LES THÉORIES DU SUCCESSEUR ET DE LA COPRIMARITÉ

ANNIE CHATEAU
DEA de logique et fondements de l'informatique
Université Paris 7
1999 - 2000

sous la direction de

DENIS RICHARD
Laboratoire de Logique, Algorithmique et Informatique de Clermont-Ferrand
Université Clermont 1
IUT Dept.Informatique - BP 86 - 63172 AUBIERE Cedex

Contents

1 Généralités sur les p-destinées **11**
1.1 Définitions 11
1.1.1 p-destinées 11
1.1.2 Destinées exhaustives et essentielles 18
1.1.3 p-transversale essentielle 20
1.2 L'outil pour les problèmes de définissabilité : les graphes de Skolem 21
1.2.1 Présentation 21
1.2.2 Théorème des graphes de Skolem (énoncé) 23
1.3 Lien avec la décidabilité 24
1.3.1 Distinguante : passage d'une satisfaction à une autre . 24
1.3.2 Théorème de "passage" (énoncé) 25

2 $Th\{\mathbb{N}, S, \perp\}$ et la conjecture de Woods-Erdös **27**
2.1 Autour de la conjecture de Woods-Erdös 27
2.1.1 Introduction 27
2.1.2 Résultats préliminaires de théorie des nombres 28
2.1.3 La méthode de codage ZBV et la réduction du problème de Julia Robinson à la définissabilité de $x \mapsto 5^x$ dans $\{S, \perp\}$ 29
2.1.4 La conjecture de Julia Robinson est vraie en restriction aux primaires 32
2.1.5 Equivalence entre les deux conjectures 33
2.2 Résultats de F.Nézondet : quelques modèles de l'arithmétique 36
2.2.1 Notion d'abritement 36
2.2.2 Rappels sur les ultraproduits 39
2.2.3 Définition d'un premier modèle 40
2.2.4 Autre modèle plus proche d'un modèle de la théorie complète 44

3 Les théories à nombre borné de quantificateurs **48**
3.1 $Th_2\{\mathbb{N}, S, \perp\}$ 48
3.1.1 Formalisme et conventions pour les figures 48

3.1.2 Construction de la destinée 49
3.2 Etude préalable de la 3-transversale essentielle de $Th_3\{\mathbb{N}, S, \perp\}$ 51
3.2.1 Présentation générale 51
3.2.2 Les cas de base sur les supports de n et k 51
3.2.3 Cas possibles pour les branches (k, u) proches de k ou n, lorsque k et n sont éloignés 53
3.2.4 Etude des branches (n, k) pour k proche de n 55
3.2.5 Cas pathologiques $k = 0$, $k = 1$, $k = 2$ 61
3.2.6 Conclusion sur le nombre de destinées possibles 63
3.3 Résultats sur les ensembles X et Y 63
3.3.1 Présentation 63
3.3.2 Quelques résultats 64

4 Perspectives 65
4.1 Un algorithme de vérification 65
4.1.1 Choix des structures de données 65
4.1.2 Outils autour de ces structures 67
4.1.3 Le corps de la fonction d'évaluation 67
4.1.4 Mode d'emploi du programme 68
4.1.5 Exemple de session 69
4.1.6 Améliorations envisagées 70
4.2 Un algorithme pour déduire l'axiomatisation de la transversale 70
4.3 Un codage dans les entiers sous forme binaire 70
4.4 Lien avec les bases de données 71

A Résultats autour de la conjecture Woods-Erdös 72
A.1 Démonstration du corollaire 1: 72
A.2 Synonymie de $< \mathbb{N}, +, \times, = >$ et $< \mathbb{N}, S, | >$ 74

B Etude des branches (n, k) pour k éloignés de n 77
B.1 Rappel des 80 cas possibles 77
B.2 $Supp(k) = Supp(n)$ 78
B.3 $Supp(n) \subsetneq Supp(k)$ 78
B.4 $Supp(n) \cap Supp(k) = \emptyset$ 80
B.4.1 n pair 80
B.4.2 n impair 80
B.5 $Supp(n) \backslash Supp(k) \neq \emptyset$, $Supp(k) \backslash Supp(n) \neq \emptyset$ et $n \not\perp k$ 80
B.5.1 n pair 80
B.5.2 n impair 81
B.6 $Supp(k) \subsetneq Supp(n)$ 81
B.6.1 n pair 81
B.6.2 n impair 81
B.7 Conclusion sur le nombre de destinées 82

C Programme de vérification: code source en Caml **83**
C.1 L'analyse syntaxique et sémantique 83
C.2 Les outils de manipulation de destinées et de formules 85
C.3 La fonction d'évaluation 88
C.4 L'interface avec l'utilisateur 91

Notations

- $\mathcal{P}$ désigne l'ensemble des entiers premiers de $\mathbb{N}$;
- $\perp$ est le symbole de la coprimarité: $x \perp y$ signifie x et y sont premiers entre eux;
- $Th_3\{< \mathbb{N}, S, \perp >\}$ désigne la théorie de $\mathbb{N}$ muni du langage du successeur et de la coprimarité, restreinte aux formules à trois quantificateurs;
- $\equiv_A$ désigne la relation d'équivalence $x \equiv_A y$ si et seulement si $\forall k \in A\ SUPP(x+k) = SUPP(y+k)$;
- $[a..b]$ est la notation adoptée pour les intervalles d'entiers naturels;
- $SUPP(x)$, où x est un entier, désigne l'ensemble des diviseurs premiers de x: c'est le support de x;
- $V_p(x)$ est la valuation p-adique de x, c'est-à-dire le plus grand exposant α tel que p^α divise x;
- $p^\alpha \mid\mid x$ signifie "$p^\alpha \mid x$ et $p^{\alpha+1} \not| x$", ou encore $V_p(x) = \alpha$.
- $Ord(p, q)$ est l'ordre de p modulo q.

Introduction

Les nombres entiers naturels étaient considérés par les Grecs anciens comme les seuls véritables "nombres". Ils ont fasciné des générations de mathématiciens, à la fois parce qu'ils semblent être des objets simples, et parce que la théorie arithmétique est si vaste qu'on est loin d'en avoir fait le tour.

La logique, et en particulier la théorie des modèles, a apporté un sang nouveau à l'étude des entiers: considérant les théories arithmétiques des entiers, et plus particulièrement la théorie arithmétique de Peano, on s'est aperçu qu'on pouvait construire des modèles "plus gros" que $\mathbb{N}$, les modèles non standard, vérifiant exactement les mêmes axiomes que les entiers naturels. Ces modèles font intervenir des "entiers infinis", ou non standard, qui sont infinis pour un regard extérieur au modèle, mais sont des entiers au même titre que les autres pour un observateur qui vivrait dans le modèle. Ces modèles apportent des outils performants pour comprendre les entiers, et sont également utilisés en analyse par exemple.

Une autre approche, qui est celle employée ici, consiste non pas à s'intéresser aux modèles de l'arithmétique en tant qu'objets d'étude, mais à étudier les théories de $\mathbb{N}$ muni de différents langages, et les liens entre ces théories. Cela fait bien entendu appel aux outils de la théorie des modèles, mais pas uniquement.

Nous nous intéresserons plus particulièrement à un langage composé de deux relations binaires, le successeur et la coprimarité. Nous ne pourrons pas explorer de manière exhaustive tout ce qui a été fait sur le sujet, mais le but de ce travail est de présenter deux voies qui ont été et/ou sont en train d'être parcourues:

Synonymie : On peut se demander quel est le "pouvoir expressif" du langage choisi, le successeur et la coprimarité, par rapport au langage de l'arithmétique classique, $\{+, \times, =\}$, dans le sens ou l'on recherche si les théories $Th\{<\mathbb{N}, S, \perp>\}$ et $Th\{<\mathbb{N}, +, \times, =>\}$ sont équivalentes, ou si la première est strictement moins forte que la seconde. C'est le but des travaux réalisés autour des conjectures de Woods-Erdös et de Julia Robinson, que nous verrons au chapitre 2.

Décidabilité des théories restreintes : On peut également se demander, sachant que la théorie de $< \mathbb{N}, S, \perp >$ est indécidable et que ses sous-théories réduites à un nombre de quantificateurs fixé sont, elles, décidables, s'il est possible de trouver effectivement un algorithme prenant en entrée une formule à un nombre fixé de quantificateurs (dont dépend l'algorithme) et donnant au bout d'un temps fini la réponse à la question "Cette formule est-elle un théorème de $< \mathbb{N}, S, \perp >$?". On sait que cet algorithme existe, mais il nous importe ici de le trouver afin de l'utiliser dans la pratique. Nous nous intéresserons plus particulièrement dans le chapitre 3 au cas à trois quantificateurs : L'algorithme pour le cas à 2 quantificateurs est en effet connu, et on en propose une implémentation en Caml en 4.1, mais l'algorithme pour le cas à 3 quantificateurs n'est pas encore déterminé.

Dans cette deuxième approche, on peut faire intervenir un outil introduit récemment par Francis Nézondet (voir [9]), et qui apporte de nombreuses possibilités, tant théoriques que sur le plan pratique: les p-destinées. Ces structures arborescentes seront présentées au chapitre 1.

Avertissement : Dans tout ce qui suit on supposera le lecteur familiarisé avec la logique mathématique de base, et l'arithmétique élémentaire. Nous renvoyons à [5] et [6] en cas de problème ...

Chapter 1

Généralités sur les p-destinées

Dans cette partie nous allons faire connaissance avec un outil qui semble tout à fait pertinent pour l'étude de la décidabilité des théories à nombre borné de quantificateurs. Introduites par Francis Nézondet en 1997 (voir [9]), les p-destinées en sont encore à leurs débuts, et sans doute qu'elles peuvent être le cadre d'études théoriques et pratiques dans d'autres domaines.

1.1 Définitions

Comme tout outil relativement récent, les p-destinées nécessitent au moins dans les débuts de leur "entrée sur la scène internationale" toute une panoplie de définitions plus ou moins formelles indispensables pour la compréhension.

1.1.1 p-destinées

1.1.1.1 Définition non formelle et enjeux

Nous allons commencer par essayer de donner une intuition de ce que sont les p-destinées. Tout d'abord, que signifie ce p, et pourquoi ce nom très métaphorique de "destinée"?

Il s'agit, très grossièrement, d'un arbre de possibilités de hauteur p. Prenons tout de suite un exemple. On prend l'ensemble des entiers naturels, et le langage de l'ordre $<$ et de la primalité: 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 construit une forêt d'arbres de hauteur 2, comme dans la figure 1.1.

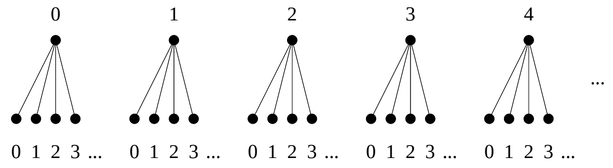


Figure 1.1: Forêt d'entiers de hauteur 2

Ensuite, on veut répertorier toutes les situations possibles, de manière à ce que deux cas "isomorphes" ne se représentent pas. 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 1.2. (Les termes apparaissant dans les légendes de ces figures seront explicités dans la sous-section suivante).

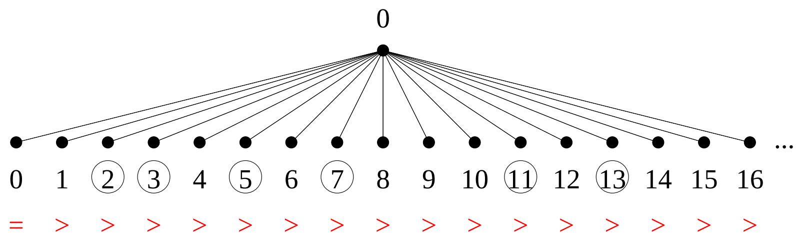


Figure 1.2: Destinée exhaustive mais non essentielle de racine 0

On remarque déjà qu'il y a des redondances, 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 1.3.

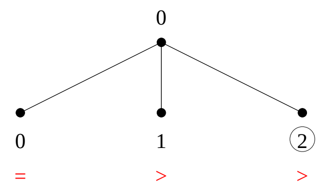


Figure 1.3: Destinée exhaustive essentielle de racine 0

On a désormais un arbre fini, sans redondance infinie de branches "isomorphes" (figurée par " ... " sur la figure). Continuons le même genre de construction pour les autres entiers. On aura ainsi quelque chose comme la figure 1.4.

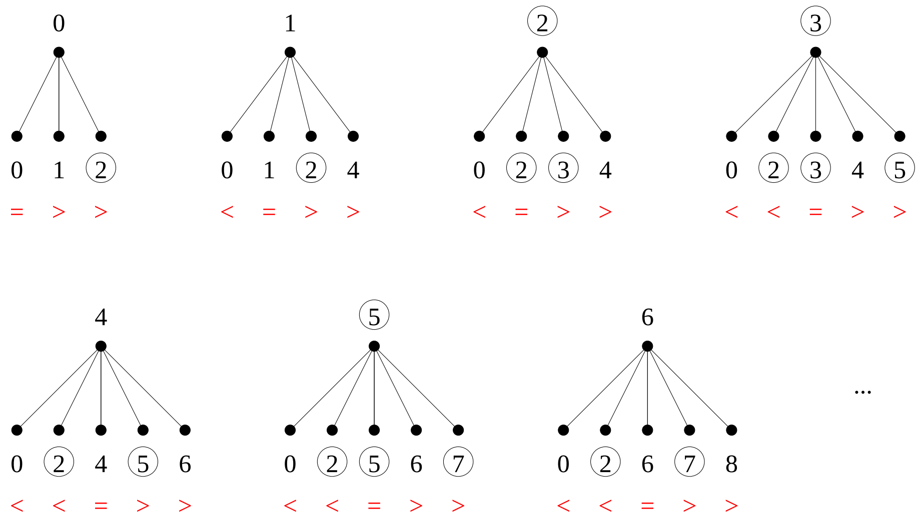


Figure 1.4: Destinée exhaustive, simplifiée au rang 2

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. 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 ($n!$)
  - 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 ($n!$)
- 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 la destinée de la figure 1.5.

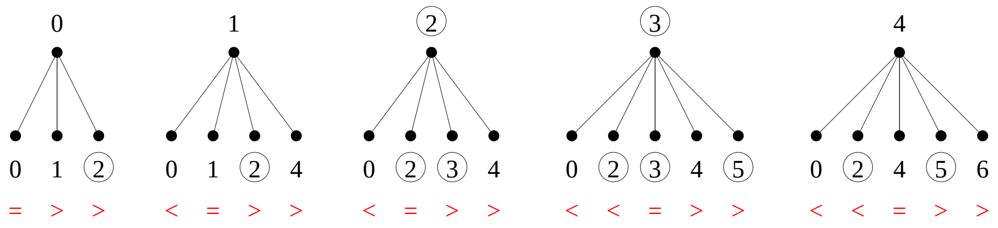


Figure 1.5: Destinée exhaustive essentielle

Continuons notre "découverte intuitive" des p-destinées, et voyons maintenant à quoi cela peut nous servir d'avoir construit un tel objet.

On s'intéresse à la théorie complète des entiers munis du langage de l'ordre et de la primarité, restreinte à deux quantificateurs. On sait qu'il existe un nombre fini de formules à deux quantificateurs à équivalence logique près dans ce langage, puisqu'il s'agit d'un langage relationnel. 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 destinée précédente 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 quantificateurs dans ce langage. Cela nous donne "effectivement" un algorithme de décision, ce qui est vraiment ce qui est plus utile dans la pratique que le simple fait de savoir qu'il existe un algorithme.

On peut également s'intéresser concernant ces destinées à ce qu'elles nous apportent en matière d'axiomatisation de la théorie. En effet la transversale essentielle représente en principe toutes les formules de la théorie, à équivalence logique près. On peut déduire de la structure arborescente une transversale des formules en examinant les branches. Dans notre exemple, ainsi, la destinée de 0 nous donne les formules:

- $\exists x(\exists y(\neg(y > x) \wedge \neg(x < y)))$, en examinant la branche $(0, 0)$
- $\exists x(\exists y(\neg(P(y)) \wedge (x < y)))$ en examinant la branche $(0, 1)$
- $\exists x(\exists y((P(y)) \wedge (x < y)))$ en examinant la branche $(0, 2)$
- $\exists x(\forall y \neg(y < x))$ en examinant toutes les branches à la fois

- ...

On peut examiner ainsi toutes les formules, à équivalence logique près, fournies par la transversale, et on obtiendrait ainsi l'axiomatique de la théorie. C'est l'opération inverse, en quelque sorte, de la vérification de formules grâce à la transversale. Bien entendu, l'axiomatique a été utilisée en vue d'établir la transversale, mais ceci de façon non formelle, alors qu'ici on peut espérer construire automatiquement l'axiomatique à partir de la transversale. Remarquons au passage que cette transversale constitue alors un moyen de stockage intéressant pour la théorie: il suffit de communiquer celle-ci pour donner tout à la fois l'algorithme de vérification des formules et l'axiomatique de la théorie.

1.1.1.2 Définition formelle

L'exemple précédent est évidement très simple, et nous allons formaliser dans cette section et les suivantes les notions abordées précédemment de façon informelle.

On supposera le lecteur habitué à la terminologie des arbres. On se fixe un langage relationnel donné$\mathcal{L}$.

Définition 1 (Destinée)
On appelle p-destinée un arbre régulier (X, P, r), de hauteur p muni d'une application de structuration c:

- *X est l'ensemble de base de l'arbre*
- *P est la relation de paternité entre les noeuds*
- *$r \in X$ est la racine de l'arbre*
- *c associe à un noeud n une $\mathcal{L}$-structure notée $c(n)$, de domaine la branche $[r, n]$ et telle que si n_1 et n_2 sont sur une même branche et que n_1 est un aïeul de n_2 alors $c(n_1)$ est une sous-structure de $c(n_1)$.*

1.1.1.3 Exemple

On peut reprendre comme exemple la destinée construite en 1.1.1.1. Notons bien que sur les schémas, les étiquettes des nœuds étaient entiers, alors que la définition de destinée suppose que ce soient des $\{<, P\}$-structures. On peut coller à la définition en disant que sur le schéma, l'entier associé au noeud est en fait une représentation pour la structure de domaine les entiers présents sur la branche allant du noeud à la racine, et où l'interprétation des symboles de relation est la restriction à ce domaine de l'interprétation usuelle sur $\mathbb{N}$.

1.1.2 Destinées exhaustives et essentielles

1.1.2.1 Destinée exhaustive - Exemple

Pour pouvoir nommer les noeuds de façon correcte, on les étiquette par des listes. Pour pouvoir associer une destinée de manière formelle à une structure donnée, on associe à un ensemble M donné (qui sera le domaine de la structure en question), et à un élément fixé x, un arbre des listes d'éléments de M de longueur p et de dernier élément x:

Définition 2 (Arbre M-complet de hauteur p)
On appelle arbre M-complet de hauteur p de x l'arbre régulier $(LIST(M,p,x), P, (x))$, où:

- *$LIST(M,p,x)$ est l'ensemble des listes d'éléments de M de longueur p aboutissant en x*
- *P est la relation binaire définie sur l'ensemble précédent par: $x_1 P x_2$ si et seulement si x_1 est la liste étêtée (i.e. à qui on a ôté son premier élément) de x_2*
- *La racine (x) est la liste composée du seul élément x.*

On met une application de structuration sur l'arbre obtenu précédemment, afin d'obtenir une destinée à partir de la structure $\mathcal{M}$ de domaine M.

Définition 3 (Structuration canoniquement induite)
Soit $\mathcal{M}$ une $\mathcal{L}$-structure, et $x \in M$. On définit une application de structuration c sur $(LIST(M,p,x), P, (x))$ par:
Pour tout prédicat R d'arité k de $\mathcal{L}$, pour toutes familles $x_1 = (x'_1, \ldots)$, $x_2 = (x'_2, \ldots)$, $\ldots$, $x_k = (x'_k, \ldots)$ de listes de têtes de listes respectives $x'_1, \ldots, x'_k$, appartenant au support d'une même branche de feuille z de l'arbre $(LIST(M,p,x), P, (x))$, 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 peut maintenant définir la p-destinée exhaustive d'une $\mathcal{L}$-structure $\mathcal{M}$:

Définition 4 (p-destinée exhaustive)
Soit $\mathcal{M}$ une $\mathcal{L}$-structure, p un entier, x un élément de M. On définit la p-destinée exhaustive de x comme étant la p-destinée:

$$(LIST(M,p,x), P, (x), c),$$

où $(LIST(M,p,x), P, (x))$ est l'arbre M-complet de hauteur p et de racine x, et c l'application de structuration canoniquement induite sur cet arbre par M.

<u>Exemple:</u> On peut reprendre la destinée de 0 construite au paragraphe 1.1.1.1

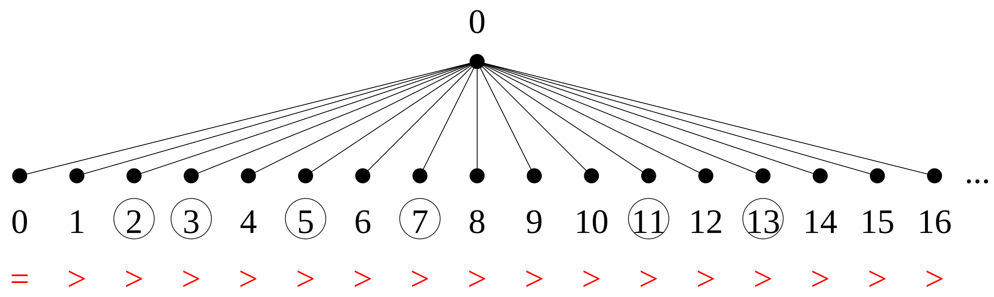


Figure 1.6: Destinée exhaustive de racine 0 dans $\mathbb{N}$ muni de $<$ et P

1.1.2.2 Destinée essentielle : isomorphisme de pétiole, superposition et simplification

Maintenant qu'on dispose d'une collection de p-destinées (éventuellement infinies) pouvant rendre compte de ce qui se passe dans une structure donnée, on aimerait pouvoir disposer d'un formalisme pour exprimer les simplifications que nous avons effectuées dans la première partie 1.1.1.1. (Enlever les " ... ")

Précisons qu'un pétiole est un sous-arbre de la destinée, muni de la structuration induite par celle de la destinée globale.

Définition 5 (Isomorphisme de pétiole)
Un isomorphisme de pétioles entre deux arbres (resp. deux destinées) est une bijection qui conserve la structure d'arbres (i.e. la relation de filiation) (resp. la structure d'arbres et c'est un isomorphisme pour l'application de structuration).

Définition 6 (Superposition au rang k)
Etant donnée une p destinée (A, P, r, c), on dit que (A', P', r', c') est une superposition au rang k $(0 < k < p)$ de cette destinée si pour chaque noeud x de rang k, on a choisi un représentant par classe d'isomorphisme de pétioles pour les pétioles ayant pour racines les fils de ce noeud x.

