

INFO-F-302 : Logique informatique

Projet : Logique du Premier Ordre et Utilisation de l'Outil Paradox

L'objectif de ce projet est de modéliser un problème en logique du premier ordre, et de le résoudre avec l'outil *Paradox*. Cet outil cherche des modèles aux formules du premier ordre. Le problème considéré est une énigme dont la définition est attribuée à Einstein¹.

1 L'outil Paradox

L'outil *Paradox* est disponible à l'adresse suivante :

<http://www.cs.chalmers.se/~koen/folkung/>

Il est conseillé de l'utiliser en ligne via l'interface suivante :

<http://www.cs.miami.edu/~tptp/cgi-bin/SystemOnTPTP>

Paradox prend en entrée une formule de la logique du premier ordre au format TPTP² et, en fonction du but spécifié, tentera de trouver un modèle à la formule, ou un modèle qui ne satisfait pas la formule. L'algorithme est *incrémental* : *Paradox* cherche d'abord des modèles dont le domaine contient exactement un élément, puis deux, puis trois, etc... S'il existe un modèle fini à la formule, alors *Paradox* finira par le trouver (même s'il lui faut des milliards d'années). Par contre, si la formule n'a pas de modèle, alors il ne sera pas en général capable de conclure que la formule est insatisfiable et cherchera un modèle à l'infini. Dans certains cas particuliers pourtant, il pourra décider si la formule est insatisfiable. C'est par exemple le cas lorsqu'on spécifie que tout modèle de la formule a au plus k éléments par une formule du type :

$$\forall x, x = a_1 \vee x = a_2 \vee \dots \vee x = a_k$$

où a_1, \dots, a_k sont des symboles de constantes. C'est aussi le cas lorsqu'on n'utilise pas de symboles de fonction, car dans ce cas, le système peut calculer une taille maximale pour la taille des domaines. Dans le cadre du projet, on vous demande de **ne pas utiliser de symboles de fonction**, ainsi l'insatisfiabilité peut être décidée par *Paradox*.

La syntaxe complète du format TPTP est détaillée à l'adresse suivante dans la section *The Formulae Section* :

<http://www.cs.miami.edu/~tptp/TPTP/TR/TPTPTR.shtml>

Format TPTP Nous donnons ici les éléments de syntaxe utiles pour le projet. Une spécification au format TPTP est une suite de déclarations de formules de la forme :

fof(nom de la formule, type de la formule, formule).

nom de la formule est un nom que vous donnez à la formule, *type de la formule* est un type à choisir parmi *axiom, conjecture, theorem, lemma, hypothesis, etc....* Dans le cadre du projet, on indiquera toujours *axiom*. Finalement, *formule* est une formule dans la syntaxe TPTP. Commençons par un exemple :

¹pas de sources fiables

²<http://www.cs.miami.edu/~tptp/>

```
fof(au_moins_deux, axiom, ? [X,Y] : X != Y).
```

```
fof(tous_rouges, axiom, ! [X] : rouge(X)).
```

Cette spécification est composée de deux formules (n'oubliez pas le point après chaque déclaration). Ces deux formules sont prises en conjonction. La première formule s'appelle `au_moins_deux` est correspond à $\exists X \exists Y \cdot X \neq Y$. La deuxième formule correspond à $\forall X \cdot \text{rouge}(X)$ où rouge est interprété comme un prédicat unaire. Quelques remarques :

- les variables commencent toujours par une majuscule
- les prédicats et symboles de fonction commencent toujours par une minuscule
- les symboles de constantes commencent toujours par une minuscule
- les espaces n'ont pas d'importance (sauf pour la lisibilité)
- les commentaires sont toujours précédés de %

Voici une liste de symboles logiques du format TPTP :

quantification existentielle	? [liste variables] : formule
quantification universelle	! [liste variables] : formule
disjonction	
conjonction	&
négation	~ (tilde)
égalité	=
différence	!=
implication	=>
équivalence	<=>

Il ne faut pas spécifier le langage utilisé. Tout mot de la formule qui commence par une majuscule est considéré comme une variable. Tout mot qui commence par une minuscule est considéré comme une constante, un prédicat, ou une fonction. Selon l'utilisation qui en ai faite, l'outil décide automatiquement si c'est une constante, un prédicat, ou une fonction. Par exemple, dans la formule ci-dessus, `rouge(X)` est utilisé comme une valeur vrai/faux, c'est donc nécessairement un prédicat. **Remarque importante** : toute faute de frappe donnera potentiellement de nouveaux symboles de prédicat/fonction/constante. Par exemple, si vous taper `rouge(X)` et plus loin `rrouge(X)`, alors `rrouge` sera interprété comme un nouveau symbole de prédicat. Attention donc aux fautes de frappes.

