DEA Informatique
Université de Rennes I
ENS Cachan -antenne de Bretagne-

Approximation de Stratégies de Preuves en Réécriture

Rapport de stage

Yann Hodique

Encadrant :
Thomas Genet

Introduction

L'augmentation constante du besoin de fiabilité logicielle, notamment dans les secteurs dits critiques, a permis le développement d'un nombre important d'assistants de preuve (Coq, PVS, Isabelle, ...) dont la finalité est de produire une garantie, sous forme de preuve mathématique, de l'adéquation entre le comportement d'un programme et ce que l'on attend de lui. Dans ces outils, les théories équationnelles (ensembles d'équations) sont utilisées comme des règles de réécriture pour la simplification des preuves. Cependant, l'application systématique de ces règles de réécriture peut diverger, et gêner l'utilisateur au lieu de l'assister dans sa preuve. L'algorithme de complétion de Knuth-Bendix permet de transformer une théorie équationnelle $\mathcal{E}$ en une procédure de décision pour $\mathcal{E}$, c'est à dire en un système de réécriture équivalent mais toujours convergent. Cependant la complétion de Knuth-Bendix diverge assez fréquemment en pratique. De plus, elle nécessite pour fonctionner de disposer d'un ordre de terminaison, et la recherche d'un tel ordre est un problème difficile [4]. Nous proposons de recourir à une méthode d'approximation pour éviter la divergence de la réécriture dans les preuves équationnelles. En particulier nous verrons comment construire une stratégie approchée qui puisse guider l'application des règles. Cette stratégie doit permettre de faire converger plus rapidement les preuves par réécriture dans les assistants de preuve.

Première partie

Étude bibliographique

Chapitre 1

Fondements théoriques

1.1 Présentation du problème

Le rôle des assistants de preuve étant de démontrer des propriétés dans un certain contexte, le schéma rencontré pour les problèmes à résoudre est toujours le même : à partir d'un certain nombre d'hypothèses, on tente d'aboutir à une conclusion. La séquence d'utilisation des hypothèses amenant à cette conclusion constitue la preuve de la propriété.

Exemple Voici un exemple typique de ce que l'on pourra vouloir démontrer (ici avec l'assistant de preuve PVS). Les lignes numérotées négativement sont les axiomes d'une théorie ensembliste classique, et la ligne numérotée positivement est la propriété que l'on cherche à prouver : $A \cap (A \cup B) = A$.

```
{-1}  FORALL (A: Ensemble): inter(A, Vide) = Vide
[-2]  FORALL (A: Ensemble): union(A, Vide) = A
[-3]  FORALL (A, B, C: Ensemble):
        union(A, inter(B, C)) = inter(union(A, B), union(A, C))
[-4]  FORALL (A: Ensemble): union(A, A) = A
[-5]  FORALL (A, B: Ensemble): inter(A, B) = inter(B, A)
[-6]  FORALL (A, B: Ensemble): union(A, B) = union(B, A)
  |-------
[1]   inter(A!1, union(A!1, B!1)) = A!1
```

Pour démontrer une conclusion, on peut appliquer de manière systématique l'ensemble des équations disponibles jusqu'à se ramener au cas simple, à savoir celui des axiomes de la théorie que l'on considère.

Si, pour certaines théories simples, l'application brutale des règles disponibles suffit, rendant ainsi la démonstration des propriétés totalement automatique, dans la majeure partie des cas le processus n'aboutit pas. Dans l'exemple considéré, l'application de la commande `grind`, qui ré-écrit les hypothèses et la

conclusion à l'aide des axiomes sans stratégie particulière, s'arrête en produisant le résultat suivant :

```
Rule? (grind)
Trying repeated skolemization, instantiation, and if-lifting,
this simplifies to:
theo :

{-1} inter(inter(inter(inter(inter(Vide, Vide), inter(Vide, Vide)),
                       inter(inter(Vide, Vide), inter(Vide, Vide))),
                 inter(inter(inter(Vide, Vide), inter(Vide, Vide)),
                       inter(inter(Vide, Vide), inter(Vide, Vide)))),
           inter(inter(inter(inter(Vide, Vide), inter(Vide, Vide)),
                       inter(inter(Vide, Vide), inter(Vide, Vide))),
                 inter(inter(inter(Vide, Vide), inter(Vide, Vide)),
                       inter(inter(Vide, Vide), inter(Vide, Vide)))))
      =
      inter(inter(inter(inter(Vide, Vide), inter(Vide, Vide)),
                  inter(inter(Vide, Vide), inter(Vide, Vide))),
            inter(inter(inter(Vide, Vide), inter(Vide, Vide)),
                  inter(inter(Vide, Vide), inter(Vide, Vide))))
{-2} union(inter(inter(inter(inter(Vide, Vide), inter(Vide, Vide)),
                       inter(inter(Vide, Vide), inter(Vide, Vide))),
                 inter(inter(inter(Vide, Vide), inter(Vide, Vide)),
                       inter(inter(Vide, Vide), inter(Vide, Vide)))),
           inter(inter(inter(inter(Vide, Vide), inter(Vide, Vide)),
                       inter(inter(Vide, Vide), inter(Vide, Vide))),
                 inter(inter(inter(Vide, Vide), inter(Vide, Vide)),
                       inter(inter(Vide, Vide), inter(Vide, Vide)))))
      =
      inter(inter(inter(inter(Vide, Vide), inter(Vide, Vide)),
                  inter(inter(Vide, Vide), inter(Vide, Vide))),
            inter(inter(inter(Vide, Vide), inter(Vide, Vide)),
                  inter(inter(Vide, Vide), inter(Vide, Vide))))
{-3} Vide = inter(Vide, Vide)
  |-------
[1]  inter(A!1, union(A!1, B!1)) = A!1
```

On observe que les règles de réécriture ont été appliquées de façon aléatoire, et qu'en poursuivant ainsi, la probabilité d'aboutir à une preuve de la propriété est nulle, alors que cette démonstration peut être menée à bien en cinq étapes de réécriture :

$$\underline{A} \cap (A \cup B) =_{(-2)} \underline{(A \cup \emptyset) \cap (A \cup B)} =_{(-3)} A \cup (\underline{\emptyset \cap B})$$
$$=_{(-5)} A \cup (\underline{B \cap \emptyset}) =_{(-1)} \underline{A \cup \emptyset} =_{(-2)} A$$

La finalité de ce stage sera de tenter d'apporter de nouvelles méthodes, plus efficaces que la recherche brute et plus pragmatiques que les procédures de décision automatique, qui sont limitées à un petit nombre de théories équationnelles.

1.2 Réécriture et théories égalitaires

Soit $\mathcal{A}$ un ensemble fini de symboles et f une fonction d'arité : $f : \mathcal{A} \to \mathbb{N}^+$
Dans la suite de ce document on représentera par $\mathcal{F}$ l'ensemble fini

$$\{(a, f(a)) \mid a \in \mathcal{A}\}$$

et par $\mathcal{X}$ un ensemble dénombrable de variables. $\mathcal{T}(\mathcal{F}, \mathcal{X})$ représentera l'ensemble des termes construits sur $\mathcal{F}$ et $\mathcal{X}$, et $\mathcal{T}(\mathcal{F})$ l'ensemble des termes clos (termes sans variable). L'ensemble des variables d'un terme t sera noté $\mathcal{V}ar(t)$.

Exemple 1. *Soient* $\mathcal{F} = \{(f, 2), (g, 1), (a, 0)\}$ *et* $\mathcal{X} = \{x, y\}$

$$\mathcal{T}(\mathcal{F}, \mathcal{X}) = \{a, x, y, g(a), \ldots, f(x, g(a)), \ldots\} \tag{1.1}$$

$$\mathcal{T}(\mathcal{F}) = \{a, g(a), f(a, a), \ldots, f(f(a, g(a)), g(a)), \ldots\} \tag{1.2}$$

Soit $t = f(x, g(f(a, x)))$ *alors* $\mathcal{V}ar(t) = \{x\}$

Définition 1 (Substitution). *Une substitution est une application* σ *de* $\mathcal{X}$ *dans* $\mathcal{T}(\mathcal{F}, \mathcal{X})$ *qui peut être étendue de manière unique en un endomorphisme sur* $\mathcal{T}(\mathcal{F}, \mathcal{X})$

Exemple 2. *Soient* $\mathcal{F} = \{(f, 2), (g, 1), (a, 0)\}$ *et* $\sigma = \{x \mapsto f(a, g(a))\}$ *À* $t = f(x, a)$ *on associe* $t\sigma = f(f(a, g(a)), a)$

Définition 2 (Équations et théories équationnelles). *Une équation est une expression de la forme* $l = r$*, avec* $(l, r) \in \mathcal{T}(\mathcal{F}, \mathcal{X})^2$*. Une théorie équationnelle est une théorie dont les axiomes sont un ensemble d'équations.*

Définition 3 (Égalité modulo $\mathcal{E}$). *La relation* $=_\mathcal{E}$ *est la plus petite relation vérifiant :*

$$\begin{cases} \forall s \in \mathcal{T}(\mathcal{F}, \mathcal{X}) : s =_\mathcal{E} s \text{ (réflexive)} \\ \forall (s, t, u) \in \mathcal{T}(\mathcal{F}, \mathcal{X})^3 : s =_\mathcal{E} t \wedge t =_\mathcal{E} u \implies s =_\mathcal{E} u \text{ (transitive)} \\ \forall (s, t) \in \mathcal{T}(\mathcal{F}, \mathcal{X})^2 : s =_\mathcal{E} t \implies t =_\mathcal{E} s \text{ (symétrique)} \\ \forall f \in \mathcal{F} : (\forall i \{1 \ldots n\} : a_i =_\mathcal{E} b_i) \implies f(a_1, \ldots a_n) =_\mathcal{E} f(b_1, \ldots b_n) \text{ (compatible)} \\ \forall \sigma : (\forall (l = r) \in \mathcal{E} : l\sigma =_\mathcal{E} r\sigma) \end{cases}$$

Définition 4 (Système de réécriture). *Un système de réécriture* $\mathcal{R}$ *est un ensemble de règles de réécriture de la forme* $l \to r$*, où*

$$\begin{cases} l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \\ \mathcal{V}ar(l) \supseteq \mathcal{V}ar(r) \end{cases}$$

