Récurrence noethérienne pour le raisonnement de premier ordre

Sorin Stratulat

Récurrence nœthérienne pour le raisonnement de premier ordre

Sorin Stratulat [1]

La récurrence nœthérienne est un des principes les plus généraux de raisonnement formel. Dans le cadre du raisonnement de premier ordre, nous présentons une classification de ses instances pouvant être partagées en instances basées sur des termes et des formules. Nous donnons un aperçu du raisonnement par récurrence nœthérienne basée sur des termes et sur des formules, puis nous établissons des relations entre eux. Enfin, nous présentons une méthodologie pour la certification du raisonnement basé sur des formules à l'aide de l'assistant de preuve Coq.

Introduction

Le raisonnement par récurrence est bien adapté pour valider des propriétés sur des structures de données non-bornées, comme les naturels et les listes, ainsi que des spécifications basées sur des fonctions récursives (protocoles, algorithmes distribués, etc). Sa force réside dans la possibilité d'utiliser des formules dont on n'a pas besoin de montrer la validité, en tant qu'*hypothèses de récurrence*.

Un des principes les plus généraux du raisonnement par récurrence, la *récurrence nœthérienne*, utilise des *ordres nœthériens* (ou bien fondés) pour garantir l'application correcte des hypothèses de récurrence.

1. Université de Lorraine, CNRS, LORIA.

DOI:10.48556/SIF.1024.19.157 **1024** – Bulletin de la SiF, numéro 19, Avril 2022, pp. 157–170

Définition 3 (le principe de récurrence nœthérienne). *Étant donné un ensemble (potentiellement infini) $\mathscr{E}$ muni d'un ordre nœthérien $<$, le principe de récurrence nœthérienne peut être formalisé par la formule d'ordre supérieur suivante :*

$$\forall \varphi, (\forall m \in \mathscr{E}, (\forall k \in \mathscr{E}, k < m \Rightarrow \varphi(k)) \Rightarrow \varphi(m)) \Rightarrow \forall p \in \mathscr{E}, \varphi(p), \tag{1}$$

où φ est un symbole de prédicat quantifié universellement.

Le principe de récurrence nœthérienne permet de vérifier si une propriété définie sur les éléments de l'ensemble $\mathscr{E}$ est satisfaite par tous les éléments de $\mathscr{E}$. Ainsi, pour une propriété φ et un élément $m \in \mathscr{E}$ donnés, toute formule $\varphi(k)$, avec $k < m$, peut être utilisée dans la preuve de $\varphi(m)$, en tant qu'hypothèse de récurrence. L'argument de correction est donné par la propriété de l'ordre nœthérien $<$ qui interdit toute suite *infinie* strictement décroissante d'éléments de $\mathscr{E}$. Ce principe se démontre simplement par contradiction si on suppose qu'il y a un élément $x_0 \in \mathscr{E}$ qui ne satisfait pas φ, alors il existe par contraposition un élément plus petit $x_1 \in \mathscr{E}$ qui ne satisfait pas φ. On peut raisonner sur x_1 comme on avait fait pour x_0 pour déduire qu'il y a un élément x_2 de $\mathscr{E}$ plus petit que x_1 et qui ne satisfait pas non plus φ, etc. On conclut ainsi qu'il y a une suite $\ldots < x_2 < x_1 < x_0$ strictement décroissante infinie d'éléments de $\mathscr{E}$ qui ne satisfont pas φ. Ceci contredit le fait que $<$ est un ordre nœthérien.

Nous nous sommes intéressés à l'application du principe de récurrence nœthérienne dans le raisonnement de premier ordre. Ceci se résume à produire des instances de premier ordre de la formule (1). Dans le reste de l'article, nous allons d'abord classifier ces instances selon une taxonomie que nous avons proposée pour la première fois dans [35] et qui prend en considération les cas où les éléments de $\mathscr{E}$ sont soit des termes, soit des formules de premier ordre. Ensuite, nous établissons des relations entre ces deux situations, puis nous faisons un état de l'art des principes de récurrence nœthérienne basés sur des termes et sur des formules. Enfin, nous montrons comment formaliser et certifier le raisonnement basé sur des formules à l'aide de l'assistant de preuve Coq [42], en résumant la méthodologie présentée dans [37].

Instances basées sur des termes et sur des formules de premier ordre

Comme exemple illustratif, on va prouver par récurrence la conjecture

$$x + 0 = x, \text{ pour tout naturel } x \tag{2}$$

à partir des axiomes

$$0 + x = x, \text{ pour tout naturel } x \tag{3}$$

$$S(x) + y = S(x + y), \text{ pour tout naturel } x \text{ et } y \tag{4}$$

qui définissent l'addition '+' sur les naturels à l'aide des symboles de constructeurs 0 et S (la fonction 'successeur'). '=' est le seul symbole de prédicat et les formules sont des *égalités* de la forme $s = t$.

Durant les preuves, on va employer un raisonnement dit *équationnel* qui implémente le principe de Leibniz de remplacements des égaux par des égaux [2] en utilisant les parties gauche et droite des égalités. Les variables sont implicitement quantifiées universellement.

