de transformation propagent les propriétés de monotonie de la trace source à la trace transformée, *e.g.* le filtrage temporel.
- D'autres méthodes rompent évidemment la monotonie, par exemple :
  - sélection du dernier obsel `Open`,
  - sélection des obsels `Open` sans `Close` attaché.
- La fusion peut propager la monotonie temporelle, mais au pris d'un « retard » dans la mise à jour de la trace transformée (au cas ou les traces sources sont désynchronisées).

kTBS: retour d'expérience

- Méthodes implémentées :
  - filtre temporel
  - fusion (propageant la monotonie)
  - SPARQL CONSTRUCT (sans hypothèse de propagation)
  - programme externe (sans hypothèse de propagation)
  - composite
  - dérivée
- Amélioration future : exécution incrémentale certaines méthodes SPARQL et externes.

4 Cycle de vie des traces

Traces premières/transformées

Trace première :

trace générée depuis l'extérieur du SBT; la *première* d'une lignée de traces obtenues par transformations successives.

Trace transformée :

trace générée par le SBT à partir *via* une transformation.

Lignée de traces :

graphe acyclique induit par la relation *source* sur un ensemble de traces.

Collecte

Ajout d'observé à une trace première (monotonie temporelle ou uniquement logique).

NB: On est donc parfois amené à re-définir un nouveau modèle (« modèle de collecte »), puise d'instancier le modèle initialement prévu par *transformation*.

→ Un modèle n'est donc jamais intrinsèquement un « modèle de trace première ».

Obsels duratifs et modèle de collecte

Il est souvent plus complexe de collecter des obsels ayant une durée (bufferisation) que des obsels instantanés.

Dans l'exemple du modèle de trace *Chat*, on pourrait imaginer de

- collecter chaque frappe clavier,
- re-construire les obsels `Send` par ré-écriture de motif.

Amendement

Modification arbitraire d'une trace première (monotone ou non).

- « réparation » d'une trace non conforme
- censure *a posteriori* d'une partie de la trace
- « transformation » manuelle (et irréversible)

Importance du *droit de regard* de l'utilisateur sur sa trace.

Émancipation

Changement de statut d'une trace transformée en trace première.

- Cette dernière cesse donc d'être mise à jour à partir de son/ses ancienne(s) source(s);
- on peut donc éventuellement supprimer la/le sources.
- Utile par exemple pour se débarasser de la trace de collecte lorsque cette dernière est terminée.

Scindement

Changement de statut d'une trace première en trace transformée (fusion), dont les sources sont des segments temporels de la trace initiale.

- Permet de créer des lignées différentes pour chaque segment.
- Permet éventuellement d'archiver les segments les plus anciens afin d'alléger le SBT.

5 Représentation du temps

> Qu'est-ce donc que le temps ? Si personne ne me le demande, je le sais. Mais si on me le demande et que je veuille l'expliquer, je ne le sais plus.
>
> — Augustin d'Hippone

Différentes représentations du temps doivent pourvoir co-exister au sein d'un SBT.

Origine d'une trace

Toute trace est munie d'une **origine**.

Cette origine peut être :

- absolue (date du calendrier)
- relative (identifiant opaque, mais partageable entre traces d'une même lignée)

L'origine est *arbitraire*; elle ne sert qu'à interpréter les bornes temporelles des obsels (et de la trace, voir ci-après).

Extension temporelle

Toute trace est munie d'une **extension temporelle** qui indique, par rapport son origine, la période effectivement tracée.

Lorsque la trace est en cours de collecte, son extension temporelle peut être *ouverte*.

Domaines temporels chronométriques

Toute trace est fondée sur un **domaine temporel**.

Un domaine temporel *chronométrique* est définie par rapport à une unité temporelle (ms, jour, année...).

NB: avec des origines *relatives*, il n'est pas toujours possible de convertir d'une unité à l'autre (exemple : mois et jours).

Domaine temporel généralisé

Un domaine temporel est un ensemble discret et totalement ordonné d'instants.

- bijection avec **N**,
- mais les *écarts* entre instants n'ont de sens que pour les domaines chronométriques.

Autres domaines temporels possibles :

- domaine séquentiel
- distance parcourue (ABSTRACT)
- ...

Pour résumer

- Modèle
  - domaine temporel prescrit
- Trace
  - domaine temporel effectif
  - origine
  - extension temporelle

Transformations liées au temps

- changement origine (sans changer la sémantique)
- décalage
- changement unité/domaine
- changement extension temporelle (filtre)

6 Méta-modèle

Pourquoi « méta »

Parce que notre modèle (des systèmes à base de traces) parle de modèles (de traces).

Base de traces