Définition 7 (Simplification au rang k)
Si on a une famille de simplifications S_j, $S_{j+1}, \ldots, S_{p-1}$ de rang correspondant à leur indice, sur l'ensemble des p-destinées d'un langage donné, on appelle simplification générale au rang k l'application $S_j \circ S_{j+1} \circ \ldots \circ S_{p-1}$. On appelle simplification générale une simplification générale de rang 1.

On obtient par ces applications des simplifiés généraux des destinées.

Définition 8 (Destinée essentielle)
On appelle p-destinée essentielle l'image par une simplification générale d'une p-destinée.

1.1.2.3 Exemples

Un exemple de simplification est donné à travers les figures qui suivent: on considère le langage constitué du prédicat binaire "être égal", sur les entiers modulo 3. On figure par une grosse croix le fait de ne pas être égal. On effectue la simplification au rang 2 des 3-destinées: les pétioles des noeuds de rang 1 n'ont alors plus que deux branches: l'une avec un élément congru au noeud modulo 3 (lui même) et une autre avec un élément qui ne l'est pas. (figure 1.7).

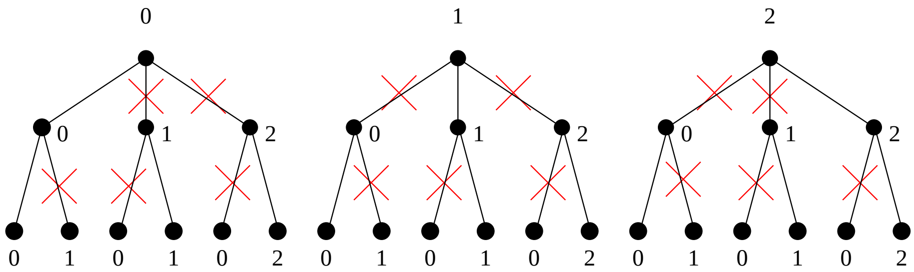


Figure 1.7: Superposition de rang 2 pour les destinées associées à la congruence modulo 3

De même, lorsqu'on effectue la simplification au rang 1, les fils des racines sont au nombre de deux. On obtient alors des destinées essentielles.(figure 1.8).

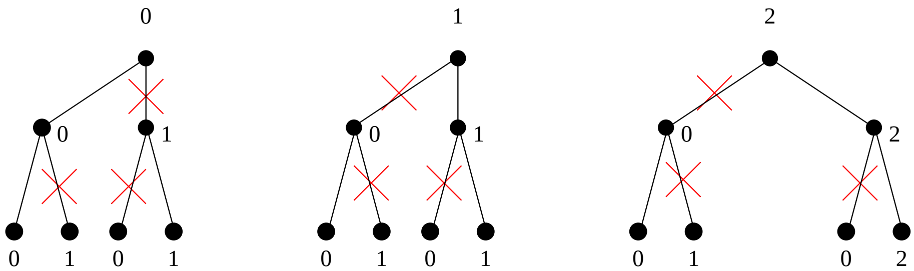


Figure 1.8: Superposition de rang 1 pour les destinées associées à la congruence modulo 3

1.1.3 p-transversale essentielle

1.1.3.1 Définition

Définition 9 (p-transversale essentielle)
On appelle p-transversale essentielle d'une $\mathcal{L}$-structure $\mathcal{M}$ donnée, pour un

langage $\mathcal{L}$ relationnel, un ensemble minimal de destinées contenant à isomorphisme près la p-destinée essentielle de chaque destinée exhaustive de $\mathcal{M}$.

La p-transversale est donc un condensé de la p-destinée exhaustive, dans le sens où elle ne contient aucune information redondante sur la structure $\mathcal{M}$.

1.1.3.2 Exemple

Reprenons l'exemple du paragraphe précédent.
On remarque que les trois destinées obtenues sont isomorphes pour cette relation: la destinée de racine 0 nous fournit une 3-transversale essentielle de la 3-destinée de $\mathbb{Z}/3\mathbb{Z}$ muni de la relation de congruence.(figure 1.9).

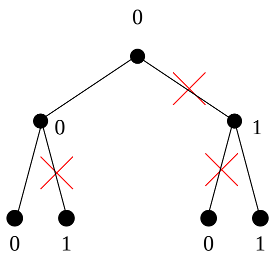


Figure 1.9: Transversale essentielle pour la congruence modulo 3

1.2 L'outil pour les problèmes de définissabilité : les graphes de Skolem

Maintenant que l'on dispose d'un outil pour étudier ce qui se passe dans une structure sur un langage relationnel, on va s'en donner un autre, les graphes de Skolem, pour voir se qui se passe au niveau de la satisfaisabilité dans la destinée, par rapport à la satisfaisabilité dans la structure initiale. Pour cela on va se donner un critère pour la satisfaisabilité d'une formule dans $\mathcal{M}$ à l'aide des graphes de Skolem. On se fixe à partir de maintenant une formule F, prise sous forme prénexe: $F = Q_1x_1Q_2x_2\dots Q_px_pR(x_1,\dots,x_p)$, avec Q_i des quantificateurs, et R une formule à p variables libres.

1.2.1 Présentation

Pour définir proprement ce qu'est un graphe de Skolem, on a besoin de définir les propriétés qu'il va vérifier:

Définition 10 (F-index universel)
Le F-index universel relatif à la formule F décrite ci-dessus est l'ensemble des indices des quantificateurs universels: $F_\forall = \{i \in [1..p] / Q_i = \forall\}$

Définition 11 (F-index existentiel)
Le F-index existentiel relatif à la formule F décrite ci-dessus est l'ensemble des indices des quantificateurs existentiels: $F_\exists = \{i \in [1..p] / Q_i = \exists\}$

On notera dans la suite Γ une partie de M^p.

Définition 12 (F-universalité)
On dit que Γ vérifie la F-universalité dans $\mathcal{M}$ si et seulement si pour toute application f de $F_\forall$ dans M, il existe un $\mu \in \Gamma$ tel que $\forall i \in F_\forall\ pr_i(\mu) = f(i)$, où $pr_i(\mu)$ désigne la $i^{ème}$ composante de μ.

Cette définition exprime que quelque soit la valeur donnée aux variables correspondant aux quantificateurs universels, on a un p-uple de Γ qui prend ces valeurs aux endroits indexés par le F-index universel.

Exemple: On prend $M = \{0, 1, 2\}$, muni de la relation d'égalité et de la relation "avoir même carré modulo 3", et $p = 3$, et la formule:

$$F = \forall x \exists y \forall z ((x = z) \vee (y^2 = z^2)).$$

On a alors $F_\forall = \{1, 3\}$ et $F_\exists \{2\}$. Γ doit contenir des éléments de la forme: $(0, ., 0)$, $(0, ., 1)$, $(0, ., 2)$, $(1, ., 0)$, $(1, ., 1)$, $(1, ., 2)$, $(2, ., 0)$, $(2, ., 1)$, $(2, ., 2)$ pour avoir la F-universalité.

Définition 13 (F-changeabilité)
On dit que Γ vérifie la F-k-changeabilité pour un entier $1 < k < p$ si et seulement si pour tous μ, μ' uples de Γ, on a: si pour tout $j \in F_\forall$ tel que $j < k$, on a $pr_j(\mu) = pr_j(\mu')$, alors il existe un μ'' tel que pour tout $j < k$ on a $pr_j(\mu'') = pr_j(\mu')$ et pour tout $j \in F_\forall$ tel que $j \geq k$, on a $pr_j(\mu'') = pr_j(\mu)$. On dit que Γ vérifie la F-changeabilité si Γ vérifie la F-k-changeabilité pour tout $k \leq p$.

Cette définition exprime le fait que si deux uples de Γ coïncident pour leurs variables universelles, alors on peut en trouver un qui ait les variables existentielles de l'un au début, et les variables existentielles de l'autre à la fin: la valeur d'une variable existentielle (i.e. d'indice dans $F_\exists$) ne dépend que des valeurs des variables la précédent.

Exemple: Dans notre exemple précédent, si Γ contient par exemple $(0, 1, 2)$ et $(0, 2, 1)$, alors il devra également contenir $(0, 1, 1)$ et $(0, 2, 2)$ pour vérifier la F-changeabilité.

Définition 14 (F-$\mathcal{M}$-satisfaction)
On dit que Γ vérifie la F-$\mathcal{M}$-satisfaction si tous les uples de Γ vérifient R.
(On rappelle que R est la formule à l'"intérieur" de F: $F = Q_1x_1Q_2x_2 \dots Q_px_pR(x_1, \dots, x_p)$.)

Exemple: $\Gamma = \{(0,0,0),(1,1,1),(2,2,2),(0,1,0),(0,1,1),(0,1,2),(0,2,1),(1,0,1),(0,2,2),(2,0,2)\}$ vérifie la F-changeabilité pour l'exemple précédent.

On vient de définir des notions que l'on trouve agréables de retrouver dans un ensemble de uples de variables, si bien que l'on rassemble ces trois propriétés sous la notion de graphe de Skolem:

Définition 15 (Graphe de Skolem)
Soit un langage fini $\mathcal{L}$. Soit $F = Q_1x_1Q_2x_2 \dots Q_px_pR(x_1, \dots, x_p)$ une formule de $\mathcal{L}$ et $\mathcal{M}$ une $\mathcal{L}$-structure.
Soit Γ un ensemble de p-uples de M. On dit que Γ est un graphe de Skolem pour F et $\mathcal{M}$ si et seulement si Γ vérifie la F-universalité, la F-changeabilité et la F-$\mathcal{M}$-satisfaction.

1.2.2 Théorème des graphes de Skolem (énoncé)

Nous sommes désormais en mesure de donner un critère de satisfaisabilité d'une formule à l'aide des graphes de Skolem:

Théorème 1 (Lien entre les graphes de Skolem et la satisfaisabilité d'une formule)
Soit un langage fini $\mathcal{L}$. Soit $\mathcal{M}$ une $\mathcal{L}$-structure et p un entier. Soit F une formule s'écrivant $F = Q_1x_1Q_2x_2 \dots Q_px_pR(x_1, \dots, x_p)$ sous forme prénexe.
Pour $n < p$, on note F_n la formule $Q_{n+1}x_{n+1} \dots Q_px_pR$. $\mathcal{M}$ réalise la formule F si et seulement si l'une des conditions suivantes est vérifiée:

- *(i). Il existe un entier n et un graphe de Skolem Γ pour la formule F_n, la structure $\mathcal{M}$ et la quantification $Q_1x_1Q_2x_2 \dots Q_nx_n$;*
- *(ii). Pour tout $n \in [1..p]$, il existe un graphe de Skolem Γ_n pour la formule F_n, la structure $\mathcal{M}$ et la quantification $Q_1x_1Q_2x_2 \dots Q_nx_n$.*

La démonstration de ce théorème figure en [9].
Exemple: La formule utilisée dans l'exemple précédent ne pourra pas admettre de graphe de Skolem, car elle n'est pas vérifiée dans M: Pour $x = 1$, un y convenable n'existe pas.

Il reste à ramener les graphes de Skolem à la satisfaction dans les p-destinées, ce que nous verrons à la section suivante.

1.3 Lien avec la décidabilité

On va établir quelques notions qui vont nous permettre de transposer dans l'univers des destinées ce qu'on sait déjà faire au niveau des structures "usuelles".

1.3.1 Distinguante : passage d'une satisfaction à une autre

Définition 16
On dit que deux destinées sont logiquement équivalentes si leurs simplifiés généraux sont isomorphes.

Il nous faut passer des destinées aux transversales, qui sont les objets qui en définitive nous intéressent:

Définition 17
Soient $\mathcal{M}$ et $\mathcal{M}'$ deux $\mathcal{L}$-structure sur un langage fini $\mathcal{L}$. On note A et A' leurs transversales essentielles respectives. On dit que A et A' sont logiquement équivalentes si et seulement si pour toute p-destinée de l'une il existe une p-destinée de l'autre qui lui est logiquement équivalente.

On va définir maintenant les formules "transformées", construites à partir des formules de $\mathcal{L}$, afin de transposer la satisfaisabilité dans $\mathcal{M}$ en la satisfaisabilité dans une transversale essentielle.

Définition 18 (Distinguante)
Soit $\mathcal{L}$ un langage fini ne comportant que des symboles de prédicat. Soit F une formule de la forme $F = Q_1x_1Q_2x_2 \dots Q_px_pR$, avec R sans quantificateurs. On appelle distinguante de F la formule F^* construite par induction sur l'écriture de la formule à partir de F de la manière suivante:
On introduit deux symboles de prédicat nouveaux, que l'on notera respectivement P, prédicat binaire noté sous forme infixe, et $rg(.) = 1$, prédicat unaire. La transformation de la formule se définit alors comme suit:

- $R^* = R$
- $(Q_ix_iG)^* = (Q_ix_i)^*G^*$
- *si* $i = 1$ *alors* $(\forall x_1)^* = \forall x_1(rg(x_1) = 1) \rightarrow$
- *si* $i = 1$ *alors* $(\exists x_1)^* = (rg(x_1) = 1)\wedge$
- *si* $i > 1$ *alors* $(\forall x_i)^* = \forall x_i(x_{i-1}Px_i) \rightarrow$
- *si* $i > 1$ *alors* $(\exists x_i)^* = \exists x_i(x_{i-1}Px_i)\wedge$

Exemple: Si $F = \forall x \exists y \exists z \forall t R(x,y,z,t)$, alors

$$F^* = \forall x(rg(x) = 1) \to (\exists y(xPy) \wedge (\exists z(yPz) \wedge (\forall t(zPt) \to R(x,y,z,t)))).$$

Il nous reste enfin à définir l'interprétation des nouveaux symboles que nous venons d'introduire, dans la structure associée à la p-transversale essentielle de $\mathcal{M}$:

Définition 19 (Structure et théorie de la transversale essentielle de $\mathcal{M}$)
Soit $\mathcal{L}$ un langage fini ne contenant pas le symbole P, $\mathcal{M}$ une $\mathcal{L}$-strusture, p un entier non nul et D une transversale p-essentielle de $\mathcal{M}$.
Soit $\mathcal{L}^*$ le langage constitué de $\mathcal{L}$, du prédicat binaire P et des prédicats unaires $rg(.) = 1$, $rg(.) = 2$, $\ldots$ $rg(.) = p$.
Soit D^* la $\mathcal{L}$-structure dont le domaine est l'ensemble N des noeuds des p-destinées essentielles de D. Le prédicat P est interprété comme la relation de filiation: xPy signifie x est le père de y. Les prédicats de rang sont interprétés naturellement par le rang dans l'arborescence.
La satisfaction dans D^* est alors définie par: Si R est un prédicat d'arité k de $\mathcal{L}$, $(x_1, \ldots, x_k)$ un k-uple de N et c l'application de structuration associée à D, alors $D^* \models R(x_1, \ldots, x_k)$ si et seulement si les x_i appartiennent à la même branche b et $c(b) \models R(x_1, \ldots, x_k)$.
D^* est appelé la structure de la p-transversale essentielle de $\mathcal{M}$ et sa théorie est appelée théorie de la p-transversale essentielle de $\mathcal{M}$.

On dispose de tous les outils pour faire le lien entre la satisfaisabilité dans $\mathcal{M}$ et celle dans sa transversale essentielle.

1.3.2 Théorème de "passage" (énoncé)

Théorème 2 (Lien entre la satisfaisabilité dans une structure et dans une transversale essentielle de cette structure)
Soit $\mathcal{L}$ un langage fini n'ayant que des symboles de prédicats. Soit F une $\mathcal{L}$ formule prénexe ayant au plus p quantificateurs, et $\mathcal{M}$ une $\mathcal{L}$-structure. Soit D une p-transversale essentielle de $\mathcal{M}$ et D^* la structure qui lui est associée. On a:

$$\mathcal{M} \models F \text{ si et seulement si } D^* \models F^*.$$

Démonstration:
On va simplement ici faire une esquisse, le détail de la preuve se trouvant dans [9].

On montre uniquement le sens direct. La réciproque s'obtient en appliquant le sens direct à $\neg F$. On suppose donc $\mathcal{M} \models F$.
Soit F^* la distinguante de F. D'après le théorème 1, on a un graphe de Skolem Γ pour F. on veut montrer qu'il existe un graphe de Skolem Γ^*

pour F^* dans D^*. On va définir ce graphe de Skolem de la façon suivante: Soit Δ la forêt des p-destinées exhaustives de $\mathcal{M}$. Alors D est l'image de Δ par un morphisme simplificateur φ. On a besoin de la définition suivante:

Définition 20 (Rupture)
Si $\sigma = (x_1, \ldots, x_k)$ est un k-uple de noeuds de D, une rupture de σ est:

- *soit un couple (x_i, x_{i+1}) tel que x_i n'est pas le père de x_{i+1},*
- *soit $\{x_1\}$ si x_1 n'est pas une racine d'une destinée de D.*

On note:

- Γ_1 est l'ensemble des p-uples σ d'éléments de $LIST(M, p)$ (les listes à p éléments de M) tels que le p-uple des têtes de listes des éléments de σ soit dans Γ.
- Γ_2 est l'image de Γ_1 par φ.
- Γ^* est l'ensemble des p-uples de Γ_2 qui n'ont pas de rupture ou qui présentent leur première rupture (indice i minimal pour une rupture (x_i, x_{i+1})) pour un indice i tel que $Q_{i+1} = \forall$.

On montre que Γ^* vérifie les trois conditions des graphes de Skolem vues plus haut.

Chapter 2

$Th\{\mathbb{N}, S, \perp\}$ et la conjecture de Woods-Erdös

2.1 Autour de la conjecture de Woods-Erdös

2.1.1 Introduction

Un nombre considérable de travaux ont été réalisés autour du langage du successeur et de la coprimarité dans le cadre de l'étude des synonymies sur les entiers.La conjecture dont nous allons parler est devenue emblématique des méthodes de l'étude des synonymies.

Ce qu'on appelle conjecture d'Erdös-Woods (ou Woods-Erdös selon l'ordre choisi sur l'ensemble des noms de mathématiciens ...) est la conjecture suivante due à Alan Woods:

> " Existe-t-il un entier k tel que, pour tous entiers x et y, il y ait égalité entre x et y si et seulement si pour chaque i=0,...,k, les entiers x+i et y+i ont les mêmes diviseurs premiers ?"

Cette conjecture est un affaiblissement de la suivante due à P.Erdös:

> " N'y a-t-il qu'un nombre fini d'entiers m, n, h, p avec p≥h≥3 et m≠n tels que les produits (m+1)(m+2)... (m+p) et (n+1)(n+2)... (n+h) aient les mêmes diviseurs premiers ?"

En effet, si la conjecture de P.Erdös est vraie: on pose $k = \max(p) + 1$ et si x et y sont tels que pour tout $i \in \{0 \dots k\}$, $SUPP(x + i) = SUPP(y + i)$, alors on a *a fortiori* : $SUPP((x+1)(x+2)\dots(x+k)) = SUPP((y+1)(y+2)\dots(y+k))$. Aucun h tel que $k \geq h \geq 3$ n'est tel que (x, y, h, k) vérifie les conditions données dans la conjecture, par choix de k. Or pour $h = k$, la condition $SUPP((x+1)(x+2)\dots(x+k)) = SUPP((y+1)(y+2)\dots(y+k))$ est vérifiée, c'est donc que c'est $x \neq y$ qui est en défaut, donc $x = y$.

Cette conjecture est équivalente à un problème de définissabilité soulevé par J.Robinson, en les termes suivants:

> “ Peut-on définir au premier ordre toute l'arithmétique (prédicats et fonctions usuelles $+$, $\times$, Exp, ...) avec seulement la fonction successeur S et le prédicat de coprimarité $\perp$?” ¿

Des avancées parallèles ont été menées en théorie des nombres et en logique concernant ces conjectures. Nous allons nous intéresser à quelques unes d'entre elles, en particulier:

- La conjecture de Julia Robinson équivaut à la définissabilité de la fonction $x \mapsto 5^x$ dans le langage $\{S, \perp\}$;
- La concture d'Erdös-Woods et celle de Julia Robinson sont vraies en restriction aux entiers primaires (puissances d'un nombre premier);
- Elles sont équivalentes.

Nous ne présentons ici qu'un échantillon des travaux réalisés autour de ce sujet. (Voir [7], [11], [12], [14], ...)

2.1.2 Résultats préliminaires de théorie des nombres

Nous allons utiliser par la suite une méthode de codage des parties finies de l'ensemble des nombres premiers appelée ZBV, des noms de Zsigmondy, Birkhoff et Vandiver, et qui se fonde sur les résultats suivants:

Théorème 3 (Zsigmondy - Birkhoff - Vandiver)
Soient x et y des entiers premiers entre eux tels que
$0 < y < x$. *Pour tout $n > 0$, il existe au moins un diviseur premier de $x^n - y^n$ qui ne divise pas $x^m - y^m$ pour $0 < m < n$. (un tel diviseur est dit* <u>*primitif*</u> *pour $x^n - y^n$) excepté dans les cas suivants:*
i). $n = 1$, $x - y = 1$, $x - y$ n'a alors aucun diviseur premier;
ii). $n = 2$, $x + y = 2^u$ où $u > 0$;
iii).$n = 6$, $x = 2$, $y = 1$.

Théorème 4 (Lucas - Carmichael)
Soient x et y des entiers premiers entre eux tels que
$0 < y < x$. *Pour tout $n > 0$, il existe au moins un diviseur premier de $x^n + y^n$ qui ne divise pas $x^m + y^m$ pour $0 < m < n$. (un tel diviseur est dit* <u>*caractéristique*</u> *pour $x^n + y^n$) excepté dans le cas où $n = 3$, $x = 2$ et $y = 1$.*

Théorème 5
$\gcd(\frac{x^m-y^m}{x-y}, \frac{x^n-y^n}{x-y}) = \frac{x^{\gcd(n,m)}-y^{\gcd(n,m)}}{x-y}$

Ces résultats peuvent être trouvés dans [1] et [3].
C'est principalement le corollaire suivant qui va nous servir:

Corollaire 1
Pour tout entier $x > 1$ et tous les entiers α et β, on a:
i). $SUPP(x^\alpha - 1) = SUPP(x^\beta - 1)$ si et seulement si $\alpha = \beta$ ou $x = 2^u - 1$ avec $u > 1$ et $\{\alpha, \beta\} \subseteq \{1, 2\}$.
ii). $SUPP(x^\beta - 1) \subseteq SUPP(x^\alpha - 1)$ si et seulement si $\beta \mid \alpha$ ou $x = 2^u - 1$ avec $u > 1$ et $\beta = 2$.
iii). p est un diviseur primitif de $x^\alpha - 1$ si et seulement si $p \mid x^\alpha - 1$ et ou bien $\alpha = 1$, ou bien pour tout $\beta \neq \alpha$ tel que $p \mid x^\beta - 1$ on a $SUPP(x^\alpha - 1) \subsetneq SUPP(x^\beta - 1)$.
iv). $SUPP(x^\alpha + 1) = SUPP(x^\beta + 1)$ si et seulement si $\alpha = \beta$ ou $x = 2$ et $\{\alpha, \beta\} \subseteq \{1, 3\}$.