On va prouver $n+0=n$ pour un naturel n arbitraire qui peut être soit 0, soit le successeur d'un autre naturel n'. Dans le premier cas, $0+0=0$ est une instance de l'axiome (3) lorsque x est 0. Dans le deuxième cas, $S(n')+0=S(n')$ est équivalent à $S(n'+0)=S(n')$ si on remplace la partie gauche par la partie droite de l'égalité $S(n')+0=S(n'+0)$, qui est une instance de (4) lorsque x est n' et y est 0. L'hypothèse de récurrence $n'+0=n'$ permet de remplacer $S(n'+0)$ par $S(n')$ afin d'obtenir l'identité $S(n')=S(n')$ et de finir la preuve.

L'usage correct de $n'+0=n'$ peut être justifié par deux instances différentes du principe de récurrence nœthérienne, lorsque les éléments de l'ensemble $\mathscr{E}$ de la Definition 3 sont i) des (vecteurs de) termes, ou ii) des formules de premier ordre. Pour faire la distinction entre les ordres employés dans les deux cas, on va dénoter $<$ par $<_t$ pour le premier cas, et par $<_f$ pour le deuxième cas.

Récurrence nœthérienne basée sur des termes (RNT). Lorsque $\mathscr{E}$ est un ensemble de vecteurs de termes, (1) devient une formule de premier ordre si φ est fixé à un symbole de prédicat de premier ordre donné. L'instance de (1) obtenue ainsi est :

$$(5) \qquad (\forall \overline{m} \in \mathscr{E}, (\forall \overline{k} \in \mathscr{E}, \overline{k} <_t \overline{m} \Rightarrow \varphi(\overline{k})) \Rightarrow \varphi(\overline{m})) \Rightarrow \forall \overline{p} \in \mathscr{E}, \varphi(\overline{p})$$

où les symboles supralignés représentent des vecteurs de termes.

Un exemple de principe de RNT est celui de *Peano* : pour prouver une formule $P(x)$ pour tout naturel x, il est suffisant de la prouver pour x étant 0, puis $S(x')$, où x' est une nouvelle variable naturelle. Dans le deuxième cas, le principe de Peano permet d'utiliser $P(x')$ en tant qu'hypothèse de récurrence parce que $x' <_t S(x')$ lorsqu'on prend $<_t$ comme étant la relation « plus petit que » sur les naturels.

Exemple 1 (La preuve de $x+0=x$ par récurrence de Peano)**.** *Dans notre exemple, on définit $P(x)$ en tant que l'égalité $x+0=x$. L'hypothèse de récurrence fournie par le principe de Peano dans la preuve de $P(S(n'))$ est $P(n')$, i.e., $n'+0=n'$. C'est ce qui a été demandé.*

D'autre part, $n'+0=n'$ peut être aussi définie comme hypothèse de récurrence légitimée par des principes de récurrence nœthérienne basés sur des formules, comme c'est expliqué ci-dessous.

Récurrence nœthérienne basée sur des formules (RNF). Cette fois-ci, les éléments de $\mathscr{E}$ de (1) sont des formules de premier ordre. Dans ce cas, φ est un symbole de prédicat de second ordre puisqu'il a des formules de premier ordre comme arguments. Pour que (1) devienne une formule de premier ordre, une solution serait de définir φ comme étant la relation d'identité de second ordre telle que $\varphi(x) = x$, pour toute formule de premier ordre $x \in \mathscr{E}$. L'instance de la formule (1) est dans ce cas :

$$(\forall m \in \mathscr{E}, (\forall k \in \mathscr{E}, k <_f m \Rightarrow k) \Rightarrow m) \Rightarrow \forall p \in \mathscr{E}, p \tag{6}$$

Le principe de RNF ainsi obtenu dit qu'on a le droit d'utiliser une formule k dans la preuve d'une autre formule m si k est plus petite que m.

Exemple 2. *Le principe de RNF peut s'appliquer lorsque la preuve $x+0=x$ a été générée en utilisant un ordre $<_f$ qui respecte les conditions suivantes*[2] *:*

- $x+0=x <_f S(x+0)=S(x)$*, pour tout naturel x,*
- $S(x+0)=S(x) <_f S(x)+0=S(x)$*, pour tout naturel x, et*
- $<_f$ *est nœthérien et* stable par substitutions*, i.e., si $s <_f t$ alors $s\sigma <_f t\sigma$, pour toute substitution σ qui remplace des variables par des termes.*

On définit $\mathscr{E}$ comme étant l'ensemble de toutes les formules produites pendant la preuve, i.e., $\{n+0=n, 0+0=0, S(n')+0=S(n'), S(n'+0)=S(n'), S(n')=S(n')\}$. Dans la preuve de $S(n')+0=S(n')$, on a le droit d'utiliser en tant qu'hypothèse de récurrence toute formule de $\mathscr{E}$ plus petite. Dans notre cas, ce serait $n'+0=n'$, i.e., un renommage de la première formule pour lequel $n'+0=n' <_f S(n'+0) = S(n') <_f S(n')+0=S(n')$.

