Automatisation de l'application de l'hypothèse de récurrence dans la preuve des formules implicatives

Inès Mouakher, Francis Alexandre, Khaled Bsaïes

Automatisation de l'application de l'hypothèse de récurrence dans la preuve des formules implicatives

I. Mouakher[1] **F. Alexandre**[2] **K. Bsaïes**[1]

[1] Faculté des Sciences de Tunis, DSI, Campus Universitaire 2092 Tunis, Tunisie
[2] LORIA BP 239, 54506 Vandoeuvre-lès-Nancy, France
ines_mouakher@yahoo.fr alexandr@loria.fr khaled.bsaies@fst.rnu.tn

Résumé

L'objectif général de ce travail est de prouver les propriétés des programmes logiques (ensemble de clauses de Horn). Ces propriétés sont des formules de la forme $\forall\bar{x}(\exists\bar{y}\ \Gamma \leftarrow \Delta)$ où Γ et Δ sont des conjonctions d'atomes. Nous disposons d'un ensemble de règles de déductions tels que le pliage, le dépliage et la simplification. La preuve consiste à appliquer l'une de ces règles sur la formule à prouver jusqu'à aboutir à un ensemble de formules triviales. Une étape essentielle dans le processus de la preuve est la réussite du pliage qui peut être vue comme l'application d'une hypothèse de récurrence dans une preuve inductive. Nous proposons des stratégies pour automatiser partiellement le processus de preuve en nous basant sur une analyse statique des formules et des programmes.

Abstract

The aim of this study is to prove the properties of the logical programs (set of Horn clauses). These properties are the implicative formulas of the form $\forall\bar{x}(\exists\bar{y}\ \Gamma \leftarrow \Delta)$ where Γ and Δ are conjunctions of atoms. We use deductive rules like fold, unfold and simplification. The proof consists in applying one of these rules on the formula to prove until we have a set of trivial formulas. An important step in an inductive proof attempt is the application of induction hypothesis. We propose some strategies to automate partially the process of proof by using a static analysis of the formulas and programs.

1 Introduction

La preuve par récurrence est une technique très puissante pour raisonner sur des structures récursives. Plusieurs systèmes et approches pour automatiser le processus de preuve par récurrence ont été proposés. Ces systèmes s'intéressent à la preuve de certaines classes de formules en se basant sur deux paradigmes : la récurrence explicite et la récurrence implicite [5, 4].

Nous considérons le problème de la preuve des propriétés des programmes logiques. Ces propriétés sont des formules de la logique des prédicats ayant la forme suivante : $\forall\bar{x}(\exists\bar{y}\ \Gamma \leftarrow \Delta)$. Ces formules sont assez particulières puisque très peu de systèmes se sont intéressés à l'automatisation de leur preuve à cause des variables existentielles. Ainsi, nous avons opté pour un système de preuve par pliage et dépliage [11, 3, 2] qui utilise l'induction non explicite pour prouver ces propriétés. Il a permis de prouver un grand nombre de formules implicatives non triviales, que peu de systèmes sont capables de traiter.

Ce système se base sur des règles de déduction telles que le pliage et le dépliage. Ces règles sont bien connues tant dans le domaine de la démonstration de théorèmes que dans celui de la transformation de programmes fonctionnels et logiques.

L'un des problèmes rencontré dans les systèmes de preuve est l'explosion combinatoire de la preuve et les retours arrière qui sont très coûteux. Pour cette raison, il est important de disposer d'heuristiques et de tactiques afin d'automatiser totalement ou partiellement le processus de preuve.

Prenons l'exemple des systèmes Clam et nqthm qui se basent sur la récurrence explicite. Une étape essentielle pour accomplir une preuve est le choix de la bonne règle de récurrence et par la suite le choix des règles à utiliser pour réussir à appliquer l'hypothèse de récurrence. En effet, ces systèmes effectuent une analyse de récursivité [4] pour choisir la règle de récurrence appropriée, ensuite ils utilisent des heuristiques

tels que le « rippling » pour Clam.

Dans le système de preuve par pliage-dépliage, le problème crucial est aussi lié à l'indéterminisme du processus de preuve. En d'autres termes, il construit les preuves pas à pas et nécessite alors une fréquente intervention de l'utilisateur. En effet, nous nous intéressons dans ce travail à la découverte des stratégies permettant de réduire l'indéterminisme et de réussir la preuve. Une étape essentielle dans le processus de la preuve est la réussite du pliage qui peut être vue comme l'application d'une hypothèse de récurrence dans une preuve inductive.

L'approche proposée est basée sur l'analyse statique de la formule et du programme considéré. Tout d'abord, nous nous limitons à une classe particulière de formules et de programmes et nous caractérisons les formules à traiter par des schémas. Nous menons ensuite une étude, en nous basant sur ces schémas, sur la réussite du pliage. Finalement, nous déterminons des conditions nécessaires pour la réussite du pliage. Cette étude permet de détecter soit la réussite de l'application du pliage en donnant d'une manière précise les différentes étapes à suivre, soit de détecter l'échec de leur application.

2 Préliminaires

Dans ce paragraphe nous présentons les définitions et les notations utilisées dans la suite. Ces concepts peuvent être trouvés avec plus de détails dans [9].

- Un programme logique défini est un ensemble de clauses définies.
- Une clause récursive linéaire est une clause de la forme $P(\overline{x}) \leftarrow \Delta, P(\overline{x}), \Gamma$, où $P(\overline{x})$ est un atome et Γ et Δ sont des conjonctions d'atomes qui ne contiennent pas le prédicat P.
- $\mathcal{M}(\mathcal{S})$ dénote le plus petit modèle de Herbrand du programme logique $\mathcal{S}$.
- $\mathcal{V}ar(F)$ dénote l'ensemble des variables d'une expression F.
- Les formules implicatives considérées sont de la forme $\forall \bar{x}(\exists \bar{y}\Gamma \leftarrow \Delta)$. On les note par $\Gamma \leftarrow \Delta$ où les variables universelles sont notées par des lettres minuscules et les variables existentielles par des lettres majuscules.
- Une formule implicative triviale s'écrit sous la forme : $\Delta \leftarrow faux$ ou $vrai \leftarrow \Gamma$. Elle se réduit trivialement à vrai.
- Une variable est interne à un membre (respectivement à une conjonction d'atome) dans une formule si cette variable n'apparaît pas ailleurs dans la formule.
- Une substitution θ est un ensemble fini de couples noté par $\{v_1/t_1, ..., v_n/t_n\}$, où $\forall i \in [1..n]$ v_i est une variable, t_i est un terme et $v_i \neq t_i$ et si $i \neq j$ alors $v_i \neq v_j$.
- L'image et le domaine d'une substitution θ sont respectivement : $Im(\theta) = \{t_1, ..., t_n\}$ et $Dom(\theta) = \{v_1, ..., v_n\}$.
- Une substitution existentielle est une substitution telle que son domaine est un ensemble de variables existentielles.
- pgu(E,E') dénote le plus général unificateur de E et E'.