La démonstration est faite en annexe A.1.
Notation: on note $\cong_A$ la relation d'équivalence: $x \cong_A y$ si et seulement si pour tout $k \in A$, $SUPP(x + k) = SUPP(y + k)$.
On aura également besoin du théorème chinois des restes et du théorème de Dirichlet:

Théorème 6 (Restes chinois)
Soient $(b_0, \ldots, b_n)$ des entiers premiers entre eux et $(\alpha_0, \ldots, \alpha_n)$ des entiers, alors il existe $a \in \mathbb{N}$ tel que $\forall i \in \{0 \ldots n\}, a \equiv \alpha_i [b_i]$.

Une présentation agréable en est faite dans [6].

Théorème 7 (Dirichlet)
il existe une infinité de nombres premiers dans les progressions arithmétiques $u_n = an + b$ pour $a \perp b$.

Une preuve complète est présentée dans [13]. On pourra également consulter [2] pour une preuve de ce théorème à l'intérieur de la théorie de Peano.

2.1.3 La méthode de codage ZBV et la réduction du problème de Julia Robinson à la définissabilité de $x \mapsto 5^x$ dans $\{S, \perp\}$

2.1.3.1 $\{S, \perp\}$-définissabilité de $x \mapsto 5^x$

Les mathématiciens sont toujours heureux de pouvoir réduire un problème donné à une autre formulation d'apparence plus simple. Mais outre l'effet purement psychologique du théorème suivant, la réduction du problème de J.Robinson à la définissabilité de $x \mapsto 5^x$ dans $\{S, \perp\}$ a l'avantage de donner un point de départ pour attaquer la conjecture d'Erdös-Woods en restriction aux entiers primaires, grâce au déplacement de structure que cela permet.

Théorème 8
L'application $x \mapsto 5^x$ transforme la structure $< \mathbb{N}, +, \times, = >$ en une structure isomorphe $< 5^{\mathbb{N}}, NA, NM, =_{PP} >$ qui est entièrement définissable dans $< \mathbb{N}, S, \perp >$.
En particulier, $< \mathbb{N}, S, \perp, x \mapsto 5^x >$ et $< \mathbb{N}, +, \times, = >$ définissent les mêmes relations et fonctions.

Notation: NA et NM sont les images respectives de $+$ et $\times$ par cet isomorphisme, et $=_{PP}$ est de manière plus générale l'égalité restreinte aux entiers primaires.
Démonstration:(partielle)
On se sert du résultat suivant de synonymie, dû à Julia Robinson:

Théorème 9
Les structures $< \mathbb{N}, S, | >$ et $< \mathbb{N}, +, \times, = >$ définissent les mêmes relations et fonctions.

Ce résultat est démontré en Annexe A.2.
Les images de S et $|$ par $x \mapsto 5^x$ sont respectivement:

- NS: $5^n \mapsto 5^{n+1}$
- NDIV: $\{(5^n, 5^m)/n \mid m\}$

Si on part du principe qu'elles sont définissables dans $< \mathbb{N}, S, \perp >$, on a alors dans cette structure une sous-structure $< 5^{\mathbb{N}}, NS, NDIV >$ isomorphe à $< \mathbb{N}, S, | >$ et donc dans laquelle on peut définir NA et NM, ainsi que $=_{pp}$ restreinte à $5^{\mathbb{N}}$, et par conséquent on aura $DEF(\mathbb{N}, +, \times, =) \subseteq DEF(\mathbb{N}, S, \perp)$
L'autre inclusion est très facile:

- S est définissable avec $+$
- $\perp$ est définissable par la formule:
  $\forall u(\neg(u = 0) \wedge \neg(u = 1) \wedge ((u \mid x) \rightarrow \neg(u \mid y))$
- $x \mapsto 5^x$ est récursive, donc représentable dans $< \mathbb{N}, +, \times, = >$. On peut même exhiber une formule, grâce à la fonction β de Gödel: On rappelle la propriété remarquable de la fonction de Gödel: "Pour toute suite finie d'entiers $(\alpha_0 \ldots \alpha_p)$ il existe des entiers c et d tels que $\forall i \in \{0 \ldots p\}$, $\beta(c, d, i) = \alpha_i$":
  $\forall x \forall y((z = 5^x) \leftrightarrow \exists c \exists d(B(c, d, 0, 1) \wedge (\forall i\ 0 < i \leq x\ \exists \gamma B(c, d, i, 5 \times \gamma) \wedge B(c, d, i-1, \gamma)) \wedge B(c, d, x, z)))$ est une formule définissant $x \mapsto 5^x$, avec $B(c, d, x, y)$ représentant β.

En résumé, il reste à prouver la $\{S, \perp\}$-définissabilité de $5^{\mathbb{N}}$, de NS et de $NDIV$ pour conclure cette démonstration, ce que nous ferons au paragraphe suivant.

2.1.3.2 Codage ZBV

Pour manipuler des objets tels que $5^{\mathbb{N}}$, on va commencer par coder des parties finies de l'ensemble des nombres premiers dans $< \mathbb{N}, S, \perp >$: on code une partie finie de $\mathcal{P}$ par un entier ayant cette partie pour support. On peut ainsi définir les opérations ensemblistes sur ces parties en utilisant des formules faisant intervenir leurs codes. Remarquons qu'un tel code existe toujours, quitte à prendre pour code d'une partie le produit des éléments de cette partie, mais que ce code n'est pas unique.

- la relation d'inclusion entre deux parties se traduit par l'inclusion des supports de leurs codes: $SUPP(x) \subseteq SUPP(y)$, autrement dit dans le langage $\{\perp\}$ par la formule: $\forall z((z \perp y) \to (z \perp x)$;

- L'égalité: $SUPP(x) = SUPP(y)$ est $(SUPP(x) \subseteq SUPP(y)) \wedge (SUPP(y) \subseteq SUPP(x))$;

- L'union: $SUPP(z) = SUPP(x) \cup SUPP(y)$ est $(SUPP(x) \subseteq SUPP(z)) \wedge (SUPP(y) \subseteq SUPP(z)) \wedge \forall t((SUPP(x) \subseteq SUPP(t)) \wedge (SUPP(y) \subseteq SUPP(t)) \to (SUPP(z) \subseteq SUPP(t)))$;

- L'intersection: $SUPP(z) = SUPP(x) \cap SUPP(y)$ est $(SUPP(z) \subseteq SUPP(x)) \wedge (SUPP(z) \subseteq SUPP(y)) \wedge \forall t((SUPP(t) \subseteq SUPP(x)) \wedge (SUPP(t) \subseteq SUPP(y)) \to (SUPP(t) \subseteq SUPP(z)))$;

- La différence: $SUPP(z) = SUPP(x) - SUPP(y)$ est $(SUPP(y) \subseteq SUPP(x)) \wedge (SUPP(z) \subseteq SUPP(x)) \wedge \forall u(SUPP(z) \cap SUPP(y) \subseteq SUPP(u)) \wedge \forall t(((SUPP(t) \subseteq SUPP(x)) \wedge \forall u(SUPP(t) \cap SUPP(y) \subseteq SUPP(u))) \to (SUPP(t) \subseteq SUPP(z)))$.

- on peut définir ensuite l'ensemble des primaires (un primaire est un entier caractérisé par l'inclusion de son support dans celui de tout entier non premier avec lui), par l'intermédiaire des formules $PP(x)$ (x est primaire) et $PP(x, y)$ (x et y sont primaires d'un même premier): $PP(x)$ est $\forall y((\neg(x \perp y)) \to SUPP(x) \subseteq SUPP(y))$ et $PP(x, y)$ est $PP(x) \wedge PP(y) \wedge \neg(x \perp y)$;

- les constantes 1 et 0 peuvent être également définies dans ce même langage: $EGAL_1(x) \leftrightarrow (\forall y(y \perp x))$ et $EGAL_0(x) \leftrightarrow (\forall y((y \perp x) \to EGAL_1(y)))$;

- toutes les constantes entières, en particulier 5, en découlent grâce au successeur;

- on peut définir la fonction prédécesseur restreinte aux primaires: $x = PRED_{pp} y$ si et seulement si $(ZERO(x) \to ZERO(y)) \wedge (\neg(ZERO(x)) \to (x =_{PP} Sy))$

Constatons rapidement que nous avons le résultat suivant:

Théorème 10
L'égalité coïncide avec $\cong_{\{0,1,2\}}$ sur les entiers primaires.

Démonstration: La seule classe de $\cong_{\{0,1\}}$ dont la trace sur les primaires n'est pas réduite à un seul élément est celle de 2, d'après le point *iv*). du corollaire 1.
Comme $SUPP(2+2) = \{2\}$ et $SUPP(2^3+2) = \{2,5\}$, on a $2 \not\cong_{\{0,1,2\}} 8$.

Remarque: On vient sans s'en apercevoir de démontrer la conjecture d'Erdös-Woods sur les entiers primaires, avec $k = 2$.

- Le théorème précédent permet de définir l'égalité restreinte aux primaires: $x =_{PP} y$ si et seulement si $SUPP(x) = SUPP(y) \wedge SUPP(Sx) = SUPP(Sy) \wedge SUPP(SSx) = SUPP(SSy)$;
- $5^{\mathbb{N}}$ est définissable comme étant les entiers primaires de même support premier que 5;
- $5^n \mapsto 5^{n+1}$ est définissable car on a le fait suivant: 5^{n+1} est le seul primaire de base 5, donc de la forme 5^α, tel que $SUPP(5^\alpha - 5) = \{5\} \cup SUPP(5^n - 1)$. En effet, $SUPP(5^\alpha - 5) = \{5\} \cup SUPP(5^{\alpha-1} - 1)$ et les deux unions sont disjointes. On a donc $SUPP(5^{\alpha-1} - 1) = SUPP(5^n - 1)$. Comme 5 n'est pas de la forme $2^u - 1$ (c'est pour cela qu'on a choisi ce chiffre), on a donc d'après le point *i*). du corollaire 1, $\alpha - 1 = n$, c'est-à dire $\alpha = n + 1$. Comme le prédécesseur est définissable, on peut exprimer cette caractérisation dans $\{S, \perp\}$;
- $\{(5^n, 5^m)/n \mid m\}$ est définissable car, toujours grâce au corollaire 1, on a $n|m$ si et seulement si $SUPP(5^n - 1) \subseteq SUPP(5^m - 1)$, les cas pathologiques étant évités par choix de 5.

2.1.3.3 Conclusion

On vient donc d'achever la preuve du théorème 8, et l'un des corollaires de ce théorème est l'indécidabilité de la théorie $\mathcal{T}\mathrm{h}(\mathbb{N}, S, \perp)$, ce qui est une condition nécessaire à la conjecture de Julia Robinson. On pourra trouver ce résultat d'indécidabilité dans [7] et [14].

2.1.4 La conjecture de Julia Robinson est vraie en restriction aux primaires

On va esquisser une preuve du fait que $x \mapsto 5^x$ restreinte aux entiers primaires est $\{S, \perp\}$-définissable. En fait on peut même montrer que toutes les fonctions et relations récursives définies sur l'ensemble des primaires sont $\{S, \perp\}$-définissables dans $\mathbb{N}$, et en particulier la fonction qui nous intéresse.

On a pour cela besoin de définir l'ensemble des nombres premiers: Notons $X = \{x \text{ primaire}/SUPP(x-1) \subseteq SUPP(y-1)$ pour tout primaire y de même base que $x\}$. X est définissable par la formule $X(x)$:

$$PP(x) \wedge \forall y(PP(x,y) \to (SUPP(Pred_{PP}(x)) = SUPP(Pred_{PP}(y))))$$

D'après le point ii). du corollaire 1, on a: $X = \mathcal{P} \cup \{(2^u - 1)^2/u \geq 2$ et $2^u - 1$ premier $\}$
On se débarasse du deuxième ensemble en remarquant que: $(2^u - 1) + 1$ est une puissance de 2, mais pas $(2^u - 1)^2 + 1 = 2(2^u(2^{u-1} - 1) + 1)$. D'où: $\mathcal{P} = \{x \in X/$ s'il existe un $y \in X$ de même base que x et distinct de x, alors $x+1$ est une puissance de $2\}$ c'est-à dire:

$$P(x) : X(x) \wedge \exists y((X(y) \wedge PP(x,y) \wedge \neg(x = y)) \to PP(Sx, 2))$$

On peut alors définir en un certain nombre d'étapes égal à n pour n grand un morceau de l'exponentielle: $(p,q) \mapsto q^{ord(q,p)} - 1$ où $ORD(q,p)$ est l'ordre de q dans $(\mathbb{Z}/\mathbb{Z}_p)^*$, cette fonction étant définie sur l'ensemble des couples de premiers distincts.
On utilise ce bout d'exponentielle pour prouver successivement les deux théorèmes suivants:

Théorème 11
La fonction $x \mapsto 5^x$ restreinte aux premiers est $\{S, \perp\}$-définissable.

Théorème 12
La fonction $x \mapsto 5^x$ restreinte aux primaires est $\{S, \perp\}$-définissable.

Tout cela est détaillé dans [11].

2.1.5 Equivalence entre les deux conjectures

Mentionnons tout d'abord un résultat qui peut paraître surprenant, car on ramène la conjecture d'Erdös-Woods à un problème qui semble à première vue plus faible que la conjecture de Julia Robinson.

2.1.5.1 Equivalence entre E-W et la $\{S, \perp\}$-définissabilité de l'égalité

Théorème 13
Il y a équivalence entre la conjecture d'Erdös-Woods et la $\{S, \perp\}$-définissabilité de l'égalité.

Démonstration: (esquissée)
$\Rightarrow$: Si Erdös-Woods est vraie, pour un certain k, alors la formule suivant définit l'égalité: $x = y$ si et seulement si $\forall z((z \perp x) \leftrightarrow (z \perp x)) \wedge ((z \perp Sx) \leftrightarrow (z \perp Sy)) \wedge ((z \perp SSx) \leftrightarrow (z \perp SSy))$.
$\Leftarrow$: On utilise la notion de saturation: une relation binaire R est saturée pour un ensemble A si pour tous x et y, on a $x \cong_A y \Rightarrow R(x,y)$. On a le résultat suivant, dont la preuve est donnée dans [7]:

Théorème 14 *Si ρ est une relation définissable dans $< \mathbb{N}, S, \perp >$, alors ρ est saturée pour un A fini.*

Il y a en fait équivalence, mais nous n'avons besoin que de ce sens là. Si la relation d'égalité est $\{S, \perp\}$-définissable, elle est saturée pour un ensemble A fini, donc a fortiori pour un ensemble de la forme $\{0..k\}$, et alors on aura pour tout x, pour tout y, $x \cong_{\{0..k\}} y \Rightarrow x = y$, c'est-à dire Erdös-Woods.

2.1.5.2 Equivalence E-W / JR

Théorème 15
$+$ et $\times$ sont définissables dans $< \mathbb{N}, S, \perp >$ si et seulement si on a un k tel que tout $x \in \mathbb{N}$ est entièrement déterminé par les supports de x, $x+1$, $x+2$ $\ldots$ $x+k$.

Pour démontrer ce théorème, nous aurons besoin de quelques notions préalables:

Définition 21
Dans un modèle $\mathcal{M}$ de l'arithmétique de Peano, on appelle coupure toute partie non vide I de M close par la fonction successeur ($\overline{S}^{\mathcal{M}}(I) \subseteq I$ et $\forall x \forall y (x \leq y$ et $y \in I \rightarrow x \in I)$.

Proposition 1
Toute coupure propre dans $\mathcal{M}$ modèle non standard de l'arithmétique de Peano est non définissable par une formule du langage de l'arithmétique dans cette structure.

Démonstration:
Dans le cas contraire, soit $\varphi(x, \vec{a})$ une formule définissant I:
$I = \{i \in M / \mathcal{M} \models \varphi(i, \vec{a})\}$. Puisque I est clos par S et $\overline{0}^{\mathcal{M}} \leq b$ pour tout $b \in \mathcal{M}$, on aurait $\mathcal{M} \models \varphi(0, \vec{a}) \wedge \forall x(\varphi(x, \vec{a}) \rightarrow \varphi(Sx, \vec{a}))$, d'où $\mathcal{M} \models \forall x \varphi(x, \vec{a})$ et donc $I = M$, contradiction.

Théorème 16 (Débordement supérieur)
Soit I une coupure propre de $\mathcal{M}$, modèle non standard de l'arithmétique de Peano. Soit $\vec{a} = (a_1 \ldots a_n) \in M^n$ et soit $\varphi(x, x_1, \ldots, x_n)$ une formule du langage de l'arithmétique.

- *(i) Si $\mathcal{M} \models \varphi(i, \vec{a})$ pour tout $i \in I$, il existe $\alpha \in M \backslash I$ majorant strict de I tel que $\mathcal{M} \models \forall x \leq \alpha\, \varphi(x, \vec{a})$.*
- *(ii) Si pout tout $i \in I$ il existe $j \geq i$ tel que $\mathcal{M} \models j \geq i \wedge \varphi(j, \vec{a})$, alors pour tout $\alpha \in M$ tel que $\alpha > I$, il existe β tel que $I < \beta < \alpha$ et $\mathcal{M} \models \varphi(\beta, \vec{a})$.*

<u>Démonstration:</u>
(i). Si un tel $\alpha > I$ n'existe pas, on a $\mathcal{M} \models \forall x \leq \alpha\, \varphi(x, \vec{a}) \Rightarrow \alpha \in I$. Mais $\alpha \in I \Rightarrow \mathcal{M} \models \forall x \leq \alpha\, \varphi(x, \vec{a})$. Alors φ serait une définition dans le langage de l'arithmétique de I, ce qui est impossible.
(ii). Soit α quelconque de M satisfaisant $\alpha > I$. Par hypothèse, pour tout $i \in I$, $\mathcal{M} \models \exists j(i \leq j < \alpha \wedge \varphi(j, \vec{a}))$. On prend $\psi(i, \vec{a}, \alpha)$ la formule $\forall j(i \leq j < \alpha \wedge \varphi(j, \vec{a}))$.
Comme ψ est vraie pour tout $i \in I$, alors d'après (i), il existe $\beta > I$ tel que $\mathcal{M} \models \forall x \leq \psi(i, \vec{a}, \alpha)$, c'est-à-dire:
$\mathcal{M} \models \forall x \leq \beta \exists (x \leq j < \alpha \wedge \varphi(j, \vec{a}))$ donc $\mathcal{M} \models \exists j(\beta \leq j < \alpha \wedge \varphi(j, \vec{a}))$, et en conséquence il existe un β' tel que $\mathcal{M} \models (\beta \leq \beta' < \alpha) \wedge \varphi(\beta', \vec{a})$, d'où il existe $I < \beta' < \alpha$ tel que $\mathcal{M} \models \varphi(\beta', \vec{a})$, et en définitive β' convient.

On rappelle brièvement la structure des modèles non standard de l'arithmétique de Peano: ils sont de type d'ordre "$\mathbb{N} + \mathbb{Q} \times \mathbb{Z}$", c'est-à-dire qu'on a une section initiale isomorphe au modèle standard, puis des "galaxies" isomorphes à $\mathbb{Z}$, ces galaxies étant ordonnées comme l'ensemble des rationnels.

Nous sommes maintenant en mesure de démontrer le résultat annoncé au début de cette section:
<u>Démonstration:</u> (du théorème 15)
$\Rightarrow$: On suppose que pour tout $k \in \mathbb{N}$, il existe x et y distincts tels que pour tout $i \in \{0..k\}$, $SUPP(x+i) = SUPP(y+i)$.
Soit $\mathcal{M}$ un modèle non standard de $\mathcal{T}\mathrm{h}(<\mathbb{N}, +, \times, =>)$ et $\alpha \in \mathcal{M} \backslash \mathbb{N}$. Par débordement supérieur, on peut trouver a et b distincts dans $\mathcal{M}$ tels que:
$\mathcal{M} \models \forall i < \alpha(SUPP(a+i) = SUPP(b+i))$.
On considère alors la bijection f de $\mathcal{M}$ dans lui-même définie par:

- $\forall z \in \mathbb{N},\ f(a+z) = b+z$
- $\forall z \in \mathbb{N},\ f(b+z) = a+z$
- f coïncide avec l'identité ailleurs

f échange deux demi-galaxies, $a+\mathbb{N}$ et $b+\mathbb{N}$, elle ne peut donc pas respecter l'ordre. D'autre part, c'est un $\{S, \perp\}$-automorphisme: c'est évident pour S, et pour $\perp$, cela vient de la formule vérifiée dans $\mathcal{M}$ par les supports des $a+i$ et $b+i$.
Dès lors, si $+$ et $\times$ étaient définissables dans $\{S, \perp\}$, l'ordre le serait également et serait alors préservé par f: contradiction.
$\Leftarrow$: Si un tel k existe, 5^x est le seul entier z primaire de base 5, i.e. de la forme 5^α, vérifiant: pour tout $i \in \{0..k\}$, pour tout premier p, p divise $(x+i)$ si et seulement si p divise $\log_5(z)+i$. Or $z \mapsto \log_5(z)+i$ est récursive, donc sa restriction aux entiers primaires est définissable dans $<\mathbb{N}, S, \perp>$, d'où sa restriction à $5^{\mathbb{N}}$ l'est également. On peut alors définir $x \mapsto 5^x$, ce qui entraîne le résultat voulu.

2.2 Résultats de F.Nézondet : quelques modèles de l'arithmétique

Le but de cette partie est de présenter quelques résultats récents en ce qui concerne la conjecture de Woods-Erdös. On s'intéresse à des modèles non standard d'une portion de l'arithmétique de Peano, que l'on va essayer d'étendre au maximum, dans lesquels la conjecture n'est pas vérifée.

Les outils intervenant dans cette partie sont définis dans les deux premières sections.

Nous verrons ensuite un exemple de modèle de l'arithmétique finie (c'est-à-dire les axiomes de base, sans le schema d'axiomes d'induction), avec division euclidienne, dans lequel la conjecture n'est pas vérifiée pour un entier k donné. Puis nous pourrons grâce aux ultraproduits construire un modèle d'un plus gros morceau de l'arithmétique qui contredit la conjecture pour tout k.

Ces modèles sont présentés par Francis Nézondet dans sa thèse ([9]). Nous y renvoyons pour ce qui concerne les démonstrations de cette section.

2.2.1 Notion d'abritement

L'idée de la notion d'abritement est de transporter certaines propriétés liées au support d'un intervalle à un autre intervalle translaté du premier.