Théorème 1. *La relation d'identité de second ordre est la plus générale instanciation de φ qui permettrait de faire de (1) une formule de premier ordre.*

Démonstration. Soient $\mathscr{E} = \{\varphi_1, \ldots, \varphi_n, \ldots\}$ un ensemble de formules de premier ordre, p un symbole de prédicat de second ordre et $p(\varphi)$ une formule de premier ordre pour tout $\varphi \in \mathscr{E}$, et $<_p$ un ordre nœthérien tels que :

$$\forall \varphi_i \in \mathscr{E}, (\forall \varphi_j \in \mathscr{E}, \varphi_j <_p \varphi_i \Rightarrow p(\varphi_j)) \Rightarrow p(\varphi_i)) \Rightarrow \forall \varphi \in \mathscr{E}, p(\varphi) \tag{7}$$

On définit

- le nouvel ensemble de formules $\mathscr{E}' = \{p(\varphi) \mid \varphi \in \mathscr{E}\}$, et
- l'ordre $<_{p'}$ tel que $p(\varphi_i) <_{p'} p(\varphi_j)$ si $\varphi_i <_p \varphi_j$, pour toutes $\varphi_i, \varphi_j \in \mathscr{E}$.

On peut remarquer que $<_{p'}$ est nœthérien et que la formule (7) implique

$$\forall \varphi_i \in \mathscr{E}', (\forall \varphi_j \in \mathscr{E}', \varphi_j <_{p'} \varphi_i \Rightarrow \varphi_j) \Rightarrow \varphi_i) \Rightarrow \forall \varphi \in \mathscr{E}', \varphi$$

qui est une instance de (6) lorsque $\mathscr{E}$ est $\mathscr{E}'$ et $<_f$ est $<_{p'}$. □

2. Un tel ordre peut être défini sur des égalités en tant qu'extension multi-ensemble d'un ordre multi-ensemble sur les chemins basé sur la précédence croissante sur les symboles de fonction 0, S, et $+$ [2].

Principes de RNT

Les principes de RNT définissent les hypothèses de récurrence de manière *explicite* dans le cadre des *schémas de récurrence* [1], souvent issus de l'analyse des fonctions et des structures de données récursives. Étant donné une formule φ, un schéma de récurrence identifie d'abord un sous-ensemble de variables de φ à instancier, appelées des *variables de récurrence*, puis définit des instances de φ en termes d'hypothèses de récurrence et de *conclusions de récurrence* en spécifiant les hypothèses qui peuvent être utilisées dans la preuve de chaque conclusion. L'ensemble de conclusions doit être choisi de manière que sa validité implique celle de φ.

Un exemple de principe de récurrence basé sur des schémas de récurrences, issus des spécifications sortées et basées sur des constructeurs, est la *récurrence structurelle* [28], qui généralise le principe de Peano ainsi que les principes de récurrence mathématiques usuels.

Définition 4 (récurrence structurelle)**.** *Soit φ une formule à vérifier pour tout élément d'une sorte S à l'aide d'un schéma de récurrence à n conclusions. Le principe de la récurrence structurelle est défini par la formule :*

$$(\bigwedge_{i=1}^{n}(\forall x_1, \ldots, x_{n_i}, \varphi(x_{i_1}) \wedge \ldots \wedge \varphi(x_{i_k}) \Rightarrow \varphi(f_i(x_1, \ldots, x_{n_i}))) \Rightarrow \forall x, \varphi(x)$$

où les variables $x_{i_1}, \ldots, x_{i_k}$ sont celles parmi $x_1, \ldots, x_{n_i}$ qui ont la sorte S et toute fonction f_i ($i \in [1..n]$) est d'arité n_i et de codomaine S.

L'ordre de récurrence est implicite mais il existe en tant que relation « plus petite que » sur les naturels représentant la profondeur des arbres syntaxiques associés aux termes donnés comme arguments à φ.

A son tour, la récurrence structurelle est généralisée par la *récurrence sur des ensembles couvrants* [44] qui a été inspirée par l'idée proposée dans [8] selon laquelle les schémas de récurrence sont issus des définitions récursives des fonctions qui apparaissent dans la conjecture à prouver. La récurrence sur des ensembles couvrants suppose que chaque sorte S est caractérisée, ou *couverte*, par un ensemble fini de termes de sorte S appelé *ensemble couvrant* (de termes). La notion d'ensemble couvrant peut être généralisée par un ensemble de vecteurs de termes qui couvrent un produit de sortes $S_1 \times S_2 \times \ldots$

Définition 5 (récurrence sur des ensembles couvrants)**.** *Soient Ψ un ensemble couvrant non-vide $\{\overline{t_1}, \ldots, \overline{t_m}\}$ de vecteurs de termes de sorte $\mathscr{S}$ et φ une formule à vérifier pour tout élément de $\mathscr{S}$. Le principe de la récurrence sur des ensembles couvrants est défini par la formule :*

$$(\bigwedge_{i=1}^{m}(\bigwedge_{j=1}^{m_i} \forall \overline{k_i^j} \in \mathscr{S}, \overline{k_i^j} <_t \overline{t_i} \Rightarrow \varphi(\overline{k_i^j})) \Rightarrow \varphi(\overline{t_i})) \Rightarrow \forall \overline{t} \in \mathscr{S}, \varphi(\overline{t})$$

Les inconvénients les plus importants des principes de récurrence basés sur des schémas sont liés à la gestion des hypothèses de récurrence, lorsqu'elles sont i) définies mais inutilisées, et ii) nécessaires mais non incluses dans le schéma ou impossibles à produire par la méthode basée sur l'analyse de la récursivité. D'autre part, son avantage principal réside dans la définition locale de l'ordre de récurrence, qui se passe au niveau du schéma de récurrence. Ceci permet plus de flexibilité dans la gestion des ordres pendant le déroulement d'une preuve ; les contraintes d'ordre sont vérifiées une seule fois, au moment de la définition des schémas de récurrence. De plus, le raisonnement par récurrence basé sur des schémas peut être facilement intégré dans les *systèmes d'inférence* des démonstrateurs en termes de *règles d'inférence* correctes.