3 Système de preuve par pliage/dépliage

3.1 Processus de preuve

Le système de preuve par pliage/dépliage considéré permet de prouver qu'une formule implicative est une propriété valide d'un programme. Autrement dit, Soit $\mathcal{S}$ un programme définit et π une formule, il permet de prouver $\mathcal{M}(\mathcal{S}) \models \pi$. La preuve consiste à appliquer pas à pas l'une des règles de déduction sur la formule à prouver. Ce qui génère un ensemble de formules à démontrer et ainsi de suite jusqu'à aboutir à un ensemble de formules triviales donc à une preuve de la formule initiale.

Ce système utilise un ensemble de règle de déduction : dépliage droit(NFI), dépliage gauche(DCI), pliage droit (CUT_R), pliage gauche(CUT_L), et simplification.

3.2 Règles de déduction

Nous commençons par présenter les deux règles de dépliage droit(NFI) et gauche(DCI) qui consistent à évaluer un atome d'une formule implicative.

Définition 3.1 ***(Dépliage droit(NFI))***
Soient $\mathcal{S}$ un programme défini, $\pi : \Gamma \leftarrow \Delta, A$ une formule implicative et $c_1, \cdots, c_k$ les clauses de $\mathcal{S}$ de la forme $c_j : B_j \leftarrow \Delta_j$ telles qu'il existe $\theta_j = pgu(B_j, A)$.

$$\begin{array}{rcl} < \pi : \Gamma & \leftarrow & \Delta, A > \\ & \downarrow & NFI \\ < \pi_j : (\Gamma & \leftarrow & \Delta, \Delta_j)\theta_j >,\ j \in [1, k] \end{array}$$

Les nouvelles variables introduites sont universelles.

Définition 3.2 ***(Dépliage gauche(DCI))***
Soient $\mathcal{S}$ un programme défini, $\pi : \Gamma, A \leftarrow \Delta$ une formule implicative et $c_1, \cdots, c_k$ les clauses de $\mathcal{S}$ de la forme $c_j : B_j \leftarrow \Delta_j$ telles qu'il existe une substitution existentielle $\theta_j = pgu(B_j, A)$.

$$\begin{array}{rcl} < \pi : \Gamma, A & \leftarrow & \Delta > \\ & \downarrow & DCI \\ < \pi_j : (\Gamma, \Delta_j)\theta_j & \leftarrow & \Delta >,\ j \in [1, k] \end{array}$$

Les nouvelles variables introduites sont existentielles.

Remarque 3.1 *Le dépliage d'un atome du membre gauche d'une formule est possible si seulement si les arguments contenant des variables universelles sont suffisamment instanciés. En effet, la substitution engendrée par les dépliages gauches doit être existentielle et ne doit pas modifier les variables universelles.*

Définition 3.3 ***(Pliage droit (CUT_ R))***
Considérons la branche de l'arbre de preuve suivante :

$$\begin{array}{c} < \pi_i : \Lambda \leftarrow \Sigma > \\ \vdots \\ \downarrow \\ < \pi_{i+n} : \Gamma \leftarrow \Delta_1, \Delta_2 > \\ \downarrow \qquad\qquad CUT\_R(\pi_i)^* \\ < \pi_{i+n+1} : \Gamma \leftarrow \Lambda\theta, \Delta_2 > \end{array}$$

*(* : pliage de π_i dans π_{i+n}, $n > 0$)*
avec :

1. $\Sigma\theta = \Delta_1$,
2. *θ substitue les variables internes de Σ par des variables distinctes,*
3. *les variables $x\theta$ qui substituent les variables internes de Σ ne doivent apparaître ni dans Γ ni dans Δ_2 et*
4. *la formule π_{i+n} est générée à partir de π_i par l'application d'une suite de règles. Cette suite contient au moins un dépliage droit.*

Les nouvelles variables introduites sont universelles.

Définition 3.4 ***(Pliage gauche(CUT_ L))***
Considérons la branche de l'arbre de preuve suivante :

$$\begin{array}{c} < \pi_i : \Lambda \leftarrow \Sigma > \\ \vdots \\ \downarrow \\ < \pi_{i+n} : \Gamma_1\Gamma_2 \leftarrow \Delta > \\ \downarrow \qquad\qquad CUT\_L(\pi_i)^* \\ < \pi_{i+n+1} : \Sigma\theta, \Gamma_2 \leftarrow \Delta > \end{array}$$

*(* : pliage de π_i dans π_{i+n}, $n > 0$)*
avec :

1. $\Lambda\theta = \Gamma_1$,
2. *θ substitue les variables internes de Λ par des variables distinctes,*
3. *les variables $x\theta$ qui substituent les variables internes de Λ ne doivent apparaître ni dans Γ_2 ni dans Δ et*
4. *la formule π_{i+n} est générée à partir de π_i par l'application d'une suite de règles. Cette suite contient au moins un dépliage gauche.*

Les nouvelles variables introduites sont existentielles.

Remarque 3.2 *Les substitutions des dépliages du membre droit des variables quantifiées universellement vont être aussi appliquées sur le membre gauche. D'où parfois la nécessité de déplier les atomes du membre droit afin de réussir le pliage du membre gauche.*

3.3 Correction des règles de déduction

Définition 3.5 ***(Règle valide et règle conservative)***
Soient $\mathcal{S}$ un programme et une règle de déduction qui génère $E_{i+1} = (E_i \backslash \{\pi\}) \bigcup \{E'\}$, cette règle est :
- *Valide : si $\mathcal{M}(\mathcal{S}) \models E_{i+1}$ implique $\mathcal{M}(\mathcal{S}) \models E_i$.*
- *Conservative : si $\mathcal{M}(\mathcal{S}) \models E_{i+1}$ est équivalent à $\mathcal{M}(\mathcal{S}) \models E_i$.$\diamondsuit$*

Énonçons les propriétés des règles de déduction [8, 7, 6] :
- La simplification est valide, mais n'est généralement pas conservative.
- La règle de dépliage droit est conservative.
- La règle de dépliage gauche est valide, mais en général elle n'est pas conservative.
- Les règles de pliage droit et gauche ne sont pas conservatives, elles peuvent être valides sous certaines conditions.