Utilisation de l'outil Vous pouvez soit installer l'outil sur votre machine, ou l'utiliser en ligne. Dans les deux cas, je vous conseille d'écrire votre spécification dans un fichier `nomdefichier.tptp`. Si vous avez installé Paradox, il faut taper la commande :

```
paradox3 --model --tstp nomdefichier.tptp
```

Pour l'utilisation en ligne, rendez-vous à l'adresse mentionnée plus haut. Vous avez le choix entre rentrer la spécification dans la fenêtre prévue à cet effet, ou *uploader* un fichier.³ Normalement, vous devez laisser les options par défaut. En particulier, *output mode* doit être sur *system*. Dans la liste de systèmes (qui regroupe différents outils de résolution), sélectionner Paradox. Finalement, lancer l'évaluation en cliquant sur *RunSelectedSystems*.

Sortie générée par l'outil Par défaut, l'outil dit si la formule est satisfaisable. Dans le cas contraire, il se peut qu'il ne termine pas et continue à chercher des modèles. Parfois, il est capable de conclure que la formule n'est pas satisfaisable. Dans le cas où elle est satisfaisable, l'option `--model --tstp` (par défaut sur l'interface en ligne) permet d'afficher un modèle au format TPTP. Prenons la sortie du programme de l'interface en ligne pour l'exemple donné plus haut :

```
% START OF SYSTEM OUTPUT
Paradox, version 3.0, 2008-07-29.
```

³cette solution est conseillée car vous pouvez utiliser votre éditeur de texte préféré et en revenant à la page précédente dans votre navigateur web, vous pouvez resoumettre votre spécification modifiée sans avoir à refaire un copier/coller dans une fenêtre

```

+++ PROBLEM: /tmp/SystemOnTPTP1874/SOT_sMx9Yr.tptp:short
Reading '/tmp/SystemOnTPTP1874/SOT_sMx9Yr.tptp:short' ... OK
+++ SOLVING: /tmp/SystemOnTPTP1874/SOT_sMx9Yr.tptp:short
domain size 1
domain size 2
+++ BEGIN MODEL
SZS output start FiniteModel for /tmp/SystemOnTPTP1874/SOT_sMx9Yr.tptp:short
% domain size is 2
fof(domain, fi_domain,
  (![X] : (X = "1" | X = "2")))
).

fof(rouge, fi_predicates,
  ( ![X1] : rouge(X1) <=> $true
  )
).
SZS output end FiniteModel for /tmp/SystemOnTPTP1874/SOT_sMx9Yr.tptp:short
+++ END MODEL
+++ RESULT: Satisfiable
SZS status Satisfiable for /tmp/SystemOnTPTP1874/SOT_sMx9Yr.tptp:short

% END OF SYSTEM OUTPUT
% RESULT: SOT_sMx9Yr - Paradox---3.0 says Satisfiable - CPU = 0.0 WC = 0.0 Size = 2
% OUTPUT: SOT_sMx9Yr - Paradox---3.0 says FiniteModel - CPU = 0.0 WC = 0.0

```

La partie SOLVING donne la taille des modèles qui ont été essayés (nombre d'éléments du domaine). Dans ce cas, il a essayé de trouver des modèles de taille 1. Comme il n'existe pas de modèle de taille 1, il a testé des modèles de taille 2, et bien sûr il a trouvé. La réponse se trouve après le mot clef RESULT. La description du modèle se trouve entre le mot clef BEGIN MODEL et le mot clef END MODEL. Le domaine est toujours décrit par une disjonction :

```

fof(domain, fi_domain,
  (![X] : (X = "1" | X = "2"))).