Un autre inconvénient serait présenté par la difficulté de traiter naturellement la *récurrence mutuelle*, adaptée au raisonnement sur des spécifications basées sur des définitions mutuellement récursives des fonctions ou des structures de données. Ceci est dû au fait que les hypothèses et les conclusions des schémas de récurrence sont des instances d'une *même* formule. Cependant, quelques solutions partielles ont été proposées pour palier cet inconvénient. Boyer et Moore [9] construisent des schémas de récurrence à partir d'une nouvelle fonction qui appelle les différentes fonctions définies mutuellement en fonction de la valeur d'un argument supplémentaire. Dans d'autres circonstances, la conjecture doit être généralisée ou des lemmes auxiliaires sont rajoutés par les utilisateurs pour spécifier des propriétés cruciales des fonctions mutuellement définies. Une solution plus automatique a été proposée par Kapur et Subramaniam [23] pour gérer une classe de fonctions mutuellement récursives par leur transformation dans des fonctions (simplement) récursives. L'idée est de déplier les définitions des fonctions lorsque les arguments sont des ensembles couvrants afin d'arriver aux appels simplement récursifs. Une autre idée serait de proposer des schémas de récurrence multi-prédicats [7] en associant un prédicat à chacune des fonctions mutuellement récursives, ce qui élimine le besoin de dépliage de fonctions. Cependant, si la conjecture intègre plusieurs symboles de fonction récursive, les schémas de récurrence doivent être combinés [8].

Principes de RNF

La technique de *preuve par récurrence sans récurrence*, aussi connue sous le nom de *preuve par cohérence*, est la première qui intègre une implémentation du principe de RNF. Proposée par Musser [29], elle utilise l'*algorithme de complétion* de Knuth-Bendix [25] pour prouver des propriétés inductives. La méthode peut prouver qu'un ensemble d'égalités est une conséquence d'un ensemble cohérent d'axiomes égalitaires orientables en règles de réécriture par i) leur ajout aux axiomes, ii) leur orientation dans des règles de réécriture, et iii) par leur cohérence avec les axiomes lorsque le processus de complétion *sature*, i.e, aucune nouvelle égalité n'est générée par rapport à un certain critère de redondance. Le processus de saturation demande que la

stratégie de preuve soit *équitable* afin de garantir le traitement ultérieur de toute nouvelle conjecture. Avec le temps, la méthode a été améliorée [20, 16, 17, 21, 22, 3, 27]. Pour un aperçu sur cette technique de preuve, le lecteur peut consulter [12, 13].

Sans utiliser la saturation, la *récurrence implicite* est une technique de preuve qui sépare les axiomes des conjectures à prouver. Comme la technique de preuve par cohérence, elle est basée sur la réécriture, ce qui permet de construire des règles d'inférence *réductives* qui remplacent à chaque étape de preuve une conjecture φ par un ensemble potentiellement vide de nouvelles conjectures qui sont soit trivialement vraies (e.g. tautologies), soit plus petites par rapport à (des instances de) φ. Dans ce dernier cas, les règles de réécriture utilisées doivent être issues soit des axiomes soit des autres conjectures plus petites que φ. Au début de son évolution, Reddy [31] a proposé la méthode appelée *la récurrence par réécriture de termes*, et mis au point une procédure qui calcule des *ensembles couvrants de formules*, obtenus par l'instanciation de certaines variables d'une conjecture avec les ensembles couvrants de termes qu'on associe aux produit de leurs sortes sous la forme de *substitutions couvrantes*. Bachmair [3] avait montré que les ensembles couvrants de formules sont fondamentaux pour les preuves par cohérence. Ceci reste aussi vrai pour les preuves par récurrence implicite.

Afin de prouver des propriétés inductives, la procédure de récurrence par réécriture de termes n'a pas besoin d'être *réfutationnellement complète* pour garantir la détection en temps fini des conjectures fausses, comme c'est demandé dans [3], ni des systèmes de réécriture qui soient *confluents sur les termes clos* [3]. Kounalis et Rusinowitch [26] sont allés plus loin et ont proposé une technique de preuve par récurrence basée sur des *ensembles tests* [22] qu'on peut définir comme des ensembles couvrants de termes qui garantissent la réduction par réécriture des ensembles couvrants de formules générés.

