Décision des théories à nombre borné de variables sur les langages relationnels finis : un algorithme utilisant les destinées

Annie Chateau

Laboratoire de Logique, Algorithmique et Informatique de Clermont 1 (LLAIC1)
IUT de Clermont-Ferrand
Département Informatique — BP86
63172 Aubière Cédex
email : chateau@llaic3.u-clermont1.fr

Résumé

Dans cet article est présenté un algorithme de décision pour les formules de la logique du premier ordre sur un langage relationnel fini, avec un nombre borné de variables. Lorsqu'on se fixe un tel langage $\mathcal{L}$, et un nombre de variables fixé p, on sait qu'il existe un algorithme qui, prenant une formule en entrée, termine et indique si cette formule est vraie ou non. On dit que la théorie (ensemble de ces formules) est décidable. L'algorithme présenté ici réalise cette décision en s'appuyant sur un outil logique puissant, les p-destinées.

Mots clés

décidabilité effective, nombre borné de variables, destinées

Introduction

Les destinées ont été introduites par Francis Nézondet dans sa thèse (voir (Nézondet, 1997)). Encore peu étudiées à l'heure actuelle, elles sont cependant les héritières de plusieurs domaines fondamentaux de la logique mathématique. Tout d'abord, elles s'inscrivent en plein dans le domaine de la théorie des modèles, car elles représentent elles-mêmes un modèle "adapté" d'une théorie complète sur un langage relationnel fini avec un nombre borné de variables (Si le lecteur n'est pas familier avec la notion de théorie, qu'il se rassure, les objets présentés se comprennent très bien intuitivement sur des exemples). Ensuite, elles touchent également de près la théorie de la décision, car elles fournissent un algorithme extrêmement simple de décision de ces théories, que nous exposons ici (au paragraphe 2). On peut également leur trouver des ressemblances conceptuelles avec les jeux d'Ehrenfeucht-Fraïssé, ou encore les bases de données relationnelles. La présentation qui en est faite ici se veut très simplifiée, une présentation succinte peut être trouvée en annexe A, et une présentation complète dans (NÉZONDET, 1997). L'algorithme de décision n'a en lui-même rien de compliqué, mais a ceci de très intéressant qu'il peut s'adapter à toute théorie, *pourvu que les destinées de cette théorie aient été construites*, ce qui peut être en revanche très difficile.

1 Présentation des destinées à travers un exemple

1.1 Les p-destinées de l'ordre et de la primarité, introduction au problème

Afin de comprendre les destinées, nous allons construire un exemple de 2-transversale, c'est-à-dire que nous allons construire un objet qui va résumer tout ce que l'on peut exprimer avec seulement deux variables dans le contexte suivant : on considère l'ensemble des entiers naturels et le langage de l'ordre $<$ et de la primarité (P est le prédicat unaire "être premier"). On a envie d'avoir un archétype de toutes les situations possibles qui peuvent se produire avec deux variables. Pour cela on aligne les entiers naturels, et on commence par construire une forêt d'arbres de hauteur 2, comme dans la figure 1.

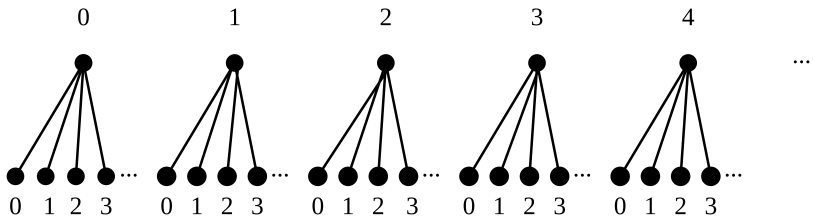

FIG.1 – *Forêt d'entiers de hauteur 2*

Ensuite, on veut répertorier toutes les situations possibles, c'est-à-dire que l'on va indiquer les relations existant (dans l'ensemble des entiers naturels, et ne concernant que l'ordre et la primarité) entre les racines des arbres et leurs fils. On commence à travailler sur le premier arbre, celui de zéro. On exprime que le nombre est premier par un cercle, et la relation d'ordre par un symbole $>$, $=$ ou $<$. Cela nous donne la figure 2.