2.2.1.1 Définitions

On généralise la notion de support en application de type support:

Définition 22 (Application de type support)
Soit $[a..b]$ un intervalle dans $\mathbb{N}$ de longueur $l = b - a + 1$. Une application $\varphi : [a..b] \to \mathcal{P}$ est de type support si:

- *(i) Pour tous x, x' dans $[a..b]$, pour tout entier premier $q \leq l$ tel $q \in \varphi(x)$, alors $q \in \varphi(x')$ si et seulement si $q \mid\mid x - x' \mid$*
- *(ii) Pour tout $q \in \mathcal{P}$, $q \leq l$, il existe $x \in [a..b]$ tel que $q \in \varphi(x)$*
- *(iii) Pour tout $q \in \mathcal{P}$, $q > l$, il y a au plus un $x \in [a..b]$ tel que $q \in \varphi(x)$.*

Exemples:

1. La fonction support est l'exemple qui a inspiré cette notion
2. Les applications composées du support par une translation sont également de type support

3. Sur $[0..k]$, on va définir une application φ: On pose $Q = \mathcal{P} \cap [1..k+1]$, et $k+1 < p_0 < p_1 < \ldots < p_k$ des premiers.
   Pour tout $0 < i \leq k$, on pose $\varphi(i) = \{p_i\} \cup SUPP(i)$, et $\varphi(0) = \{p_0\} \cup Q$. On va montrer que c'est une application de type support:
   - (i): Soient x et x' dans $[0..k]$, et $q \in \varphi(x)$, $q \leq l = k+1$. Alors nécessairement, si $x \neq 0$, on a $q \in SUPP(x)$, et si $x' \neq 0$, on a $q \in \varphi(x') \Leftrightarrow q \in SUPP(x) \cap SUPP(x') \Leftrightarrow q \mid\mid x - x' \mid$. Si l'un des deux est nul, c'est évident.
   - (ii): Soit $q \in \mathcal{P}$, tel que $q \leq k+1$ alors si $q \leq k$ on a $\varphi(q)$ et si $q = k+1$, alors $q \in Q \cap [1..k+1] = \varphi(0)$
   - (iii): Si $q \in \mathcal{P}$ et $q > k+1$, alors on a au plus un $x \in [0..k]$ tel que $q \in \varphi(x)$.

Ce dernier exemple n'est pas innocent: il nous servira plus tard.

Définition 23 (Abritement d'une application de type support)
Soit $\varphi : [a..b] \to \mathcal{P}$ de type support, $[a..b]$ de longueur l. On dit qu'un intervalle $[a'..b']$ de même longueur que $[a..b]$ abrite φ si $\forall i \in [0..l-1]$, on a $\varphi(a+i) \subseteq SUPP(a'+i)$

Cette notion peut se représenter sous la forme du schema suivant, où $\varphi(x)$ est représenté par le patatoïde surmontant x et $SUPP(y)$ par le rectangle surmontant y.

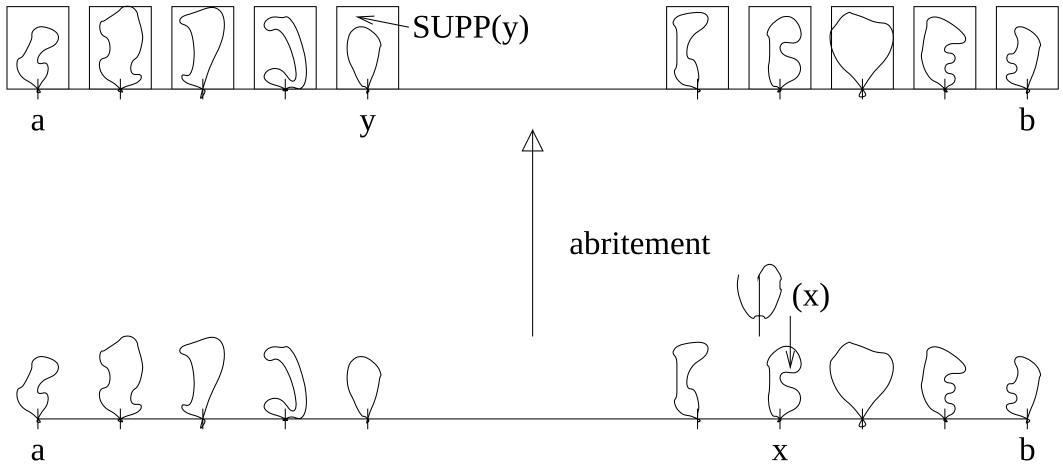


Figure 2.1: Abritement d'une application de type support

Le terme abritement illustre le fait qu'on "abrite" l'ensemble $\varphi(x)$ dans un support qu'on peut formaliser dans le langage $\{S, \perp\}$, au prix d'une translation bénigne.

2.2.1.2 Résultats

Le premier résultat est fondamentalement sympathique, puisqu'il confirme l'intérêt de la notion d'abritement, que l'on va pouvoir utiliser pour la suite:

Proposition 2 (Abritement)
Soit $[a..b]$ un intervalle de $\mathbb{N}$ et φ une application de type support sur $[a..b]$. il existe un intervalle translaté $[a+t..b+t]$ qui abrite φ, c'est-à-dire tel que $\forall x \in [a..b]$, on a $\varphi(x) \subseteq SUPP(x+t)$.

La démonstration de ce résultat se fait par l'absurde.

Lemme 1 (Abritement éliminatoire)
Soit $[a..b]$ un intervalle dans les entiers naturels, Q un ensemble fini de premiers strictement supérieurs à la longueur l de l'intervalle, et disjoint de $SUPP([a..b])$.
Alors il existe une infinité d'entiers t tels que $\forall x \in [a..b]$:

- *(i) $SUPP(x) \subseteq SUPP(x+t)$*
- *(ii) $SUPP([a+t..b+t]) \cap Q = \emptyset$*

On dit que l'intervalle translaté, noté $tr_{[a..b]}$, est obtenu par élimination de Q dans $[a..b]$.

Remarque: Dans ce lemme, on prend comme cas particulier d'application de type support la fonction support elle-même.

Proposition 3 (Abritement éliminatoire avec maîtrise de l'exposant n.)
On se fixe un entier n, un intervalle $[a..b]$ de $\mathbb{N}$ et $\varphi : [a..b] \to \mathcal{P}$ une application de type support. On suppose que: pour tout $x \in [a..b]$, il existe au moins un entier premier $q_x > l = b-a+1$ appartenant à $\varphi(x)$ (les q_x sont donc deux à deux disjoints d'après la propriété (iii) des applications de type support).
On se donne également un ensemble Q de premiers strictement supérieurs à l, disjoint de $\bigcup_{x\in[a..b]} \varphi(x)$.
Alors il existe un intervalle translaté par t de $[a..b]$ qui abrite φ et tel que pour tout $x \in [a..b]$, on a: $q_x^n \mid\mid x+t$ et $SUPP([a+t..b+t]) \cap Q = \emptyset$.

La démonstration fait intervenir des systèmes de congruence et le théorème des restes chinois.

2.2.2 Rappels sur les ultraproduits

2.2.2.1 Ultrafiltre

On se place dans une algèbre de Boole $< \mathcal{A}, \leq, 0, 1 >$. On rappelle qu'un ultrafiltre $\mathcal{U}$ est un filtre maximal, caractérisé par les propriétés suivantes:

1. $0 \notin \mathcal{U}$ et $1 \in \mathcal{U}$
2. $\forall x,\, y \in \mathcal{U}\ x \frown y \in \mathcal{U}$
3. $\forall x \in \mathcal{U},\ \forall y \in \mathcal{A},\ y \geq x \Rightarrow y \in \mathcal{U}$
4. $\forall x \in \mathcal{A},\ x \in \mathcal{U}$ ou $x^c \in \mathcal{U}$

Par la suite, on se fixera un ultrafiltre sur l'algèbre de Boole des parties de $\mathbb{N}$, car $\mathbb{N}$ sera l'ensemble des indices des suites que nous considérerons.
On supposera que $\mathcal{U}$ contient le filtre de Fréchet des parties cofinies de $\mathbb{N}$. Un tel ultrafiltre existe d'après le théorème de Krull. (On pourra consulter [5] pour les notions de base sur les ultrafiltres)

2.2.2.2 Ultraproduit

Si $\mathcal{L}$ est un langage, $(\mathcal{M}_i)_{i\in I}$ une famille de $\mathcal{L}$-structures, et $\mathcal{U}$ un ultrafiltre sur l'algèbre de Boole des parties de I, on peut définir une $\mathcal{L}$-structure appelée ultraproduit des $(\mathcal{M}_i)_{i\in I}$ modulo $\mathcal{U}$, notée $\mathcal{M} = (\prod_{i\in I} \mathcal{M}_i)/\mathcal{U}$, de la manière suivante:

- Les éléments de base sont les classes d'équivalences de suites d'éléments $(a_i)_{i\in I}$ tels que $\forall i \in I, a_i \in M_i$, pour la relation d'équivalence induite par $\mathcal{U}$: $(a_i)_{i\in I} \sim (b_i)_{i\in I}$ si et seulement si $\{i \in I/a_i = b_i$ dans $\mathcal{M}_i\} \in \mathcal{U}$;

- Les symboles de constantes sont interprétés comme la classe d'équivalence de la suite des interprétations de cette constante dans les $\mathcal{M}_i$;

- Les symboles de fonctions: si f est un symbole de fonction d'arité n, on interprète f dans $\mathcal{M}$ comme la fonction qui à un uple $(a_1, \ldots, a_n)$ d'éléments de M associe la classe d'équivalence modulo $\mathcal{U}$ de la suite $(f_i(a_1^i, \ldots, a_n^i))_{i\in I}$, où f_i, a_k^i sont les interprétations respectives de f, a_k dans $\mathcal{M}_i$;

- Les symboles de relation: si R est un symbole de relation d'arité n, on l'interprète dans $\mathcal{M}$ par:

$$\{(a_1, \ldots, a_n) \in M^n/\{i \in I/(a_1^i, \ldots, a_n^i) \in R_i\} \in \mathcal{U}\}$$

  où R_i est l'intepprétation de R dans $\mathcal{M}_i$.

On vérifie facilement qu'il s'agit d'une $\mathcal{L}$-structure, que l'on appelle ultra-produit des $\mathcal{M}_i$ modulo $\mathcal{U}$.

Lorsque tous les $\mathcal{M}_i$ sont égaux à une même structure $\mathcal{M}'$, on dit que $\mathcal{M}$ est une ultrapuissance de $\mathcal{M}'$, et on note: $\mathcal{M} = \mathcal{M}'^I/\mathcal{U}$.

2.2.2.3 Théorème de Łos

Ce qui va nous servir par la suite est la conservation de la satisfaction des formules par une ultrapuissance. En effet le théorème de Łos nous dit:

Théorème 17 (Łos)
Soit $(\mathcal{M}_i)_{i\in I}$ une famille de $\mathcal{L}$-structures et $\mathcal{U}$ un ultrafiltre sur l'ensemble des parties de I. Pour toute formule $F[x_1, \dots, x_n]$ de $\mathcal{L}$ et tout uple $(a_1, \dots, a_n)$ d'éléments de $\prod_{i\in I} M_i$, on a:

$$\prod_{i\in I} \mathcal{M}_i/\mathcal{U} \models F[\bar{a_1}, \dots, \bar{a_n}] \ ssi\ \{i \in I/\mathcal{M}_i \models F[a_1^i, \dots, a_n^i]\} \in \mathcal{U}.$$

On voit que dans le cas d'une ultrapuissance, si on note $\bar{a}$ la classe d'équivalence modulo $\mathcal{U}$ de la suite constante égale à a indexée par I, on a:
$\begin{array}{ccc} \mathcal{N} & \to & \mathcal{M} \\ a & \mapsto & \bar{a} \end{array}$ est un morphisme élémentaire de $\mathcal{L}$-structures.
Ainsi, un ultraproduit va permettre de construire un modèle d'une infinité de $\mathcal{L}$-formules (F_k), à partir du moment où l'on sait construire un modèle $\mathcal{M}_1$ de F_1, un modèle $\mathcal{M}_2$ de F_1 et F_2, ..., un modèle $\mathcal{M}_k$ de F_1 et ... F_k, ...

Définition 24 (Ultra-entier)
Dans le cas d'une ultrapuissance, on appelera ses éléments des ultra-entiers.

2.2.3 Définition d'un premier modèle

On va construire un modèle de l'arithmétique finie avec division euclidienne qui contredit la conjecture d'Erdös-Woods pour un entier $k > 2$, c'est-à-dire qu'on va chercher à construire un intervalle $[a..a + k]$ et un entier $t \neq 0$ tels que pour tout $i \in [0..k]$, on a $SUPP(a + i) = SUPP(a + t + i)$. Ainsi on aura deux entiers distincts a et $a+t$ tels que $a \cong_{\{0,\dots,k\}} a+t$, ce qui contredit Erdös-Woods pour k et pour tous les $k' \leq k$.

Pour construire cette structure, on va s'appuyer sur un entier non standard m qui sera un ultra-entier, et on prendra comme modèle les valeurs "positives" prises par les polynômes à coefficients rationnels en m, dans une ultrapuissance de $\mathbb{Z}$.

2.2.3.1 Construction d'un ultra-entier idoine

On va en premier lieu construire l'ultra-entier m. Pour cela, on a besoin de définir une suite d'entiers m_i, en utilisant les notions liées à l'abritement.
D'après les lemmes d'abritement avec le troisième exemple d'application de type support, on obtient un entier a tel que chaque nombre a, $a+1$, $\ldots$, $a+k$ est divisible par un entier premier qui ne divise aucun autre $a+i$.
On construit $(m_i)_{i\in\mathbb{N}}$ par récurrence sur i:
On pose $m_0 = a$, et si on considère la suite construite jusqu'au rang $i-1$, et vérifiant pour tous $r < r' < i$, $(m_{r'}/m_r) > 3^{r'(r'-r)}$.
On définit l'ensemble $C_i = \{p^\alpha/p \in P,\, p < i^i,\, p^\alpha \leq i^i$ et $p^{\alpha+1} > i^i\}$, et m_i vérifiant:

1. m_i est un élément de l'ensemble des solutions du système de congruences suivant: $\{x \equiv a\,[\text{mod } p^\alpha]\}_{p^\alpha \in C_i}$.

2. m_i est la plus petite solution de ce système telle que pour tout $r < i$, on ait $(m_i/m_r) > 3^{i(i-r)}$.

Cet entier vérifie une propriété qui nous sera bien utile par la suite, et qui nous permet d'éloigner vers l'infini les diviseurs premiers de $m_i + j$ qui ne sont pas des diviseurs premiers de $a + j$, pour $j \in \{0..k\}$:

Proposition 4
Il existe un indice i_0 tel que pour tout $i > i_0$ et tout $j \in \{0..k\}$, on ait $SUPP(m_i + j) \cap [1..i^i] = SUPP(a + j)$.

Démonstration:
Soit i_0 le plus petit entier tel que $i_0^{i_0} > a + k$. Soit $i > i_0$, $0 \leq j \leq k$ et $q \in SUPP(m_i + j) \cap [1..i^i]$. D'après la construction de m_i (congruences), on a $m_i \equiv a\,[\text{mod } q]$. Comme on a en outre $m_i + j \equiv 0\,[\text{mod } q]$, on a également $a + j \equiv 0\,[\text{mod } q]$, d'où $q \in SUPP(a + j)$.
Réciproquement, si $q \in SUPP(a + j)$, on a $q \leq a + k < i_0^{i_0} < i^i$, donc par construction de m_i, on a $m_i \equiv a\,[\text{mod } q]$, d'où $a + j \equiv 0\,[\text{mod } q]$ entraîne que $m_i + j \equiv 0\,[\text{mod } q]$, et finalement on a bien $q \in SUPP(m_i + j) \cap [1..i^i]$.

2.2.3.2 Construction du modèle

On va maintenant construire une structure du langage de l'arithmétique $\mathcal{L}_A = \{+, \times, =, \leq, 0, 1\}$.
On construit dans un premier temps une structure, notée $\mathbb{Z}_k^\omega$, sous-structure de $\mathbb{Z}^\mathbb{N}/\mathcal{U}$.
Le domaine est défini comme suit:
On pose $W = \{(z_n)_{n\in\mathbb{N}} \in \mathbb{Q}^\mathbb{N}/\exists P(X) \in \mathbb{Q}[X]$ tel que $\{n \in \mathbb{N}/z_n = P(m_n)$ et $z_n \in \mathbb{Z}\} \in \mathcal{U}\}$. On prend donc en quelque sorte les polynômes à coefficients rationnels qui ont des valeurs presque partout entières en m modulo $\mathcal{U}$.

$\mathbb{Z}_k^\omega$ est $W/\simeq$ où $\simeq$ est la relation d'équivalence modulo $\mathcal{U}$. On confondra dans la suite $\mathbb{Z}_k^\omega$ avec une famille composée d'un représentant de chaque classe, contenant $\mathbf{0}$ et $\mathbf{1}$, les suites constantes égales à 0 et 1 respectivement. On interprète les symboles de la façon suivante:

- Constantes:
  $0 \mapsto \mathbf{0}$
  $1 \mapsto \mathbf{1}$
- Fonctions:
  $(x_i) + (y_i) = (z_i) \Leftrightarrow \{i \in \mathbb{N}/x_i + y_i = z_i\} \in \mathcal{U}$
  $(x_i) \times (y_i) = (z_i) \Leftrightarrow \{i \in \mathbb{N}/x_i \times y_i = z_i\} \in \mathcal{U}$
- Relations:
  $(x_i) \leq (y_i) \Leftrightarrow \{i \in \mathbb{N}/x_i \leq y_i\} \in \mathcal{U}$
  $(x_i) = (y_i) \Leftrightarrow \{i \in \mathbb{N}/x_i = y_i\} \in \mathcal{U}$

Définition 25
On dit qu'un ultra-entier $x = (x_i)$ est fini s'il existe $a \in \mathbb{Z}$ tel que $\{i \in \mathbb{N}/x_i = a\} \in \mathcal{U}$.

On vérifie facilement que $\mathbb{Z}_k^\omega$ ainsi défini est une $\mathcal{L}$-structure, grâce à la stabilité par les opérations $+$ et $\times$ dans respectivement $\mathbb{Q}[X]$ et $\mathbb{Z}$, et à la compatibilité avec $\simeq$.

On définit maintenant la structure qui nous intéresse: $< \mathbb{N}_k^\omega, \mathcal{L}_A >$, sous-structure de $< \mathbb{Z}_k^\omega, \mathcal{L}_A >$ de domaine $\mathbb{N}_k^\omega = \{z \in \mathbb{Z}_k^\omega / z \geq 0\}$.

2.2.3.3 C'est un modèle de l'arithmétique finie avec division euclidienne ...

On travaille sur $\mathbb{Z}_k^\omega$, et on se ramène ensuite à $\mathbb{N}_k^\omega$.
On rappelle les axiomes de l'arithmétique finie des entiers relatifs:

(A2)$\forall x \exists y (y + 1 = x)$
(A3)$\forall x \forall y \forall z (x + z = y + z \Rightarrow x = y)$
(A4)$\forall x (x + 0 = x)$
(A5)$\forall x \forall y \forall z (x + (y + z) = (x + y) + z)$
(A6)$\forall x (x \times 0 = 0)$
(A7)$\forall x \forall y (x \times (y + 1) = x \times y + x)$
(A8)$\forall x \forall y (x \leq y + 1 \Leftrightarrow (x \leq y) \vee (x = y + 1))$
(A9)$\forall x \forall y (x \leq y) \vee (y \leq x)$

On considérera également le schéma d'axiomes suivants, pour la division euclidienne: pour tout $n > 1$,

$$\textbf{(S)} \forall x \exists y ((0 \leq r < n) \wedge (x = n \times y + r)).$$

Si on montre que les axiomes universels sont vrais dans $\mathbb{Z}_k^\omega$, alors il seront automatiquement vrais dans $\mathbb{N}_k^\omega$. Pour les autres axiomes, il faudra simplement vérifier que les "$\exists$" ne font pas sortir de la sous-structure.

Comme $\mathbb{Z}_k^\omega$ est une sous-structure de $\mathbb{Z}^{\mathbb{N}}/\mathcal{U}$ qui vérifie tous les axiomes ci-dessus, on a automatiquement les axiomes universels vérifiés dans $\mathbb{Z}_k^\omega$ et donc dans $\mathbb{N}_k^\omega$. en définitive on aura à régler les cas de **(A2)** et **(S)**.

(A2) $\forall x \exists y (y+1=x)$

Soit $x=(x_i)_{i\in\mathbb{N}}$ un élément de $\mathbb{Z}_k^\omega$. On pose $y'=(x_i-1)_{i\in\mathbb{N}}$. Si x est fini, on constate immédiatement que y' est celui qu'on cherche.
Sinon $\exists P(X)\in\mathbb{Q}[X]$ tel que $\{i\in\mathbb{N}/P(m_i)\in\mathbb{Z} \text{ et } P(m_i)\in\mathbb{Z}\}\in\mathcal{U}$. On considère $P(X)-1\in\mathbb{Q}[X]$:
$\{i\in\mathbb{N}/P(m_i)-1\in\mathbb{Z} \text{ et } P(m_i)-1=x_i-1=y'_i\}\in\mathcal{U}$, donc y' a un représentant de sa classe, y, dans W. Et il est évident que $y+1=x$ dans $\mathbb{Z}_k^\omega$ car $\{i/y_i=y'_i\}$, $\{i/P(m_i)=x_i\}$ et $\{i/P(m_i)-1=y'i\}$ sont dans $\mathcal{U}$ et donc $\{i/y_i+1=x_i\}\in\mathcal{U}$.

Adaptation à $\mathbb{N}_k^\omega$: on veut que cette structure vérifie l'axiome modifié: **(A2 bis)**$\forall x\exists y((x\neq 0)\to(y+1=x))$. On considère donc un élément non nul x de $\mathbb{N}_k^\omega$. On pose $y=(x_n-1)$. On a vu précédemment que l'on avait l'égalité $y+1=x$ dans $\mathbb{Z}_k^\omega$, et pour cela on montre que $y\leq 0$. Si x est fini, c'est immédiat. Sinon, on a un polynôme $P(X)\in\mathbb{Q}[X]$ qui correspond à x. La suite des (m_i) tend vers $+\infty$ donc si le polynôme est de degré non nul (donc non constant), la suite $(P(m_i))$ tend également vers $+\infty$. Donc $\{i/P(m_i)<1\}$ est fini, et par suite, comme $\mathcal{U}$ contient les ensembles cofinis, $\{i/P(m_i)>0\}\in\mathcal{U}$. D'où $\{i/P(m_i)-1=y_i \text{ et } P(m_i)-1>0\}\in\mathcal{U}$, c'est ce qu'on voulait.