Définition 6 (récurrence basée sur la réécriture de termes)**.** *Soit* $\{\varphi\sigma_1, \ldots, \varphi\sigma_n\}$ *les instances de la formule* $\forall \bar{x} \in \mathscr{E}, \varphi(\bar{x})$ *construites avec les substitutions couvrantes* $\sigma_1, \ldots, \sigma_n$*, respectivement. Le principe de la récurrence basé sur la réécriture de termes est défini par la formule :*

$$(\bigwedge_{i=1}^{n}(\bigwedge_{j=1}^{m_i} \varphi\theta_{m_i} <_f \varphi\sigma_i \Rightarrow \varphi\theta_{m_i}) \Rightarrow \varphi\sigma_i) \Rightarrow \forall \bar{x} \in \mathscr{E}, \varphi(\bar{x})$$

Comme les implémentations du principe de RNT, les implémentations du principe de récurrence basé sur la réécriture de termes ne peut pas gérer naturellement le raisonnement par récurrence mutuelle parce que les formules manipulées sont toutes des instances d'une seule formule. Cet inconvénient a été écarté par la proposition de la méthode de preuve par récurrence implicite, dont l'idée a été suggérée dans [26], puis présentée formellement dans [10].

3. Cependant, ces propriétés sont nécessaires pour *réfuter* les fausses conjectures.

Définition 7 (récurrence implicite)**.** *Soit $\mathscr{E}$ l'ensemble des conjectures produites par une procédure de récurrence implicite et qui termine par un ensemble vide de conjectures. Le principe de la récurrence implicite est une application immédiate du principe de RNF où la condition $(\forall m \in \mathscr{E}, (\forall k \in \mathscr{E}, k <_f m \Rightarrow k) \Rightarrow m)$ de (6) est garantie* implicitement *par la procédure de preuve.*

Pour une formule m arbitraire, si elle est trivialement vraie on n'a pas besoin d'hypothèses de récurrence pour sa preuve. Si par absurde m est fausse, alors on peut construire une suite strictement décroissante infinie de fausses formules de $\mathscr{E}$, ce qui contredit le fait que $<_f$ est nœthérien. Puisque la conjecture à prouver fait partie de $\mathscr{E}$, on conclut qu'elle est vraie.

La récurrence implicite peut être aussi mise en œuvre avec une généralisation des ensembles couvrants de formules appelés *ensembles couvrants contextuels* [33]. Le nombre de contraintes d'ordre à respecter peut également être réduit si on utilise un système de preuves *cyclique* [35], où on impose des contraintes d'ordre seulement aux conjectures qui, d'un point de vue de la théorie des modèles, dépendent mutuellement les unes des autres. Des travaux plus récents [36, 38, 40], laissent penser que la récurrence cyclique utilisée dans des cadres logiques de premier ordre autres que la logique équationnelle, comme celui qui inclut des prédicats inductifs [11], peut être expliquée par des principes de RNF. Quelquefois, les preuves par récurrence implicite sont plus automatiques que celles basées sur la récurrence explicite [6], dans d'autres cas il se produit le contraire [23]. D'autres analyses et comparaisons ont été faites dans [18, 21, 24, 30, 12, 43].

La RNF peut mieux gérer la récurrence mutuelle puisqu'elle accepte l'usage d'instances de différentes formules [34]. Elle permet aussi la récurrence *paresseuse* où les hypothèses de récurrence sont fournies sur demande et sont connues seulement au moment de leur application. En contrepartie, elle est peu utilisée par les systèmes de preuve actuels à cause de la difficulté de l'implémenter par une seule règle d'inférence et de gérer les dépendances d'ordre au niveau des preuves.

Relations entre les instances basées sur des termes et des formules

Nous allons maintenant étudier les rapports entre les deux types d'instances de premier ordre du principe de récurrence nœthérienne. Le théorème suivant est important aussi bien d'un point de vue théorique que pratique.

Théorème 2. *Tout principe de RNT peut être aussi représenté comme instance basée sur des formules.*

Démonstration. Soient $\mathscr{E}$ un ensemble non-vide de vecteurs de termes et φ une propriété vérifiée pout tout élément de $\mathscr{E}$ avec le principe de RNT suivant :

$$(\forall \overline{m} \in \mathscr{E}, (\forall \overline{k} \in \mathscr{E}, \overline{k} <_t \overline{m} \Rightarrow \varphi(\overline{k})) \Rightarrow \varphi(\overline{m})) \Rightarrow \forall \overline{p} \in \mathscr{E}, \varphi(\overline{p})$$

Soit $\mathscr{E}'$ l'ensemble $\{\varphi(\overline{p}) \mid \overline{p} \in \mathscr{E}\}$, on peut définir le principe de RNF comme suit :

$$(\forall \varphi(\overline{m}) \in \mathscr{E}', (\forall \varphi(\overline{k}) \in \mathscr{E}', \varphi(\overline{k}) <_f \varphi(\overline{m}) \Rightarrow \varphi(\overline{k})) \Rightarrow \varphi(\overline{m})) \Rightarrow \forall \varphi(\overline{p}) \in \mathscr{E}', \varphi(\overline{p})$$

où $\varphi(\overline{k}) <_f \varphi(\overline{m})$ si $\overline{k} <_t \overline{m}$. □

Corollaire 1. *Toute preuve par RNT peut être reconstruite en tant que preuve par RNF.*

D'un point de vue pratique, le raisonnement par RNT peut être *directement* intégré dans les systèmes de preuves qui implémentent le principe de RNF.