On parle alors d'une destinée de 0 pour désigner l'arbre avec les indications des relations entre les nœuds. On dit qu'elle est *exhaustive*, car toutes les situations faisant intervenir deux entiers dont l'un est 0 et exprimables uniquement avec l'ordre et le prédicat "être premier" apparaissent dans cet arbre.

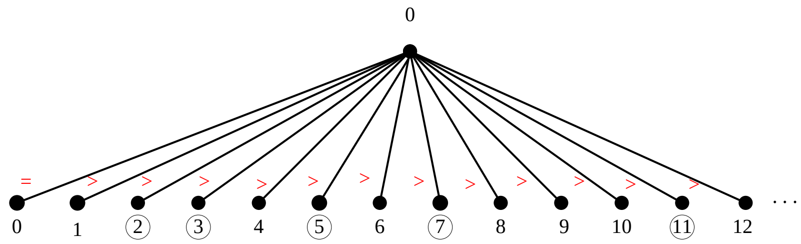

Fig.2 – *Destinée exhaustive mais non essentielle de racine 0*

1.2 Essentialité : élimination des redondances

On voit en fait qu'une même situation peut être représentée plusieurs fois, par exemple tous les nombres au-dessus de 1 vont être strictement supérieurs à 0, et il y en a de deux catégories : ceux qui sont premiers, et ceux qui ne sont pas premiers. On peut donc garder uniquement 1 et 2, les autres entiers supérieurs auront le même type de comportement vis-à-vis de zéro que l'un de ces deux nombres. Ce qui nous donne la figure 3.

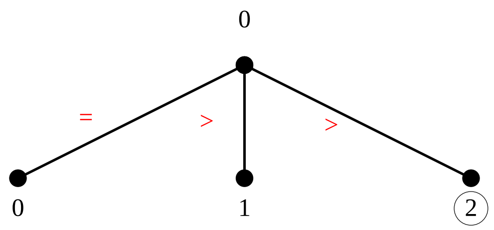

Fig.3 – *Destinée exhaustive essentielle de racine 0*

On a désormais un arbre fini, sans redondance infinie de branches "isomorphes" (figurée par ". . ." sur la figure). On dit alors qu'on a simplifié la destinée de 0, et on obtient alors une destinée *essentielle*.

Continuons le même genre de construction pour les autres entiers. On aura ainsi la forêt représentée sur la figure 4.

1.3 Transversale

On remarque que les arbres de racines 3 et 5 ont la même "tête", de même que les arbres de racines 4 et 6. Avoir la même "tête" signifie être $\{<, P\}$-isomorphes. On peut donc éliminer les arbres de racines 5 et 6 sans perdre d'information sur la théorie à deux variables des entiers naturels munis de l'ordre et de la

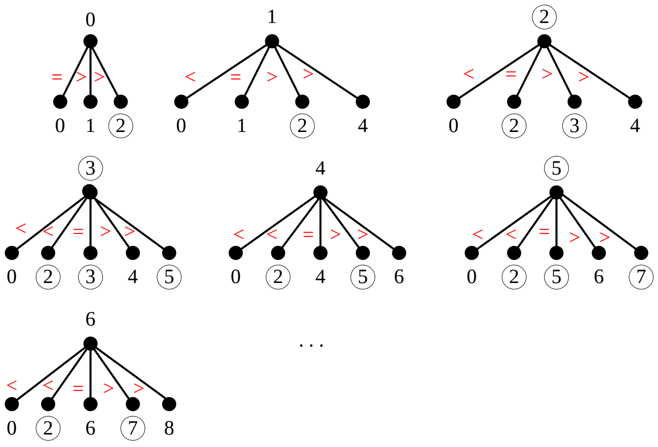

FIG.4 – *Destinées exhaustives, simplifiées au rang 2 de* $Th_2\{\mathbb{N}, <, P\}$

primarité. Si on poursuit la construction pour tous les entiers, on s'aperçoit qu'on a en fait toujours des schémas qu'on avait parmi les 5 premiers. En effet, si on prend un nombre entier quelconque $n \geq 4$:

- Si n n'est pas premier : un entier par rapport à n peut être
  - non-premier et strictement inférieur (0)
  - premier et strictement inférieur (2)
  - non-premier et égal (n)
  - le cas premier et égal est impossible
  - non-premier et strictement supérieur ($2n$)
  - premier et strictement supérieur (il en existe d'après le théorème d'Euler)

  Et ce sont tous les cas possibles. Alors la destinée essentielle (non redondante) de racine n est isomorphe à celle de 4.
- si n est premier : un entier par rapport à n peut être
  - non-premier et strictement inférieur (0)
  - premier et strictement inférieur (2)
  - premier et égal (n)
  - le cas non-premier et égal est impossible
  - non-premier et strictement supérieur ($2n$)
  - premier et strictement supérieur (il en existe d'après le théorème d'Euler)

  Et ce sont tous les cas possibles. Alors la destinée essentielle (non redondante) de racine n est isomorphe à celle de 3.