Définition 5 (Réécriture). *La relation $\rightarrow_R$ est la plus petite relation vérifiant :*

$$\begin{cases} \forall \sigma : (\forall (l \rightarrow_{\mathcal{R}} r) \in \mathcal{R} : l\sigma \rightarrow_{\mathcal{R}} r\sigma) \\ \forall f \in \mathcal{F} : a \rightarrow_{\mathcal{R}} b \implies f(\ldots, a, \ldots) \rightarrow_{\mathcal{R}} f(\ldots, b, \ldots) \end{cases}$$

Définition 6 (Positions de réécriture). *Une position de réécriture représente un emplacement dans un terme, en vu de l'application d'une règle à cet endroit. Une telle position est notée suivant une convention hiérarchique pointée. Par convention on marquera par ϵ la fin d'une position.*

Exemple 3. *Dans le terme $f(g(x, y))$, la position* $1.1.2.\epsilon$ *désigne l'emplacement marqué par le symbole y.*

Définition 7 (Linéarité). *Une règle de réécriture $l \rightarrow r$ est linéaire à gauche (resp. à droite) si chaque variable de l (resp. r) apparaît une seule fois. Une règle est linéaire si elle est linéaire à gauche et à droite. Un système de réécriture est linéaire (resp. linéaire à gauche, à droite) si chacune des règles qui le composent est linéaire (resp. linéaire à gauche, à droite).*

Définition 8 (Descendants). *La clôture réflexive transitive de $\rightarrow_{\mathcal{R}}$ est notée $\rightarrow^*_{\mathcal{R}}$. L'ensemble des $\mathcal{R}$-descendants d'un ensemble de termes clos $\mathcal{E}$ est $\mathcal{R}^*(\mathcal{E}) = \{t \in \mathcal{T}(\mathcal{F}) \mid \exists s \in \mathcal{E} \text{ t.q. } s \rightarrow^*_{\mathcal{R}}\}$.*

Définition 9 (Confluence). *Un système de réécriture est dit confluent si* $\forall (s,t,u) \in \mathcal{T}(\mathcal{F})^3 \begin{cases} s \rightarrow^*_{\mathcal{R}} t \\ s \rightarrow^*_{\mathcal{R}} u \end{cases} \implies \exists v \in \mathcal{T}(\mathcal{F}) \text{ t.q. } \begin{cases} t \rightarrow^*_{\mathcal{R}} v \\ u \rightarrow^*_{\mathcal{R}} v \end{cases}$

Définition 10 (Terminaison). *Un système $\mathcal{R}$ est dit terminant s'il n'existe pas de suite infinie de réécriture $s \rightarrow_{\mathcal{R}} s_1 \rightarrow_{\mathcal{R}} s_2 \rightarrow_{\mathcal{R}} \ldots$ Autrement dit, si tout terme possède une forme normale.*

1.3 Décision de l'égalité

1.3.1 Résultats

Théorème 1 (Théorème de Birkoff). *Soit $\mathcal{E}$ une théorie équationnelle. $\mathcal{R} = \{l \rightarrow r, r \rightarrow l \mid l = r \in \mathcal{E}\}$.*

$$s =_{\mathcal{E}} t \iff s \rightarrow^*_{\mathcal{R}} t \tag{1.3}$$

Ce théorème ne permet pas d'obtenir une procédure de décision de l'égalité à cause du problème de la terminaison : celle-ci ne peut être assurée à cause de la présence systématique d'une règle de réécriture et de sa réciproque.

Exemple 4. *Soit la théorie équationnelle $\mathcal{E} = \{x + 0 = x\}$. $\mathcal{E}$ induit le système de réécriture $\mathcal{R} = \{0 + x \rightarrow x, x \rightarrow 0 + x\}$. Ainsi pour démontrer $0 + 0 = 0$ on pourra aboutir à la suite infinie de réécritures $0 + 0 \rightarrow 0 + (0 + 0) \rightarrow 0 + (0 + (0 + 0)) \ldots$*

Proposition 1. *Soit $\mathcal{E}$ une théorie équationnelle $\{l_1 = r_1, l_2 = r_2, \ldots, l_n = r_n\}$ telle que $\mathcal{R} = \{l_1 \to r_1, l_2 \to r_2, \ldots, l_n \to r_n\}$ soit un système de réécriture terminant et confluent. Soit $(X, Y) \in \mathcal{T}(\mathcal{F}, \mathcal{X})^2$.*

$$X =_{\mathcal{E}} Y \iff \exists Z \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \text{ t.q. } \begin{cases} X \to_{\mathcal{R}}^* Z \\ Y \to_{\mathcal{R}}^* Z \end{cases} \tag{1.4}$$

Ainsi, pour toute théorie équationnelle $\mathcal{E}$ orientable en un système de réécriture $\mathcal{R}$ confluent et terminant, il est possible de décider $X =_{\mathcal{E}} Y$ en ré-écrivant X et Y par $\mathcal{R}$ jusqu'à atteindre une forme normale de chacun de ces deux termes, et en vérifiant que ces deux formes normales sont identiques.

Pour produire un système de réécriture confluent et terminant à partir d'une théorie équationnelle $\mathcal{E}$, on introduit la complétion de Knuth-Bendix.

1.3.2 Complétion de Knuth-Bendix

Définition 11 (Confluence locale). $\forall (s, t, u) \in \mathcal{T}(\mathcal{F})^3$ *t.q.*

$$\begin{cases} s \xrightarrow{\mathcal{R}} t \\ s \xrightarrow{\mathcal{R}} u \end{cases}$$

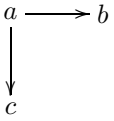
$\exists v \in \mathcal{T}(\mathcal{F})$ *t.q.*

$$\begin{cases} t \xrightarrow{\mathcal{R}}{}^* v \\ u \xrightarrow{\mathcal{R}}{}^* v \end{cases}$$

Définition 12 (Paire critique). *Soient $s \to t$ et $l \to r$ deux règles de réécriture (non nécessairement distinctes). Soit μ l'unificateur le plus général de l et d'un sous-terme s_1 de s, avec $s_1 \notin \mathcal{X}$. L'équation $s\mu[r\mu] = t\mu$, où $r\mu$ remplace $s_1\mu$ $(= l\mu)$ dans $s\mu$, est une paire critique pour ces règles.*

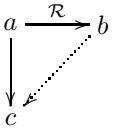
Exemple 5. *$\mathcal{E} = \{a = b, a = c\}$ orienté en $\mathcal{R} = \{a \to b, a \to c\}$ On a bien $b =_{\mathcal{E}} a =_{\mathcal{E}} c$ mais $b \not\to_{\mathcal{R}}$ et $c \not\to_{\mathcal{R}}$*

Algorithme 1. *Soit $\mathcal{R}$ un système de réécriture. Tant que $\mathcal{R}$ n'est pas confluent, on sélectionne une paire critique*



et on ajoute à $\mathcal{R}$ la règle $b \to c$ (ou $c \to b$).

Exemple 6. *Dans l'exemple précédent, en ajoutant*



on obtient le système de réécriture $\mathcal{R}' = \{a \to b, a \to c, b \to c\}$ *confluent, terminant et décidant* $=_\mathcal{E}$

1.3.3 La décision de l'égalité modulo $\mathcal{E}$ en pratique

La procédure de complétion de Knuth-Bendix est rarement utilisée dans les assistants de preuve car elle demande une grande expertise et diverge très fréquemment.

Le constat est donc le suivant : étant donnée une théorie équationnelle $\mathcal{E}$, on ne dispose quasiment d'aucun outil pour savoir si $s =_\mathcal{E} t$ dans le cas général. Pour tenter de remédier à cette situation, on va utiliser directement le théorème de Birkoff avec des stratégies de recherche : on s'intéressera donc à un problème plus général : savoir si $s \to_\mathcal{R}^* t$ pour un système de réécriture donné.

Chapitre 2

Stratégie et complétion d'automates

2.1 Recherche brute

Considérons le système de réécriture suivant :

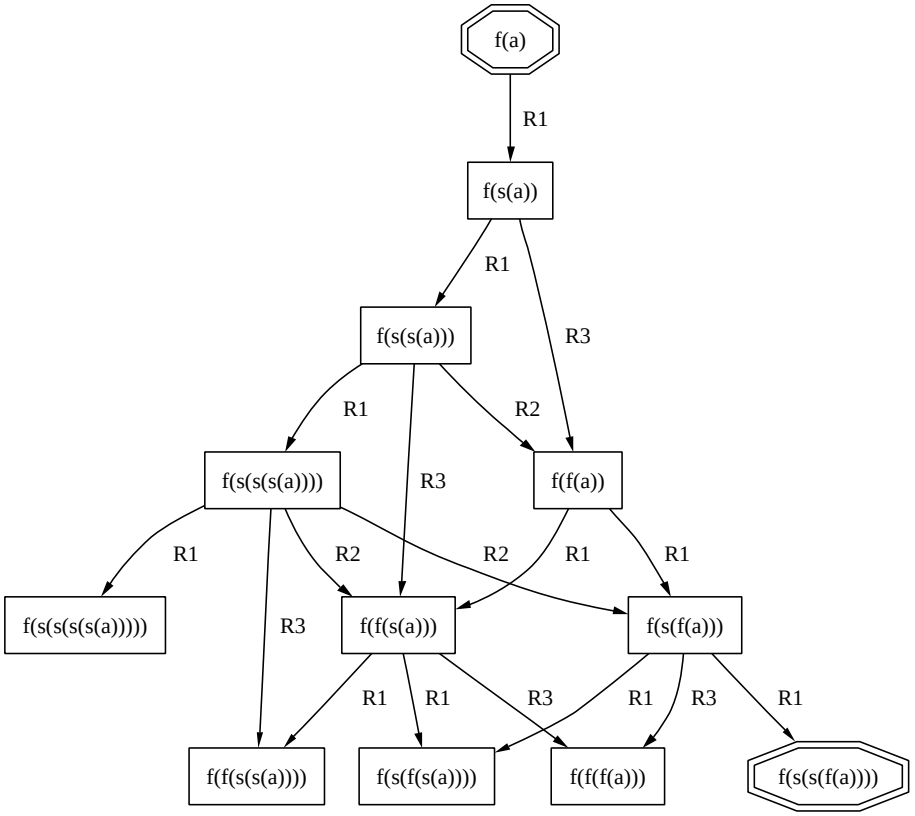
$$\mathcal{R} = \begin{cases} f(x) \to f(s(x)) \\ s(s(x)) \to f((x)) \\ f(s(x)) \to f(f(x)) \end{cases}$$

On désire savoir si $f(a) \to_{\mathcal{R}}^{*} f(s(s(f(a))))$ est prouvable.