La traduction dans le sens inverse est aussi valable pour le cas suivant :

Théorème 3. *Le principe de RNF, définit par (6), peut être aussi représenté comme une instance basée sur des termes lorsque $\mathscr{E}$ est formé seulement par des instances d'une même formule.*

Démonstration. Soit le principe de RNF défini par la formule :

$$(\forall m \in \mathscr{E}, (\forall k \in \mathscr{E}, k <_f m \Rightarrow k) \Rightarrow m) \Rightarrow \forall p \in \mathscr{E}, p$$

où $\mathscr{E} = \{\varphi(\overline{x_1}), \ldots, \varphi(\overline{x_n}), \ldots\}$. Soit $\mathscr{E}'$ le nouvel ensemble $\{\overline{p} \mid \varphi(\overline{p}) \in \mathscr{E}\}$. On peut définir ainsi le principe de RNT :

$$(\forall \overline{m} \in \mathscr{E}', (\forall \overline{k} \in \mathscr{E}', \overline{k} <_t \overline{m} \Rightarrow \varphi(\overline{k})) \Rightarrow \varphi(\overline{m})) \Rightarrow \forall \overline{p} \in \mathscr{E}', \varphi(\overline{p})$$

où $\overline{k} <_t \overline{m}$ si $\varphi(\overline{k}) <_f \varphi(\overline{m})$. □

La validité du Théorème 3 pour le cas général reste un problème ouvert :

Conjecture 1. *Tout principe de RNF peut être aussi représenté en tant qu'instance basée sur des termes.*

Une conjecture similaire a été proposée dans [11], dans le cadre de la logique de premier ordre avec des définitions inductives, mais a été invalidée ultérieurement [4].

Certification avec Coq du raisonnement par RNF

La formalisation en Coq du principe de RNF suit des idées présentées dans [34, 41] et utilisées pour certifier automatiquement des preuves par récurrence implicite développées avec le démonstrateur SPIKE [39]. D'abord, on associe à chaque formule une *mesure* qui va nous permettre de comparer des formules et de définir l'ordre de récurrence. Étant donné une liste LF de paires de la forme $(\varphi,\ \mu_\varphi)$, où φ est une formule et μ_φ sa mesure, (6) peut être reformulée comme suit :

$$(\forall p \in \mathsf{LF}, (\forall p' \in \mathsf{LF}, snd(p') <_f snd(p) \Rightarrow fst(p')) \Rightarrow fst(p)) \Rightarrow \forall f \in \mathscr{E}, fst(f),$$

où *fst* (resp., *snd*) est la fonction qui retourne la première (resp., deuxième) projection d'une paire. Dans Coq, la partie conditionnelle de l'implication la plus extérieure peut être formalisée par le lemme suivant :

`Lemma` main : $\forall p$, In p LF $\rightarrow$ ($\forall p'$, In p' LF $\rightarrow$ less (snd p') (snd p)) $\rightarrow$ fst p') $\rightarrow$ fst p.

L'ordre de récurrence, noté less, est une implémentation de $<_f$ en tant qu'extension multi-ensemble d'un ordre rpo [2] sur les termes. La mesure d'une formule peut être représentée par un multi-ensemble de ses sous-termes. Étant donné deux paires $(\varphi_1, \mu_{\varphi_1})$ et $(\varphi_2, \mu_{\varphi_2})$, on dit que φ_1 est *plus petit que* φ_2 si less μ_{φ_1} μ_{φ_2}.

Les étapes de récurrence principales pour prouver la validité d'une formule φ de chaque paire de LF sont : i) la partie déductive : choisir des (instances de) formules de LF en tant qu'hypothèses de récurrence à utiliser dans la preuve de φ, et ii) la partie des contraintes d'ordre : montrer que ces hypothèses de récurrence sont plus petites que φ. La partie ii) peut être omise si la partie i) ne nécessite pas de raisonnement par récurrence.

L'ordre less a été mis en œuvre avec les représentations syntaxiques des termes fournies par la bibliothèque COCCINELLE [14, 15] qui permet de modéliser des notions mathématiques pour la réécriture, comme les algèbres de termes et les ordres de récurrence, ainsi que par la bibliothèque CoLoR [5], utilisée pour implémenter la relation « extension multi-ensemble ».

Les mesures doivent suivre les instanciations des formules. C'est pour cette raison que la liste LF du lemme main a été adaptée pour inclure des fonctions anonymes à la place des paires afin de pouvoir partager les variables communes entre les formules et leurs mesures :

`Definition` type_LF := *argument_sort* $\rightarrow$ (`Prop` $\times$ (**List.list term**)),

où *argument_sort* est la sorte écrite sous forme curryfiée de la plus générale version du vecteur de variables partagées par chaque paire de LF. La définition du lemme main dans la nouvelle mouture est :

`Lemma` main : $\forall f$, In f LF $\rightarrow$ $\forall u$, ($\forall f'$, In f' LF $\rightarrow$ $\forall u'$, less (snd (f' u')) (snd (f u)) $\rightarrow$ fst (f' u')) $\rightarrow$ fst (f u).

Toute formule de la liste LF, dont les conjectures initiales, peut être certifiée si on prouve le théorème all_true. Ceci nécessite l'utilisation du lemme main et du principe de récurrence nœthérienne (1) qui est intégré nativement dans Coq :

`Theorem` all_true : $\forall f$, In f LF $\rightarrow$ $\forall u$: **nat**, fst (f u).