On décrit donc tous les cas avec l'ensemble de destinées de la figure 5, que l'on appelle la 2-transversale de la théorie.

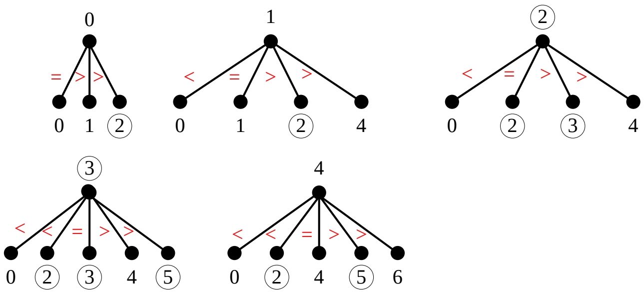

FIG.5 – *transversale de* $Th_2\{\mathbb{N}, <, P\}$

2 Algorithme de décision de la théorie à partir de la p-transversale

2.1 Exemple

Restons dans le cadre de la théorie complète des entiers munis du langage de l'ordre et de la primarité, à deux variables. On sait qu'il existe un nombre fini de formules à deux variables, à équivalence logique près, dans ce langage, puisqu'il s'agit d'un langage relationnel fini. On veut décider ces formules, c'est-à-dire disposer d'un algorithme permettant de savoir s'il s'agit de théorèmes ou non. On va voir que la 2-transversale construite précédemment nous fournit naturellement un tel algorithme. Découvrons-le sur un exemple : On choisit une formule à deux quantificateurs : $\forall x \exists y (P(y) \wedge (P(x) \vee (x < y)))$. Imaginons quelques minutes que nous ne savons pas s'il s'agit d'un théorème ou non, et notons $F[x, y] = P(y) \wedge (P(x) \vee (x < y))$.
Décomposons cette formule :

- le premier quantificateur est un $\forall$, il va donc falloir donner à x successivement toutes les valeurs des racines de la destinée et faire tourner l'algorithme sur chacun de ces arbres : si tous les résultats sont positifs (la formule est un théorème quand on attribue à x la valeur de la racine), c'est que la formule est un théorème ; si l'un des résultats est négatif, c'est que la formule n'est pas un théorème, et on a un contre-exemple.
- le deuxième quantificateur est un $\exists$, on va donc encore une fois, pour une valeur donnée de x parmi les valeurs des racines, donner successivement à y toutes les valeurs des fils de x dans la destinée. Cette fois les critères de "réussite" sont différents : si l'un des résultats est positif, c'est que la formule est vraie quand on attribue à x la valeur de la racine, et on a un y explicite ; si tous les résultats sont négatifs, alors c'est que la formule est fausse quand x prend la valeur de la racine.

Faisons tourner ce pseudo algorithme sur la formule choisie :

- Lorsque x prend la valeur 0 :

- Lorsque y prend la valeur 0 : 0 n'est pas premier donc $F[x, y]$ est fausse : $\Rightarrow$ Faux
  - Lorsque y prend la valeur 1 : 1 n'est pas premier donc $F[x, y]$ est fausse : $\Rightarrow$ Faux
  - Lorsque y prend la valeur 2 : 2 est premier et $0 < 2$ donc $F[x, y]$ est vraie : $\Rightarrow$ Vrai

  L'un des cas est "Vrai" donc $\exists y F[0, y]$ est vraie : $\Rightarrow$ Vrai
- Lorsque x prend la valeur 1 :
  - Lorsque y prend la valeur 0 : 0 n'est pas premier donc $F[x, y]$ est fausse : $\Rightarrow$ Faux
  - Lorsque y prend la valeur 1 : 1 n'est pas premier donc $F[x, y]$ est fausse : $\Rightarrow$ Faux
  - Lorsque y prend la valeur 2 : 2 est premier et $0 < 2$ donc $F[x, y]$ est vraie : $\Rightarrow$ Vrai

