

Mémoire intermédiaire PresTaf

Bourgeois Adrien, Marbois Bryce, Roque Maxime, Turnherr Jérémy
Université UFR Collégium des Sciences et Technique d'Orléans-la-Source

11 mars 2017

Table des matières

1 Résumé du projet	1
2 Domaine	2
2.1 Arithmétique de Presburger	2
2.2 Logique monadique du second ordre	2
2.3 Automates et minimisation	2
2.4 PresTaf	2
3 Analyse de PresTaf	2
4 Besoins fonctionnels	3
5 Besoins non fonctionnels	4
6 Prototypes et tests préparatoires	5
7 Planning	5

1 Résumé du projet

Notre projet s'inscrit dans la mis-à-jour de PresTaf, logiciel d'analyse de formules logique et de leur transformation en automate minimal. Nous intervenons ici dans la création d'un interfaçage [Lua](#), ainsi qu'une reprise du code source PresTaf pour donner une version plus optimisé, plus claire et propre.

Dans un second temps nous chercherons à intégrer de nouvelles logiques telle que la logique monadique ou encore l'interprétation des formules de Presburger en base -2.

2 Domaine

2.1 Arithmétique de Presburger

L'**Arithmétique de Presburger** à été introduite par Mojżesz Presburger en 1929. Cette arithmétique du premier ordre dispose de deux constante 0 et 1 ainsi qu'un symbole binaire +. Ce langage est limité aux entiers naturels et est défini par les lois suivante :

1. $\forall x, \neg(0 = x + 1)$
2. $\forall x, \forall y, x + 1 = y + 1 \rightarrow x = y$
3. $\forall x, x + 0 = x$
4. $\forall x, \forall y, x + (y + 1) = (x + y) + 1$
5. $\forall P(x, y_1, \dots, y_n) \in \text{Formule du premier ordre}, \forall y_1 \dots \forall y_n [(P(0, y_1, \dots, y_n) \vee \forall x (P(x, y_1, \dots, y_n) \rightarrow P(x + 1, y_1, \dots, y_n))) \rightarrow \forall y P(y, y_1, \dots, y_n)]$

2.2 Logique monadique du second ordre

La **Logique monadique du second ordre**, aussi connu sous le nom de *Monadic Second Order* ou *MSO*, est notamment utilisé dans un autre programme de M.Couvreur : VeriTaf. VeriTaf permet de vérifier des formules CTL (Computation Tree Logic) et des formules LTL (Linear Temporal Logic)

2.3 Automates et minimisation

Les automates que nous utilisons sont générés à partir de la formule de Presburger ou la formule monadique passée en entrée. Cet automate est fini et déterministe. Cependant ils ne sont pas minimaux, pour se faire il faut appliquer l'algorithme d'Hopcroft [?]. Nous implémenteront une variante de cet algorithme, l'algorithme de Blum [?]. La difficulté de cet algorithme n'est pas son implémentation qui elle est très simple mais la structure de données.

2.4 PresTaf

PresTaf permettrait d'avoir une représentation visuel, sous forme d'automates, d'une formule de Presburger ou monadique. En d'autres termes l'utilisateur rentrerait sa formule, et PresTaf lui fournira l'automate solution de sa formule. Ce programme pourrait notamment être utilisé par les étudiant de l'université.

3 Analyse de PresTaf

PresTaf est programme codé par Couvreur, qui prend des formules de Presburger en entrées et les résous. Les automates générés sont déterministes et finis mais non minimaux. Il faut donc les minimiser, et pour se faire il faut utiliser

l'algorithme d'Hopcroft. La où M.Couvreur avait fait une implémentation intelligence de l'algorithme, mais elle malgré son temps d'exécution très performant, il n'est pas facile à lire, et l'implémentation est assez vieille.

4 Besoins fonctionnels

Nom : Transformation de formules arithmétiques de Presburger

Description : Ensemble des fonctions qui prennent en entrée une formule arithmétique et retourne l'automate acceptant cette formule.

Justification : Besoin initial.

Priorité : 1

Nom : Minimisation d'automate (Blum)

Description : Ensemble de fonctions qui prennent un automate (fini, complet) et déterministe en entrée et retourne l'automate minimal équivalent.

Justification : Besoin initial.

Priorité : 1

Nom : Interfaçage Lua

Description : Permet le codage des automates en Lua, ainsi que l'utilisation de chaque fonction qui seront ensuite exécutés en Java.

Justification : Le lua est un langage de script simple à prendre en main et qui permet facilement d'écrire des automates et d'utiliser des fonctions.

Priorité : 2

Nom : Acceptation de formules en base -2

Description : La base -2 est défini par : $\sum_{i=0}^n (-2)^i * k_i$. Par exemple $2_{decimal} = 0 * (-2)^0 + 1 * (-2)^1 \Rightarrow 2_{decimal} = -2_{base-2}$. Pour déterminer un nombre en base -2, il suffit de déterminer son écriture binaire et ensuite d'appliquer le calcul base -2. Si l'on a le nombre $10110010_{binaire}$ alors on aura en base -2 : $0 * (-2)^0 + 1 * (-2)^1 + 0 * (-2)^2 + 0 * (-2)^3 + 1 * (-2)^4 + 1 * (-2)^5 + 0 * (-2)^6 + 1 * (-2)^7 = -146_{base-2}$

Justification :

Priorité : 3

Nom : Logique monadique du second ordre

Description :

Justification :

Priorité : 3

5 Besoins non fonctionnels

Nom : Machine visée

Description :

Justification :

Priorité : 1

Nom : Optimisation

Description :

Justification :

Priorité : 1

6 Prototypes et tests préparatoires

Implémentation de l'algorithme de Blum en python

7 Planning

1. Implémentation de l'algorithme de Blum.
2. Révision du code PresTaf.
3. Interface en Lua.
4. Rédaction du mémoire intermédiaire

Glossaire

Arithmétique de Presburger L'arithmétique de Presburger a été introduite par Mojżesz Presburger en 1929. Cette arithmétique du premier ordre dispose de deux constantes 0 et 1 ainsi qu'un symbole binaire $+$. Ce langage est limité aux entiers naturels et est défini par les lois suivantes :

1. $\forall x, \neg(0 = x + 1)$
2. $\forall x, \forall y, x + 1 = y + 1 \rightarrow x = y$
3. $\forall x, x + 0 = x$
4. $\forall x, \forall y, x + (y + 1) = (x + y) + 1$
5. $\forall P(x, y_1, \dots, y_n) \in \text{Formule du premier ordre}, \forall y_1 \dots \forall y_n [(P(0, y_1, \dots, y_n) \vee \forall x (P(x, y_1, \dots, y_n) \rightarrow P(x + 1, y_1, \dots, y_n))) \rightarrow \forall y P(y, y_1, \dots, y_n)]$

2

Logique monadique du second ordre aussi connu sous le nom de *Monadic Second Order* ou *MSO*, est notamment utilisé dans un autre programme de M.Couvreur : VeriTaf. VeriTaf permet de vérifier des formules CTL (Computation Tree Logic) et des formules LTL (Linear Temporal Logic) 2