Conclusions et travaux à venir

Nous avons présenté une taxonomie du raisonnement par récurrence nœthérienne de premier ordre, ainsi qu'un survol de ses instances les plus emblématiques : basées sur des termes et des formules. La traduction des preuves intégrant des instances basées sur des formules $\rightarrow$ termes reste un problème ouvert. On a montré sa validité seulement pour certains cas (voir Théorème 3). Nous sommes en train d'étudier d'autres cas, par exemple, les preuves avec des séquents dans des logiques multi-sortées de premier ordre avec des prédicats inductifs. Nous n'avons pas discuté des problématiques importantes de la preuve par récurrence, comme la réfutation des fausses conjectures, le choix des variables de récurrence, l'usage des hypothèses de récurrence, l'ajout de nouveaux lemmes, la généralisation des conjectures, ainsi que le choix des stratégies de preuve. Le lecteur intéressé trouvera plus de détails ailleurs [19].

La formalisation en Coq du principe de RNF nous a permis de certifier des preuves par récurrence implicite générées automatiquement par SPIKE, concernant entre autres des lemmes cruciaux pour valider la conformité d'un protocole de télécommunications [32]. On envisage d'appliquer la même méthodologie de certification sur des preuves cycliques, comme celles produites avec E-CYCLIST [40].

Références

[1] R. AUBIN : Mechanizing structural induction. *Theor. Comput. Sci.*, 9:329–362, 1979.

[2] F. BAADER et T. NIPKOW : *Term Rewriting and All That*. Cambridge University Press, 1998.

[3] L. BACHMAIR : Proof by consistency in equational theories. *Logic in Computer Science, 1988. LICS '88., Proceedings of the Third Annual Symposium on*, pages 228–233, 1988.

[4] S. BERARDI et M. TATSUTA : Classical system of Martin-Lof's inductive definitions is not equivalent to cyclic proofs. *Logical Methods in Computer Science*, 15(3), 2019.

[5] F. BLANQUI et A. KOPROWSKI : CoLoR : a Coq library on well-founded rewrite relations and its application to the automated verification of termination certificates. *MSCS*, 21(4):827–859, 2011.

[6] A. BOUHOULA et M. RUSINOWITCH : Implicit induction in conditional theories. *Journal of Automated Reasoning*, 14(2):189–235, 1995.

[7] R. BOULTON et K. SLIND : Automatic derivation and application of induction schemes for mutually recursive functions. *In* J. LLOYD, V. DAHL, U. FURBACH, M. KERBER, K.-K. LAU, C. PALAMIDESSI, L. PEREIRA, Y. SAGIV et P. STUCKEY, éditeurs : *Computational Logic — CL 2000*, volume 1861 de *Lecture Notes in Computer Science*, pages 629–643. Springer Berlin / Heidelberg, 2000.

[8] R. S. BOYER et J S. MOORE : *A Computational Logic*. Academic Press, New York, NY, 1979.

[9] R. S. BOYER et J S. MOORE : *A Computational Logic Handbook*. Academic Press Professional, 1988.

[10] F. BRONSARD, U.S. REDDY et R. HASKER : Induction using term orderings. *In CADE (Conf. on Automated Deduction)*, volume 814 de *LNCS*, pages 102–117. Springer, 1994.

[11] J. BROTHERSTON et A. SIMPSON : Sequent calculi for induction and infinite descent. *Journal of Logic and Computation*, 21(6):1177–1216, 2011.

[12] H. COMON : Inductionless induction. *In* A. ROBINSON et A. VORONKOV, éditeurs : *Handbook of Automated Reasoning*, pages 913–962. Elsevier and MIT Press, 2001.

[13] H. COMON et R. NIEUWENHUIS : Induction= I-axiomatization+ first-order consistency. *Information and Computation(Print)*, 159(1-2):151–186, 2000.

[14] E. CONTEJEAN, P. COURTIEU, J. FOREST, O. PONS et X. URBAIN : Certification of automated termination proofs. *Frontiers of Combining Systems*, pages 148–162, 2007.

[15] E. CONTEJEAN, A. PASKEVICH, X. URBAIN, P. COURTIEU, O. PONS et J. FOREST : A3PAT, an approach for certified automated termination proofs. *In* J. P. GALLAGHER et J. VOIGTLÄNDER, éditeurs : *PEPM - Proceedings of the 2010 ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation, PEPM 2010, Madrid, Spain*, pages 63–72. ACM, 2010.

[16] N. DERSHOWITZ : Applications of the Knuth-Bendix completion procedure. *In Seminaire d'Informatique Theorique*, pages 95–111, 1982.

[17] L. FRIBOURG : A strong restriction of the inductive completion procedure. *Journal of Symbolic Computation*, 8(3):253 – 276, 1989.

[18] S. J. GARLAND et J. V. GUTTAG : Inductive methods for reasoning about abstract data types. *Proceedings of the 15th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 219–228, 1988.

[19] B. GRAMLICH : Strategic issues, problems and challenges in inductive theorem proving. *Electronic Notes in Theoretical Computer Science*, 125(2):5–43, 2005.

[20] G. HUET et J.M. HULLOT : Proofs by induction in equational theories with constructors. Rapport technique 0028, INRIA, 1980.

[21] J.-P. JOUANNAUD et E. KOUNALIS : Automatic proofs by induction in theories without constructors. *Information and Computation*, 82(1):1 – 33, 1989.