Une recherche naïve, à savoir l'application systématique de toutes les règles de réécriture disponibles, aboutit en 4 pas de réécriture en produisant l'arbre 2.1 On constate que sur cet exemple particulièrement simple, l'arbre développé est déjà complexe. On imagine donc sans peine à quel point, sur des exemples réels, cette méthode se révèle impraticable

Nous étudierons donc la possibilité d'utiliser la complétion d'automates [5] pour améliorer cette méthode. En effet, pour un terme t donné, la complétion d'automate permet de calculer une sur-approximation de $\mathcal{R}^*(\{t\})$ pour un système de réécriture $\mathcal{R}$ quelconque. Ceci est déjà utilisé pour prouver que deux termes s et t ne sont pas joignables (i.e. $s \not\to_{\mathcal{R}}^{*} t$). Nous proposons d'exploiter l'information produite par la complétion d'automate pour optimiser l'algorithme de recherche brute. Dans le cas de notre exemple, nous souhaitons procéder à la complétion d'automate pour approcher $\mathcal{R}^*(\{f(a)\})$ et utiliser le résultat de la complétion pour construire une stratégie approchée menant à $f(s(s(f(a))))$.

Fig. 2.1 Recherche exhaustive



2.2 Complétion d'automates

2.2.1 Définitions

Définition 13 (Transition). *Soit $\mathcal{Q}$ un ensemble fini de symboles d'arité 0 appelés états. $\mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ est appelé ensemble des configurations. Une transition est un règle de réécriture $c \to q$, où $c \in \mathcal{T}(\mathcal{F} \cup \mathcal{Q})$ et $q \in \mathcal{Q}$. Une transition normalisée est une transition $c \to q$ où*

$$\begin{cases} c = q' \in \mathcal{Q} \text{ ou } c = f(q_1, \ldots, q_n) \\ f \in \mathcal{F} \\ ar(f) = n \\ q_1 \ldots q_n \in \mathcal{Q}. \end{cases}$$

Définition 14 (Automate d'arbre fini). *Un automate d'arbre fini sur $\mathcal{F}$ est un quadruplet $\mathcal{A} = \langle \mathcal{Q}, \mathcal{F}, \mathcal{Q}_f, \Delta \rangle$ avec $\mathcal{Q}$ un ensemble d'états, $\mathcal{Q}_f \subseteq \mathcal{Q}$ l'ensemble des états finaux, et Δ l'ensemble des transitions, de la forme :*

$$f(q_1, \ldots, q_n) \to q$$

avec $n \geq 0$, $f \in \mathcal{F}_n$, $q, q_1, \ldots, q_n \in \mathcal{Q}$.

Définition 15 (Automate d'arbre déterministe). *Un automate d'arbre est dit déterministe si $\forall l_1 \to r_1, l_2 \to r_2 \in \Delta : l_1 \neq l_2$*

La relation de réécriture induite par les transitions de $\mathcal{A}$ est notée $\to_{\mathcal{A}}$. Le langage reconnu par $\mathcal{A}$ est $\mathcal{L}(\mathcal{A}) = \{t \in \mathcal{T}(\mathcal{F}) \mid \exists q \in \mathcal{Q}_f \text{ t.q. } t \to_{\mathcal{A}}^* q\}$.

Exemple 7. *Soit $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ avec $\mathcal{F} = \{(f, 1), (a, 0)\}$, $\mathcal{Q} = \{q_a\}$, $\mathcal{Q}_f = \{q_a\}$ et $\Delta = \{a \to q_a, f(q_a) \to q_a\}$. Le langage reconnu par $\mathcal{A}$ est $\mathcal{L}(\mathcal{A}) = f^*(a)$*

2.2.2 Algorithme de complétion

Motivation À partir d'un automate d'arbre $\mathcal{A}_0 = \langle \mathcal{F}, \mathcal{Q}_0, \mathcal{Q}_f, \Delta_0 \rangle$ et d'un système de réécriture $\mathcal{R}$, le but est de calculer un automate d'arbre $\mathcal{A}_k$ tel que $\mathcal{L}(\mathcal{A}_k) \supseteq \mathcal{R}^*(\mathcal{L}(\mathcal{A}_0))$

Démarche On calcule une suite d'automates d'arbre $(\mathcal{A}_k)_{k \geq 1}$ telle que $\forall i \geq 0 : \mathcal{L}(\mathcal{A}_i) \subseteq \mathcal{L}(\mathcal{A}_{i+1})$; et si $s \in \mathcal{L}(\mathcal{A}_i)$ tel que $s \to_{\mathcal{R}} t$ alors $t \in \mathcal{L}(\mathcal{A}_{i+1})$ jusqu'à stabilisation du processus, à savoir obtention d'un $k \in \mathbb{N}$ vérifiant $\mathcal{L}(\mathcal{A}_k) = \mathcal{L}(\mathcal{A}_{k+1})$

La construction de A_{i+1} à partir de A_i est effectuée à l'aide de l'algorithme suivant.

Algorithme

- découverte d'une paire critique entre $\to_{\mathcal{R}}$ et $\to_{\mathcal{A}_i}$:
  soit une règle $l \to r$ de $\mathcal{R}$, soit σ une substitution, on a une paire critique si $\exists q \in \mathcal{Q}$ tel que $l\sigma \to_{\mathcal{A}_i}^* q$, $l\sigma \to_{\mathcal{R}} r\sigma$ et $r\sigma \not\to_{\mathcal{A}_i}^* q$
- normalisation de $r\sigma \to q$
- ajout des nouvelles règles à Δ

2.2.3 Exemple de complétion d'automate

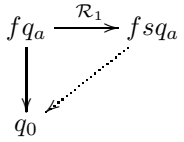
Initialement, on a l'automate $\mathcal{A} = \langle \mathcal{F}, \mathcal{Q}, \mathcal{Q}_f, \Delta \rangle$ tel que $\mathcal{F} = \{(f,1),(a,0)\}$, $\mathcal{Q} = q_a$, $\mathcal{Q}_f = q_0$ et $\Delta = \left\{ \begin{array}{l} f(q_a) \to q_0 \\ a \to q_a \end{array} \right.$ Cet automate reconnaît le langage $L(A) = \{f(a)\}$. Décrivons maintenant la complétion de cet automate par le système

$$\mathcal{R} = \left\{ \begin{array}{ll} f(x) \to f(s(x)) & (\mathcal{R}_1) \\ s(s(x)) \to f((x)) & (\mathcal{R}_2) \\ f(s(x)) \to f(f(x)) & (\mathcal{R}_3) \end{array} \right.$$

Étapes de complétion :

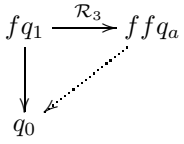
Remarque 1. *la complétion d'automates menée sur $\mathcal{A}$ et $\mathcal{R}$ produit un automate reconnaissant un sur-ensemble de $\mathcal{R}^*(\mathcal{L}(\mathcal{A}))$, cependant par souci de simplification et de clarté, on se limitera à un sous-ensemble des étapes de complétion qui contribuent réellement à la résolution du problème.*

Étape 1



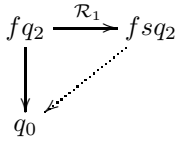
On complète le système avec : $\left\{ \begin{array}{l} s(q_a) \to q_1 \\ f(q_1) \to q_0 \end{array} \right.$. qui est une normalisation possible de $fsq_a \to q_0$
$\mathcal{A}_1 = \langle \mathcal{F}, \mathcal{Q}_1, \mathcal{Q}_f, \Delta_1 \rangle$ avec
$\mathcal{Q}_1 = \mathcal{Q} \cup \{q_1\}$, $\Delta_1 = \Delta \cup \{s(q_a) \to q_1, f(q_1) \to q_0\}$.

Étape 2



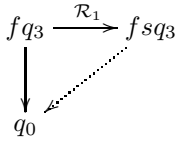
On complète le système avec : $\left\{ \begin{array}{l} f(q_a) \to q_2 \\ f(q_2) \to q_0 \end{array} \right.$.
$\mathcal{A}_2 = \langle \mathcal{F}, \mathcal{Q}_2, \mathcal{Q}_f, \Delta_2 \rangle$ avec
$\mathcal{Q}_2 = \mathcal{Q}_1 \cup \{q_2\}$, $\Delta_2 = \Delta_1 \cup \{f(q_a) \to q_2, f(q_2) \to q_0\}$.

Étape 3



On complète le système avec : $\begin{cases} s(q_2) \to q_3 \\ f(q_3) \to q_0 \end{cases}$.
$\mathcal{A}_3 = \langle \mathcal{F}, \mathcal{Q}_3, \mathcal{Q}_f, \Delta_3 \rangle$ avec
$\mathcal{Q}_3 = \mathcal{Q}_2 \cup \{q_3\}$, $\Delta_3 = \Delta_2 \cup \{s(q_2) \to q_3, f(q_3) \to q_0\}$.

Étape 4



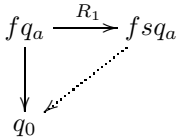
On complète le système avec : $\begin{cases} s(q_3) \to q_4 \\ f(q_4) \to q_0 \end{cases}$.
$\mathcal{A}_4 = \langle \mathcal{F}, \mathcal{Q}_4, \mathcal{Q}_f, \Delta_4 \rangle$ avec
$\mathcal{Q}_4 = \mathcal{Q}_3 \cup \{q_4\}$, $\Delta_4 = \Delta_3 \cup \{s(q_3) \to q_4, f(q_4) \to q_0\}$.

Conclusion On a bien $f(s(s(f(a)))) \in L(\mathcal{A}_4)$. Ici, afin de simplifier l'exemple, la complétion aboutissant à $\mathcal{A}_4$ est menée de façon exacte, ce qui nous permet de répondre à la question posée : $f(a) \to_{\mathcal{R}}^* f(s(s(f(a))))$ est prouvable. Par contre, ce que nous cherchons véritablement, à savoir la séquence précise des règles à appliquer et des positions auxquelles les appliquer, reste inconnue.