(S) Division euclidienne

On procède de manière analogue. Le détail est exposé dans [9]. Regardons simplement comment adapter le cas $\mathbb{Z}_k^\omega$ au cas $\mathbb{N}_k^\omega$: On suppose qu'on a la division euclidienne pour tout $q>1$ dans $\mathbb{Z}_k^\omega$. On se fixe x dans $\mathbb{N}_k^\omega$. On obtient un y et un r *a priori* dans $\mathbb{Z}_k^\omega$. On a $0\leq r<q$, donc r est positif et donc dans $\mathbb{N}_k^\omega$. D'autre part, comme $\mathcal{U}$ est un ultrafiltre, si y n'est pas positif, il est nécessairement négatif, donc si $y\notin\mathbb{N}_k^\omega$, on a, dans $\mathbb{Z}_k^\omega$:
$y<0,\ 0\leq r<g\Rightarrow q\times y<-q$ et donc $q\times y+r\leq -q+r<0$, ce qui est impossible puisque $x=q\times y+r>0$.

2.2.3.4 ... qui contredit la conjecture de Woods-Erdös

On va montrer que $a+j$ et $m+j$ ont même support pour $0\leq j\leq k$, dans $\mathbb{N}_k^\omega$.
On rappelle qu'on avait le résultat suivant: $\exists i_0\ \forall i>i_0\ SUPP(m_i+j)\cap\{1\dots i^i\}=SUPP(a+j)$ pour tout $0\leq j\leq k$. (Proposition 4)

On suppose que $q \in SUPP(a+j)$: alors $\exists i_0\, \forall i > i_0\, q \in SUPP(m_i + j)$, donc $\{i / q \in SUPP(m_i + j)\} \in \mathcal{U}$, d'où en identifiant q et la suite constante égale à q modulo $\mathcal{U}$, $q \in SUPP(m + j)$.

On suppose maintenant que $q \in SUPP(m+j)$. On a $\{i / q \leq i^i\} \in \mathcal{U}$, et $\{i / q \in SUPP(m_i + j)\} \in \mathcal{U}$, donc $\{i / q \leq i^i$ et $q \in SUPP(m_i + j)\} \in \mathcal{U}$. en particulier, cet ensemble est infini, et donc pour un i bien choisi avec $i > i_0$, on a $q \in SUPP(a + j)$.

On a donc deux éléments différents qui vérifient la relation $a \cong_{\{0,\ldots,k\}} m$. La conjecture d'Erdös-Woods est donc fausse dans la structure $\mathbb{N}_k^\omega$, au rang k.

2.2.3.5 D'où un modèle qui contredit Erdös-Woods pour tout rang k fini.

Si on réalise un ultraproduit $\mathbb{N}_\omega^\omega$ des $\mathbb{N}_k^\omega$ pour un ultrafiltre $\mathcal{U}'$ englobant le filtre de Fréchet (par exemple notre bon vieil ultrafiltre arbitraire $\mathcal{U}$): comme chaque $\mathbb{N}_k^\omega$ est un modèle de l'arithmétique finie avec division euclidienne, $\mathbb{N}_\omega^\omega$ en sera un également. D'autre part, pour tout entier k fini, la conjecture d'Erdös-Woods au rang k est fausse dans tous les $\mathbb{N}_{k'}^\omega$ pour $k' \geq k$. Donc elle sera fausse également dans $\mathbb{N}_\omega^\omega$.

2.2.4 Autre modèle plus proche d'un modèle de la théorie complète

On va montrer dans cette partie qu'on peut construire un modèle de l'arithmétique finie avec division euclidienne d'une part, et de la théorie complète de $< \mathbb{N}, +, \times, \perp, 0, 1 >$ (sans égalité, et où $+$ et $\times$ sont des symboles de fonction), qui réfute la conjecture d'Erdös-Woods. Cela ne nous permet pas d'affirmer que cette conjecture est fausse dans $\mathbb{N}$, mais on sait d'ors et déjà que la théorie complète de l'arithmétique ne démontre pas cette conjecture.

2.2.4.1 Ultra-ultra-entiers: quelques propriétés

Pour chaque k, on a défini un ultra-entier $m^{(k)}$ et un entier a_k permettant de construire le modèle $\mathbb{N}_k^\omega$ présenté précédemment. On définit maintenant l'ultra-ultra-entier m_ω par la classe d'équivalence modulo $\mathcal{U}$ de la suite $(m^{(k)})_{k\in\mathbb{N}}$.
On définit également un ultra-ultra-entier à base de la suite $(a_k)_{k\in\mathbb{N}}$: a_w est la classe d'équivalence modulo $\mathcal{U}$ de la suite de suites pseudo-stationnaires à a_k modulo $\mathcal{U}$.
On a les propriétés suivantes:

Proposition 5

1. *$\forall k \in \mathbb{N}$ $\forall p^\alpha$ primaire de $\mathbb{N}$, $\exists i_0$ tel que $\forall i > i_0$, on a:*
   (a) $a_k \equiv m_i^{(k)}$ $[p^\alpha]$
   (b) *Donc par compatibilité de la congruence avec les opérations arithmétiques + et ×: $\forall P(X)$ à coefficients entiers finis: $P(a_k) \equiv P(m_i^{(k)})$ $[p^\alpha]$*
   (c) *Donc par passage à l'ultrapuissance: pour tout polynôme $P[X]$ à coefficients entiers finis: $P(a_k) \equiv P(m^{(k)})$ $[p^\alpha]$*
2. *Donc, par passage à l'ultraproduit: pour tout polynôme $P[X]$ à coefficients ultra-entiers, pour tout primaire q^α ultra-entier: $P(a_\omega) \equiv P(m_\omega)$ $[q^\alpha]$.*

2.2.4.2 Description du modèle

On construit un ensemble PM de polynômes à une variable, à coefficients ultra-entiers tel que:

1. PM est clos par addition, multiplication, soustraction d'entiers finis positifs sauf quand le résultat est un polynôme constant négatif;
2. Pour tout polynôme de $P(X)$ de $PM[X]$, les ultra-ultra-entiers $P(a_\omega)$ et $P(w_\omega)$ sont positifs, et $P(a_\omega) = 1 \Leftrightarrow P(w_\omega) = 1$;
3. Si $P(X) \in PM[X]$ est un polynôme non constant, alors $P(a_\omega)$ est infini.

A partir de cet ensemble de polynômes, on va construire le domaine du modèle $\mathcal{M}$ recherché:

Définition 26 (Domaine du modèle)
$M = \{P(m_\omega) \backslash P(X) \in PM[X]\}$. De la même façon que précédemment, on déplace par passage à l'ultraproduit les relations d'addition et de multiplication sur $(\mathbb{N}^\omega/\mathcal{U})^\omega/\mathcal{U}$.

On munit cet ensemble du langage $\mathcal{L} = \{0, 1, +, \times, \leq, =\}$.
On construit également un ensemble dans les ultra-ultra-rationnels:

Définition 27 (Ensemble CM)
$CM \subseteq \mathbb{Q}^{\omega^2}/\mathcal{U}$ est l'ensemble des classes d'équivalence de suites de classes de suites constantes de rationnels.

On note CME^+ le sous-ensemble de CM constitué des éléments de CM qui sont des entiers. On a le résultat suivant:

Lemme 2
La structure $< CME^+, 0, 1, +, \times, \leq, => $ *est un modèle de l'arithmétique complète du premier ordre.*

Démonstration:
On se déplace isomorphiquement dans $\mathbb{N}^\omega/\mathcal{U}$, en associant à une suite de suites constantes de rationnels (en l'occurrence des entiers) la suite des rationnels correspondants: par exemple, à la suite ((1, 1, ...), (2, 2, ...), (3, 3, ...), ...), on fait correspondre la suite (1, 2, 3, ...) (le tout modulo $\mathcal{U}$).

On va devoir redéfinir la notion d'entier premier, de façon à la restreindre dans notre gros modèle, mais de façon à ce que la restriction de la notion ainsi définie corresponde avec les entiers premiers sur $\mathbb{N}$:

Définition 28 (Notion d'entier premier, super-premier dans M)
Un nombre premier dans M est un nombre qui n'est divisible que par 1 et par lui-même (définition classique).
Un nombre super-premier dans M est un nombre q premier tel que pour tout n dans M premier et strictement inférieur à $2q$, le produit $q \times n$ admet une seule décomposition en facteurs non triviaux.

Ces deux notions sont définissables au premier ordre dans $\mathcal{M}$, par deux formules qui sont équivalentes dans $\mathbb{N}$. On peut alors montrer le lemme suivant:

Lemme 3
Tous les nombres super-premiers dans M appartiennent à CME^+.

2.2.4.3 C'est un modèle de l'arithmétique finie avec division euclidienne qui contredit Erdös-Woods

Tout comme avec le modèle précedent, la tâche est simplifiée par le fait que tous les axiomes sauf deux sont universels: comme $\mathcal{M}$ est une sous-structure de $< \mathbb{N}^{\omega^2}/\mathcal{U}, \mathcal{L} >$, qui est un modèle de l'arithmétique, $\mathcal{M}$ va vérifier tous les énoncés universels vérifiés par $< \mathbb{N}^{\omega^2}/\mathcal{U}, \mathcal{L} >$. Il reste à prouver les axiomes **(A2 bis)** et **(S)**, ce qui se fait de manière relativement analogue au cas précédent, en manipulant adroitement les propriétés de $\mathcal{U}$, $P(a_\omega)$ et $P(m_\omega)$ (voir [9]).

Pour montrer que ce modèle contredit la conjecture d'Erdös-Woods, on montre la proposition suivante:

Proposition 6
Pour tout entier k fini, $m_\omega + k$ et $a_\omega + k$ ont les mêmes diviseurs super-premiers dans M.

<u>Démonstration:</u>(idée)
Soit q un diviseur super-premier de $a_\omega + k$. On veut montrer que q divise $m_\omega + k$ dans $\mathcal{M}$. On a la relation $q \mid a_\omega + k$, donc il existe $r_0 \in m$ tel que $a_\omega + k = r_0 \times q$. Donc, pour presque tout i, on a: $a^{(i)} + k = q^{(i)} \times r_0^{(i)}$. En appliquant le lemme 4, on a, pour r assez grand, $SUPP(m_r^{(i)} + k) \cap [1..r^r] = SUPP(a^{(i)} + k)$. Si on fixe i, pour $r^r > i$, on a $q^{(i)} \in [1..r^r]$ d'où:

$$q^{(i)} \in SUPP(m_r^{(i)} + k) \Leftrightarrow q^{(i)} \in SUPP(a^{(i)} + k).$$

Comme $q^{(i)}$ est premier et divise $a^{(i)} + k$, on a donc $q^{(i)}$ divise $m_r^{(i)} + k$. Donc il existe $r'^{(i)} \in \mathbb{N}^\omega/\mathcal{U}$ tel que $m_r^{(i)} + k = q^{(i)} \times r'^{(i)}$. D'où, en passant à l'ultrapuissance: $m_\omega + k = q \times r'$. Il reste à montrer que $r' \in M$, c'est-à-dire qu'il est de la forme $P(m_\omega)$ avec $P(X) \in PM[X]$. On montre que $P(X) = (1/q)X + (k/q)$ convient. (voir [9]).

Réciproquement, si q est un diviseur super-premier de $m_\omega + k$, alors on a un r tel que $m_\omega + k = q \times r$. Comme q est super-premier, d'après le lemme 3, q est dans CME^+, et donc c'est un ultra-entier identifié à un ultra-ultra-entier. D'après la proposition 5: *Pour tout polynôme P(X) à coefficients ultra-entiers, pour tout primaire q^r, on a* $P(a_\omega) \equiv P(m_\omega)\ [q^r]$, appliquée au polynôme $X + k$ et au primaire q, on a alors q divise $a_\omega + k$.
Dès lors, il existe un r' entier tel que $a_\omega + k = q \times r'$, avec $q \in CME^+$. Comme $a_\omega + k$ et q sont dans CME^+, r' également, et donc $r' \in M$. donc q divise $a_\omega + k$ dans $\mathcal{M}$, et on a donc montré que la conjecture était fausse dans ce modèle.

Chapter 3

Les théories à nombre borné de quantificateurs

3.1 $Th_2\{\mathbb{N}, S, \perp\}$

3.1.1 Formalisme et conventions pour les figures

Nous allons voir sur cet exemple comment construire “à la main” la 2-destinée (sous-entendu la 2-transversale essentielle) de la théorie des entiers naturels avec successeur et coprimarité, et nous verrons combien il est facile de le faire dans le cas de deux niveaux. Tout d’abord, nous aurons besoin d’introduire un formalisme pour les figures: la racine x de chaque destinée doit figurer les relations de x avec x, et dans ce cas on sait que x ne peut pas être le successeur de x, mais il peut être copremier avec lui même (cas de 1) ou non. Pour cela on indique par un vecteur à une seule composante si on est dans le premier cas ou le deuxième. (voir figure 3.1)

Ensuite, une feuille k doit figurer les relations de k avec lui-même d’une

Figure 3.1: Cas de figures rencontrés pour les racines

part (une composante), et les relations de k avec x, son père, d’autre part (trois composantes: x et k peuvent être copremiers ou non, x peut être le successeur de k ou non, k peut être le successeur de x ou non). Les successions sont formalisées par des flèches: de gauche à droite pour dire que x est le successeur de k, de droite à gauche pour dire que k est le successeur de x. Tous les cas que l’on peut rencontrer sont résumés dans la figure 3.2.

Pour les branches à trois niveaux, ce formalisme s’étend aisément, comme

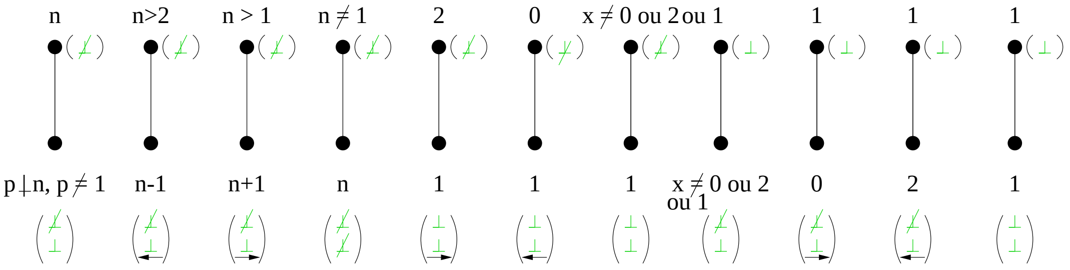


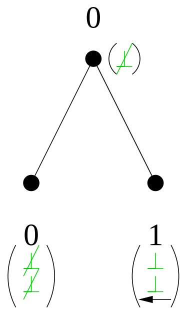
Figure 3.2: Cas de figures rencontrés pour les branches père-fils

on le verra à la section 3.2

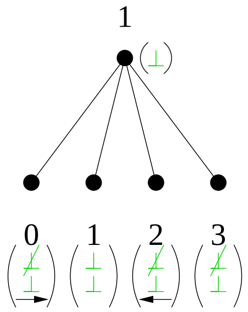
3.1.2 Construction de la destinée

On va faire cette construction progressivement, en commençant par les entiers 0, 1, etc., puis on prouvera qu'on a ainsi tout trouvé.

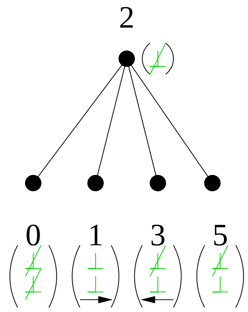
On commence, pour la destinée de 0: 0 n'est pas premier avec lui-même. Ses fils potentiels: 0 n'est pas premier avec lui-même, ni avec 0, et n'a pas de relation de successeur avec 0; 1 est premier avec lui-même, premier avec 0 et est son successeur; 2 n'est pas premier avec lui-même, ni avec 0, et n'a pas de relation de successeur avec 0, il est donc comme le fils "0", on ne le note pas; les entiers suivant vont être également dans cette situation: ils ne sont pas premiers avec eux-mêmes (1 étant le seul et l'unique dans cette situation), ni avec 0 par propriété de 0, et n'ont pas de relation de successeur avec 0: ils sont donc comme le fils "0". Finalement la destinée de 0 contient deux branches, elle est indiquée sur la figure ci-contre.



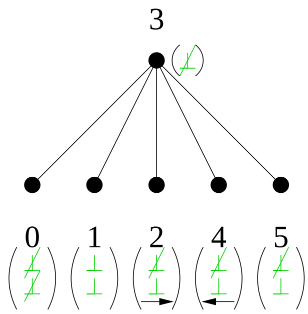
On fait ensuite la même chose avec 1: 1 est premier avec lui-même. Ses fils: 0 n'est pas premier avec lui-même, il est premier avec 1 et c'est son prédéceseur; 1 est premier avec lui-même et avec 1, il n'a pas de relation de succession avec 1; 2 est non premier avec lui-même, il est premier avec 1 et c'est son successeur; 3 est non premier avec lui-même, il est premier avec 1 et n'a pas de relation de successeur avec 1; les entiers supérieurs sont également dans ce dernier cas, on a donc épuisé toutes les possibilités avec 1 comme racine.



La destinée de 2: 2 est non premier avec lui-même. Ses fils: 0 n'est pas premier avec lui-même, ni avec 2, et n'a pas de lien de succession avec 2; 1 est premier avec lui-même et 2, et 2 est le successeur de 1; 2 n'apparaît pas car il vérifie les mêmes prédicats vis-à-vis de 2 que 0; 3 est premier avec 2, pas avec lui-même, et c'est le successeur de 2; 4 est comme 0, de même que tous les nombres pairs supérieurs à 4, donc n'apparaît pas; 5 est non premier avec lui-même, premier avec 2 et n'a pas de relation de succession avec 2, et tous les nombres impairs supérieurs à 5 vérifient également cela.

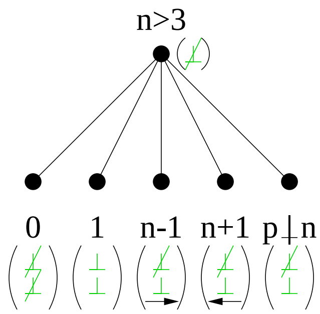


La destinée de 3: 3 est non premier avec lui-même. Ses fils: 0 n'est pas premier avec lui-même, ni avec 3, et n'a pas de lien de succession avec 3; 1 est premier avec lui-même et avec 3, il n'a pas de relation de succession avec 3; 2 est non premier avec lui-même, il est premier avec 3 et 3 est le successeur de 2; 3 est comme 0 donc n'apparaît pas; 4 est non premier avec lui-même, il est premier avec 3 et est le successeur de 3; 5 est non premier avec lui-même, premier avec 3 et n'a pas de lien de succession avec 3; les nombres supérieurs n'ont pas de relation de succession avec 3, ils sont non premiers avec eux-mêmes, et se subdivisent en deux catégories: ceux qui sont premiers avec 3, et alors leur branche est isomorphe à celle de 5, et ceux qui ne sont pas premiers avec 3, et alors leur branche est isomorphe à celle de 0: on a donc tout épuisé.



On va montrer par l'étude du cas $n \geq 3$ général, que l'on a épuisé toutes les possibilités:

La destinée de $n \geq 3$: n est non premier avec lui-même. Ses fils: 0 n'est pas premier avec lui-même, ni avec n, et n'a pas de lien de succession avec n; 1 est premier avec lui-même et avec n, il n'a pas de relation de succession avec n; $n-1$ est non premier avec lui-même, il est premier avec n et n est le successeur de $n-1$; $n+1$ est non premier avec lui-même, il est premier avec n et est le successeur de n; $2n+1$ est non premier avec lui-même, premier avec n et n'a pas de lien de succession avec n; les autres entiers sont non premiers avec eux-mêmes, n'ont pas de relation de succession avec n, et se subdivisent en deux catégories: ceux qui sont premiers avec n, et alors leur branche est isomorphe à celle de $2n+1$, et ceux qui ne sont pas premiers avec n, et alors leur branche est isomorphe à celle de 0: on a donc épuisé tous les cas, et on retrouve une destinée isomorphe à celle de 3.



Voici donc la 2-transversale essentielle de la théorie de $\mathbb{N}$ muni du successeur et de la coprimarité avec deux quantificateurs, présentée en figure

3.3.

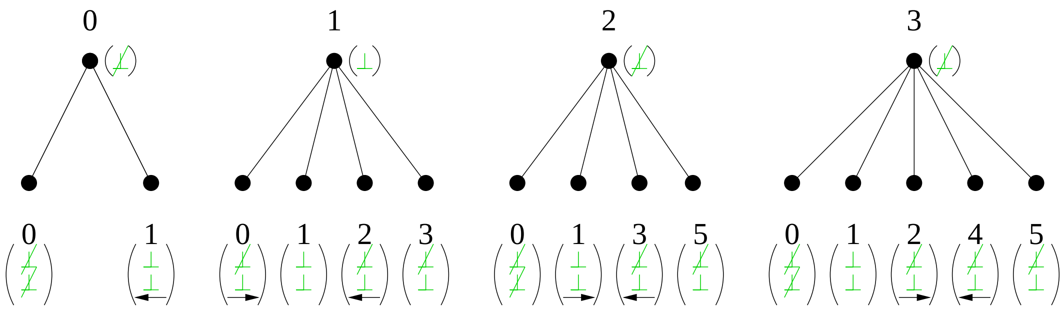


Figure 3.3: 2-transversale essentielle de $Th_2\{\mathbb{N}, S, \perp\}$

3.2 Etude préalable de la 3-transversale essentielle de $Th_3\{\mathbb{N}, S, \perp\}$

3.2.1 Présentation générale

Le cas de la 3-transversale est beaucoup plus compliqué, comme on va le voir, et n'est pas complètement résolu dans le sens où il reste des cas possibles dont on ne sait pas s'il existe un exemple ou au contraire s'ils ne peuvent apparaître. Les travaux existants sur cette 3-transversale ont été réalisés par Denis Richard, Marcel Guillaume et Ji Lei Yin au LLAIC de Clermont-Ferrand.

On va introduire quelques notions:

- On appelera n la racine d'une destinée quelconque, k un fils de cette racine, et u un fils de k
- On dira que k est loin de n s'il n'appartient pas à l'ensemble $\{n-2, n-1, n, n+1, n+2\}$, et que u est loin de k s'il appartient pas à l'ensemble $\{k-1, k+1\}$, et loin de n s'il n'appartient pas à l'ensemble $\{n-1, n+1\}$.
- Sinon on parlera de proximité.
- On supposera $n \geq 3$ et $k \geq 3$, les autres cas seront considérés ultérieurement de manière exhaustive.