[22] D. KAPUR, P. NARENDRAN et H. ZHANG : Proof by induction using test sets. *In 8th International Conference on Automated Deduction*, volume 230 de *Lecture Notes Computer Science*, pages 99–117. Springer, 1986.

[23] D. KAPUR et M. SUBRAMANIAM : Automating induction over mutually recursive functions. *In Algebraic Methodology and Software Technology*, volume 1101 de *LNCS*, pages 117–131. Springer, 1996.

[24] D. KAPUR et H. ZHANG : Automating induction : Explicit vs. inductionless. *Proc. Third International Symposium on Artificial Intelligence and Mathematics, Fort Lauderdale, Florida*, pages 2–5, 1994.

[25] D.E. KNUTH et P.B. BENDIX : Simple word problems in universal algebras. *In Computational Problems in Abstract Algebra*, pages 263–297, 1970.

[26] E. KOUNALIS et M. RUSINOWITCH : Mechanizing inductive reasoning. *In Proceedings of the Eighth National Conference on Artificial Intelligence - Volume 1*, AAAI'90, pages 240–245. AAAI Press, 1990.

[27] W. KÜCHLIN : Inductive completion by ground proof transformation. *In* H. AIT-KACI et M. NIVAT, éditeurs : *Resolution of Equations in Algebraic Structures (Volume II) : Rewriting Techniques*, pages 211–244. Academic Press, London, 1989.

[28] J. MCCARTHY et J. PAINTER : Correctness of a compiler for arithmetic expressions. *In Mathematical Aspects of Computer Science*, pages 33–41. American Mathematical Society, 1967.

[29] D. R. MUSSER : On proving inductive properties of abstract data types. *In POPL*, pages 154–162, 1980.

[30] D. NAIDICH : On generic representation of implicit induction procedures. Rapport technique CS-R9620, CWI, 1996.

[31] U.S. REDDY : Term rewriting induction. *Proceedings of the 10th International Conference on Automated Deduction*, pages 162–177, 1990.

[32] M. RUSINOWITCH, S. STRATULAT et F. KLAY : Mechanical verification of an ideal incremental ABR conformance algorithm. *Journal of Automated Reasoning*, 30(2):153–177, 2003.

[33] S. STRATULAT : A general framework to build contextual cover set induction provers. *Journal of Symbolic Computation*, 32(4):403–445, 2001.

[34] S. STRATULAT : Integrating implicit induction proofs into certified proof environments. *In IFM'2010 (8th International Conference on Integrated Formal Methods)*, volume 6396 de *Lecture Notes in Computer Science*, pages 320–335, 2010.

[35] S. STRATULAT : A unified view of induction reasoning for first-order logic. *In* A. VORONKOV, éditeur : *Turing-100 (The Alan Turing Centenary Conference)*, volume 10 de *EPiC Series*, pages 326–352. EasyChair, 2012.

[36] S. STRATULAT : Cyclic proofs with ordering constraints. *In* R. A. SCHMIDT et C. NALON, éditeurs : *TABLEAUX 2017 (26th International Conference on Automated Reasoning with Analytic Tableaux and Related Methods)*, volume 10501 de *LNAI*, pages 311–327. Springer, 2017.

[37] S. STRATULAT : Mechanically certifying formula-based Noetherian induction reasoning. *Journal of Symbolic Computation*, 80, Part 1:209–249, 2017.

[38] S. STRATULAT : Validating back-links of $\mathrm{FOL}_{\mathrm{ID}}$ cyclic pre-proofs. *In* S. BERARDI et S. van BAKEL, éditeurs : *CL&C'18 (Seventh International Workshop on Classical Logic and Computation)*, numéro 281 de EPTCS, pages 39–53, 2018.

[39] S. STRATULAT : SPIKE, an automatic theorem prover – revisited. *In SYNASC 2020 : Proceedings of the 22nd International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pages 93–96. IEEE Computer Society, 2020.

[40] S. STRATULAT : E-Cyclist : Implementation of an efficient validation of $\mathrm{FOL}_{\mathrm{ID}}$ cyclic induction reasoning. *In* T. KUTSIA, éditeur : *9th International Symposium on Symbolic Computation in Software Science*, volume 342 de *Electronic Proceedings in Theoretical Computer Science*, pages 129–135, juin 2021.

[41] S. STRATULAT et V. DEMANGE : Automated certification of implicit induction proofs. *In CPP'2011 (First International Conference on Certified Programs and Proofs)*, volume 7086 de *Lecture Notes Computer Science*, pages 37–53. Springer Verlag, 2011.

[42] THE COQ DEVELOPMENT TEAM : *The Coq Reference Manual*. INRIA, 2020. `http://coq.inria.fr/doc`.

[43] C.-P. WIRTH : History and future of implicit and inductionless induction : Beware the old jade and the zombie ! *In Mechanizing Mathematical Reasoning : Essays in Honor of Jörg H. Siekmann on the Occasion of His 60th Birthday*, numéro 2605 de Lecture Notes in Artificial Intelligence, pages 192–203. Springer, 2005.

[44] H. ZHANG, D. KAPUR et M. S. KRISHNAMOORTHY : A mechanizable induction principle for equational specifications. *In Proceedings of the 9th International Conference on Automated Deduction*, pages 162–181, London, UK, 1988. Springer-Verlag.