3.4 Arbre de preuve

Le processus de preuve peut être schématisé par un arbre de preuve

Définition 3.6 ***(Arbre de preuve)***
Soient $\mathcal{S}$ un programme défini et ϕ une formule implicative. Un arbre de preuve pour ϕ est un arbre qui vérifie les conditions suivantes :
- *La racine est étiquetées par ϕ.*
- *Chaque nœud contient une formule et la règle appliquée pour obtenir ses descendants.*
- *Si un noœd contient une formule « Vrai » ou « Faux », alors c'est une feuille.*
- *Les descendants de chaque nœud sont construits à partir de la transformation qui lui est appliquée.*

L'interprétation de l'arbre de preuve est comme suit :
- Si toutes les feuilles de l'arbre de preuve sont Vrai alors la formule à la racine de l'arbre est un théorème.

– Si l'arbre de preuve contient un nœud étiquetés par « Faux » et toutes les transformations utilisées entre ce nœud et la racine sont des transformations conservatives, alors la formule à la racine de l'arbre n'est pas un théorème.
– Si l'arbre de preuve contient un nœud marqué par « Faux » et au moins une des transformations utilisées entre ce nœud et la racine est une transformation non conservative, alors on ne peut rien dire sur ϕ.

3.5 Réussite de l'application de l'hypothèse de récurrence

Les deux règles de pliage droit et gauche sont très importantes dans le processus de preuve, puisqu'elles sont le pendant de l'application de l'hypothèse de récurrence dans une preuve par récurrence classique. En effet, le pliage à droite (respectivement gauche) permet de remplacer l'instance d'un membre droit (respectivement gauche) d'une formule par l'instance correspondante de son membre gauche (respectivement droit), donc en général la règle de pliage permet de rapprocher les deux membres d'une formule.

La réussite de l'application de ces deux règles dépend de plusieurs conditions et critères. Par exemple, soit un membre Δ (droit ou gauche) à plier dans le membre Δ_1 obtenu après l'application sur Δ de certaines règles de preuve dont au moins un dépliage. Il faut qu'il existe une substitution θ telle que :
– Au moins un dépliage doit être effectué sur Δ avant d'obtenir Δ_1.
– $\Delta\theta = \Delta_1$ qui dépend :
  – de la définition des différents atomes de Δ
  – de l'instanciation des termes : les termes de Δ_1 doivent être des instances des termes de Δ afin de les filtrer
  – des relations entre les termes (c-à-d les termes qui ont des variables communes) de Δ doivent être conservées pour les termes de Δ_1 afin de trouver un filtre pour les deux membres.
– Les variables qui substituent des variables internes de Δ doivent être deux à deux distinctes et elles n'ont d'occurrence que dans Δ_1.

De plus, nous constatons que les dépliages sur le membre droit, dans certains cas, engendrent l'application d'une substitution sur des variables universelles du membre gauche. La substitution des dépliages d'un membre gauche est existentielle et n'affecte que les variables de ce membre.

En conséquence, pour étudier la réussite du pliage d'un membre droit nous n'avons pas besoin des informations sur le membre gauche. Par contre, les dépliages d'un membre droit d'une formule influent sur le membre gauche de cette formule et permettent, dans certains cas, une application de la règle de pliage gauche.

4 Schémas et formules candidates

Les schémas ont été introduits par F. Alexandre[1] dans le cadre de transformation de programmes logiques par pliage et dépliage. R. Salem [12] a repris et a étendu l'utilisation des schémas dans le cadre de la démonstration de théorèmes en utilisant le système de preuve par pliage et dépliage. Les schémas sont une sorte de compilation de l'information pertinente. En effet l'information pertinente dépend :
– de la classe des formules et programmes considérée
– du but de l'étude (exemple : réussite du pliage d'un membre d'une formule, ou simplification).

En se basant sur ces travaux, nous avons défini un schéma associé à une classe de formules (formules candidates). Ces schémas sont les données d'entrée pour l'étude statique de la réussite de la simplification et du pliage.

Notation 4.1 *Nous considérons deux types de termes :*
– *ceux construits avec des constructeurs unaires tels que les entiers naturels de la forme* $s(s(...(s(x))...)$
– *ceux construits avec des constructeurs binaires tels que les listes de la forme* $[a_1|[a_2...|[a_n|x]...]$.

Ces termes sont notés par $cons^a(x)$ *avec :*
– x *est une variable de type liste ou entier naturel*
– $cons$ *est le constructeur du type de* x
– a *est le nombre d'application du constructeur.*$\diamondsuit$

Dans ce qui suit, nous présentons la classe de programmes considérée. Nous caractérisons les clauses récursives de ces programmes par des schémas.

Définition 4.1 *Les clauses des programmes traités sont des clauses récursives linéaires de la forme :* $p(T_1, \cdots, T_k) \leftarrow \Delta, p(t_1, \cdots, t_k), \Theta$ *où chaque couple* (T_i, t_i) *peut avoir l'une des deux formes suivantes :*
– $(T_i, t_i) = (cons^{k+r}(x), cons^k(x))$ *avec* $r \geq 0$.
– $(T_i, t_i) = (cons^k(x), z)$, *avec* z *une variable.*

De plus, une variable apparaît toujours dans des termes construits avec le même constructeur cons.

Exemple 4.1 *Soit* $(\mathcal{S})$ *le programme logique suivant :*

$$\mathcal{S}\begin{cases} (1)\ len([],0) & \leftarrow \\ (2)\ len([x|y],s(z)) & \leftarrow len(y,z) \\ (3)\ app([],x,x) & \leftarrow \\ (4)\ app([a|x],y,[a|z]) & \leftarrow app(x,y,z) \end{cases}$$

Définition 4.2 ***(Schéma d'une clause récursive linéaire)***
Soit c une clause récursive linéaire de la forme $p(T_1, \cdots, T_n) \leftarrow \Delta p(t_1, \cdots, t_n)\Theta$. Le schéma de cette clause est défini par le n_uplet $(\zeta_c(1), \cdots, \zeta_c(n))$ tel que $\zeta_c(i) = (T_i, t_i)$.