2.3 Le projet

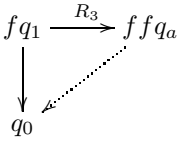
Les étapes de complétion de l'algorithme peuvent fournir un supplément d'information concernant les règles à appliquer ainsi que la façon de les appliquer :

Étape 1



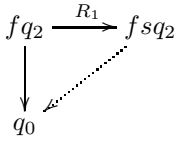
On a complété le système avec : $\begin{cases} s(q_a) \to q_1 \\ f(q_1) \to q_0 \end{cases}$.
Cette étape n'apporte pas d'information utile puisque la seule règle applicable en partant de $f(a)$ est R_1 et qu'elle ne peut s'appliquer qu'à une seule position dans ce terme.

Étape 2



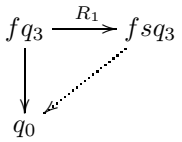
On a complété le système avec : $\begin{cases} f(q_a) \to q_2 \\ f(q_2) \to q_0 \end{cases}$.
L'information apportée est que la règle appliquée sur le terme $f(s(a))$ est R_3. Comme R_3 ne peut s'appliquer qu'à une seule position, aucune autre information n'est nécessaire.

Étape 3



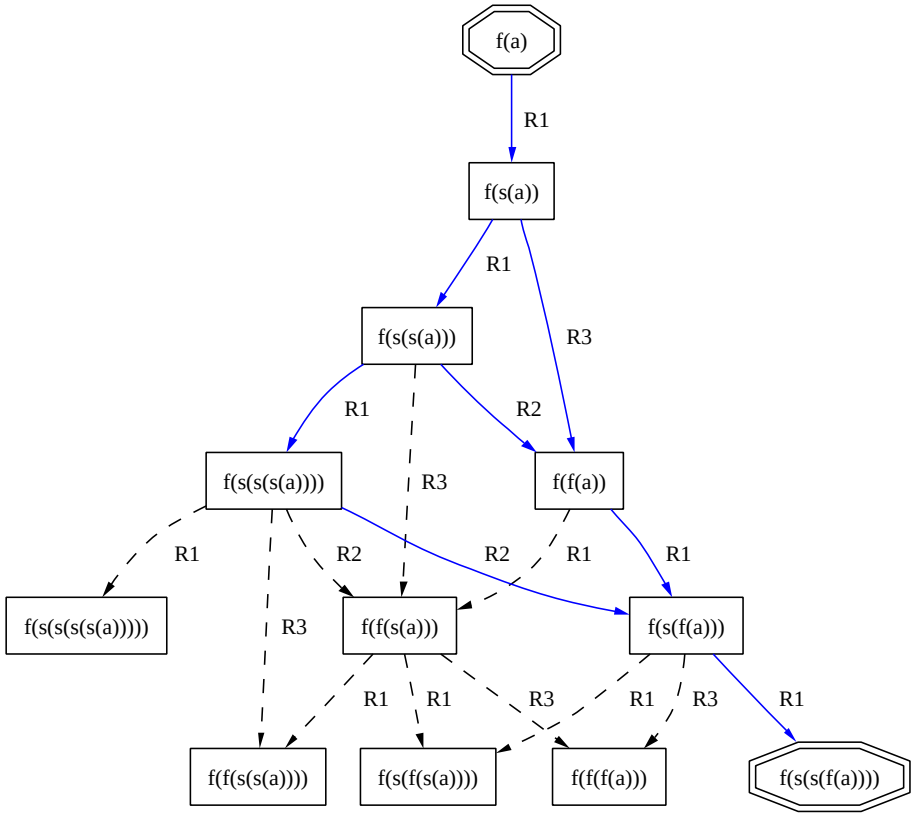
On a complété le système avec : $\begin{cases} s(q_2) \to q_3 \\ f(q_3) \to q_0 \end{cases}$.
Dans ce cas l'information est moins évidente : on sait qu'une application de la règle R_1 est nécessaire, mais on ne sait pas à quelle position. On peut cependant la déduire des états mis en jeux dans la paire critique : le fait d'aboutir en q_0 ne nous laisse pas d'autre choix que d'appliquer la règle au sommet du terme (une application sur un sous-terme aurait donné lieu à une paire critique aboutissant en un autre état : q_2).

Étape 4



On a complété le système avec : $\begin{cases} s(q_3) \to q_4 \\ f(q_4) \to q_0 \end{cases}$.
Ceci achève la complétion dans les mêmes conditions qu'à l'étape précédente : les états mis en jeu ici ne laissent pas d'autre possibilité que d'appliquer la règle en début de terme.

Fig. 2.2 Recherche orientée



Dans ce cas simple, on voit que l'information récupérable est non-négligeable : séquence précise des réécritures à effectuer, position à laquelle appliquer la réécriture en cas de positions de réécriture multiples. En effectuant toutes les étapes de l'algorithme de complétion, on en arrive à restreindre l'arbre de la manière représentée en 2.2 (représentée par des flèches pleines).

Dans l'exemple précédent, nous avons mis en évidence la présence d'information dans l'algorithme de complétion. Nous avons cependant passé sous silence le cas où l'on introduit une approximation dans la complétion. Ceci ayant pour effet de fusionner des noeuds de l'arbre de réécritures, la conséquence est une perte d'information sur les étapes de réécriture à effectuer pour atteindre le terme cible. L'objectif de ce stage sera de déterminer comment exploiter au maximum l'information dont on dispose pendant le déroulement de la complétion de l'automate, afin de reconstituer le plus précisément possible une stratégie approchée de preuve qui nous permette de démontrer les théorèmes souhaités.

Deuxième partie

Stage

Introduction

Dans ce document, nous tenterons de restituer le plus fidèlement possible la démarche suivie durant le stage proprement dit, en procédant par étapes successives. Dans un premier temps nous nous intéresserons au cas de la construction d'une stratégie à partir d'une complétion d'automate exacte. Ensuite nous reviendrons dans un cadre plus général et moins favorable pour prendre en compte du mieux possible le caractère approché de la complétion, qui fait tout l'intérêt de la méthode.

Chapitre 3

Complétion exacte

3.1 Motivations

Se placer dans le cas exact présente l'avantage de bénéficier du fait que la complétion reflète exactement la réécriture : à un état correspond un terme. Cette restriction doit permettre de préciser les données véhiculées par l'algorithme de complétion. Pour l'instant il n'est pas question de donner un algorithme efficace. En effet, comme on l'a souligné précédemment, la complétion sans approximation ne converge pas en général.

3.2 Algorithme

Considérons donc un algorithme naïf, en deux étapes, consistant simplement en la complétion de l'automate (sans ajout d'informations supplémentaires) puis en l'application inverse des règles de réécriture apparaissant dans les paires critiques sur le terme cible.

Soit `t` le terme cible, `orig` le terme de départ, `A` l'automate initial et `S` un accumulateur pour la stratégie calculée (exprimée sous forme de liste).

On utilisera les fonctions suivantes, supposées définies :

- `complete(A[i]))` qui renvoie l'automate construit grâce à un pas de complétion à partir de `A[i]`
- `reach(A[i],t)` qui renvoie la règle de réécriture utilisée par la complétion pour créer la transition $t \to q_0$, avec $orig \to_A^* q_0$
- `inverse(regle,t)` qui renvoie le terme t' obtenu par réécriture inverse : $t' \to_{regle} t$

```
i = 0;
A[0] = A;
while (!(t in A[i]) and !(A[i] == A[i-1])) {
   A[i+1] = complete(A[i]);
   i = i+1;
```

```
}
if(!(t in A[i]))
   return nil;
while (t != orig) {
   regle = reach(A[i],t)
   t = inverse(regle,t)
   S = (regle).S
}
return S;
```

3.3 Exemple détaillé

3.3.1 Énoncé

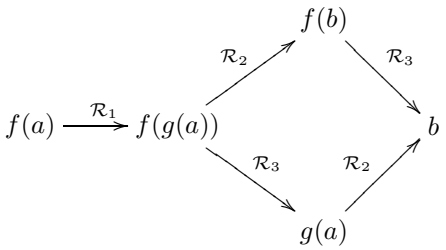
Considérons le système de réécriture suivant :

$$\mathcal{R} = \begin{cases} f(x) \to f(g(x)) & (\mathcal{R}_1) \\ g(a) \to b & (\mathcal{R}_2) \\ f(x) \to x & (\mathcal{R}_3) \end{cases}$$

On cherche à déterminer si $f(a) \to^*_{\mathcal{R}} b$

3.3.2 Solutions

Les solutions (minimales) sont les suivantes :



L'objectif que l'on se fixe est donc de retrouver l'une au moins de ces deux solutions minimales à la fin de nos manipulations.