```

Ceci indique qu'il y a deux éléments dans le domaine, appelés "1" et "2". Il ne faut pas les voir comme des entiers, mais plutôt comme des noms sous forme de chaîne de caractères.

Le prédicat est défini par :

```

fof(rouge, fi_predicates,
  ( ![X1] : rouge(X1) <=> $true)).

```

Ceci signifie que pour tout élément du domaine, le prédicat est vrai.

Avant de passer aux questions, prenons un exemple plus compliqué.

```

% l'élément special n'est pas rouge
fof(element_special, axiom, ~rouge(special)).

% il existe un élément différent de special
fof(element_diff_special, axiom, ? [X] : X != special).

% la fonction f retourne toujours l'element special sauf
% pour l'element special
fof(fonction_f, axiom, ! [X]: (X!=special <=> f(X)=special)).

```

Le modèle retourné par le système est :

```

fof(domain, fi_domain,

```

```

(![X] : (X = "1" | X = "2"))).

fof(f, fi_functors,
  ( f("1") = "2"
  & f("2") = "1")).

fof(rouge, fi_predicates,
  ( ![X1] : rouge(X1) <=> $false)).

fof(special, fi_functors,
  ( special = "1")).

```

La formule appelée **domain** définit le domaine, la formule appelée **f** définit la fonction **f**, la formule appelée **rouge** définit le prédicat **rouge** (qui est faux partout), et enfin la formule **special** définit le symbole de constante **special** comme étant l'élément "1".

Parenthésage des quantifications Attention, $!\ [X] : p(X) \ \& \ q(X)$ est équivalent à $(!\ [X] : p(X)) \ \& \ q(X)$. La variable **X** apparaît donc libre dans la deuxième partie. Ceci est valable avec la quantification existentielle et les autres connecteurs logiques (ou, implique, etc...). N'hésitez donc pas à parenthéser pour lever toute ambiguïté!

2 Prise en Main de Paradox

Question 1 *Ecrire une formule au format TPTP telle que tout modèle de cette formule possède au moins quatre éléments. Vérifier que Paradox retourne bien un modèle qui contient au moins quatre éléments.*

En théorie des langages, un *mot* est une suite de caractères sur un alphabet fini. Par exemple, sur l'alphabet $\{0, 1\}$, 0001011101100110 est un mot. Un mot sur $\{0, 1\}$ peut être représenté par une structure sur le langage $\{\text{pluspetit}, \text{zero}, \text{un}\}$, où **pluspetit** est un prédicat binaire représentant une relation d'ordre stricte et totale (toute paire d'éléments est comparable). **zero** et **un** sont des prédicats unaires qui représentent les éléments dont les lettres sont 0 et 1 respectivement. Par exemple, le mot 011 est représentée par la structure M qui a trois éléments $\{e_1, e_2, e_3\}$ et les relations suivantes :

$$\begin{aligned}
\text{pluspetit}^M &= \{(e_1, e_2), (e_1, e_3), (e_2, e_3)\} \\
\text{zero}^M &= \{e_1\} \\
\text{un}^M &= \{e_2, e_3\}
\end{aligned}$$

Question 2 *Ecrire une formule au format TPTP telle que tout modèle de cette formule est un mot de longueur au moins 4 sur l'alphabet $\{0, 1\}$. Donner un mot satisfaisant cette formule trouvé par Paradox. **Attention** : chaque élément du domaine n'est assigné qu'à une et une seule lettre.*

Une relation *successeur* pour l'ordre **pluspetit** fait correspondre chaque élément à son élément suivant dans l'ordre. Nous noterons **succ** le prédicat binaire représentant la fonction successeur. Par exemple, pour le mot 011, ce prédicat est interprété dans M par :

$$\text{succ}^M = \{(e_1, e_2), (e_2, e_3)\}$$

Question 3 *Ecrire une formule au format TPTP définissant le prédicat successeur, à partir de la relation d'ordre.*

3 L'énigme d'Einstein

Dans cette section, il faut résoudre une énigme qu'on attribue à Einstein. L'objectif est de formaliser cette énigme en logique du premier ordre, et d'utiliser Paradox pour trouver un modèle.

Ce modèle donnera la réponse à l'énigme.