Notation 4.2 *Pour chaque couple (T_i, t_i), les notations suivantes sont adoptées :*
- *$(cons^k(x), cons^k(x))$ est noté par ι_k,*
- *$(cons^k(x), w)$ est noté par ω_k,*
- *$(cons^{k+r}(x), cons^k(x))$ est noté par $\tau_{k,r}$.*

Exemple 4.2 *Le schéma de la clause 4 du programme $\mathcal{S}$ est : $(\tau_{0,1}, \iota_0, \tau_{0,1})$.*$\diamondsuit$

Dans ce qui suit, nous présentons les formules considérées dans cette étude et nous allons les caractériser par des schémas. Ces schémas sont déduits à partir des clauses définissant les atomes de ces formules.

Définition 4.3 ***(Formule implicative candidate)***
Une formule implicative candidate est de la forme $\phi : \Gamma \leftarrow \Delta$ où Δ et Γ sont des conjonctions d'atomes telles que :
- *Pour toute variable $x \in \mathcal{V}ar(\phi)$, x apparaît toujours dans des termes construits avec le même constructeur cons.*
- *Pour chaque atome $p_i(\overline{t}_i)$ de Δ ou de Γ, $\overline{t}_i$ est un vecteur de termes qui sont de la forme $cons^k(x)$, $k \geq 0$. Les atomes $p_i(\overline{t}_i)$ peuvent être non linéaires.*
- *Pour chaque prédicat p_i de Δ ou de Γ, les définitions de p_i sont récursives linéaires.*

Définition 4.4 ***(Schéma d'un membre)*** *Soit un membre $\Delta = p_1(\overline{t}_1), \cdots, p_k(\overline{t}_k)$ (Δ est un membre gauche ou droit d'une formule à démontrer), si x est une variable d'un terme t_{ij} de $p_i(t_{i_1}, \cdots, t_{ij}, \cdots, t_{in_i})$ alors le schéma qui lui est associé est : $\psi_{ij} = \zeta_{p_i}(j)(t_{ij})$. Le schéma de Δ est défini par l'ensemble des schémas de ses variables.*

Pour chaque variable interne à un membre à la position j d'un atome p_i tels que :
- *la définition de p_i de la forme : $p_i(T_1, ..., T_j, ...T_n) \leftarrow p_i(t_1, ..., t_j, ...t_n), \Delta$ avec Δ une conjonction d'atomes*
- *une variable $w \in \mathcal{V}ar(t_j) \cap \mathcal{V}ar(\Delta)$,*

trouve son schéma marqué par un plus.

Notation 4.3 *Si le terme $t_{ij} = cons^a(x)$, ψ_{ij} se note selon les valeurs de $\zeta_{p_i}(j)$:*
- *Si $\zeta_{p_i}(j) = \tau_{k,r}$ alors $\psi_{ij} = \tau_{k,r}(cons^a(x))$ ou bien $\psi_{ij} = \tau_{k,r}(x^a)$.*
- *Si $\zeta_{p_i}(j) = \iota_k$ alors $\psi_{ij} = \iota_k(cons^a(x))$ ou bien $\psi_{ij} = \iota_k(x^a)$.*
- *Si $\zeta_{p_i}(j) = \omega_k$ alors $\psi_{ij} = \omega_k(cons^a(x))$ ou bien $\psi_{ij} = \omega_k(x^a)$.*

Définition 4.5 ***(Schéma d'une formule implicative candidate)*** *Soit une formule implicative candidate, l'ensemble des schémas de ses deux membres droit et gauche définit le schéma de la formule implicative candidate.*

Notation 4.4 *Dans la suite, nous présentons les schémas des formules implicatives candidates sous forme de tableaux.*

$$\phi : app(x, y, Z)len(Z, t) \leftarrow app(y, x, r)len(r, t)$$

Le schéma de cette formule est présenté par le tableau qui suit :
Soit la formule à prouver :

	x	y	$\underline{Z}$	$\underline{r}$	t
1 *app*	$\tau_{0,1}(x)$	$\iota_0(y)$	$\tau_{0,1}(Z)$		
2 *len*			$\tau_{0,1}(Z)$		$\tau_{0,1}(t)$
3 *app*	$\iota_0(x)$	$\tau_{0,1}(y)$		$\tau_{0,1}(r)$	
4 *len*				$\tau_{0,1}(r)$	$\tau_{0,1}(t)$

La première colonne du tableau est formée des numéros des atomes de la formule à prouver. La deuxième colonne est formée des symboles de prédicats correspondant à ces atomes. Les autres colonnes correspondent aux variables ayant des occurrences dans la formule. Au point de la ligne p_i et de la colonne x on trouve l'ensemble des $\psi_{ij}(x)$.

Les schémas du membre gauche et du membre droit sont représentés respectivement par les premières lignes du tableau et les deuxièmes lignes et ils sont séparés par une ligne blanche. Ainsi les atomes 1 et 2 sont les atomes du membre gauche et les atomes 3 et 4 sont ceux du droit.

Les variables internes à un membre sont soulignées.$\diamondsuit$

Dans ce qui suit, nous définissons le schéma élémentaire relatif à une variable d'un membre d'une formule qui est un sous ensemble du schéma d'une formule, dans le but de pouvoir par la suite associer certaines conditions nécessaires à la réussite d'un pliage.

Définition 4.6 ***(Schéma élémentaire d'une variable)***
Soit un membre $\Delta : p_1(\overline{t}_1), \cdots, p_k(\overline{t}_k)$ avec $\overline{t}_i$ un vecteur de termes $(t_{i1}, \cdots, t_{ih_i})$. Soit x une variable de Δ qui a r occurrences, l'ensemble $\{\psi_{i_1 j_1}, \cdots, \psi_{i_r j_r}\}$ est un sous-ensemble de schémas du membre Δ contenant les schémas des r occurrences de x. C'est le schéma élémentaire du membre Δ par rapport à la variable x.

5 Conditions de filtrage associées aux schémas

Nous nous intéressons dans cette section à la réussite du pliage d'un membre Δ dans un membre Δ_1 et en particulier à la vérification de la condition 1 du pliage : $\Delta_1 = \Delta\theta$.

Nous avons considéré des programmes élémentaires en utilisant la notion de projection par rapport à un argument.