3.3.3 Automate initial

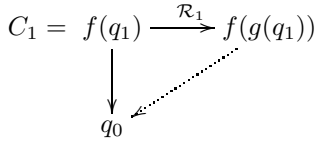
$$\begin{cases} a \to q_1 \\ f(q_1) \to q_0 \end{cases}$$

q_0 est donc implicitement l'état chargé de reconnaître l'ensemble des termes accessibles par la relation de réécriture $\to^*_{\mathcal{R}}$

3.3.4 Méthode exacte

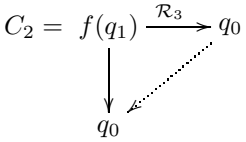
Étapes de complétion

Première étape



La normalisation de $f(g(q_1))$ sans approximation demande la création d'un nouvel état q_{new_0} et des transitions :

- $g(q_1) \to q_{new_0}$
- $f(q_{new_0}) \to q_0$

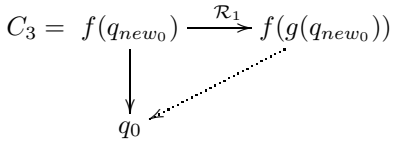


- $q_1 \to q_0$

Les transitions ciblant q_1 sont donc «redirigées» vers q_0 :

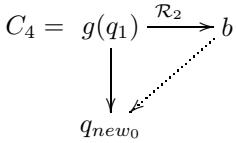
$$a \to q_0$$

Deuxième étape

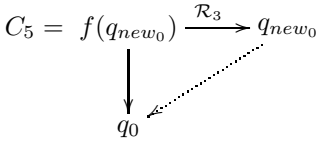


La normalisation de $f(g(q_{new_0})) \to q_0$ sans approximation demande la création d'un nouvel état q_{new_1} et des transitions :

- $g(q_{new_0}) \to q_{new_1}$
- $f(q_{new_1}) \to q_0$



- $b \to q_{new_0}$



– $q_{new_0} \to q_0$

Les transitions ciblant q_{new_0} sont donc «redirigées» vers q_0 :

$$\begin{cases} g(q_1) \to q_0 \\ b \to q_0 \end{cases}$$

Troisième étape

$$C_6 = \begin{array}{ccc} f(q_{new_1}) & \xrightarrow{\mathcal{R}_1} & f(g(q_{new_1})) \\ \downarrow & \swarrow & \\ q_0 & & \end{array}$$

La normalisation de $f(g(q_{new_1})) \to q_0$ sans approximation demande la création d'un nouvel état q_{new_2} et des transitions :

– $g(q_{new_1}) \to q_{new_2}$
– $f(q_{new_2}) \to q_0$

$$C_7 = \begin{array}{ccc} g(q_1) & \xrightarrow{\mathcal{R}_2} & b \\ \downarrow & \swarrow & \\ q_0 & & \end{array}$$

– $b \to q_0$

$$C_8 = \begin{array}{ccc} f(q_{new_1}) & \xrightarrow{\mathcal{R}_3} & q_{new_1} \\ \downarrow & \swarrow & \\ q_0 & & \end{array}$$

– $q_{new_1} \to q_0$

Les transitions ciblant q_{new_1} sont donc «redirigées» vers q_0 :

$$g(q_{new_0}) \to q_0$$

Suite ...

Étant donné que le terme est atteint, on ne détaillera pas la suite de la complétion. Par ailleurs, comme la complétion est sans approximation, il est possible de s'arrêter avant d'atteindre un automate complet puisque l'on sait que le terme est effectivement atteignable par réécriture. Dans le pire des cas, on risque simplement de ne pas trouver certains chemins de réécriture menant au terme, mais ceux-ci seraient nécessairement non-optimaux.

Construction de la stratégie

$$S = ()$$

– $f(a) \rightarrow^*_{\mathcal{R}} b\,?$
Les transitions de l'automate permettant la reconnaissance de b en q_0 sont les suivantes (directe, et par identification)

$$b \rightarrow q_0 \quad ou \quad q_{new_0} \rightarrow q_0$$

La première est triviale, et la transition $b \rightarrow q_{new_0}$ provenant de C_4 a engendré un $b \rightarrow q_0$ grace à la seconde.

$$C_5 : \mathcal{R}_3^{-1}(b, \epsilon) = f(b)$$

$$C_7 : \mathcal{R}_2^{-1}(b, \epsilon) = g(a)$$

d'où la stratégie partielle :

$$S = \{(\mathcal{R}_3, \epsilon) \mid (\mathcal{R}_2, \epsilon)\}.()$$

– $f(a) \rightarrow^*_{\mathcal{R}} f(b)\,?$
Une seule solution dans l'automate :

$$\begin{cases} b \rightarrow q_{new_0} \\ f(q_{new_0}) \rightarrow q_0 \end{cases}$$

Or pour obtenir la transition $b \rightarrow q_{new_0}$ il a été nécessaire d'appliquer $\mathcal{C}_4$

$$\mathcal{C}_4 : \mathcal{R}_2^{-1}(f(b), 1.\epsilon) = f(g(a))$$

On peut donc compléter $\mathcal{S}$ de la façon suivante :

$$S = \{(\mathcal{R}_2, 1.\epsilon).(\mathcal{R}_3, \epsilon) \mid (\mathcal{R}_2, \epsilon)\}.()$$

– $f(a) \rightarrow^*_{\mathcal{R}} g(a)\,?$

$$\begin{cases} g(q_1) \rightarrow q_{new_0} \\ q_{new_0} \rightarrow q_0 \end{cases}$$

$$S = \{(\mathcal{R}_2, 1.\epsilon).(\mathcal{R}_3, \epsilon) \mid (\mathcal{R}_3, \epsilon).(\mathcal{R}_2, \epsilon)\}.()$$

$$\mathcal{R}_2^{-1}(g(a), \epsilon) = f(g(a))$$

– $f(a) \rightarrow^*_{\mathcal{R}} f(g(a))\,?$

$$\begin{cases} g(q_1) \rightarrow q_{new_0} \\ f(q_{new_0}) \rightarrow q_0 \end{cases}$$

$$S = (\mathcal{R}_1, \epsilon).\{(\mathcal{R}_2, 1.\epsilon).(\mathcal{R}_3, \epsilon) \mid (\mathcal{R}_3, \epsilon).(\mathcal{R}_2, \epsilon)\}.()$$

$$\mathcal{R}_2^{-1}(f(g(a)), \epsilon) = f(a)$$

– $f(a) \rightarrow_{\mathcal{R}}^{*} f(a)$?
ok.

$$S = (\mathcal{R}_1, \epsilon).\{(\mathcal{R}_2, 1.\epsilon).(\mathcal{R}_3, \epsilon) \mid (\mathcal{R}_3, \epsilon).(\mathcal{R}_2, \epsilon)\}.()$$

correspond bien à l'ensemble des solutions déterminées en 3.3.2

Commentaires

Dans cette méthode, une grande quantité du travail de réécriture est en quelque sorte effectuée à deux reprises : à la fois lors de la complétion, et lors de la reconstruction de la stratégie. Les termes successifs sont recalculés pour reconstruire le chemin de réécriture emprunté, ce qui ne devrait pas être nécessaire à condition de conserver davantage d'information au cours de l'algorithme de complétion.

Par ailleurs, en introduisant des approximations dans les étapes de complétion, il ne sera plus possible de compter sur la forte correspondance entre un terme et un état (toute approximation a pour effet de similariser des termes distincts dans un même état) pour inverser aussi simplement le chemin de réécriture suivi.

Chapitre 4

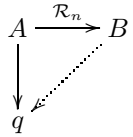
Construction des dépendances

La première étape pour la construction de la stratégie est de déterminer quelles sont les dépendances entre les paires critiques successivement construites.

4.1 Algorithme

4.1.1 Complétion

La complétion s'effectue par calcul itéré de paires critiques de la forme



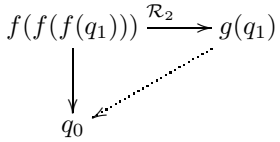
où B est le résultat de l'application de la règle de réécriture $\mathcal{R}_n$ au terme A, et qui permet d'atteindre l'état q à partir de B.

Notation 1. *Pour une telle paire critique C on notera :*
- $left(C)$ *le terme A*
- $right(C)$ *le terme B*
- $rule(C)$ *la règle de réécriture $\mathcal{R}_n$*
- $target(C)$ *l'état cible q*

Définition 16 (Ensemble des pré-requis). *Soit C une paire critique. Soit $\mathcal{E}$ l'ensemble des transitions de l'automate considéré à l'étape de complétion courante. On notera $pre(C)$ l'ensemble des règles de transition de l'automate qui rendent envisageable C.*

$$\begin{cases} pre(C) \subseteq \mathcal{E} \\ left(C) \rightarrow^*_{pre(C)} target(C) \end{cases}$$

Exemple 8. *Soit la paire critique C*



On aura par exemple (suivant l'automate)

$$pre(C) = \{f(q_1) \to q_0, f(q_0) \to q_0\}$$

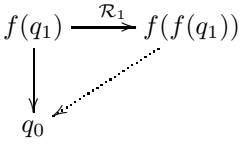
car

$$\begin{aligned} f(f(f(q_1))) &\to f(f(q_0)) \\ &\to f(q_0) \\ &\to q_0 \end{aligned}$$

Définition 17 (Ensemble des conséquences). *Soit C une paire critique. On notera $post(C)$ l'ensemble des règles de transition de l'automate qui sont créées par C.*

$$\begin{cases} post(C) \subseteq \mathcal{E} \\ right(C) \to^*_{post(C)} target(C) \end{cases}$$

Exemple 9. *Soit la paire critique C*



On aura par exemple (suivant les approximations)

$$post(C) = \{f(q_1) \to q_0, f(q_0) \to q_0\}$$

car

$$\begin{aligned} f(f(q_1)) &\to f(q_0) \\ &\to q_0 \end{aligned}$$

Définition 18 (Réciproques). *On définit la fonction $post^{-1}$ sur l'ensemble des règles de transitions :*

$$post^{-1}(r) = \{C \mid r \in post(C)\}$$

Définition 19 (Dépendance de paires critiques). *Soit C une paire critique.*

$$dep(C) = \{c \in post^{-1}(r) \mid r \in pre(C)\}$$

On peut voir l'ensemble des dépendances d'une paire critique C comme l'ensemble des paires critiques nécessaires à la construction de C

4.2 Exemple 2

4.2.1 Énoncé

Considérons le système de réécriture suivant :

$$\mathcal{R} = \begin{cases} f(x) \to f(f(x)) & (\mathcal{R}_1) \\ f(f(f(x))) \to g(x) & (\mathcal{R}_2) \\ f(g(x)) \to b & (\mathcal{R}_3) \end{cases}$$

On cherche à déterminer si $f(a) \to_{\mathcal{R}}^{*} b$

4.2.2 Automate initial

$$\begin{cases} a \to q_1 \\ f(q_1) \to q_0 \end{cases}$$

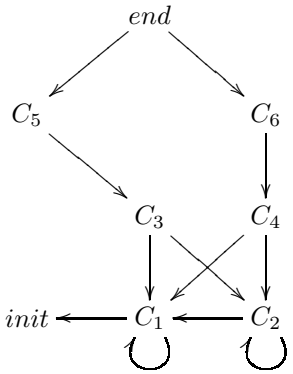
4.2.3 Déroulement de l'algorithme

Sur chacune des lignes, le champs «Requirements» correspond au calcul de $pre(C)$ pour la paire critique courante C. «Dependencies Map» associe à chaque transition t de l'automate l'ensemble $post^{-1}(t)$.