3.2.2 Les cas de base sur les supports de n et k

On va définir cinq cas de base dans le cas où k et n sont éloignés, et donc où le successeur n'intervient pas. On définit d'abord les configurations de base: il y a cinq possibilités de configuration associée à u:

1. $u = 0$ n'est premier avec personne
2. $u = 1$ est premier avec tout le monde
3. u est non-premier avec lui-même, et premier avec k et avec n
4. u est non-premier avec lui-même, et premier avec k mais pas avec n
5. u est non-premier avec lui-même, et premier avec n mais pas avec k

Il y a donc au plus cinq branches (n, k, u) éloignées de k et n pour k et n donnés. Le nombre de ces branches va dépendre de la position relative des supports de n et k. On supposera que $Supp(n) = \{p_1, \dots, p_r\}$ et $Supp(k) = \{q_1, \dots, q_s\}$

$Supp(k) \subsetneq Supp(n)$
: Alors on peut réaliser 4 configurations:

  - Le cas 1 est vérifié par $u = 0$
  - Le cas 2 est vérifié par $u = 1$
  - Le cas 3 est vérifié par une solution du système de congruence suivant: $x = 1\ [p_i]$
  - Le cas 4 est vérifié par une solution du système de congruence suivant:
    $\left\{\begin{array}{lcll} x & = & 1 & [q_i] \\ x & = & 0 & [p_j \notin Supp(k)] \end{array}\right.$
  - Le cas 5 est impossible

$Supp(n) \subsetneq Supp(k)$
: Alors on peut réaliser 4 configurations:

  - Le cas 1 est vérifié par $u = 0$
  - Le cas 2 est vérifié par $u = 1$
  - Le cas 3 est vérifié par une solution du système de congruence suivant: $x = 1\ [q_i]$
  - Le cas 4 est impossible
  - Le cas 5 est vérifié par une solution du système de congruence suivant:
    $\left\{\begin{array}{lcll} x & = & 1 & [p_i] \\ x & = & 0 & [q_j \notin Supp(n)] \end{array}\right.$

$Supp(n) = Supp(k)$
: Alors on peut réaliser 3 configurations:

  - Le cas 1 est vérifié par $u = 0$

- Le cas 2 est vérifié par $u = 1$
- Le cas 3 est vérifié par une solution du système de congruence suivant: $x = 1\ [q_i]$
- Le cas 4 est impossible
- Le cas 5 est impossible

$Supp(n) \backslash Supp(k) \neq \emptyset$ **et** $Supp(k) \backslash Supp(n) \neq \emptyset$

Ce cas en regroupe deux, selon que $k \perp n$ ou non. Mais dans ces deux sous-cas, on peut réaliser les 5 configurations:

- Le cas 1 est vérifié par $u = 0$
- Le cas 2 est vérifié par $u = 1$
- Le cas 3 est vérifié par une solution du système de congruence suivant: $\left\{ \begin{array}{lll} x & = & 1 \quad [q_i] \\ x & = & 1 \quad [p_i \notin Supp(k)] \end{array} \right.$
- Le cas 4 est vérifié par une solution du système de congruence suivant:
$\left\{ \begin{array}{lll} x & = & 1 \quad [q_i] \\ x & = & 0 \quad [p_j \notin Supp(k)] \end{array} \right.$
- Le cas 5 est vérifié par une solution du système de congruence suivant:
$\left\{ \begin{array}{lll} x & = & 1 \quad [p_i] \\ x & = & 0 \quad [q_j \notin Supp(n)] \end{array} \right.$

Ces cas sont résumés dans la figure 3.4

3.2.3 Cas possibles pour les branches (k, u) proches de k ou n, lorsque k et n sont éloignés

On suppose tout d'abord que $k \geq 3$, pour éviter que le successeur intervienne dans les pétioles $(n, k, 0)$ et $(n, k, 1)$. On a donc, pour un couple (k, n) donné, au plus cinq fils de k qui sont éloignés à la fois de k et de n. Il reste à étudier ce qui se passe pour les fils de k qui sont soit proches de k, soit proches de n, c'est-à-dire les éléments de l'ensemble $\{k-1, k+1, n-1, n+1\}$. Pour chacun de ces éléments, il peut être premier avec n ou non, et premier avec k ou non. Pour k et n éloignés, les interventions de la relation de succession sont bien déterminées, et donc ne portent pas à discussion:

- $k-1$ est le prédécesseur de k, et n'a aucun lien de succession avec n
- $k+1$ est le successeur de k, et n'a aucun lien de succession avec n
- $n-1$ est le prédécesseur de n, et n'a aucun lien de succession avec k
- $n+1$ est le successeur de n, et n'a aucun lien de succession avec k

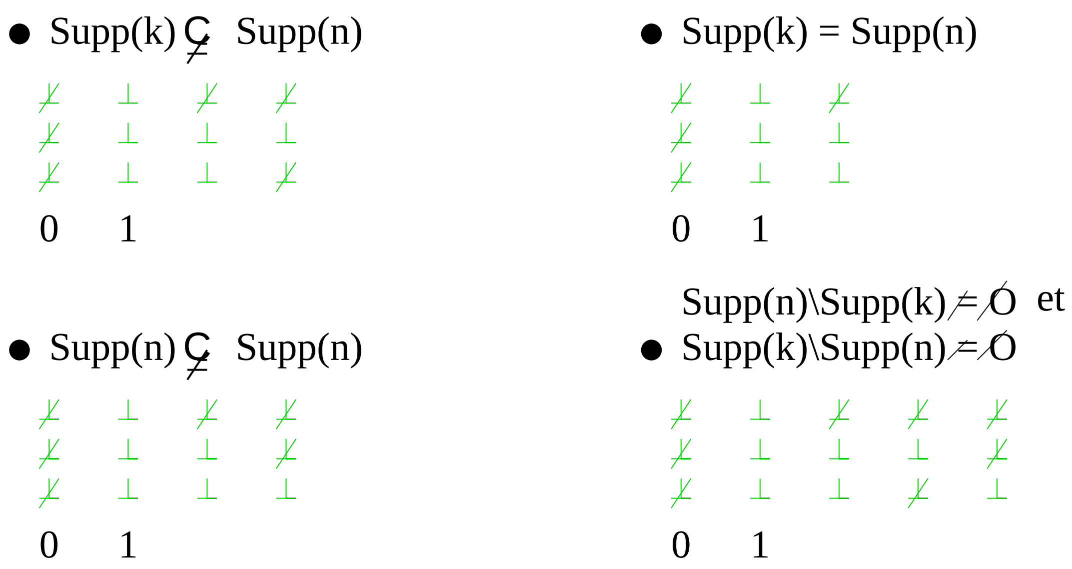


Figure 3.4: Les cas u éloignés de k et n

D'autre part, on sait déjà que certaines relations de coprimarité sont également fixées, car s'il y a une relation de succession entre deux nombres, ils sont nécessairement premiers entre eux:

- $k-1 \perp k$
- $k+1 \perp k$
- $n-1 \perp n$
- $n+1 \perp n$

Il reste donc à discuter des liens de coprimarité de:

- $k-1$ avec n
- $k+1$ avec n
- $n-1$ avec k
- $n+1$ avec k

Pour chacun de ces cas, on a deux possibilités: soit les deux nombres sont premiers entre eux, soit ils ne le sont pas. Cela porte donc à 16 le nombre de cas possibles pour les configurations de branches (n, k, u) pour u proche de n ou k. Ces configurations apparaissent en figure 3.5.

Si on effectue un premier pointage des possibilités:
Pour chaque couple (n, k) éloignés, il y a 5 possibilités pour les branches

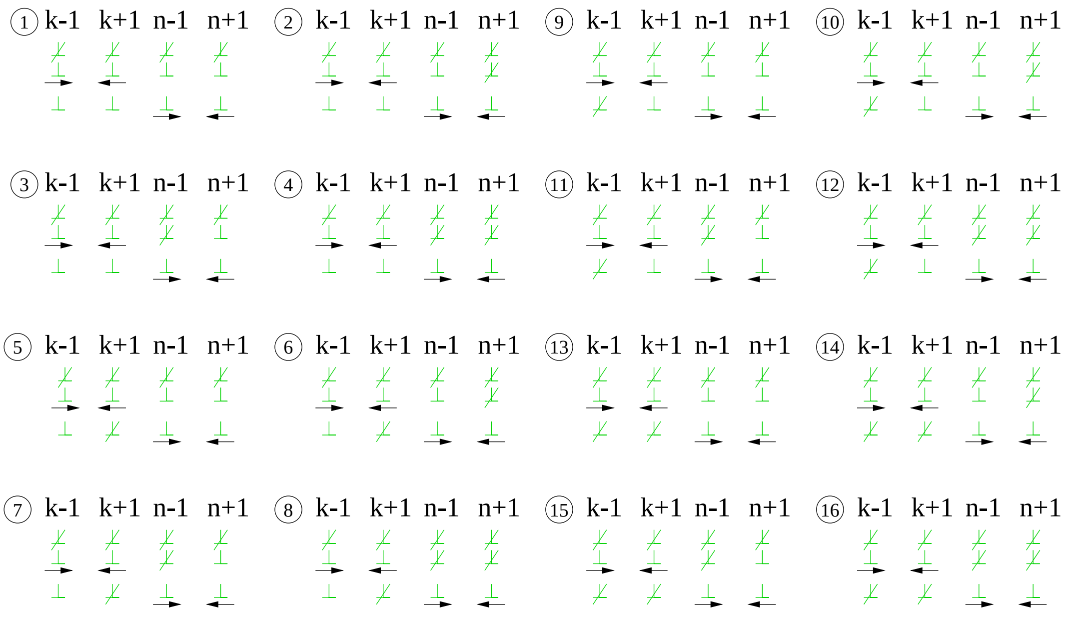


Figure 3.5: Configurations pour u proche de k ou n, k et n éloignés

(n, k, u) pour u éloigné de k et n, et 16 possibilités pour les branches (n, k, u) pour u proche de k ou de n, ce qui porte donc à $5 \times 16 = 80$ le nombre de pétioles (n, k) avec n et k éloignés qu'il est possible de rencontrer dans la 3-transversale essentielle.

Chacune de ces branches pouvant apparaître ou non dans la destinée d'un n donné, il y a a priori 2^{80} possibilités. Nous verrons plus tard qu'il est possible de réduire considérablement ce nombre.

3.2.4 Etude des branches (n, k) pour k proche de n

On étudie maintenant ce qui se passe pour n et k proches, en supposant $n > 3$ (pour que $n - 2$ soit différent de 0 ou 1). On va trouver différents cas selon la forme de n. Sur les figures sont notées à gauche les branches où u est éloigné de k et n, à droite les branches où u est proche de k ou n, et les cas nécessitant discussion sont marqués par des lettres entourées.

3.2.4.1 Pétiole où $k = n - 2$

Commençons la discussion sur ce qui se passe entre n et $n - 2$ au niveau de la coprimarité:

- $n \perp n - 2$ si n est impair et alors on a les 5 possibilités pour les branches (n, k, u) pour u éloigné de k et n réalisées
- $n \not\perp n - 2$ si n est pair, et alors la discussion porte sur les supports

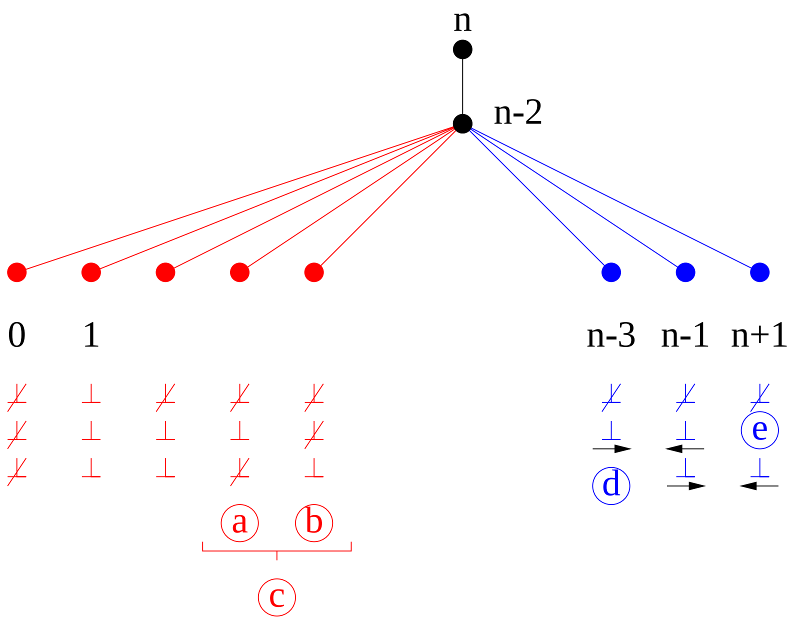


Figure 3.6: Pétiole $(n, n-2)$

Plaçons nous dans le cas où n est pair: remarquons que $Supp(n) \cap Supp(n-2) = \{2\}$. On a alors quatre cas possibles:

- $Supp(n-2) \subsetneq Supp(n)$: Ce cas est possible si et seulement si n est de la forme $2(2^{\alpha}+1)$, avec $\alpha \geq 0$ (car alors $Supp(n-2) = \{2\}$): le schéma "a" est absent;

- $Supp(n) \subsetneq Supp(n-2)$: Ce cas est possible si et seulement si n est de la forme 2^{α}, avec $\alpha > 0$: le schéma "b" est absent;

- $Supp(n) = Supp(n-2) = \{2\}$: Ce cas correspond à $n = 4$. On exclut désormais ce cas de la discussion générale, on le traitera à part;

- Le dernier cas, quand chaque support contient au moins un élément qui ne se trouve pas dans l'autre, intervient pour n d'une forme différente que pour les cas précédents. Les cinq possibilités pour les branches (n, k, u) pour u éloigné de k et n sont alors réalisées: le shéma "c" est présent.

On ôte le cas de 4, qui sera traité à part. Les trois cas restant s'excluent mutuellement.
On regarde maintenant les cas des branches proches:

- Cas "d": on a $n-3 \not\perp n$ si et seulement si $n \equiv 0[3]$;

- Cas "e": on a $n+1 \not\perp n-2$ si et seulement si $n \equiv 2[3]$;

- On a $n-3 \perp n$ et $n+1 \perp n-2$ si et seulement si $n \equiv 1[3]$.

Ces trois derniers cas s'excluent.
En éliminant le cas particulier $n = 4$, on arrive donc à sept cas de figure différents, correspondant chacun à un pétiole $(n, n-2)$ possible:

$n = 2(2^\alpha + 1)$	et	$n \equiv 0[3]$
$n = 2(2^\alpha + 1)$	et	$n \equiv 1[3]$
$n = 2^\alpha$	et	$n \equiv 2[3]$
$n = 2^\alpha$	et	$n \equiv 1[3]$
$n \neq 2^\alpha$ et $n \neq 2(2^\alpha + 1)$	et	$n \equiv 0[3]$
$n \neq 2^\alpha$ et $n \neq 2(2^\alpha + 1)$	et	$n \equiv 2[3]$
$n \neq 2^\alpha$ et $n \neq 2(2^\alpha + 1)$	et	$n \equiv 1[3]$

Remarquons que les trois derniers cas englobent les cas où n est impair.

3.2.4.2 Pétiole où $k = n - 1$

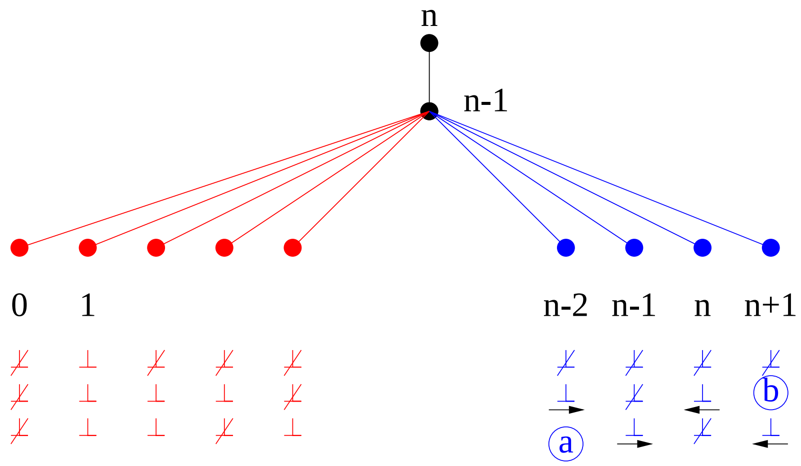


Figure 3.7: Pétiole $(n, n-1)$

Ici, n et $n-1$ sont nécessairement premiers entre eux, donc la discussion sur leurs support n'a pas lieu d'être: les cinq cas de branches éloignées sont présents.
En revanche, la figure 3.7 nous montre deux cas à étudier selon la parité de n:

- Si n est pair, alors pour le cas "a", on a $n-2 \not\perp n$ et le cas "b", $n+1 \perp n-1$;
- Si n est impair, alors pour le cas "a", on a $n-2 \perp n$ et le cas "b", $n+1 \not\perp n-1$.

On a donc dans ce cas, seulement deux pétioles possibles.

3.2.4.3 Pétiole où $k = n$

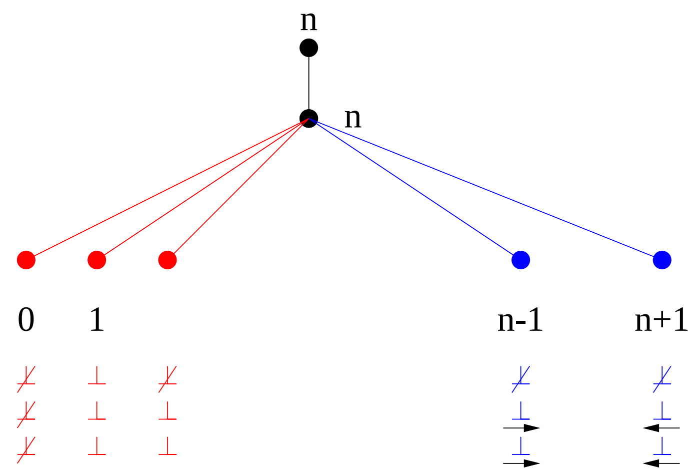


Figure 3.8: Pétiole (n, n)

Dans ce cas, il n'y a pas de discussion: les supports sont égaux donc il y a trois branches éloignées, et comme $k = n$, seulement deux branches proches, pour lesquelles on sait ce qui se passe (voir figure 3.8).

3.2.4.4 Pétiole où $k = n + 1$

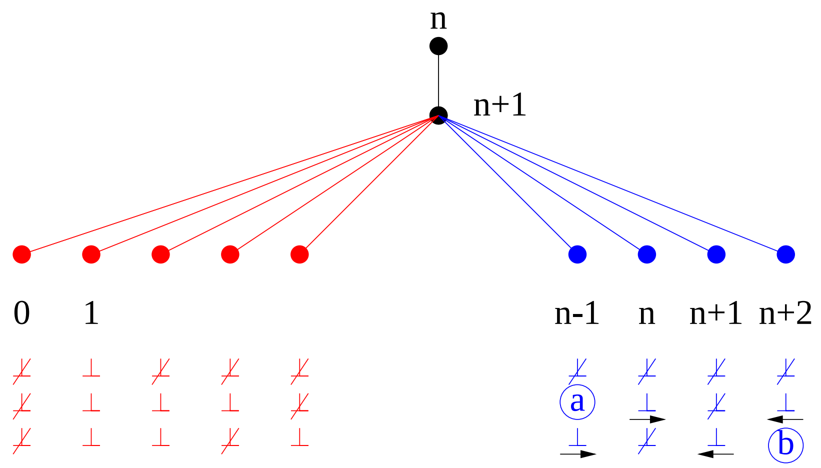


Figure 3.9: Pétiole $(n, n+1)$

Ce cas est semblable à celui de 3.2.4.2: n et $n + 1$ sont premiers entre eux, et la discussion porte sur la parité de n.

- Si n est pair, alors pour le cas "a", on a $n-1 \perp n+1$ et le cas "b", $n+2 \not\perp n$;
- Si n est impair, alors pour le cas "a", on a $n-1 \not\perp n+1$ et le cas "b", $n+2 \perp n$.

3.2.4.5 Pétiole où $k = n+2$

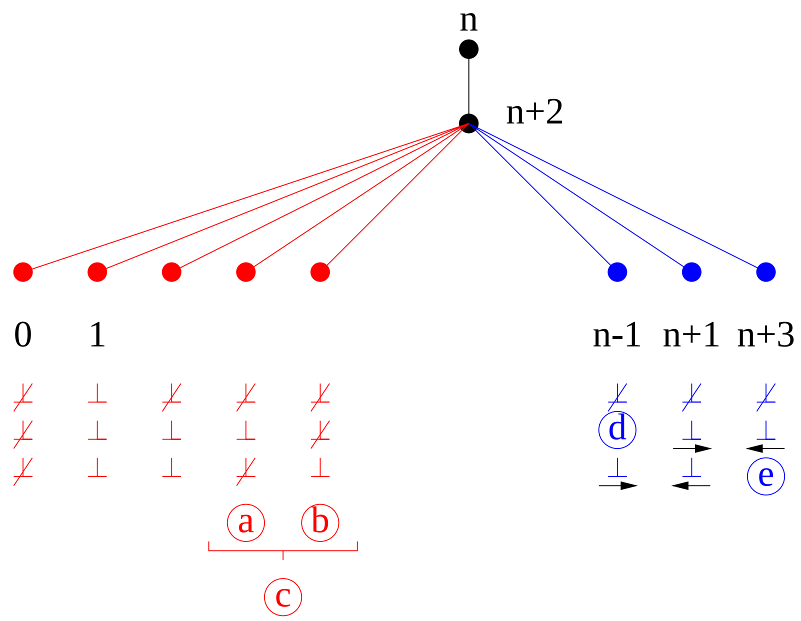


Figure 3.10: Pétiole $(n, n+2)$

- $n \perp n+2$ si n est impair et alors on a les 5 possibilités pour les branches (n,k,u) pour u éloigné de k et n réalisées
- $n \not\perp n+2$ si n est pair, et alors la discussion porte sur les supports

Plaçons nous dans le cas où n est pair:

- $Supp(n+2) \subsetneq Supp(n)$: Ce cas est possible si et seulement si n est de la forme $2(2^\alpha - 1)$, avec $\alpha \geq 0$ (car alors $Supp(n+2) = \{2\}$): le schéma "a" est absent;
- $Supp(n) \subsetneq Supp(n+2)$: Ce cas est possible si et seulement si n est de la forme 2^α, avec $\alpha > 0$: le schéma "b" est absent;
- $Supp(n) = Supp(n+2) = \{2\}$: Ce cas correspond à $n = 2$. Ce cas est traité à part;

- Le dernier cas, quand chaque support contient au moins un élément qui ne se trouve pas dans l'autre, intervient pour n d'une forme différente que pour les cas précédents. Les cinq possibilités pour les branches (n, k, u) pour u éloigné de k et n sont alors réalisées: le shéma "c" est présent.

Les trois cas restant s'excluent mutuellement si on ôte le cas $n = 2$.
On regarde maintenant les cas des branches proches:

- Cas "d": on a $n - 3 \not\perp n$ si et seulement si $n \equiv 0[3]$;

- Cas "e": on a $n + 1 \not\perp n - 2$ si et seulement si $n \equiv 2[3]$;

- On a $n - 3 \perp n$ et $n + 1 \perp n - 2$ si et seulement si $n \equiv 1[3]$.