Par exemple, soit Δ un membre d'une formule candidate au pliage, tel que x apparaît dans deux atomes p_i et p_j respectivement dans les argument k et l. Nous considérons les clauses c_1 et c_2 qui définissent les projections des deux prédicats p_i et p_j par rapport à la position k et l.

$$\mathcal{S}_1 \left\{ \begin{array}{l} c_1 \ : \ p_i^k(T_k) \leftarrow p_i^k(t_k) \\ c_2 \ : \ p_j^l(T_l) \leftarrow p_j^l(t_l) \end{array} \right.$$

Soit $\{\psi_{ik}(cons^a(x)), \psi_{jl}(cons^b(x))\}$ le schéma élémentaire de Δ par rapport à x. En effet, selon la valeur des éléments du schéma élémentaire, on définit les conditions sur les dépliages des atomes p_i et p_j nécessaires pour la réussite du pliage de Δ.

Dans les sections 5.1 et 5.2 nous présentons les conditions nécessaires associées aux différentes valeurs du schéma élémentaire. Ces conditions ont été prouvées dans [10]. L'idée générale de ces preuves est donnée dans la section 5.4.

5.1 Conditions associées aux variables du membre droit et aux variables existentielles

5.1.1 Les variables à une seule occurrence

Schémas	*Conditions*
$\{\tau_{k_1,r_1}(x^a)\}$	$a > k_1$ $n = 0$
	$a \leq k_1$ Pas de conditions
$\{\iota_{k_1}(x^a)\}$	Pas de conditions
$\{\omega_{k_1}(x^a)\}$	$a = 0$

n est le nombre des dépliages de l'atome contenant x que l'on doit effectuer.

5.1.2 Les variables à plusieurs occurrences

Soit un schéma élémentaire de la variable x : $\{\psi_{i_1 j_1}, \cdots, \psi_{i_d j_d}\}$ avec $d \geq 1$.

- S'il existe $\psi_{ij} = \iota_{k_{ij}}(x^{a_{ij}})$ alors pour tout schéma $\psi_{hg}(x) = \tau_{k_{hg},r_{hg}}(x^{a_{hg}})$, $n_h = 0$.
- S'il existe $\psi_{ij} = \omega_{k_{ij}}(x^{a_{ij}})$ alors $n_i = 0$ et pour tout schéma $\psi_{hg}(x) = \tau_{k_{hg},r_{hg}}(x^{a_{hg}})$, $n_h = 0$.
- Si tous les schémas sont de la forme $\psi_{ij}(x) = \tau_{k_{ij},r_{ij}}(x^a)$ alors si tous les $a_{ij} > k_{ij}$ alors $n_{i_1} = \cdots = n_{i_d} = 0$. S'il existe au moins un a_{ij} tel que : $a_{ij} \leq k_{ij}$ alors $n_{i_1} * r_{i_1 j_1} = \cdots = n_{i_d} * r_{i_d j_d} \geq 0$.

5.2 Conditions associées aux variables universelles du membres gauche

Dans l'étude des conditions pour les schémas associés aux variables universelles du membre gauche (section 5.2) nous avons ajouté des conditions de la forme $m \geq val$. De telles conditions seront vérifiées s'il existe des dépliages du membre droit de la formule qui génère des substitutions $\{x/cons^m(x)\}$ permettant d'instancier la variable x du membre gauche. Dans la section 5.3, nous donnons selon le schéma d'une variable d'un atome p_i dans un membre droit la substitution engendrée par les dépliages effectués sur p_i.

5.2.1 Les variable à une seule occurrence

Schémas	*Conditions*
$\{\tau_{k,r}(x^a)\}$	$a \geq k$ $m \geq n * r$
	$a < k$ $m \geq k + n * r - a$
$\{\iota_k(x^a)\}$	$a \geq k$ $m \geq 0$
	$a < k$ $m \geq k - a$
$\{\omega_k(x^a)\}$	$a = k = 0$ $m \geq 0$
	$a < k \wedge a = 0$ $m \geq k - a$

n est le nombre des dépliages de l'atome contenant x que l'on doit effectuer.

5.2.2 Les variables à plusieurs occurrences

Soit un schéma élémentaire de la variable x : $\{\psi_{i_1 j_1}, \cdots, \psi_{i_d j_d}\}$ avec $d \geq 1$.

- S'il existe $\psi_{ij} = \iota_{k_{ij}}(x^{a_{ij}})$ alors pour tout schéma $\psi_{hg}(x) = \tau_{k_{hg},r_{hg}}(x^{a_{hg}})$, $n_h = 0$ et la valeur de m_x est comme suit :
  - si parmi les schémas qui sont sous la forme $\psi_{ij} = \iota_k(x^{a_{ij}})$, nous avons $a_{ij} < k_{ij}$ et $n_i > 0$ alors $m \geq Max(k_{ij} - a_{ij})$
  - sinon $m_x \geq 0$.
- S'il existe $\psi_{ij} = \omega_{k_{ij}}(x^{a_{ij}})$ alors $n_i = 0$ et pour tout schéma $\psi_{hg}(x) = \tau_{k_{hg},r_{hg}}(x^{a_{hg}})$, $n_h = 0$.

– Si tous les schémas sont sous la forme $\psi_{ij}(x) = \tau_{k_{ij},r_{ij}}(x^{a_{ij}})$ alors $n_{i_1} * r_{i_1 j_1} = \cdots = n_{i_d} * r_{i_d j_d}$ et la valeur de m_x est comme suit :
  – si tous les $a_{ij} \geq k_{ij}$ et $n_i > 0$ alors $m_x \geq n_{i_1} * r_{i_1 j_1}$
  – s'il existe des schémas tel que $a_{ij} < k_{ij}$ alors $m_x \geq Max(k_{ij} - a_{ij} + n_{i_i} * r_{ij})$ avec $\psi_{ij} = \tau_{k_{ij},r_{ij}}(x^{a_{ij}})$, $a_{ij} < k_{ij}$ et $n_i > 0$.

5.3 Étude de l'influence des dépliages droits sur le membre gauche

Soit une formule $\pi : \Delta \leftarrow \Gamma$, soit $\pi' : \Delta\theta \leftarrow \Delta_1\Gamma'$ une formule obtenue en appliquant des dépliages droits à partir de π et soit x une variable universelle telle que $x \in \mathcal{V}ar(\Delta \cap \Gamma)$. Si $x \in Dom(\theta)$, alors il existe une substitution $\theta_i = \{x/cons^m(x)\} \subset \theta$. La substitution θ_i plus particulièrement m dépend du schéma élémentaire relatif à x dans Γ.