Step	Critical Pair	Requirements	Dependencies Map
0	–	–	$a \to q_1$ $\{init\}$ $f(q_1) \to q_0$ $\{init\}$
1	$\mathcal{C}_1 = f(q_1) \xrightarrow{\mathcal{R}_1} f(f(q_1))$ $\downarrow \quad \swarrow$ q_0	$f(q_1) \to q_0$	$a \to q_1$ $\{init\}$ $f(q_1) \to q_0$ $\{init, \mathcal{C}_1\}$ $f(q_0) \to q_0$ $\{\mathcal{C}_1\}$
2	$\mathcal{C}_2 = f(q_0) \xrightarrow{\mathcal{R}_1} f(f(q_0))$ $\downarrow \quad \swarrow$ q_0	$f(q_0) \to q_0$	$a \to q_1$ $\{init\}$ $f(q_1) \to q_0$ $\{init, \mathcal{C}_1\}$ $f(q_0) \to q_0$ $\{\mathcal{C}_1, \mathcal{C}_2\}$
2	$\mathcal{C}_3 = f(f(f(q_1))) \xrightarrow{\mathcal{R}_2} g(q_1)$ $\downarrow \quad \swarrow$ q_0	$f(q_1) \to q_0$ $f(q_0) \to q_0$	$a \to q_1$ $\{init\}$ $f(q_1) \to q_0$ $\{init, \mathcal{C}_1\}$ $f(q_0) \to q_0$ $\{\mathcal{C}_1, \mathcal{C}_2\}$ $g(q_1) \to q_0$ $\{\mathcal{C}_3\}$
2	$\mathcal{C}_4 = f(f(f(q_0))) \xrightarrow{\mathcal{R}_2} g(q_0)$ $\downarrow \quad \swarrow$ q_0	$f(q_0) \to q_0$	$a \to q_1$ $\{init\}$ $f(q_1) \to q_0$ $\{init, \mathcal{C}_1\}$ $f(q_0) \to q_0$ $\{\mathcal{C}_1, \mathcal{C}_2\}$ $g(q_1) \to q_0$ $\{\mathcal{C}_3\}$ $g(q_0) \to q_0$ $\{\mathcal{C}_4\}$
3	$\mathcal{C}_5 = f(g(q_1)) \xrightarrow{\mathcal{R}_3} b$ $\downarrow \quad \swarrow$ q_0	$g(q_1) \to q_0$ $f(q_0) \to q_0$	$a \to q_1$ $\{init\}$ $f(q_1) \to q_0$ $\{init, \mathcal{C}_1\}$ $f(q_0) \to q_0$ $\{\mathcal{C}_1, \mathcal{C}_2\}$ $g(q_1) \to q_0$ $\{\mathcal{C}_3\}$ $g(q_0) \to q_0$ $\{\mathcal{C}_4\}$ $b \to q_0$ $\{\mathcal{C}_5\}$
3	$\mathcal{C}_6 = f(g(q_0)) \xrightarrow{\mathcal{R}_3} b$ $\downarrow \quad \swarrow$ q_0	$g(q_0) \to q_0$ $f(q_0) \to q_0$	$a \to q_1$ $\{init\}$ $f(q_1) \to q_0$ $\{init, \mathcal{C}_1\}$ $f(q_0) \to q_0$ $\{\mathcal{C}_1, \mathcal{C}_2\}$ $g(q_1) \to q_0$ $\{\mathcal{C}_3\}$ $g(q_0) \to q_0$ $\{\mathcal{C}_4\}$ $b \to q_0$ $\{\mathcal{C}_5, \mathcal{C}_6\}$

4.2.4 Dépendances produites

Il est possible de produire le diagramme de dépendances suivant à l'aide de la relation $dep(C)$



Les éléments *end* et *init* représentent respectivement les points d'entrée (en l'occurrence la transition cible $b \rightarrow q_0$, les flèches sortantes matérialisant $post^{-1}(b \rightarrow q_0)$) et de sortie (successeur des paires critiques sans dépendances)

Malheureusement cette construction se révèle insuffisante pour fournir une stratégie exacte. (l'exemple 5.6 en particulier verrait une telle stratégie échouer)

Chapitre 5

Ordre des dépendances

5.1 Problèmes

Le calcul des dépendances entre les paires critiques ne suffit pas à fournir une stratégie de réécriture. Nous avons besoin d'ordonner ces dépendances au maximum.

5.2 Structures de stratégies

On peut exprimer relativement finement les stratégies calculées à l'aide de deux structures :
- choix
- séquence

5.3 Algorithme

5.3.1 Structures

Définition 20 (Terme de stratégie). *Soit $\mathcal{L}$ le langage de termes considéré. Un terme de stratégie $\mathcal{T}$ est une fonction :*

$$\begin{array}{rl} \mathcal{T} : & \mathcal{L} \to \mathcal{L} \cup \{\bot\} \\ & t \mapsto \mathcal{T}(t) \end{array}$$

Un terme de stratégie correspond à l'application d'une règle de réécriture à un terme. En cas d'échec (règle inapplicable) $\bot$ est renvoyé.

Nous verrons dans la suite comment les termes de stratégie son exprimés en fonction des étapes de complétion.

Notation 2. *On notera Id le terme de stratégie :*

$$\begin{array}{rl} Id : & \mathcal{L} \to \mathcal{L} \\ & t \mapsto t \end{array}$$

La structure suivante permet de combiner les termes de stratégie, de façon éventuellement non-déterministe.

Définition 21 (Stratégie). *Soit $\mathcal{L}$ le langage de termes considéré. Une stratégie $\mathcal{S}$ est une fonction :*

$$\begin{aligned} \mathcal{S} : \quad & \mathcal{L} \to \mathcal{P}(\mathcal{L}) \\ & t \mapsto \mathcal{S}(t) \end{aligned}$$

Une stratégie inapplicable (dans laquelle il n'existe pas de chemin applicable) renvoie $\emptyset$

Remarque 2. *Par abus de notation, on écrira $\mathcal{T}$ pour désigner à la fois le terme de stratégie et la stratégie composée de cet unique terme.*

5.3.2 Opérateurs

Définition 22. *L'opérateur de concaténation . opérant sur deux stratégies $\mathcal{S}_1$ et $\mathcal{S}_2$ est l'analogue de la composition $\times$. Il est défini comme l'application de $\mathcal{S}_2$ au résultat de l'application de $\mathcal{S}_1$.*

Exemple 10. *Soient deux stratégies $\mathcal{S}_1$ et $\mathcal{S}_2$ et un terme t*

$$\mathcal{S}_1(t) = \{t_1, \ldots, t_n\} \implies (\mathcal{S}_1.\mathcal{S}_2)(t) = \mathcal{S}_2(t_1) \cup \ldots \cup \mathcal{S}_2(t_n)$$

Définition 23. *L'opérateur de choix $\vee$ opérant sur deux stratégies $\mathcal{S}_1$ et $\mathcal{S}_2$ est défini comme l'union des résultats des applications de $\mathcal{S}_1$ et $\mathcal{S}_2$.*

Exemple 11. *Soient les stratégies $\mathcal{S}_1$ et $\mathcal{S}_2$ alors*

$$\mathcal{S}_1 \vee \mathcal{S}_2 = (t \mapsto \mathcal{S}_1(t) \cup \mathcal{S}_2(t))$$

Définition 24. *L'opérateur de composition $\wedge$ opérant sur deux stratégies $\mathcal{S}_1$ et $\mathcal{S}_2$ est défini comme l'ensemble des arrangements possibles des résultats des applications de $\mathcal{S}_1$ et $\mathcal{S}_2$.*

Il vérifie les axiomes suivants :

$$\begin{cases} \mathcal{S}_1 \wedge \mathcal{S}_2 = \mathcal{S}_2 \wedge \mathcal{S}_1 & (1) \\ \mathcal{S}_1 \wedge Id = \mathcal{S}_1 & (2) \\ \mathcal{S}_1 \wedge \mathcal{S}_2 = \bigvee_{i \in I} ((\mathcal{S}_i.\mathcal{S}_1^{'}) \wedge \mathcal{S}_2) & (3) \\ \quad si\ \mathcal{S}_1 = (\bigvee_{i \in I} \mathcal{S}_i).\mathcal{S}_1^{'} & \\ \mathcal{S}_1 \wedge \mathcal{S}_2 = (\mathcal{T}_1.(\mathcal{S}_1^{'} \wedge \mathcal{S}_2)) \vee (\mathcal{T}_2.(\mathcal{S}_1 \wedge \mathcal{S}_2^{'})) & (4) \\ \quad si \begin{cases} \mathcal{S}_1 = \mathcal{T}_1.\mathcal{S}_1^{'} \\ \mathcal{S}_2 = \mathcal{T}_2.\mathcal{S}_2^{'} \end{cases} & \\ \mathcal{T} \wedge \mathcal{T} = \mathcal{T} \vee (\mathcal{T}.\mathcal{T}) & (5) \end{cases}$$

Remarque 3. *On a défini deux opérateurs commutatifs $\vee$ et $\wedge$ avec $\wedge$ distributif par rapport à $\vee$*

Proposition 2. *Il existe toujours une forme normale disjonctive pour toute expression construite à partir des opérateurs $\vee$, wedge et .*

Démonstration. Cette proposition peut être démontrée en remarquant que les opérations (2), (3), (4) et (5) permettent toujours de faire diminuer le nombre d'occurrences de $\wedge$ ou le niveau hiérarchique du $\wedge$ situé le plus haut dans l'arbre de l'expression. □

Définition 25. *On notera $\mathcal{S} \lhd \mathcal{S}^{'}$ si :*

$$\begin{cases} \mathcal{S}^{'} = \mathcal{S} \vee \mathcal{S}^{''} \\ \mathcal{S} = \mathcal{T}_1 . \ldots . \mathcal{T}_n \end{cases}$$

où $\mathcal{T}_1 \ldots \mathcal{T}_n$ sont des termes de stratégies

Exemple 12. *Soient les stratégies*

$$\begin{cases} \mathcal{S}_1 = \mathcal{T}_0 . \mathcal{T}_1 \\ \mathcal{S}_2 = \mathcal{T}_2 . \mathcal{T}_3 \end{cases}$$

$$\begin{aligned} \mathcal{S}_1 \wedge \mathcal{S}_2 = \; & \mathcal{T}_0 . \mathcal{T}_1 . \mathcal{T}_2 . \mathcal{T}_3 \vee \mathcal{T}_0 . \mathcal{T}_2 . \mathcal{T}_1 . \mathcal{T}_4 \vee \\ & \mathcal{T}_0 . \mathcal{T}_2 . \mathcal{T}_4 . \mathcal{T}_1 \vee \mathcal{T}_2 . \mathcal{T}_4 . \mathcal{T}_0 . \mathcal{T}_1 \vee \\ & \mathcal{T}_2 . \mathcal{T}_0 . \mathcal{T}_4 . \mathcal{T}_1 \vee \mathcal{T}_2 . \mathcal{T}_0 . \mathcal{T}_1 . \mathcal{T}_4 \end{aligned}$$

$$et \; (\mathcal{T}_0 . \mathcal{T}_1 . \mathcal{T}_2 . \mathcal{T}_3) \lhd (\mathcal{S}_1 \wedge \mathcal{S}_2)$$

