Projet 2 L3IF — Quatrième rendu, apprentissage de clauses

à rendre pour le **mardi 17/05/2016 à 23h59**

envoyez une archive *qui compile* à `olga.kupriianova@ens-lyon.fr`, `antoine.plet@ens-lyon.fr` et `daniel.hirschkoff@ens-lyon.fr`

1 Au menu, déclinaisons

Voici les ingrédients de ce quatrième et dernier rendu :

1. Il s'agit du dernier rendu : repassez sur tout le code depuis le début, nettoyez-le, assurez-vous qu'aucun bug connu ne traîne, etc.
2. Terminez l'apprentissage de clauses si nécessaire (autrement dit, si vous n'avez traité que la partie 2.1 dans le rendu 3, la partie 2.2 est au menu).
3. Une quantité variable de SMT est également au menu.
   Suivant les cas, cela peut prendre la forme suivante :
   (a) SMT hors ligne pour l'égalité.
   (b) SMT en ligne pour deux théories : l'égalité, et une théorie au choix entre congruence et différences.
   Il faudra reprendre votre code de manière à avoir une version de DPLL(T), autrement dit que la partie "DPLL" ne soit pas dupliquée, mais partagée, le basculement d'une théorie à l'autre se faisant de manière naturelle.
   (c) SMT en ligne pour les trois théories décrites ci-dessous.
   Comme ci-dessus (point 3b), le code devra implémenter DPLL(T).
   Voir ci-dessous (partie 2) des informations à propos des diverses théories envisagées.
4. Préparez un rapport et une présentation— voir partie 4. Joignez le rapport au rendu 4.

Développez SMT sans activer l'apprentissage de clauses. Si vous en avez le temps, vérifiez que SMT continue à bien marcher lorsque l'on active l'apprentissage de clauses.

Un *point important* est que la partie SMT s'appuie sur des structures de données qui peuvent être codées de manière indépendante. Cela permet de commencer à travailler rapidement, point sur lequel bon nombre de binômes ont peché dans les rendus précédents.

Vous recevrez par mail une description plus spécifique de ce qui est attendu de votre part pour le rendu 4.

2 SMT : diverses théories

2.1 Égalité

Spécification. Ajoutez l'option `-smte` au solveur pour indiquer que l'on prend en entrée des formules exprimées dans la théorie de l'égalité.

Pour spécifier ces formules, on adopte les mêmes conventions que pour Tseitin, à ceci près que les atomes (les "constituants de base" des formules logiques) ne sont plus des entiers positifs ou négatifs (et différents de zéro). À la place, on peut écrire

`k = n` ou `k != n` où `k` et `n` sont des entiers strictement positifs.

Exemple :

```
12 = 5 => (7 != 8)
~(2 != 3) /\ 2 != 5  0
```

représente la formule

$$\big((x_{12} = x_5) \Rightarrow (x_7 \neq x_8)\big) \ \wedge \ \neg(x_2 \neq x_3) \wedge (x_2 \neq x_5)$$

Algorithme. L'algorithme union find (ou ensembles de Tarjan) sera à coder à part, sous forme d'une librairie capable de répondre à des questions formulées sous forme de conjonctions d'égalités ou diségalités.

Vous pourrez, le cas échéant, réfléchir à des enrichissements pour faciliter le traitement en lien avec DPLL(T) (explication d'un conflit, incrémentalité, possibilité de retours en arrière).

2.2 Congruence

Spécification. L'option à passer est `-smtc`

La syntaxe est comme pour la théorie de l'égalité, à ceci près que l'on peut écrire des termes. Formellement, les termes sont décrits par t ::= `x`$_i$ | `f`$(t_1, \ldots, t_k)$, en tenant compte du cas particulier où $k = 0$ (constante). À noter que dans le cas présent, une constante d'arité zéro (`a`, `b`, ...) est traitée comme une variable (x_i). Voici des exemples d'atomes pour la théorie de la congruence : `12 = f(a,7)`, `g(4) != g(a)`.

La signature (c'est-à-dire l'arité des diverses constantes, `a`, `b`, `f`, ...) ne sera pas spécifiée explicitement dans la formule en entrée, mais sera inférée par le solveur. Un message d'erreur sera affiché s'il y a des problèmes d'arité (comme dans `f(a) = f(b,a)`).

Algorithme. Vous trouverez sur la page www du cours des références indiquant comment traiter la congruence, en s'appuyant sur une variante de *union find*.

2.3 Logique de différences

Spécification. L'option est `-smtd`

Les atomes des formules saisies en entrées sont décrits par `x`$_i$ − `x`$_j$ *op* n | `x`*i* *op* n, où n désigne une constante entière, et avec *op* ::= `<` | `>` | `<=` | `>=` | `=` | `!=`.

Algorithme. Il s'agit de détecter la présence d'un cycle de poids négatif dans un graphe orienté. Les algorithmes de Bellman Ford et de Johnson sont de bons points d'entrée. Vous pouvez aussi vous référer à cet article.

Expliquez vos structures de données. **********

3 Et à la fin...

...cf. rendus précédents...

4 ...mais ça n'est pas tout à fait terminé

En plus du rendu : un bref compte-rendu sur le projet. On vous demande d'envoyer avec le dernier rendu un court rapport (entre 2 et 5 pages), rédigé en LaTeX, qui sera accompagné d'une présentation de 5 minutes avec des transparents lors de la séance du 18 mai.

Vous trouverez sur la page www du cours des exemples simples de fichiers desquels vous pourrez partir. Lisez ces fichiers pour savoir comment s'en servir, et comment engendrer un fichier au format pdf.

Passage obligé. Il vous faudra nécessairement mentionner minisat dans votre rapport, en faisant une citation comme celle-ci : minisat [1].

Pour cela, il vous faudra éditer le fichier `ex-biblio.bib`, en y insérant les données au bon format (le format BibTex) pour l'article de Eén et Sörensson de 2003. Comment trouver ces données ? Par une recherche internet, par exemple sur le site DBLP. Une fois trouvées les informations, copiez-collez-les directement dans le fichier .bib.

Rien de bien difficile dans tout cela, mais il faut l'avoir fait au moins une fois avant de partir en stage.

Pourquoi ce compte-rendu ? Soyons lucides, l'objectif essentiel de ce compte-rendu est de s'assurer que vous avez une certaine familiarité avec les outils mis en jeu. Du point de vue du contenu, ne perdez pas de temps à raconter ce que tout le monde sait (ce que sont les littéraux surveillés, qu'est-ce que l'apprentissage de clauses). Concentrez-vous sur ce qui est propre à votre travail sur le solveur : comment il est fabriqué, ce qu'il sait bien faire, ce qu'il devrait savoir faire, ce qui a été facile/difficile à réaliser au cours du semestre, etc. (bref, rendez-vous compte vous-même de ce qu'il est pertinent d'exposer).

Cet aspect des choses ne comptera pas pour beaucoup dans l'évaluation du projet, mais jouez quand même le jeu : encore une fois, cela vous sera utile, par exemple pour l'évaluation du stage de L3.

Références

[1] Niklas Eén and Niklas Sörensson. An extensible sat-solver. In Enrico Giunchiglia and Armando Tacchella, editors, *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.