  On peut s'arrêter là dans l'étude de cas, puisqu'il nous fallait au moins un résultat positif pour dire que $\exists y F[0, y]$ est vraie : $\Rightarrow$ Vrai
- Lorsque x prend la valeur 2 :
  - Lorsque y prend la valeur 0 : 0 n'est pas premier donc $F[x, y]$ est fausse : $\Rightarrow$ Faux
  - Lorsque y prend la valeur 2 : 2 est premier et 2 est premier donc $F[x, y]$ est vraie : $\Rightarrow$ Vrai

  On peut s'arrêter là dans l'étude de cas, puisqu'il nous fallait au moins un résultat positif pour dire que $\exists y F[0, y]$ est vraie : $\Rightarrow$ Vrai
- Lorsque x prend la valeur 3 :
  - Lorsque y prend la valeur 0 : 0 n'est pas premier donc $F[x, y]$ est fausse : $\Rightarrow$ Faux
  - Lorsque y prend la valeur 2 : 2 est premier et 3 est premier donc $F[x, y]$ est vraie : $\Rightarrow$ Vrai

  On peut s'arrêter là dans l'étude de cas, puisqu'il nous fallait au moins un résultat positif pour dire que $\exists y F[0, y]$ est vraie : $\Rightarrow$ Vrai
- Lorsque x prend la valeur 4 :
  - Lorsque y prend la valeur 0 : 0 n'est pas premier donc $F[x, y]$ est fausse : $\Rightarrow$ Faux
  - Lorsque y prend la valeur 2 : 2 est premier mais 4 n'est pas premier et $4 > 3$ donc $F[x, y]$ est fausse : $\Rightarrow$ Faux
  - Lorsque y prend la valeur 4 : 4 n'est pas premier donc $F[x, y]$ est fausse : $\Rightarrow$ Faux
  - Lorsque y prend la valeur 5 : 5 et premier et $4 < 5$ donc $F[x, y]$ est vraie : $\Rightarrow$ Vrai

  On peut s'arrêter là dans l'étude de cas, puisqu'il nous fallait au moins un résultat positif pour dire que $\exists y F[0, y]$ est vraie : $\Rightarrow$ Vrai

Dans tous les cas on a un résultat positif : la formule est donc un théorème.

On voit qu'on pourrait, en formalisant un peu mieux l'algorithme précédent à tous les cas de quantification rencontrés, décider de toutes les formules à deux variables dans ce langage.

2.2 Formalisation de l'algorithme

L'algorithme peut être décrit de la façon suivante : il prend en entrée une formule et une transversale, que l'on peut voir comme une grosse destinée de racine $\emptyset$ et de *pétioles* les destinées de la transversale, où la notion de pétiole est définie comme suit :

Définition 1 *Un* pétiole *de destinée est un sous-arbre construit en prenant tout le sous-arbre ayant pour racine l'un des nœuds.*

On analyse la syntaxe de la formule, et selon les quantifications, on adopte une stratégie légèrement différente. On peut l'écrire de la façon suivante :
Début *verification*$(F,\, dest)$

1. Si la formule F est de la forme $\forall x G[x]$ alors
   Parcourir toutes les valeurs x des racines des pétioles de la destinée, renvoyer ET[*verification*$(G(x),\, petiole(x))$, x racine de pétiole]
2. Sinon si la formule F est de la forme $\exists x G[x]$ alors
   Parcourir toutes les valeurs x des racines des pétioles de la destinée, renvoyer OU[*verification*$(G(x),\, petiole(x))$, x racine de pétiole]
3. Sinon si la formule F est de la forme GbH où b est un opérateur booléen, alors
   Renvoyer [*verification*$(G,\, dest)$]b[*verification*$(H,\, dest)$]
4. Sinon si la formule F est de la forme $\neg G$ alors
   Renvoyer $\neg$*verification*$(G,\, dest)$
5. Sinon la formule est un prédicat $R(x_1, x_2, x_3, \ldots, x_k)$ $(k \leq p)$ avec x_1, x_2, ..., x_k des valeurs instanciées précédemment, alors
   Si le prédicat est vérifié, Renvoyer vrai
6. Sinon, Renvoyer faux