Schémas	*Valeur de m*
$\{\tau_{k,r}(x^a)\}$	$a \leq k$ $m = k + i * r - a$ et $i \geq 1$
	$a = k + n * r + c > k$ $m = (i - n) * r - c$ et $i \geq n + 1$
$\{\iota_k(x^a)\}$	$a \geq k$ $m = 0$
	$a < k$ $m = k - a$ et $i \geq 1$
$\{\omega_k(x^a)\}$	$a \geq k$ $m = 0$
	$a < k$ $m = k - a$ et $i \geq 1$

TAB. 1 – Les valeurs de m

Proposition 5.1 *Soit x une variable qui a d occurrences dans le membre droit.*

Soit $\{\psi_{i_1 j_1}(x), \cdots, \psi_{i_d j_d}(x)\}$ son schéma élémentaire par rapport à x.

1. *Si ces schémas sont seulement de la forme $\iota_{k_i}(x^{a_i})$ ou $\omega_{k_i}(x^{a_i})$ alors la valeur maximale de m est $max(k_i - a_i)$ où $k_i > a_i$.*
2. *Si parmi ces schémas, il existe au moins un de la forme $\tau_{k_i,r_i}(x^{a_i})$, alors on peut obtenir une valeur arbitrairement grande.*

5.4 Preuve des conditions associées aux schémas

Dans ce qui précède, nous avons associé des conditions sur les dépliages à effectuer pour réussir un pliage. La preuve de chaque condition est détaillée dans [10].

Le processus de preuve de ces conditions est le suivant : Nous définissons la condition 5.1 ci-dessous, qui détaille la condition 1 des définitions du pliage droit et gauche et qui est spécifique à la classe de formules choisie.

Condition 5.1 ***(Conditions de filtrage relatives aux membres droit et gauche)***

Soit ϕ une formule candidate de la forme :
$\Delta \leftarrow p_1(\overline{t_1}), ..., p_i(\overline{t_i}), ..., p_k(\overline{t_k})$
supposons qu'après certains dépliages des atomes de ϕ nous obtenions la formule :
$\phi' : \Delta' \leftarrow p_1(\overline{t'_1}), ..., p_i(\overline{t'_i}), ..., p_k(\overline{t'_k})\Theta$

Afin de réussir le pliage du membre droit de ϕ dans ϕ', il faut qu'il existe une substitution θ, telle que $(p_1(\overline{t_1}), ..., p_i(\overline{t_i}), ..., p_k(\overline{t_k}))\theta = p_1(\overline{t'_1}), ..., p_i(\overline{t'_i}), ..., p_k(\overline{t'_k})$, les deux conditions suivantes doivent être vérifiées :

1. *Condition d'instanciation : Pour chaque terme $t_{ij} = cons^a(x)$ d'un atome p_i avec $i \in [1..k]$ dans le membre droit de ϕ, il existe un terme $t'_{ij} = cons^{a_1}(y)$ dans le membre droit de ϕ' tel que $a_1 \geq a$.*
2. *Conditions de lien : si $x \in t_{ij} = cons^a(x)$ et $x \in t_{i'j'} = cons^b(x)$ d'un atome $p_{i'}$ ($i' \in [1..k]$) dans le membre droit de ϕ, alors il existe dans le membre droit de ϕ' les deux termes $t'_{ij} = cons^{a_1}(y)$ et $t'_{i'j'} = cons^{b_1}(y)$ tels que $a_1 - a = b_1 - b$.*

Ces conditions restent valables pour le membre gauche.$\diamondsuit$

Ces deux conditions ont été un guide pour la recherche et la découverte des conditions exprimées dans les sections précédentes. Pour chaque schéma, nous avons listé tous les membres possibles générés par les dépliages de ce membre, ainsi seulement les suites des dépliages qui génèrent des membres qui vérifient la condition 5.1 ont été gardées. Donc selon la valeur du schéma, on déduit des conditions nécessaires sur les dépliages à appliquer.

5.5 Exemple

Prenons la formule ϕ : $app(x, y, Z)len(Z, t) \leftarrow app(y, x, r)len(r, t)$ et définissons les conditions de filtrage associées pour chaque variable pour la réussite du pliage de son membre gauche.

– x est une variable universelle qui a une seule occurrence et le schéma élémentaire associé à x est : $(\tau_{0,1}(x))$, alors on lui associe la condition suivante : $n_1 * r_1 \leq m_x$
– y est une variable universelle qui a une seule occurrence et le schéma élémentaire associé à y est : $(\iota_0(y))$, alors on lui associe la condition suivante : $m_y \geq 0$

– Z est une variable existentielle qui a deux occurrences et le schéma élémentaire associé à Z est : $(\tau_{0,1}(Z), \tau_{0,1}(Z))$, alors on lui associe la condition suivante : $n_1 * r_1 = n_2 * r_2$
– t est une variable universelle qui a une seule occurrence et le schéma élémentaire associé à t est : $(\tau_{0,1}(t))$, alors on lui associe la condition suivante : $n_2 * r_2 \leq m_t$

Les variables t, x et y sont des variable universelles appartenant aux deux membres de ϕ. Les schémas élémentaires du membre droit de ϕ associés respectivement à t, x et y sont respectivement :$(\tau_{0,1}(t))$, $(\iota_0(x))$ et $(\tau_{0,1}(y))$. Alors, d'après de la proposition 5.1 les valeurs maximales de m_t et m_y sont arbitrairement grandes et la valeur maximale de m_x est 0.

6 Conditions de pliage reliées aux variables internes

Nous nous intéressons dans cette section à la réussite du pliage d'un membre Δ dans Δ_1 et en particulier à la vérification de la condition 3 du pliage.

$$\begin{array}{ll} \Delta & \\ \vdots & \\ \downarrow & \text{Dépliage} \\ \underbrace{\Delta\theta}_{\Delta_1}\beta & \\ \downarrow & \text{Pliage ?} \end{array}$$

Soit x une variable interne à Δ, bien qu'il existe une substitution θ telle que $\Delta_1 = \Delta\theta$, le pliage n'est possible que si et seulement si θ vérifie la condition 3 du pliage. Autrement dit, la variable $x\theta$ qui substitue x doit être interne à Δ_1. Si cette condition n'est pas vérifiée, nous pouvons conclure que ceci est dû à un ou plusieurs dépliages effectués sur les atomes de Δ. En conséquence, nous devons interdire ces dépliages.