Ces trois derniers cas s'excluent.
On a donc, comme dans le cas du pétiole $(n, n - 2)$, sept possibilités:

$n = 2^\alpha$	et	$n \equiv 2[3]$
$n = 2^\alpha$	et	$n \equiv 1[3]$
$n = 2(2^\alpha - 1)$	et	$n \equiv 0[3]$
$n = 2(2^\alpha - 1)$	et	$n \equiv 2[3]$
$n \neq 2(2^\alpha - 1)$ et $n \neq 2^\alpha$	et	$n \equiv 0[3]$
$n \neq 2(2^\alpha - 1)$ et $n \neq 2^\alpha$	et	$n \equiv 2[3]$
$n \neq 2(2^\alpha - 1)$ et $n \neq 2^\alpha$	et	$n \equiv 1[3]$

3.2.4.6 Conclusion provisoire

Pour savoir combien de possibilités on obtient pour les branches proches, il faut tenir compte des faits suivants:

- Les cas $k = n - 2$ et $k = n + 2$ font apparaître des cas qui se recoupent, et les cas $n = 2(2^\alpha + 1)$ et $n = 2(2^\alpha - 1)$ sont disjoints si on exclut le cas particulier $n = 6$. On écarte donc $n = 6$ de la discussion générale, on traitera ce cas à part;

- Les cas $k = n - 1$ et $k = n + 1$ font apparaître les mêmes cas, selon la parité de n;

- Dans chacune des discussions de $k = n - 2$ et $k = n + 2$, les 4 premiers cas entraînent que n est pair.

On obtient donc les 12 cas suivants:

$n = 2(2^\alpha + 1)$	et	$n \equiv 0[3]$	(pair)
$n = 2(2^\alpha + 1)$	et	$n \equiv 1[3]$	(pair)
$n = 2(2^\alpha - 1)$	et	$n \equiv 0[3]$	(pair)
$n = 2(2^\alpha - 1)$	et	$n \equiv 2[3]$	(pair)
$n = 2^\alpha$	et	$n \equiv 1[3]$	(pair)
$n = 2^\alpha$	et	$n \equiv 2[3]$	(pair)
$n \neq 2(2^\alpha - 1)$ et $n \neq 2(2^\alpha + 1)$ et $n \neq 2^\alpha$	et	$n \equiv 0[6]$	(pair)
$n \neq 2(2^\alpha - 1)$ et $n \neq 2(2^\alpha + 1)$ et $n \neq 2^\alpha$	et	$n \equiv 4[6]$	(pair)
$n \neq 2(2^\alpha - 1)$ et $n \neq 2(2^\alpha + 1)$ et $n \neq 2^\alpha$	et	$n \equiv 2[6]$	(pair)
$n \neq 2(2^\alpha - 1)$ et $n \neq 2(2^\alpha + 1)$	et	$n \equiv 3[6]$	(impair)
$n \neq 2(2^\alpha - 1)$ et $n \neq 2(2^\alpha + 1)$	et	$n \equiv 1[6]$	(impair)
$n \neq 2(2^\alpha - 1)$ et $n \neq 2(2^\alpha + 1)$	et	$n \equiv 5[6]$	(impair)

3.2.5 Cas pathologiques $k = 0$, $k = 1$, $k = 2$

Il nous reste à étudier les cas de branches (n, k) pour les k qu'on avait rejetés pour faciliter la discussion jusqu'à présent.

3.2.5.1 Cas $k = 0$

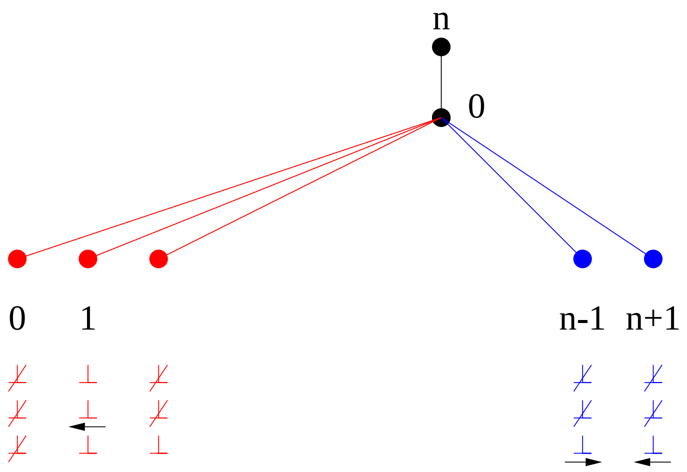


Figure 3.11: Pétiole $(n, 0)$

Ce cas ne pose pas vraiment de problème, on est dans un cas où $Supp(n) \subsetneq Supp(k)$, et les u proches de n et 0 sont entièrement déterminés (1 est premier avec n, $n-1$ et $n+1$ ne sont pas premiers avec 0). Le pétiole est dessiné en figure 3.11.

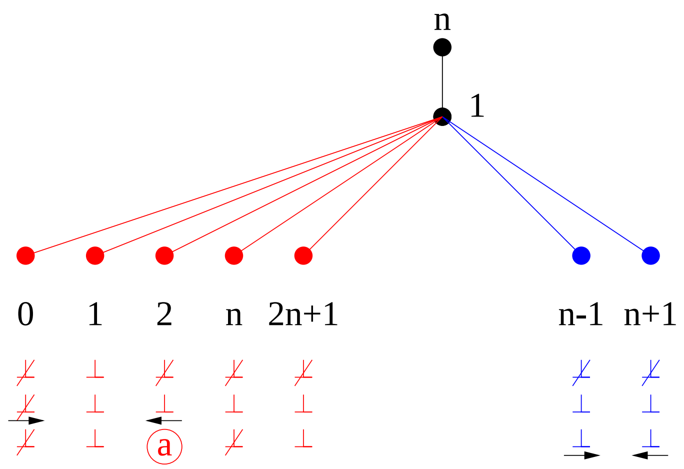


Figure 3.12: Pétiole $(n, 1)$

3.2.5.2 Cas $k = 1$

Ce cas n'apporte pas vraiment non plus de nouveauté: on a deux possibilités selon la parité de n pour le pétiole $(n, 1, 2)$, parité que l'on prend déjà en compte dans la discussion sur les branches où k est proche de n. Le pétiole est dessiné en figure 3.12.

3.2.5.3 Cas $k = 2$

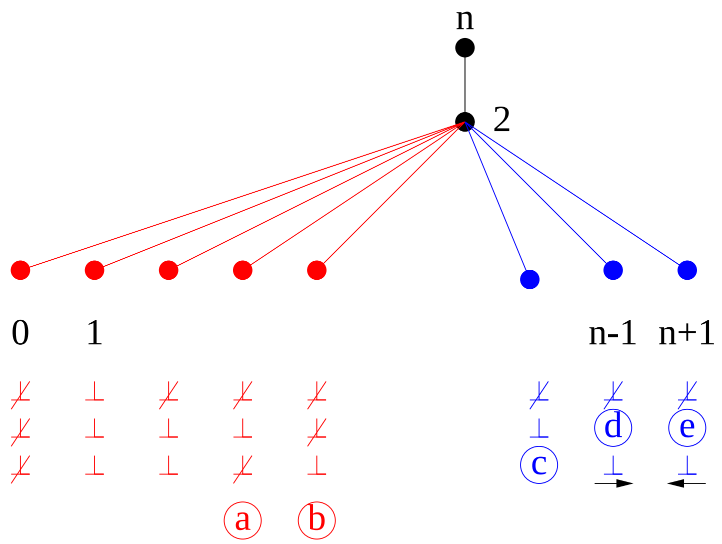


Figure 3.13: Pétiole $(n, 2)$

Ce pétiole n'apporte rien de nouveau non plus:

- Cas "a": Cette configuration apparaît si et seulement si n n'est pas une puissance de 2;
- Cas "b": Cette configuration apparaît si et seulement si n est impair (elle est alors réalisée par $u = 2$);
- Cas "c": n est non premier avec 3 si et seulement si $n \equiv 0[3]$;
- Cas "d": $n-1$ est non premier avec 2 si et seulement si n est impair;
- Cas "e": $n+1$ est non premier avec 2 si et seulement si n est impair.

Ces cas apparaissent dans les discussions déjà faites sur les branches pour k proche de n. On n'a donc rien à rajouter aux 12 cas déjà mis en évidence.

3.2.6 Conclusion sur le nombre de destinées possibles

On en est à 12 cas, si on ne considère que les k proches de n et dans $\{0, 1, 2\}$, et on avait une majoration de 80 configurations pour les k éloignés de n. Cela nous donne une majoration de $2^{80} \times 12$ destinées possibles pour le cas où $n \notin \{0, 1, 2, 3, 4, 6\}$, ce qui est énorme. Mais une étude plus détaillée des 80 cas de branches éloignées selon la forme de n permet de réduire ce nombre à au plus 56 destinées possibles, ce qui est beaucoup plus raisonnable pour éventuellement faire un parcours exhaustif par ordinateur (Voir Annexe B).

3.3 Résultats sur les ensembles X et Y

3.3.1 Présentation

Dans la discussion générale sur les 3-destinées associées au langage $\{S, \perp\}$, on est amené à s'intéresser plus particulièrement à deux ensembles d'entiers, qui correspondent à deux des 80 possibilités pour les u proches de n ou k, avec k et n éloignés. On les définit ainsi:

- $X = \{n \in \mathbb{N} / \exists k \in \mathbb{N}\, Supp(k) \subsetneq Supp(n),\ k-1 \perp N \text{ et } k+1 \not\perp n\}$
- $Y = \{n \in \mathbb{N} / \exists k \in \mathbb{N}\, Supp(k) \subsetneq Supp(n),\ k-1 \not\perp N \text{ et } k+1 \not\perp n\}$.

Le travail autour de ces ensembles consiste à trouver pour chaque forme de n mise en évidence dans le reste de la discussion: un élément dans X, un élément en dehors de Y, et un élément dans $Y \backslash X$, ou bien montrer que l'un de ces cas est impossible. Pour ce faire, on peut s'appuyer sur les caractérisations suivantes, fort utiles:

Proposition 7 (Caractérisation de Y)
Y est l'ensemble des nombres entiers naturels admettant deux facteurs premiers distincts p et q tels que $Ord(p, q)$ soit pair.

Proposition 8 (Caractérisation de $X \cap 2\mathbb{N}+1$)
Un nombre entier impair n appartient à X si et seulement s'il existe un de ses facteurs premiers p, et deux nombres entiers b et c, de supports disjoints et inclus dans $Supp(n)\backslash\{p\}$, tels que:

$$V_2(Ord(b,p)) > V_2(Ord(c,p)).$$

Ces résultats permettent de passer dans un autre univers, celui des ordres modulo un entier premier, ce qui permet d'utiliser la loi de réciprocité quadratique, les notations de Legendre et le petit théorème de Fermat, entre autres.

3.3.2 Quelques résultats

Il serait un peu long de reprendre tous les résultats obtenus par Marcel Guillaume, Ji Lei Yin et Denis Richard sur l'appartenance ou non de certains entiers à ces ensembles, nous n'en présenterons donc qu'une partie.

On a ainsi, dans Y:

- Les entiers de la forme $2^\alpha + 1$ pour $\alpha > 3$, impair;
- Les entiers de la forme $2^\alpha - 1$ pour $\alpha > 2$ pair ou $\alpha > 5$ non primaire;
- Les entiers de la forme $2^{q^\alpha} - 1$ pour q premier tel que $q \equiv -1[4]$ et $\alpha > 1$;
- Les entiers de la forme $2^{2^\alpha i}$ pour i impair et $\alpha > 0$ tel que F_α soit un premier de Fermat;
- ...

Et, de même, dans X:

- Les entiers de la forme $2^\alpha + 1$ pour $\alpha > 3$, impair tel que $|Supp(\alpha)| \geq 2$ ou $\alpha \equiv 0[3]$;
- Les entiers de la forme $2^\alpha - 1$ pour α tel que $|Supp(\alpha)| \geq 3$;
- Les entiers de la forme $2^{q^\alpha} - 1$ pour q premier impair tel que $q \equiv -1[3]$ et $\alpha > 1$;
- ...

Chapter 4

Perspectives

4.1 Un algorithme de vérification

Un algorithme de vérification de formules a été implémenté dans le langage Caml. Il 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 la 2-transversale de $\mathbb{N}$ muni du successeur et de la coprimarité, la seule actuellement disponible, indique s'il s'agit d'un théorème ou non. Le code de ce programme figure en annexe C.

4.1.1 Choix des structures de données

Le langage Caml permet de définir des types de structures de données: on a défini un type `formule` et un type `destinee` pour manipuler les objets correspondant. Ces types ont été définis de manière récursive grâce aux constructeurs:

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
;;

(*
   Le premier entier correspond au niveau courant,
   le deuxieme entier correspond à la valeur du
     noeud de la destinee,
   la liste de booleens figure les relations des
     noeuds entre eux,
   la liste de destinees est la liste des fils du
     noeud courant
*)
```

La transversale de la théorie de ℕ avec successeur et coprimarité à deux quantificateurs est alors exprimée sous forme d'une liste de destinées:

```
(*
2-transversale de la théorie du successeur et de la coprimarité,
presentée sous forme d'une liste de destinées
*)
let dest2 =
[ Node(1 , 0 , [false] ,
   [Node(2 , 0 , [false ; false ; false ; false ] , [Empty]) ;
    Node(2 , 1 , [true ; true ; false ; true ] , [Empty])
  ]) ;
  Node(1 , 1 , [true] ,
   [Node(2 , 0 , [false ; true ; true ; false  ] , [Empty]) ;
    Node(2 , 1 , [true ; true ; false ; false  ] , [Empty]) ;
    Node(2 , 2 , [false ; true ; false ; true  ] , [Empty]) ;
    Node(2 , 3 , [false ; true ; false ; false ] , [Empty])
  ]) ;
  Node(1 , 2 , [false] ,
   [Node(2 , 0 , [false ; false ; false ; false ] , [Empty]) ;
    Node(2 , 1 , [true ; true ; true  ; false ]   , [Empty]) ;
    Node(2 , 3 , [false ; true ; false ; true ]   , [Empty]) ;
    Node(2 , 5 , [false ; true ; false ; false ]  , [Empty])
  ] ) ;
```

```
    Node(1 , 3 , [false],
     [Node(2 , 0 , [false ; false ; false ; false ] , [Empty]) ;
      Node(2 , 1 , [true ; true ; false  ; false ]  , [Empty]) ;
      Node(2 , 2 , [false ; true ; true ; false ]   , [Empty]) ;
      Node(2 , 5 , [false ; true ; false ; false ]  , [Empty])
    ] )
  ]
  ;;
```

4.1.2 Outils autour de ces structures

Les outils utilisés pour manipuler les structures `formule` et `destinee` sont les suivants:

- Une fonction qui vérifie qu'une formule est close;
- Une fonction qui calcule le rang de quantification d'une formule;
- Une fonction qui calcule la hauteur d'une destinée.

4.1.3 Le corps de la fonction d'évaluation

La fonction d'évaluation prend en entrée une destinée et une formule. Elle effectue un filtrage de motif (*pattern matching*) sur la structure de la formule:

- Si la formule est `Vrai`, elle renvoie la valeur vrai (`true`);
- Si la formule est `Faux`, elle renvoie la valeur faux (`false`);
- Si la formule est de la forme $\neg F$, elle renvoie le contraire de la valeur prise par la fonction d'évaluation en F (appel récursif);
- Si la formule est de la forme $F \wedge G$, elle renvoie le "et" logique des valeurs prise par la fonction en F et G;
- ...
- Si la formule est de la forme $\forall x F(x)$, on descend d'un niveau dans la destinée, et la valeur envoyée est alors le "et" logique de toutes les valeurs prises par la fonction en F et en les sous-destinées;
- Si la formule est de la forme $\exists x F(x)$, on descend d'un niveau dans la destinée, et la valeur envoyée est alors le "ou" logique de toutes les valeurs prises par la fonction en F et en les sous-destinées;
- Si la formule est un prédicat: selon la hauteur des variables, on regarde dans la destinée au niveau de la liste des propriétés vérifiées par le noeud le plus bas, et la valeur renvoyée est alors celle correspondant

au prédicat en question. Par exemple, pour $x \perp y$, si y dépend de x, on va regarder dans la liste des propriétés vérifiées par le noeud de niveau 2, à l'endroit convenu pour le prédicat "fils premier avec père".

4.1.4 Mode d'emploi du programme

La syntaxe des formules prises en entrée est la suivante:

- Les quantificateurs sont de la forme: `Forall` et `Exists`;
- Les connecteurs se présentent sous la forme: `Non`, `Ou`, `Et`, `->`, `<->`;
- Les noms de variables sont des lettres minuscules entre 'a' et 'z';
- Les prédicats sont `Coprem(x,y)` pour signifier "x et y sont premiers entre eux", et `Succ(x,y)` pour signifier "x est le successeur de y";
- Le parenthésage est nécessaire pour lever toute ambiguité;
- Dans la version actuelle du programme, il est demandé que l'imbrication des quantificateurs soit optimale, dans le sens où un prédicat doit dépendre de la dernière variable quantifiée.

Quelques exemples de formules correctes syntaxiquement:

`Forall x (Exists y ((Coprem(x,y))Et (Succ(x,y))))`	$\forall x \exists y ((x \perp y) \wedge (S(x,y)))$
`Forall x ((Coprem(x,x))Ou(Exists y (Succ(x,y))))`	$\forall x ((x \perp x) \vee (\exists y\, S(x,y)))$
`Forall x (Exists y (Coprem(x,y)))`	$\forall x \exists y (x \perp y)$
`Exists x ((Coprem(x,x))Et(Forall y (Coprem(x,y))))`	$\exists x (x \perp x) \wedge (\forall y (x \perp y))$
`Forall x (Exists y (Succ(x,y)))`	$\forall x \exists y S(x,y)$
`Exists x (Exists y ((Coprem(y,y))Et((Coprem(x,y))Et(Succ (x,y)))))`	$\exists x \exists y (y \perp y) \wedge (x \perp y) \wedge S(x,y)$

L'interface interactive propose plusieurs commandes:

- `quit`: pour quitter l'interface
- `eval`: pour vérifier une formule. Cette commande peut s'utiliser de deux manières:
  - `eval` *formule*: vérifie la formule *formule* directement tapée en ligne de commande
  - `eval file` *nom_de_fichier*: vérifie la formule enregistrée dans le fichier *nom_de_fichier*
- `help`: fournit la syntaxe des commandes. On peut lui mettre en argument le nom d'une commande particulière

4.1.5 Exemple de session

Au lancement, l'aide est affichée automatiquement:

```
   **************************************************
   *                                                *
   *  Bonjour, vous êtes dans l'interface interactive  *
   *  de l'évaluateur de formules du successeur et de  *
   *  la coprimarité à deux quantificateurs.          *
   *                                                *
   **************************************************
eval file nom_de_fichier: lance l'exécution de l'évaluation
      de la formule apparaissant dans ce fichier
eval formule: évalue directement la formule tapée
      après la commande
quit: quitte la magnifique interface utilisateur
      et revient au top_level de OCaml
Evaluation>
```

On peut dès lors taper une formule:

```
Evaluation>eval Forall x (Exists y ((Coprem(x,y))Et (Succ(x,y))))
La formule n'est pas un théorème
Evaluation>eval Forall x ((Coprem(x,x))Ou(Exists y (Succ(x,y))))
La formule n'est pas un théorème
Evaluation>eval Forall x (Exists y (Coprem(x,y)))
La formule est un théorème
Evaluation>eval Exists x ((Coprem(x,x))Et(Forall y (Coprem(x,y)))
lexing: empty token
Evaluation>eval Exists x ((Coprem(x,x))Et(Forall y (Coprem(x,y))))
La formule est un théorème
Evaluation>eval Forall x (Exists y (Succ(x,y)))
La formule n'est pas un théorème
```

Lorsque la formule n'est pas syntaxiquement correcte, l'erreur porte sur le lexing.