Fin

2.3 Implémentation en langage Caml

L'algorithme fonctionne dans un environnement interactif. Dans la version actuellement implémentée, le programme prend en entrée une formule tapée soit dans un fichier soit directement sur la ligne de commande, et en utilisant par exemple la 2-transversale de $\mathbb{N}$ muni de l'ordre et de la primarité, indique s'il s'agit d'un théorème ou non. On définit un type `formule` et un type `destinee` pour manipuler les objets correspondants. Ces types ont été définis de manière récursive, et se présentent de la façon suivante dans la syntaxe du langage Caml :

Type `formule` :

```
type formule =
  | Vrai
  | Faux
  | Non of formule
  | Et of formule * formule
  | Ou of formule * formule
  | Implique of formule * formule
  | Equivalent of  formule * formule
  | Forall of formule * formule
  | Exists of formule * formule
  | Copremier of formule * formule
  | Successeur of formule * formule
  | Variable of string
;;
```

Type `destinee` **:**

```
type destinee =
  Empty
  | Node of int * int  * bool list * destinee list
;;
```

Dans la description d'un nœud du type `destinee` :

- le premier entier correspond au niveau courant ;
- le deuxième entier correspond à la valeur du nœud de la destinée ;
- la liste de booléens figure les relations des nœuds entre eux ;
- la liste de destinées est la liste des pétioles des fils du nœud courant.

La transversale de la théorie de $\mathbb{N}$ avec l'ordre et la primarité à deux variables est alors exprimée sous forme d'une liste de destinées.

L'implémentation consiste en plusieurs modules :

- un analyseur syntaxique qui transforme la chaîne de caractères représentant la formule en un objet de type `formule` ;
- une fonction prenant en entrée la transversale et la formule, donnant une réponse à la question "Cette formule est-elle un théorème ?" ;
- un environnement interactif avec l'utilisateur.

Le détail de cette implémentation est présenté dans (Chateau, 2000).

Conclusion

L'interêt des destinées ne se limite pas à cet algorithme de décision. Cette notion d'arbre structuré "résumant" la théorie pose un certain nombre de questions théoriques. Par exemple, on peut se demander si la taille des transversales, que l'on peut définir comme étant une notion de "degré de la transversale", est significative d'une certaine complexité de la théorie. On peut également rechercher les théories pour lesquelles on va pouvoir effectivement construire les destinées (on sait à présent qu'on peut le faire au moins pour les théories de l'ordre et l'arithmétique de Presburger à nombre borné de variables, et que cet algorithme de construction de destinées est généralisable pour toute théorie décidable dont l'algorithme de décision fournit des bornes sur la taille des transversales), c'est-à-dire écrire un algorithme qui prend en entrée le nombre de variables et engendre la p-transversale. Ces différentes pistes, qui ont certainement entre elles des liens profonds, peuvent permettre d'affiner la classification des théories décidables en théories "effectivement décidables" et "ineffectivement décidables".

A Définition des p-destinées

On peut décrire informellement les p-destinées comme des arbres dont toutes les branches ont même hauteur p, où p est le nombre de variables intervenant dans la théorie correspondante. Ces arbres sont étiquetés par des listes d'éléments d'un modèle de la théorie, et sont munis d'une opération de structuration qui permet d'exprimer sous forme arborescente ce qui se passe dans la théorie.

Les destinées peuvent être définies en tant qu'objets indépendants d'une théorie mais ici par commodité nous ne nous intéresserons qu'au cas des destinées de la théorie complète à p variables d'une $\mathcal{L}$-structure, où $\mathcal{L}$ est un langage relationnel fini.

On se fixe donc $\mathcal{X}$ une $\mathcal{L}$-structure (par exemple l'ensemble des entiers naturels $\mathbb{N}$ vu en tant que structure sur le langage $\{P, <\}$, et $\mathcal{T}_p$ la théorie complète de $\mathcal{X}$ à p variables (c'est-à-dire l'ensemble des formules à p variables vraies dans $\mathcal{X}$). Pour définir une destinée d'un élément $x \in \mathcal{X}$, on peut commencer par définir son *arbre $\mathcal{X}$-complet de hauteur* p : il s'agit de l'arbre de hauteur p ayant pour racine la liste (x) constituée du seul élément x, pour nœuds de rang k les listes de longueur $k+1$ d'éléments de $\mathcal{X}$ qui se terminent par x, et vérifiant la relation de paternité suivante : la liste l_1 est *le fils* de la liste l_2 si l_2 correspond à l_1 privée de son premier élément.