Ci-dessous, nous présentons des conditions associées au schéma d'une formule candidate afin de respecter la condition 3 du pliage. Ces conditions sont prouvées dans [10] pour des formules et des programmes élémentaires.

Condition 6.1 *S'il existe dans le schéma du membre à plier un schéma relatif à un terme de la forme : $\psi_{p_{ij}}(x) = \tau_{k,r}(x^a)^+$ ou $\iota_k(x^a)^+$ ou $\omega_k(x^a)^+$ tel que x est une variable interne (soulignée) alors $n_i = 0$.*

7 Étude statique du pliage

Nous présentons dans ce qui suit les étapes à appliquer pour l'étude statique des pliages droit et gauche.

Soit une formule candidate : ϕ de la forme $\Gamma \leftarrow \Delta$.

L'étude des pliages droit et gauche de cette formule se résume en cette suite d'étapes :

1. Calcul du schéma de la formule ϕ.
2. Calcul des schémas élémentaires du membre concerné.
3. Pour chaque schéma élémentaire trouvé, on fait correspondre la condition adéquate.
4. Ajout des conditions sur les variables internes.
5. On obtient ainsi les conditions $(cond_i)$ $(1 \leq i \leq l)$. Soit $\mathcal{S}_1$ le système suivant :

$$\mathcal{S}_1 \begin{cases} cond_i \ (1 \leq i \leq l) \\ n_1 + ... + n_k \geq 1 \end{cases}$$

   Nous dirons que $\mathcal{S}_1$ est le système associé au membre candidat au pliage.
   L'inéquation $n_1 + ... + n_k \geq 1$ signifie que l'on doit déplier au moins un atome du membre à plier avant de faire un pliage.
6. Selon le membre :
   – Membre droit : Si le système a une solution, on choisit une solution qui minimise la somme $n_1, \cdots, n_k$, et on applique les dépliages correspondants.
   – Membre gauche : Si le système a une solution, qui ne contient aucun $m > 0$, alors on choisit une solution qui minimalise la somme de $n_1, \cdots, n_k$, et on applique les dépliages correspondants.
   Si le système a toutes ses solutions qui contiennent au moins un $m > 0$, nous essayons de trouver au moins une solution parmi celles qui vérifient les valeurs des m.
7. Si le système n'a pas de solution, nous pouvons conclure qu'il est impossible de plier.

8 Exemple

Reprenons la formule :
φ : $app(x, y, Z)len(Z, t) \leftarrow app(y, x, r)len(r, t)$
– Étudions le pliage du membre gauche :

$$S = \begin{cases} n_1 * r_1 & \leq m_x \\ n_1 * r_1 = n_2 * r_2 & \\ m_y & \geq 0 \\ n_2 * r_2 & \leq m_t \\ n_1 + n_2 & > 0 \end{cases}$$

La solution minimale pour ce système est $n_1 = n_2 = 1$ avec $m_x > 0$ et $m_t > 0$.
Étudions maintenant les valeurs de m_t et m_x.

$$S = \begin{cases} m_t & = n_4 * r_4 \\ m_x & = 0 \end{cases}$$

Il est impossible de plier parce qu'on ne peut pas instancier x à partir du membre droit.

– Étudions le pliage du membre droit :

$$S = \begin{cases} n_3 * r_3 = n_4 * r_4 \\ n_3 + n_4 \quad > 0 \end{cases}$$

La solution minimale pour ce système est $n_3 = n_4 = 1$.

$$\begin{array}{rcl} app(x,y,Z)len(Z,t) & \leftarrow & \underline{app(y,x,r)}len(r,t) \\ & \Downarrow & NFI \\ app(x,[a|y],Z)len(Z,t) & \leftarrow & app(y,x,r)\underline{len([a|r],t)} \\ & \Downarrow & NFI \\ app(x,[a|y],Z)len(Z,s(t)) & \leftarrow & app(y,x,r)len(r,t) \\ & \Downarrow & CUT-R[1] \\ app(x,[a|y],Z)len(Z,s(t)) & \leftarrow & app(x,y,\ v)len(v,t) \end{array}$$

Nous obtenons alors cette nouvelle formule :

$$app(x,[a|y],Z)len(Z,s(t)) \leftarrow app(x,y,v)len(v,t)$$

son schéma est :

$$\left[\begin{array}{cccccc} & x & y & \underline{Z} & \underline{v} & t \\ 1app & \tau_{0,1}(x) & \iota_0(y^1) & \tau_{0,1}(Z) & & \\ 2len & & & \tau_{0,1}(Z) & & \tau_{0,1}(t^1) \\ & & & & & \\ 3app & \tau_{0,1}(x) & \iota_0(y) & & \tau_{0,1}(v) & \\ 4len & & & & \tau_{0,1}(v) & \tau_{0,1}(t) \end{array}\right]$$

Étudions le pliage du membre gauche :

$$S = \begin{cases} n_1 * r_1 & \leq m_x \\ n_1 * r_1 = n_2 * r_2 & \\ m_y & \geq 0 \\ n_2 * r_2 & \leq m_t \\ \Sigma n_i & > 0 \end{cases}$$

La solution minimale pour ce système est $n_1 = n_2 = 1$ avec $m_x > 0$ et $m_t > 0$.

Étudions maintenant la valeur de m_t et m_x.

$$S = \begin{cases} m_t & = n_4 * r_4 \\ m_x & = n_3 * r_3 \end{cases}$$

Alors, le pliage est possible et la solution est : $n_1 = n_2 = n_3 = n_4 = 1$.

$$\begin{array}{c} app(x,[a|y],Z)len(Z,s(t)) \leftarrow \underline{app(x,y,v)}len(v,t) \\ \Downarrow NFI \\ \underline{app([b|x),[a|y],Z)}len(Z,s(t)) \leftarrow app(x,y,v)len([b|v],t) \\ \Downarrow DCI \\ app(x,[a|y],Z)\underline{len([b|Z],s(t))} \leftarrow app(x,y,v)len([b|v],t) \\ \Downarrow DCI \\ app(x,[a|y],Z)len(Z,t) \leftarrow app(x,y,v)\underline{len([b|v],t)} \\ \Downarrow NFI \\ app(x,[a|y],Z)len(Z,s(t)) \leftarrow app(x,y,v)len(v,t) \\ \Downarrow Cut-L[4] \\ app(x,y,W)len(W,t) \leftarrow app(x,y,v)len(v,t) \\ \Downarrow \\ Vrai \end{array}$$

9 Bilan et application

Dans le cas général, le problème du pliage est indécidable. Ainsi, nous nous sommes intéressés à une classe particulière de formules et de programmes qui est une classe importante. Nous nous sommes basés sur une étude statique de la formule à prouver et du programme associé qui permet de donner comme résultat : soit la réussite du pliage avec les étapes à suivre, soit l'échec du pliage.

