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

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. $\forall P(x, y_1, \ldots, y_n) \in$ Formule du premier ordre, $\forall y_1 \ldots \forall y_n[(P(0, y_1, \ldots, y_n) \vee \forall x(P(x, y_1, \ldots, y_n) \rightarrow P(x + 1, y_1, \ldots, y_n))) \rightarrow \forall y P(y, y_1, \ldots, y_n)]$

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 forumules 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 deterministe. Cependant ils ne sont pas minimaux, pour se faire il faut appliquer l'algorithme d'Hopcroft [4]. Nous implémenteront une variante de cet algorithme, l'algorithme de Blum [1]. La difficulté de cet algorithme n'est pas son implémentation qui elle est très simple mais la structure de données.

2.4 PresTaf

PresTaf permet d'avoir une représentation visuel, sous forme d'automates, d'une formule de Presburger. En d'autres termes l'utilisateur rentre sa formule, et PresTaf lui fournit 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 M. Jean-Michel Couvreur, qui prend des formules de Presburger en entrées et les résout à l'aide d'automates minimaux. Tout d'abord il génère des automates déterministes et finis mais non minimaux. Il faut donc les minimiser, et pour se faire PresTaf utilise un algorithme d'Hopcroft modifié. L'ensemble des transitions menant de l'état initial vers un état final est solution de la formule. En outre si l'état final est l'état *zero* alors il n'y aucune solution et si l'état final est l'état *one* alors la formule est une tautologie.

Mona

Lash [2] est une bibliothèque C qui résout des formules de Presburger, mais la différence avec PresTaf est qu'il fonctionne sur des automates infini. Cette différence induit une importante baisse de performante. En effet PresTaf pour les mêmes formules était bien plus rapide à s'executer que Lash [3]

Büchi

L'algorithme d'Hopcroft est un algorithme de minimisation d'automate fini et deterministe.

Blum

4 Besoins fonctionnels

Nom : Transformation de formules arithmétiques de Presburger

Description : Ensemble de 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 determiner son écriture binaire et ensuite d'appliqué le calcule 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 : Il s'agit d'une logique du second ordre, c'est-à-dire qu'un prédicat peut avoir en argument un autre prédicat, mais celui-ci ne peut pas avoir un troisième prédicat en argument (arité un). De plus dans le cadre de la logique monadique du second ordre les quantificateurs ne peuvent être utilisé que pour les variables des prédicats du premier ordre (de type Presburger par exemple).

Justification :

Priorité : 3

5 Besoins non fonctionnels

Nom : Machine visée

Description : Windows, MacOS, Linux

Justification :

Priorité : 1

Nom : Optimisation (de PresTaf ou d'un nouveau code ?)

Description :

Justification :

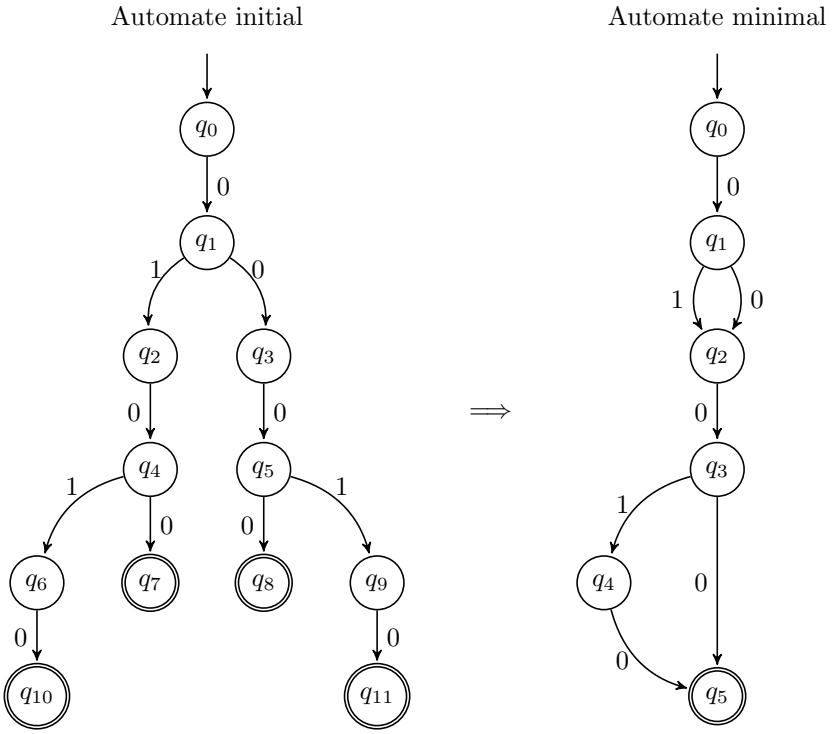
Priorité : 1

6 Prototypes et tests préparatoires

6.1 Blum

Implémentation de l'algorithme de Blum en Python.

Voici un exemple que nous avons testé avec l'automate initial et l'objectif.



6.2 LuaJava

7 Planning

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

Références

[1] Norbert Blum. An o(nlogn) implementation of the standard method for minimizing n-state finite automata. *Information Processing Letters*, 57(2) :65–69, 1996.

[2] Bernard Boigelot. Lash toolset.

[3] Jean-Michel Couvreur. A bdd-like implementation of an automata package. In *Implementation and Application of Automata, 9th International Conference, CIAA 2004, Kingston, Canada, July 22-24, 2004, Revised Selected Papers*, pages 310–311, 2004.

[4] John Hopcroft. An/n log n algorithm for minimizing states in kf in ite automaton. 1971.

Glossaire

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. $\forall P(x, y_1, \ldots, y_n) \in$ Formule du premier ordre, $\forall y_1 \ldots \forall y_n[(P(0, y_1, \ldots, y_n) \vee \forall x(P(x, y_1, \ldots, y_n) \rightarrow P(x + 1, y_1, \ldots, y_n))) \rightarrow \forall y P(y, y_1, \ldots, y_n)]$

3

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 forumules LTL (Linear Temporal Logic) 3