5.3.3 Opérations de base

Définition 26. *Soit $\mathcal{C}$ une paire critique, on notera $Term(\mathcal{C})$ le terme de stratégie correspondant, à savoir la paire $(\mathcal{R}, pos)$. Où $\mathcal{R}$ est la règle de réécriture de la paire critique, et pos la position d'application de cette règle .*

Définition 27. *Soit a une transition de l'automate complété. La stratégie $\mathcal{S}_a$ menant à la construction de cette transition est définie par :*

$$\mathcal{S}_a = \bigvee_{\mathcal{C} \in pre(a)} ((\bigwedge_{t \in req(\mathcal{C})} \mathcal{S}_t) . Term(\mathcal{C}))$$

Proposition 3. *Soient t et T deux termes, q_0 un état de l'automate initial $\mathcal{A}$ tel que $t \rightarrow_{\mathcal{A}}^{*} q_0$, $\mathcal{X}$ l'ensemble des règles de transition de l'automate complété. Soit $\mathcal{A}$ l'ensemble des parties $\mathcal{E}$ de $\mathcal{X}$ vérifiant :*

$$\begin{cases} T \rightarrow_{\mathcal{E}}^{*} q_0 \\ \forall \mathcal{E}^{'} \subset \mathcal{E}, T \not\rightarrow_{\mathcal{E}'}^{*} \end{cases}$$

La stratégie susceptible de permettre d'atteindre T à partir de t est la suivante (si elle existe) :

$$\mathcal{S} = \bigvee_{\mathcal{E} \in \mathcal{A}} (\bigwedge_{a \in \mathcal{E}} \mathcal{S}_a)$$

De plus cette stratégie fournit une chaîne de réécriture de longueur optimale.

Remarque 4. *Cette dernière étape est rendue nécessaire par le fait que la* $T \rightarrow^* q_0$ *nécessite un ensemble* $\mathcal{E}$ *de transitions de l'automate et qu'il existe plusieurs choix possibles pour* $\mathcal{E}$

Démonstration. À montrer : si il existe une chaîne de termes de stratégies $\mathcal{Z} = \mathcal{T}_1 \ldots . \mathcal{T}_n$ de longueur minimale n telle que $T \in \mathcal{Z}(t)$, alors :

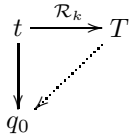
$$\begin{cases} T \in \mathcal{S}(t) \\ \exists \mathcal{Z}^{'} \lhd \mathcal{S} \text{ t.q. } |\mathcal{Z}^{'}| = n \text{ et } T \in \mathcal{Z}^{'}(t) \end{cases}$$

Procédons par récurrence sur la taille de la plus courte séquence de réécriture menant au résultat souhaité.

Pour $n = 0$, on a T trivialement.

Pour $n = 1$: soit $\mathcal{R}_k$ la règle réécriture telle que $t \rightarrow_{\mathcal{R}_k} T$ L'automate de départ est construit de telle sorte que $t \rightarrow q_0$

À la première étape de complétion on aura donc la paire critique $\mathcal{C}$



avec $T \rightarrow q_0$ normalisée sous la forme d'un système de transitions $\{t_1, \ldots, t_m\}$ et un ensemble vide de pré-requis.

Considérons l'ensemble $\mathcal{E} = \{t_1, \ldots, t_m\}$. $\mathcal{E}$ vérifie bien les hypothèses par définition.

$\forall e \in \mathcal{E}, \mathcal{C} \in pre(e)$ et $\mathcal{S}_e = Term(\mathcal{C})$ donc la stratégie construite contient bien la séquence réduite au seul terme de stratégie $Term(\mathcal{C})$

Supposons la propriété vraie pour n et T accessible en $n+1$ étapes. Alors $\exists t^{'}, t \rightarrow_{\mathcal{R}} t^{'} \rightarrow_{\mathcal{R}}^{\{n\}} T$

Il existe donc un chemin $\mathcal{C}$ de longueur 1 qui mène à $t^{'}$ Soit la stratégie $\mathcal{S}^{'} = \bigvee_{\mathcal{C}.\mathcal{S}^{''} \in \mathcal{S}} \mathcal{S}^{''}\ \mathcal{S}^{'}$ est une stratégie basée sur le terme initial $t^{'}$ plus riche que celle qui aurait été construite directement. Donc il existe un mot $\mathcal{Z}$ de longueur n dans $\mathcal{S}^{'}$ qui mène de $t^{'}$ à T.

$$\begin{cases} (\mathcal{C}.\mathcal{Z})(t) = T \\ |\mathcal{C}.\mathcal{Z}| = n+1 \end{cases}$$

□

Remarque 5. *L'utilisation intensive de l'opérateur* $\wedge$ *se justifie par le fait que l'on n'a quasiment aucun contrôle sur l'ordre dans lequel les opérations doivent se dérouler. On pourrait être tenté de penser que la relation de dépendance induit une relation d'ordre sur l'ensemble des paires critiques, mais l'information porte en fait sur une série de relations d'ordre partiel. Ainsi les dépendances devront précéder les conséquences, mais pourront être rejetées dans un passé plus «lointain».*

5.4 Élagage des branches incohérentes

La suppression des transitions «impossibles» entre termes de stratégies permettrait dans la majorité des cas de s'affranchir d'une partie non négligeable de la stratégie construite précédemment. Reste à caractériser les chaînes incohérentes.

Proposition 4. *Tout chemin dans la stratégie $\mathcal{S}$ présentant un saut en avant dans les étapes de complétion associées à chaque paire critique (champ «Step») peut être écarté.*

Démonstration. La validité de cette simplification est donnée par le fait que la réécriture d'un terme durant la complétion se fait toujours le plus tôt possible. Par conséquent générer une séquences contenant un couple de paires critiques $\mathcal{C}_i.\mathcal{C}_j$ tel que $\mathcal{C}_i$ a eu lieu lors d'une étape m et $\mathcal{C}_j$ lors d'une étape n, avec $m < n + 1$, ne peut provenir que :

- de la complétion elle-même à cause d'une approximation ;
- d'un parallélisme inhérent au terme considéré :
  ainsi un terme de la forme $g(X, Y)$ pourra, sans dommage, intercaler les paires critiques liées à X et celles liées à Y. Cependant, grâce à cette indépendance, il existe un ordonnancement équivalent qui respecte la propriété.

□

Exemple 13. *La séquence $\mathcal{C}_1.\mathcal{C}_1.\mathcal{C}_1.\mathcal{C}_3.\mathcal{C}_2.\mathcal{C}_1.\mathcal{C}_5$ appartient au langage calculé, mais est une chaîne invalide car $\mathcal{C}_1$ provient de l'étape de complétion 1 tandis que $\mathcal{C}_5$ provient de l'étape de complétion 3.*

5.5 Algorithme

Algorithme 2. *Étapes de l'algorithme :*

- *Construction de l'automate initial ;*
- *Complétion de l'automate ;*
- *Calcul de la stratégie induite ;*
- *Élagage des branches incohérentes (optionnel, mais influe grandement sur l'efficacité de la stratégie produite).*

5.6 Exemple

Dans l'exemple qui suit, nous nous plaçons volontairement dans le pire des cas de figures envisageables : une approximation maximale. Aucun nouvel état ne sera créé lors de la complétion et tous les termes seront normalisés vers q_0. En pratique, ce serait une mauvaise manière de rechercher une stratégie adéquate, car celle-ci sera nécessairement volumineuse.

5.6.1 Énoncé

Considérons le système de réécriture suivant :

$$\mathcal{R} = \begin{cases} f(x) \to f(f(x)) & (\mathcal{R}_1) \\ f(f(a)) \to b & (\mathcal{R}_2) \\ f(f(f(x))) \to g(x, f(x)) & (\mathcal{R}_3) \\ f(g(f(a), b)) \to c & (\mathcal{R}_4) \end{cases}$$