Malgré cette limitation dans les programmes et les formules considérées, notre approche peut être appliquée comme une heuristique sur toute la classe des programmes récursifs. La réussite de cette étude se base sur les conditions définies dans les sections 5 et 6.

– ces conditions sont nécessaires et suffisantes dans le cas où les clauses des programmes considérés sont récursives unaires et qu'elles ne contiennent que des termes construits avec des constructeurs unaires. Dans ce cas notre approche est déterministe.
– pour les programmes, qui contiennent des termes construits avec des constructeurs binaires tels que les liste de la forme $[a_1|[a_2...|[a_n|l]...]$, ces conditions sont nécessaires et parfois non suffisantes pour déterminer la réussite du pliage. Ceci est dû aux variables a_i qui ne sont pas prises en considération dans la définition de ces conditions.
– dans le cas où les programmes considérés contiennent des clauses récursives non unaires, alors ces conditions sont suffisantes et parfois non nécessaires pour déterminer la réussite du pliage puisque dans l'étude statique nous ne prenons pas en considération l'information obtenue à partir des nouveaux atomes introduits dans la formule candidate.

Pour les deux derniers cas, notre approche est une heuristique très forte. En effet, dans un grand nombre d'exemples traités, les résultats retournés par cette étude sont correctes. Nous avons appliqué notre approche dans la résolution d'un ensemble important d'exemples réputés difficiles à prouver par des systèmes de preuve existant (sans apport important de l'extérieur en lemmes), citons quelques exemples dans le tableau 2.

$plus(x,y,M)plus(x,v,M) \leftarrow$	$equal(y,v)$
$rev(x,T)rev(T,y) \leftarrow$	$equal(x,y)$
$mul(x,y,z) \leftarrow$	$mul(y,x,z)$
$len(z,T)plus(k,l,T) \leftarrow$	$len(x,k)len(y,l)$ $app(x,y,z)$
$len(z,k) \leftarrow$	$len(x,k)rev(x,z)$
$rev(x,w) \leftarrow$	$app(x,[y],z)$ $rev(z,[y\|w])$
$plus(k,l,M)len(t,M) \leftarrow$	$app(x,y,z)rev(z,t)$ $len(x,k)len(y,l)$
$plus(x,y,T)plus(m,n,V) \leftarrow$ $inf(T,V)$	$inf(x,n)inf(y,m)$
$perm(x,z) \leftarrow$	$perm(x,y)perm(y,z)$
$rev(x,Y)rev(Y,x) \leftarrow$	$list(x)$
$nth(i,x,y)nth(j,y,z) \leftarrow$	$nth(j,x,u)mul(i,u,z)$
$nth(i,w,Z)nth(j,Z,Y) \leftarrow$ $nth(k,Y,x)$	$nth(k,w,v)nth(j,v,l)$ $nth(i,l,x)$
$mul(x,y,t)mul(t,z,w) \leftarrow$	$mul(y,z,u)$ $mul(x,u,w)$
$palindrome(z) \leftarrow$	$rev(x,y)app(x,y,z)$
$ord(x)ord(y) \leftarrow$	$ord(z)app(x,y,z)$
$ord(t) \leftarrow$	$place(a,t,y)ord(y)$
$plus(x,y,T)plus(T,z,w) \leftarrow$	$plus(y,z,k)$ $plus(x,k,w)$
$app(z,[Y],T) \leftarrow$ $rotate(x,T,U)len(U,s(k))$	$rotate(x,z,a)len(a,k)$
$app(x,y,Z)len(Z,t) \leftarrow$	$app(y,x,r)len(r,t)$
$rev(x,Y)len(Y,t) \leftarrow$	$len(x,t)$

TAB. 2 – Exemples de formules

Références

[1] F. Alexandre. Transformations de programmes logiques. Thèse de l'université de Nancy I, février 1991.

[2] F. Alexandre, K. Bsaïes, and M. Demba. Predicate synthesis from inductive proof attempt of faulty conjectures. In *Proc. of the International Symposium on Logic-based Program Synthesis and Transformation (LOPSTR'03), Uppsala, Sweden*. Springer-Verlag, August 2003.

[3] F. Alexandre, M. Demba, and K. Bsaïes. Stratégies de preuves par récurrence basées sur le pliage et le dépliage. Technical Report TR A01-R-284, LORIA, France, Septembre 2001.

[4] A. Bundy. *Handbook of Automated Reasoning*, chapter The Automation of Proof by Mathematical Induction. In A. Robinson and A. Voronkov, editors. Elseviers Science Publishers B. V. (North-Holland), 1999.

[5] H. Comon. *Inductionless Induction*, chapter 1. Handbook of Automated Reasoning. Elsevier Science Publishers B.V., 1999.

[6] M. Demba, F. Alexandre, and K. Bsaïes. A method for patching faulty conjectures. Technical Report TR A04-R-321, LORIA, France, 2004.

[7] L. Fribourg. *Extracting Logic Programs from Proofs that Use Extended Prolog Execution and Induction, In J. M. Jaquet, editor, Constructing Logic Programs*. Chapter 2, pp. 39-66 Wiley, 1993.

[8] T. Kanamori and H. Seki. Verification of Prolog Programs Using an Extension of Execution. In *3rd International Conference on Logic Programming*, volume 225 of *Lecture Notes in Computer Science*, pages 475–489. Springer-Verlag, 1986.

[9] J.W. Lloyd. *Foundations of Logic Programming*. Springer-Verlag, 1987.

[10] I. Mouakher. Une approche basée sur l'analyse statique pour la preuve automatisée par pliage et dépliage de formules implicatives. Mémoire de mastère de l'Université de Tunis El Manar, juillet 2004.

[11] A. Sakurai and H. Motoda. Proving Definite Clauses without Explicit Use of Inductions. In K. Furukawa, H. Tanaka, and T Fujisaki, editors, *Proceedings of the 7th Conference, Logic Programming '88*, volume 383 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 1988.

[12] R. Salem and K. Bsaïes. Mechanizing proofs by unfolding, folding and simplification. *RelMiCS*, pages 133–142, 1997.