Exemple : Sur la figure 6, apparaît l'arbre $\mathbb{N}$-complet de hauteur 2 de 0 (les pointillés indiquent que toutes les listes de la forme $(n, 0)$ pour $n \in \mathbb{N}$ sont des fils de (0)).

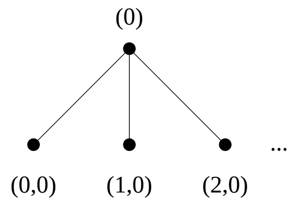

FIG.6 – *Arbre complet de hauteur 2 de 0*

On définit alors une structuration sur cet arbre, qui permet de traduire dans la destinée les relations vraies dans la théorie :

Définition 2 (Structuration induite)
Soit $\mathcal{X}$ une $\mathcal{L}$-structure, et $x \in X$. On définit une application de structuration c sur l'arbre $\mathcal{X}$-complet de hauteur p de x par :
Pour tout prédicat R d'arité k de $\mathcal{L}$, pour toutes familles $x_1 = (x'_1, \ldots)$, $x_2 = (x'_2, \ldots)$, ..., $x_k = (x'_k, \ldots)$ de listes ayant pour têtes de listes respectives $x'_1, \ldots, x'_k$, appartenant au support d'une même branche de feuille z de l'arbre $\mathcal{X}$-complet de hauteur p de x, on a l'équivalence suivante :

$$c(z) \models R(x_1, x_2, \ldots, x_k) \text{ équivaut à } \mathcal{M} \models R(x'_1, x'_2, \ldots, x'_k).$$

On obtient alors une destinée de x. Il peut y en avoir en effet plusieurs, et on a une notion de $\mathcal{L}$-isomorphisme entre deux destinées, induite par la notion de $\mathcal{L}$-isomorphisme entre deux $\mathcal{L}$-structures. Deux notions essentielles apparaissent dans l'étude des destinées :

L'exhaustivité : on dit qu'une destinée de x est *exhaustive* lorsqu'elle présente tous les cas de figure pouvant arriver à x dans $\mathcal{X}$ qui sont exprimables avec moins de p variables. Par exemple, la destinée de x construite à partir de l'arbre complet est exhaustive.

L'essentialité : une destinée de x est *essentielle* lorsqu'on a éliminé toutes les redondances entre les *pétioles* (c'est-à-dire les sous-arbres complets munis de la structuration induite). En effet, plusieurs pétioles peuvent être $\mathcal{L}$-isomorphes. Par exemple, si on considère la théorie à deux variables $Th_2\{\mathbb{N}, <\}$, une destinée de 0 est présentée en figure 6, mais elle n'est pas essentielle car du point de vue de 0, on ne peut distinguer les entiers supérieurs ou égaux à 1. En simplifiant les niveaux de la destinée un à un, en partant du niveau le plus élevé (les feuilles d'un même pétiole), on obtient, à partir d'une destinée exhaustive, un objet minimal fini, que l'on appelera une *destinée exhaustive essentielle* de x. Là encore, on peut en avoir plusieurs, il suffit de changer de choix lors des simplifications, et prendre un représentant d'une classe de $\mathcal{L}$-isomorphie à la place d'un autre pour obtenir deux destinées isomorphes.

Définition 3 *On appelle* p-transversale *d'une théorie $\mathcal{T}$ un ensemble essentiel et exhaustif de p-destinées exhaustives et essentielles de $\mathcal{T}$.*

Une p-transversale "transporte" donc en quelque sorte de manière condensée ce qui peut se passer dans la théorie si on se restreint à p variables.

Références Bibliographiques

A.Chateau (2000), "Les théories du successeur et de la coprimarité", mémoire de DEA, Université Paris 7.

F.Nézondet (1997), "p-destinées et applications à la théorie du successeur et de la coprimarité sur les entiers", Thèse de doctorat, Université d'Auvergne.