Cinq maisons, de cinq couleurs différentes appartiennent à cinq propriétaires différents. Elles sont alignées le long d'une rue. Les propriétaires ont cinq nationalités différentes. Chaque propriétaire possède un animal domestique, fume un certain type de cigares, et boit un certain type de boissons. Ils boivent tous des boissons différentes, ont tous des animaux domestiques différents, et fument tous des cigares différents. Les données sont résumées dans la liste ci-dessous :

- **nationalités** : anglais, suédois, danois, norvégien, allemand ;
- **couleurs** : rouge, vert, bleu, jaune, blanc ;
- **boisson** : eau, thé, bière, café, lait ;
- **cigares** : Pall Mall, Dunhill, Blue Master, Prince, Blend ;
- **animaux** : chien, chat, poisson, cheval, oiseaux.

A partir des indices suivants, il faut trouver **qui possède le poisson** :

1. L'Anglais vit dans une maison rouge.
2. Le Suédois a un chien comme animal domestique.
3. Le Danois boit du thé.
4. La maison verte est juste à gauche de la maison blanche⁴.
5. Le propriétaire de la maison verte boit du café.
6. La personne qui fume des Pall Mall a un oiseau.
7. Le propriétaire de la maison jaune fume des Dunhill.
8. La personne qui vit dans la maison du centre boit du lait.
9. Le Norvégien habite la première maison.
10. L'homme qui fume les Blend vit à côté de celui qui a un chat.
11. L'homme qui a un cheval est le voisin de celui qui fume des Dunhill.
12. Le propriétaire qui fume des Blue Master boit de la bière.
13. L'Allemand fume des Prince.
14. Le Norvégien vit juste à côté de la maison bleue.
15. L'homme qui fume des Blend a un voisin qui boit de l'eau.

Question 4 *Si vous deviez tester toutes les possibilités, combien y en aurait-il ? Justifier.* Attention pour cette question l'ordre des maisons a de l'importance.

Question 5 *Ecrire un ensemble de formules du premier ordre au format TPTP sans symbole de fonction dans un fichier `einstein.tptp` qui modélise le problème posé par Einstein. Exécuter *Paradox* pour qu'il trouve un modèle et donner la réponse à l'énigme.* Le but principal de cette question est d'écrire une formalisation correcte du problème en logique du premier ordre. Le modèle trouvé doit vous indiquer exactement, pour chaque propriétaire, la position de sa maison par rapport aux autres, ce qu'il fume, boit, la couleur de sa maison, sa nationalité et son animal domestique. Vous êtes libres d'utiliser le langage de prédicats que vous voulez, mais choisissez-les de telle sorte que votre spécification soit lisible. Commentez dans le fichier toutes les formules que vous écrivez en donnant leur signification intuitive. Par exemple, si vous écrivez une formule `fof(f, axiom, ? [X,Y] : X != Y)`, ajoutez au-dessus le commentaire "il existe deux éléments dans la structure". La clarté et la lisibilité de votre fichier sera prise en compte dans l'évaluation. Il est conseillé de représenter les modèles du problème comme des structures à cinq éléments ordonnés, chaque élément représentant une maison. Les autres informations (nationalité, couleur, etc...) seront représentées par des prédicats unaires..

Question 6 *Utiliser *Paradox* pour montrer qu'il existe une unique solution à l'énigme. Autrement dit, montrer qu'il n'existe pas d'autre propriétaire possédant le poisson.*

Question 7 (Pouvez-vous faire mieux qu'Einstein ?) *Est-ce que tous les indices sont nécessaires pour résoudre l'énigme ? Expliquer comment répondre à cette question avec *Paradox*. Donner les indices inutiles s'il en existe.*

⁴elles sont voisines

Modalités

Le projet se fait en **binôme**, il est à rendre au bureau de Maryka Peetroons pour le **23 Avril 16h**.

Il doit comprendre un rapport papier qui répond aux questions (mettre le contenu des fichiers .tptp dans le rapport). Envoyez également les fichiers .tptp dans un dossier compressé portant les noms de famille des deux étudiants du binôme à l'adresse **efiliot@ulb.ac.be** avec pour objet **Projet Logique : nom de famille1-nom de famille2**. Le fichier est à envoyer pour la même date que la version papier.