On cherche à déterminer si $f(a) \to^*_{\mathcal{R}} b$

5.6.2 Automate initial

$$\begin{cases} a \to q_1 \\ f(q_1) \to q_0 \end{cases}$$

5.6.3 Complétion

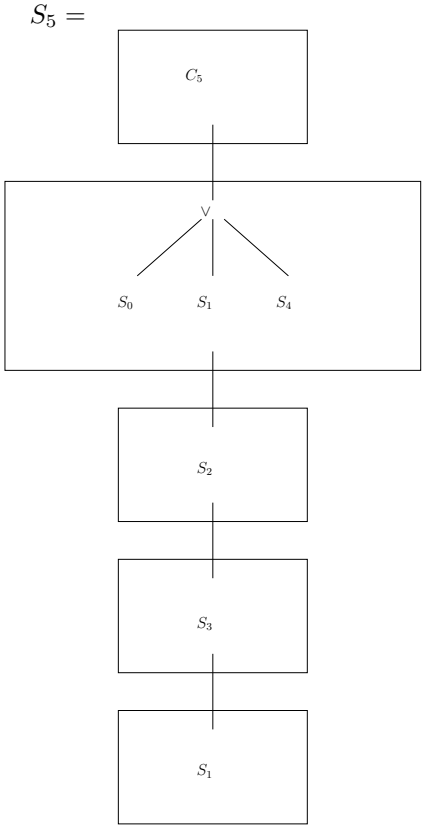
Step	Critical Pair	Requirements	Dependencies Map
0	–	–	$a \to q_1$ $\{init\}$ $f(q_1) \to q_0$ $\{init\}$
1	$\mathcal{C}_1 = f(q_1) \xrightarrow{\mathcal{R}_1} f(f(q_1))$ $\downarrow$ / $\searrow$ q_0	$f(q_1) \to q_0$	$a \to q_1$ $\{init\}$ $f(q_1) \to q_0$ $\{init, \mathcal{C}_1\}$ $f(q_0) \to q_0$ $\{\mathcal{C}_1\}$
2	$\mathcal{C}_2 = f(f(q_1)) \xrightarrow{\mathcal{R}_2} b$ $\downarrow$ / $\searrow$ q_0	$f(q_1) \to q_0$ $f(q_0) \to q_0$	$a \to q_1$ $\{init\}$ $f(q_1) \to q_0$ $\{init, \mathcal{C}_1\}$ $f(q_0) \to q_0$ $\{\mathcal{C}_1\}$ $b \to q_0$ $\{\mathcal{C}_2\}$
2	$\mathcal{C}_3 = f(f(f(q_0))) \xrightarrow{\mathcal{R}_3} g(q_0, f(q_0))$ $\downarrow$ / $\searrow$ q_0	$f(q_0) \to q_0$	$a \to q_1$ $\{init\}$ $f(q_1) \to q_0$ $\{init, \mathcal{C}_1\}$ $f(q_0) \to q_0$ $\{\mathcal{C}_1, \mathcal{C}_3\}$ $b \to q_0$ $\{\mathcal{C}_2\}$ $g(q_0, q_0) \to q_0$ $\{\mathcal{C}_3\}$
2	$\mathcal{C}_4 = f(f(f(q_1))) \xrightarrow{\mathcal{R}_3} g(q_1, f(q_1))$ $\downarrow$ / $\searrow$ q_0	$f(q_1) \to q_0$ $f(q_0) \to q_0$	$a \to q_1$ $\{init\}$ $f(q_1) \to q_0$ $\{init, \mathcal{C}_1, \mathcal{C}_4\}$ $f(q_0) \to q_0$ $\{\mathcal{C}_1, \mathcal{C}_3, \mathcal{C}_4\}$ $b \to q_0$ $\{\mathcal{C}_2\}$ $g(q_0, q_0) \to q_0$ $\{\mathcal{C}_3\}$ $g(q_1, q_0) \to q_0$ $\{\mathcal{C}_4\}$
3	$\mathcal{C}_5 = f(g(f(q_1), b)) \xrightarrow{\mathcal{R}_4} c$ $\downarrow$ / $\searrow$ q_0	$b \to q_0$ $f(q_1) \to q_0$ $g(q_0, q_0) \to q_0$ $f(q_0) \to q_0$	$a \to q_1$ $\{init\}$ $f(q_1) \to q_0$ $\{init, \mathcal{C}_1, \mathcal{C}_4\}$ $f(q_0) \to q_0$ $\{\mathcal{C}_1, \mathcal{C}_3, \mathcal{C}_4\}$ $b \to q_0$ $\{\mathcal{C}_2\}$ $g(q_0, q_0) \to q_0$ $\{\mathcal{C}_3\}$ $g(q_1, q_0) \to q_0$ $\{\mathcal{C}_4\}$ $c \to q_0$ $\{\mathcal{C}_5\}$

5.6.4 Calcul de la stratégie

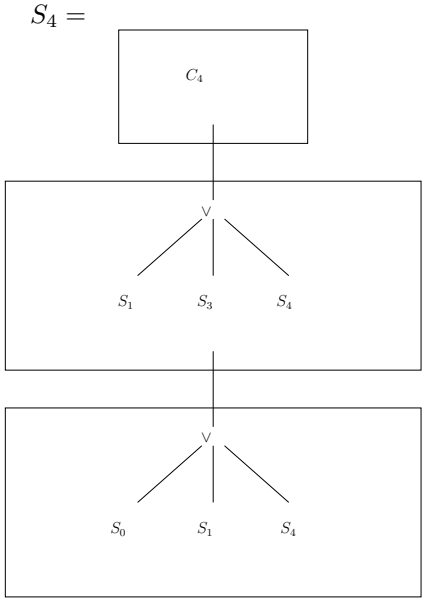
$$\begin{aligned}\mathcal{S} &= \mathcal{S}_{c \to q_0} \\ &= (\bigwedge_{t \in req(C_5)} \mathcal{S}_t).Term(C_5) \\ &= \dots\end{aligned}$$

Il est possible de mener le calcul graphiquement, en représentant les diagrammes de dépendance des paires critiques :

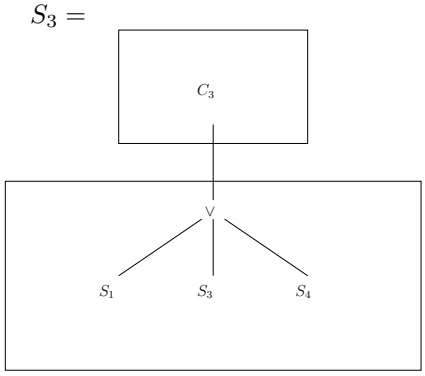
$S_5 =$



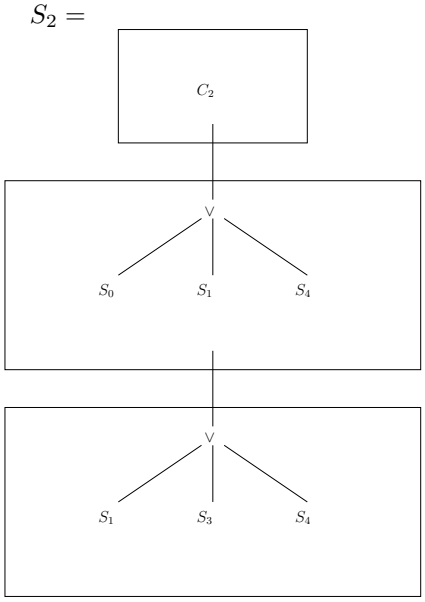
$S_4 =$



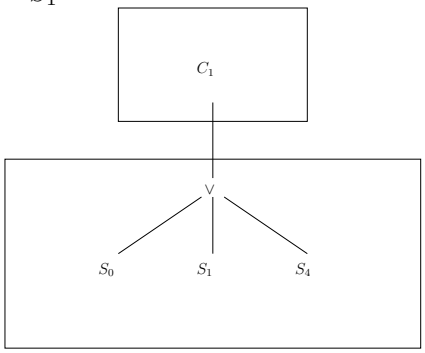
$S_3 =$



$S_2 =$



$S_1 =$



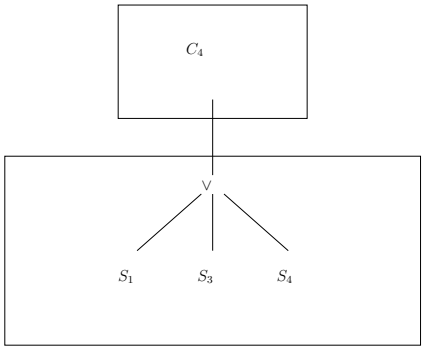
La stratégie calculée s'exprime donc sous la forme suivante :

- $S = S_5$
- $S_5 = \{(S_0 \vee S_1 \vee S_4) \wedge S_2 \wedge S_3 \wedge S_1\}\ .\ C_5$
- $S_4 = \{(S_1 \vee S_3 \vee S_4) \wedge (S_0 \vee S_1 \vee S_4)\}\ .\ C_4$
- $S_3 = \{S_1 \vee S_3 \vee S_4\}\ .\ C_3$
- $S_2 = \{(S_0 \vee S_1 \vee S_4) \wedge (S_1 \vee S_3 \vee S_4)\}\ .\ C_2$
- $S_1 = \{S_0 \vee S_1 \vee S_4\}\ .\ C_1$
- $S_0 = init = Id$

5.7 Retour sur l'élagage : optimisation

Une partie des artefacts dus aux approximations peut-être détectée assez tôt, au niveau de la construction du diagramme de dépendance des paires critiques.

Ainsi le diagramme suivant :



révélerait immédiatement une incohérence : $\mathcal{C}_4$ provenant de l'étape de l'étape 3 ne pourrait être précédé de $\mathcal{C}_1$ provenant de l'étape 1.

En conséquence, il peut être souhaitable d'inclure la construction du diagramme dans l'algorithme, et de découper la phase d'élagage en deux parties, l'une avant le calcul de la stratégie proprement dite, et l'autre après pour bénéficier de simplifications a priori difficilement détectables.

Conclusion

Nous avons proposé un algorithme qui, à l'aide de la complétion d'automates d'arbres, calcule une stratégie approchée permettant d'atteindre un terme donné si celui-ci est effectivement accessible.

L'intérêt de cette méthode est grandement conditionné par la phase de complétion, qui en est l'élément principal. D'une part, un faible nombre d'approximations rend la stratégie calculée plus précise car on se rapproche alors du cas exact évoqué en première partie. D'autre part une approximation plus grande rend la phase de complétion plus rapide, et donc plus praticable, mais réclame une politique de simplification efficace. Il conviendrait donc de trouver un juste milieu entre ces intérêts opposés.

Sans avoir véritablement étudié la question, il semble concevable de diriger l'algorithme de complétion de telle sorte qu'il se comporte de façon exacte en début de complétion («suffisamment» longtemps) puis qu'il achève la complétion par approximations. La motivation d'un tel comportement serait que faire des pas de complétion exacte sur un automate déjà approché ne permet pas de gain important, tandis que garder une base exacte aussi loin que possible restreindrait le faisceau de possibilités au départ de la stratégie produite.

Bibliographie

[1] Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. Tree automata techniques and applications, 2002.

[2] Nachum Dershowitz and David A. Plaisted. *Handbook of Automated Reasoning*. Alan Robinson and Andrei Voronkov, 2001.

[3] Guillaume Feuillade. Automates d'arbres pour l'approximation régulière des états accessibles dans les systèmes de réécriture conditionnels, Master thesis, Rennes 1, 2002.

[4] Thomas Genet. *Contraintes d'ordre et automates d'arbres pour les preuves de terminaison*. PhD thesis, nancyUHP, Nancy, 1998.

[5] Thomas Genet and Valérie Viet Triem Tong. Reachability analysis of term rewriting systems with timbuk.

[6] Sophie Tison. Tree automata and term rewriting. In *RPC'01*.

Table des matières