```
Evaluation> eval Forall x (Exists y (Forall z ((Coprem(x,z))
Ou (Coprem(y,z)))))
Il y a trop de quantificateurs
```

Le programme vérifie que le nombre de quantificateurs est inférieur à la hauteur de la destinée. Exemples d'autres erreurs possibles:

```
Evaluation> eval kjxlhgdgbd
La formule n'a pu être evaluée
Evaluation> pckjdkv
```

```
Commande non valide
Evaluation> eval Forall x (Forall y ((Coprem(x,x)) Ou (Succ(x,y))))
Formule mal construite : imbriquez mieux les quantificateurs
```

4.1.6 Améliorations envisagées

Des versions ultérieures pourront proposer des améliorations:

- Le programme pourra s'étendre à d'autres langages relationnels finis;
- Il pourra proposer un choix de la transversale selon le niveau de quantification;
- Une interface graphique pourra être construite;
- Lorsque la formule n'est pas un théorème, un contre-exemple explicite pourra être fourni.

4.2 Un algorithme pour déduire l'axiomatisation de la transversale

De la même façon qu'on a pu implémenter une fonction de vérification de formules étant donnée une transversale destinale, on peut envisager d'implémenter une fonction qui prendrait en entrée une destinée, et fournirait la théorie qu'elle représente, à savoir l'ensemble exhaustif fini des formules vérifiées dans $\mathbb{N}$ non équivalentes deux à deux. On a vu dans le chapitre de présentation des destinées, que chaque pétiole donne une formule existentielle, de manière très simple. Pour réaliser l'axiomatisation de la théorie à partir de la destinée, il faudrait introduire les quantificateurs universels. Cela pourrait se faire, par exemple, en listant toutes les formules existentielles vérifiées par la théorie, puis en remplaçant successivement les quantificateurs existentiels par des quantificateurs universels et en testant ces formules transformées. Pour deux quantificateurs, une telle démarche semble possible (cela fait 3 formules à vérifier par formule existentielle trouvée), mais si l'on veut généraliser cet algorithme, il faudra trouver des optimisations, car le nombre de formules à tester par formule existentielle trouvée est une fonction exponentielle du niveau de quantification ...

4.3 Un codage dans les entiers sous forme binaire ...

La liste des propriétés vérifiées par un noeud d'une destinée est apparue comme une liste de booléens dans l'implémentation proposée du vérificateur de formule. Mais il est également possible de coder cette liste de propriétés comme un nombre exprimé en écriture binaire: Le premier bit étant égal à 0 ou 1 selon que la racine est première avec elle-même, le deuxième exprime la propriété "le fils est premier avec lui-même", le troisième celle de la succession dans un sens: "le père est successeur du fils", le quatrième celle de la succession dans l'autre sens: "le fils est successeur du père", les bits suivants codant de manière analogue les relations du petit-fils avec respectivement lui-même, son père et son grand-père.

Prenons un exemple, dans la 2-destinée du successeur et de la coprimarité: La branche (1,2) serait alors exprimée par l'entier 1001 car 1 est premier avec lui-même, 2 n'est pas premier avec lui-même, 1 n'est pas le successeur de 2 et 2 est le successeur de 1.

Chaque branche est alors exprimée par un nombre binaire. On peut alors envisager deux perspectives:

- On peut s'intéresser aux propriétés arithmétiques de ces nombres par rapport à leur place dans la destinée, ou aux propriétés du langage;
- On peut éventuellement se servir de ce codage pour construire une destinée, en éliminant les branches impossibles par des conditions sur la forme de ce nombre.

Ces perspectives pourront faire l'objet de recherches ultérieures.

4.4 Lien avec les bases de données

Le nouvel outil que représente l'étude des p-destinées dans les langages relationnels eut également ouvrir des perspectives concernant la manipulation des bases de données relationnelles. On peut imaginer par exemple construire une destinée à partir des relations vérifiées par les éléments d'une base de données. Les consultations pourraient alors se faire par parcours de l'arbre, et les modifications, ajouts, retraits, par manipulation de la structure arborescente.

Appendix A

Résultats autour de la conjecture Woods-Erdös

A.1 Démonstration du corollaire 1:

Reprenons chaque point du corollaire comme un lemme indépendant.

Lemme 4
$SUPP(x^\alpha - 1) = SUPP(x^\beta - 1)$ si et seulement si $\alpha = \beta$ ou $x = 2^u - 1$ avec $u > 1$ et $\{\alpha, \beta\} \subseteq \{1, 2\}$.

Démonstration:
$\Rightarrow$: On suppose que $SUPP(x^\alpha - 1) = SUPP(x^\beta - 1)$. Soit on a $\alpha = \beta$, soit ce n'est pas le cas et alors vu la symétrie du problème on peut considérer $\alpha < \beta$. On est alors dans un cas d'exception du théorème 3: il n'y a pas de diviseur primitif pour $x^\beta - 1$. Donc on a l'une des trois possibilités suivantes:

- $\beta = 1$, mais alors $\alpha = 0$ et $SUPP(x^\alpha - 1) \neq SUPP(x^\beta - 1)$ car $SUPP(0) = \mathcal{P}$. C'est impossible.

- $\beta = 2$, alors $\alpha = 1$ et $x + 1 = 2^u$ avec $u > 0$. Si $u = 1$, on a $x = 1$, ce qui est contraire à l'hypothèse. Donc $u > 1$.

- $\beta = 6$, $x = 2$, $y = 1$. Or: $x^\beta - 1 = 2^6 - 1 = 63 = 3^2 \times 7$, donc $SUPP(x^\beta - 1) = \{3, 7\}$, tandis que:
  Si $\alpha = 5$: $x^\alpha - 1 = 31$ donc $SUPP(x^\alpha - 1) = \{31\} \neq \{3, 7\}$
  Si $\alpha = 4$: $x^\alpha - 1 = 15$ donc $SUPP(x^\alpha - 1) = \{3, 5\} \neq \{3, 7\}$
  Si $\alpha = 3$: $x^\alpha - 1 = 7$ donc $SUPP(x^\alpha - 1) = \{7\} \neq \{3, 7\}$
  Si $\alpha = 2$: $x^\alpha - 1 = 3$ donc $SUPP(x^\alpha - 1) = \{3\} \neq \{3, 7\}$
  Si $\alpha = 1$: $x^\alpha - 1 = 1$ donc $SUPP(x^\alpha - 1) = \emptyset \neq \{3, 7\}$
  Aucune de ces possibilités ne convient, donc ce dernier cas est exclu.

$\Leftarrow$: Si $\alpha = \beta$ on a $SUPP(x^\alpha - 1) = SUPP(x^\beta - 1)$ de manière évidente. Sinon, supposons $\beta = 2$ et $\alpha = 1$ avec $x = 2^u - 1$, on est dans un cas d'exception du théorème 3 donc tous les diviseurs premiers de $x^\beta - 1$ divisent $x^\alpha - 1$. Réciproquement, on a $x^2 - 1 = (x-1)(x+1)$ donc $SUPP(x-1) \subseteq SUPP(x^2-1)$.

Lemme 5
$SUPP(x^\beta - 1) \subseteq SUPP(x^\alpha - 1)$ si et seulement si $\beta \mid \alpha$ ou $x = 2^u - 1$ avec $u > 1$ et $\beta = 2$.

Démonstration:
$\Rightarrow$: Supposons $SUPP(x^\beta - 1) \subseteq SUPP(x^\alpha - 1)$. Si on pose $d = \gcd(\alpha, \beta)$, on a toujours $\gcd(x^\alpha - 1, x^\beta - 1) = x^d - 1$ donc $SUPP(x^d - 1) \subseteq SUPP(x^\alpha - 1)$ et $SUPP(x^d - 1) \subseteq SUPP(x^\beta - 1)$. Ici, comme $SUPP(x^\beta - 1) \subseteq SUPP(x^\alpha - 1)$, on a donc $SUPP(\gcd(x^\alpha - 1, x^\beta - 1)) = SUPP(x^\beta - 1)$ donc $SUPP(x^d - 1) = SUPP(x^\beta - 1)$. On se réfère au lemme 4, qui nous dit qu'alors soit $\beta = d$, ce qui signifie $\beta \mid \alpha$, soit $d = 1$, $\beta = 2$ et $x = 2^u - 1$ avec $u > 1$.
$\Leftarrow$: Si $\beta \mid \alpha$, on a $\beta = d$ donc $SUPP(x^d - 1) = SUPP(x^\beta - 1)$, et comme $SUPP(x^d - 1) \subseteq SUPP(x^\alpha - 1)$, on a $SUPP(x^\beta - 1) \subseteq SUPP(x^\alpha - 1)$. Si $\beta = 2$, $x = 2^u - 1$ avec $u > 1$: soit $d = \beta$, cas qu'on a déjà vu, soit $d = 1$, et alors: $x^d - 1 = 2^u - 2 = 2(2^{u-1} - 1)$, et $x^\beta - 1 = (2^u - 2) \times 2^u = 2^{u+1}(2^{u-1} - 1)$ et on a clairement $SUPP(x^d - 1) = SUPP(x^\beta - 1) = \{2\} \cup SUPP(2^{u-1} - 1)$. d'où $SUPP(x^\beta - 1) \subseteq SUPP(x^\alpha - 1)$.

Lemme 6
p est un diviseur primitif de $x^\alpha - 1$ si et seulement si $p \mid x^\alpha - 1$ et ou bien $\alpha = 1$, ou bien pour tout $\beta \neq \alpha$ tel que $p \mid x^\beta - 1$ on a $SUPP(x^\alpha - 1) \subsetneq SUPP(x^\beta - 1)$.

Démonstration:
$\Leftarrow$: Si p est un diviseur primitif de $x^\alpha - 1$, alors $p \mid x^\alpha - 1$ par définition, et on a soit $\alpha = 1$, soit $\alpha \neq 1$: si $p \mid x^\beta - 1$, on a nécessairement $\beta \geq \alpha$ puisque p est primitif. On suppose $\beta \neq \alpha$. On effectue la division euclidienne de β par α: $\beta = k\alpha + r$, avec $0 \leq r < \alpha$. On a alors: $x^\alpha \equiv 1$ $[p]$ et $x^\beta \equiv 1$ $[p]$ entraînent $x^r \equiv 1$ $[p]$. Comme p est primitif, α est l'ordre de x modulo p, donc on a $r = 0$, d'où $\alpha \mid \beta$. D'après le lemme 5, on a alors $SUPP(x^\alpha - 1) \subseteq SUPP(x^\beta - 1)$. Si on avait égalité, comme le cas $x = 2^u$ et $\alpha = 2$ est exclu du fait de l'existence de p primitif, on aurait d'après le lemme 5 $\alpha = \beta$, ce qui est contraire à l'hypothèse. Donc $SUPP(x^\alpha - 1) \subsetneq SUPP(x^\beta - 1)$
$\Rightarrow$: On suppose que $p \mid x^\alpha - 1$. Si $\alpha = 1$, il n'y a pas de β tel que $0 < \beta < \alpha$ et comme p divise tous les éléments de l'ensemble vide: p est primitif.
Sinon: on suppose que pour tous les β distincts de α tels que $p \mid x^\beta - 1$, on a $SUPP(x^\alpha - 1) \subsetneq SUPP(x^\beta - 1)$. On va montrer que nécessairement $\beta > \alpha$, ce qui prouvera que p est primitif pour $x^\alpha - 1$. Comme $SUPP(x^\alpha - 1) \subsetneq SUPP(x^\beta - 1)$, on a d'après le lemme 5:

- soit $\alpha = 2$, $x = 2^u - 1$ avec $u > 1$, auquel cas $\beta \neq 1$ sinon on serait dans le cas du lemme 4, donc on a $\beta > \alpha$;
- soit $\alpha \mid \beta$ donc $\beta > \alpha$.

Dans ces deux cas on a $\beta > \alpha$, ce qu'on voulait.

Lemme 7
$SUPP(x^\alpha + 1) = SUPP(x^\beta + 1)$ si et seulement si $\alpha = \beta$ ou $x = 2$ et $\{\alpha, \beta\} \subseteq \{1, 3\}$.

Démonstration:
$\Rightarrow$: Si $SUPP(x^\alpha + 1) = SUPP(x^\beta + 1)$, soit on a $\alpha = \beta$, soit on peut supposer $\beta > \alpha$ et alors on est dans un cas d'exception du théorème 4, donc $\beta = 3$, $x = 2$. Alors $x^\beta + 1 = 2^3 + 1 = 9 = 3^2$, d'où $SUPP(x^\beta + 1) = \{3\}$.
Et:
Si $\alpha = 2$: $x^\alpha + 1 = 5$ donc $SUPP(x^\alpha + 1) = \{5\} \neq \{3\}$
Si $\alpha = 1$: $x^\alpha + 1 = 3$ donc $SUPP(x^\alpha + 1) = \{3\}$
Donc $\alpha = 1$.
$\Leftarrow$: Si $\alpha = \beta$ alors $SUPP(x^\alpha + 1) = SUPP(x^\beta + 1)$, et si $\alpha = 1$, $\beta = 3$ et $x = 2$, on a directement $SUPP(x^\alpha + 1) = SUPP(x^\beta + 1) = \{3\}$.

A.2 Synonymie de $< \mathbb{N}, +, \times, = >$ et $< \mathbb{N}, S, | >$

Démontrons le sens difficile, le sens $DEF(S, |) \subseteq DEF(+, \times, =)$ étant très simple.

- L'égalité est définissable par: $x = y$ si et seulement si $(x \mid y) \wedge (y \mid x)$;
- Si on réussit à montrer que la multiplication est définissable, on aura également l'addition, grâce à la remarque suivante:

  $z = 0$ ou $z = x + y \Leftrightarrow (x \times z + 1) \times (y \times z + 1) = (z^2 \times (x \times y + 1)) + 1$

- Pour avoir la multiplication, on passe par le lemme suivant:

Lemme 8
Il y a équivalence entre les propositions suivantes:
i). $z = x \times y$
ii). Pour tout nombre premier p ne divisant ni x ni y, il existe des entiers x' et y' premiers entre eux et avec x, y, et z, tels que:

$$\begin{cases} xx' \equiv -1 \, [p] \\ yy' \equiv -1 \, [p] \\ zx'y' \equiv 1 \, [p] \end{cases}$$

Démonstration:

i). $\Leftarrow$ ii).: Soit p un nombre premier n'appartenant pas à l'ensemble $SUPP(x) \cup SUPP(y)$. Il existe d'après l'hypothèse des entiers x' et y' premiers entre eux et avec x, y, et z, tels que:

$$\begin{cases} xx' \equiv -1 \, [p] \\ yy' \equiv -1 \, [p] \\ zx'y' \equiv 1 \, [p] \end{cases}$$

de ce système on déduit: $xx' + zx'y' \equiv 0 \, [p] \Leftrightarrow x'(x + zy') \equiv 0 \, [p]$. Or si $p \mid x'$, on a $xx' \equiv 0 \, [p]$, ce qui contredit l'hypothèse, donc $p \not| \, x'$. D'où:
$p \mid (x + zy') \Rightarrow x + zy' \equiv 0 \, [p] \Rightarrow x \equiv -zy' \, [p] \Rightarrow xy \equiv -zy'y \, [p]$.
Comme $yy' \equiv -1 \, [p]$, on a $xy \equiv z \, [p]$. C'est vrai pour une infinité de nombres entiers donc on a bien $xy = z$.

i). $\Rightarrow$ ii).: Soit p un nombre premier n'appartenant pas à l'ensemble $SUPP(x) \cup SUPP(y)$. $p \geq 2$. On a :
$x \equiv u \, [p]$ et $\forall \alpha \; x \equiv u + \alpha p \, [p]$
$y \equiv v \, [p]$ et $\forall \beta \; y \equiv v + \beta p \, [p]$
D'après le théorème de Dirichlet, $p \perp u$ et $p \perp v$ entraînent l'existence de nombres a et b tels que $u + ap$ et $v + bp$ soient premiers.
Remarque: ces nombres premiers sont forcément différents de p, sinon $u = 0$ ou $v = 0$, ce qui signifierait $p \mid x$ ou $p \mid y$, donc contredirait l'hypothèse sur p. Par conséquent $p \perp u + ap$ et $p \perp v + bp$.
On considère alors les nombres de la forme:
$x_k = kp - (u + ap)^{p-2}$ et $x_{k'} = k'p - (v + bp)^{p-2}$.
D'après le théorème de Dirichlet, il existe des nombres k et k' tels que x_k et $x_{k'}$ soient positifs, différents et premiers. On pose alors $x' = x_k$ et $y' = x_{k'}$.
x' et y' étant premiers et différents, ils sont premiers entre eux et avec x et y. En outre:
$xx' \equiv xkp - x(u + ap)^{p-2} \equiv -x(u + ap)^{p-2} \equiv -xu^{p-2} \, [p]$
or $x \equiv u \, [p]$ donc $xu^{p-2} \equiv u^{p-1} \, [p]$. D'après le petit théorème de Fermat: $u^{p-1} \equiv 1 \, [p]$, d'où finalement: $xx' \equiv -1 \, [p]$.
Par un raisonnement analogue on obtient $yy' \equiv -1 \, [p]$.
Enfin, comme $z = xy$, on a $zx'y' \equiv xx'yy' \equiv (-1) \times (-1) \equiv 1 \, [p]$, ce qui termine la preuve.

Ramener le problème à la multiplication de deux nombres premiers entre eux permet d'utiliser la notion de ppcm (le produit de deux nombres premiers entre eux est égal à leur ppcm) qui est définissable dans $< S, | >$ par la formule z=ppcm(x,y): $(x \mid z) \wedge (y \mid z) \wedge \forall k((x \mid k) \wedge (y \mid k) \rightarrow (z \mid k))$.
On a également besoin de définir ce qu'est un nombre premier: Premier(p) est $\forall u((u \mid p) \rightarrow ((u = p) \vee Un(u)))$.
Il ne reste plus alors qu'à traduire calmement la partie ii). du lemme pour

obtenir la définition de $\times$:
$z = x \times y$ est $(\forall p(Premier(p) \wedge \neg(p \mid x) \wedge \neg(p \mid y)) \rightarrow \exists x' \exists y'(((x' \perp x) \wedge (x' \perp y) \wedge (x' \perp z) \wedge (x' \perp y') \wedge (y' \perp x) \wedge (y' \perp y) \wedge (y' \perp z)) \bigwedge((p \mid S(ppcm(x, x'))) \wedge (p \mid S(ppcm(y, y'))) \wedge (p \mid Pred(ppcm(z, ppcm(x', y'))))))) \vee (Zero(z) \leftrightarrow (Zero(x) \vee Zero(y)))$;

Appendix B

Etude des branches (n, k) pour k éloignés de n

B.1 Rappel des 80 cas possibles

Pour (n, k) avec k éloigné de n, on a cinq possibilités d'inclusion des supports:

- $Supp(k) = Supp(n)$
- $Supp(n) \subsetneq Supp(k)$
- $Supp(n) \cap Supp(k) = \emptyset$
- $Supp(n) \backslash Supp(k) \neq \emptyset$, $Supp(k) \backslash Supp(n) \neq \emptyset$ et $n \not\perp k$
- $Supp(k) \subsetneq Supp(n)$

Pour chacune de ces possibilités, on a seize cas possibles pour un fils u de k pris dans l'ensemble $\{n-1, n+1, k-1, k+1\}$, vis-à-vis des relations de u avec k et avec n. Ces cas sont présentés dans la figure B.1 Rappelons à tout fin utile que:

- le premier symbole désigne la relation de coprimarité de u avec u, ici u n'est jamais premier avec u car on a pris soin de s'éloigner de 1;
- le deuxième symbole désigne la relation de coprimarité de u avec k;
- le troisième symbole désigne la relation de succession de u par rapport à k (s'il n'apparaît pas c'est que u est éloigné de k);
- le quatrième symbole désigne la relation de coprimarité de u avec n;
- le cinquième symbole désigne la relation de succession de u par rapport à n (s'il n'apparaît pas c'est que u est éloigné de n).

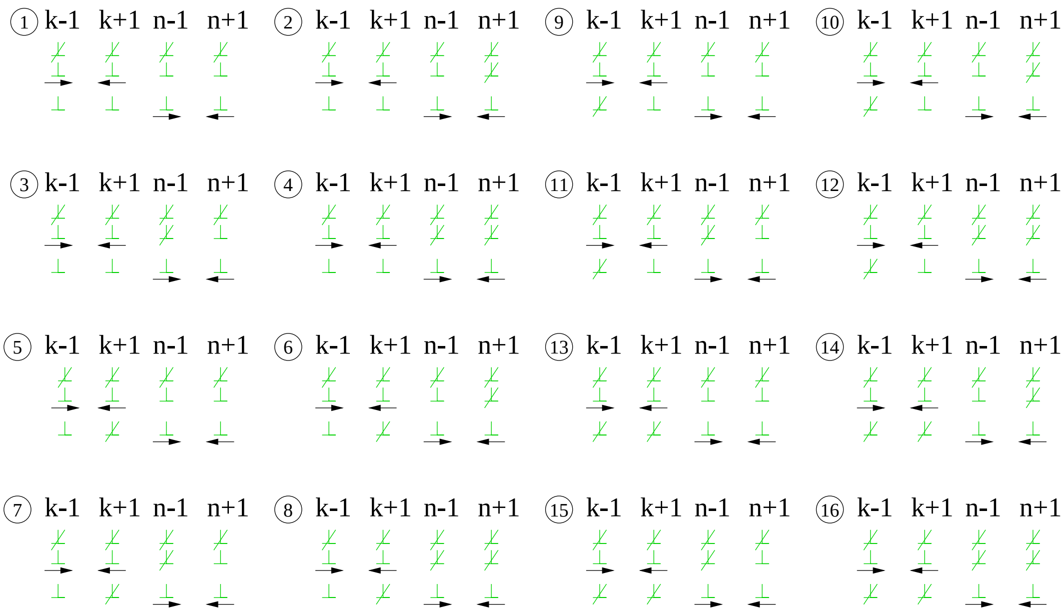


Figure B.1: Configurations pour u proche de k ou n, pour n et k éloignés

Nous allons dans ce qui suit présenter le résumé d'une étude de ces 80 cas qui permet de borner de manière plus précise le nombre de destinées possibles de cette 3-transversale essentielle.

Dans cette étude, les critères rencontrés qui vont être rajoutés par rapport aux 12 critères déjà différenciés dans le cadre de l'étude des pétioles pour k proche de n sont les suivants:

- n primaire, $|Supp(n)| = 2$, ou $|Supp(n)| > 2$;
- $n = 2^\alpha + 1$, $n = 2^\alpha - 1$ ou n d'aucune de ces formes.

Cela nous donne en tout 33 formes de n à distinguer, résumées dans le tableau B.2

B.2 $Supp(k) = Supp(n)$

Comme n et k ont les mêmes diviseurs premiers, on a donc nécessairement $k \pm 1 \perp n$, et $n \pm 1 \perp k$, donc si un cas est réalisé, c'est le cas 1. Il est réalisé pour $k = n^2$ par exemple. Donc tout est déterminé dans ce cas là.

B.3 $Supp(n) \subsetneq Supp(k)$

On a $p \mid n \Rightarrow p \mid k$ donc on a nécessairement $k \pm 1 \perp n$, ce qui nous laisse seulement 4 possibilités: les cas 1, 2, 3, 4.

n pair	$n = 2(2^\alpha - 1)$	$n \equiv 2[3]$	$\lvert Supp(n)\rvert = 2$	1
			$\lvert Supp(n)\rvert > 2$	2
		$n \equiv 0[3]$	$Supp(n) = \{2,3\}$	1
			$\lvert Supp(n)\rvert > 2$	1
	$n = 2(2^\alpha + 1)$	$n \equiv 1[3]$		1
		$n \equiv 0[3]$	$Supp(n) = \{2,3\}$	1
			$\lvert Supp(n)\rvert > 2$	1
	$n \neq 2(2^\alpha - 1)$ et $n \neq 2(2^\alpha + 1)$	$n \equiv 2[3]$	n primaire	1
			$\lvert Supp(n)\rvert \geq 2$	2
		$n \equiv 4[6]$	n primaire	1
			$\lvert Supp(n)\rvert \geq 2$	2
		$n \equiv 0[6]$	$Supp(n) = \{2,3\}$	1
			$\lvert Supp(n)\rvert > 2$	1
n impair	$n \equiv 5[6]$	$n = 2^\alpha + 1$	n primaire	1
			$\lvert Supp(n)\rvert = 2$	2
			$\lvert Supp(n)\rvert > 2$	3
		$n \neq 2^\alpha + 1$ et $n \neq 2^\alpha - 1$	n primaire	1
			$\lvert Supp(n)\rvert = 2$	2
			$\lvert Supp(n)\rvert > 2$	3
	$n \equiv 3[6]$	$n = 2^\alpha + 1$	n primaire	1
			$\lvert Supp(n)\rvert = 2$	2
			$\lvert Supp(n)\rvert > 2$	3
		$n = 2^\alpha - 1$	$\lvert Supp(n)\rvert = 2$	2
			$\lvert Supp(n)\rvert > 2$	3
		$n \neq 2^\alpha + 1$ et $n \neq 2^\alpha - 1$	n primaire	1
			$\lvert Supp(n)\rvert = 2$	2
			$\lvert Supp(n)\rvert > 2$	3
	$n \equiv 1[6]$	$n = 2^\alpha - 1$	n primaire	1
			$\lvert Supp(n)\rvert = 2$	2
			$\lvert Supp(n)\rvert > 2$	3
		$n \neq 2^\alpha + 1$ et $n \neq 2^\alpha - 1$	n primaire	1
			$\lvert Supp(n)\rvert = 2$	2
			$\lvert Supp(n)\rvert > 2$	3

Figure B.2: Formes de n donnant lieu à des destinées différentes, et nombre maximal de destinées par forme de n

Cas 1 $k \pm 1 \perp n$, $n \pm 1 \perp k$
Il est réalisé pour toute forme de n.

Cas 2 $k \pm 1 \perp n$, $n - 1 \perp k$, $n + 1 \not\perp k$
Il est impossible pour $n = 2^\alpha - 1$, et réalisé dans tous les autres cas.

Cas 3 $k \pm 1 \perp n$, $n + 1 \perp k$, $n - 1 \not\perp k$
Il est impossible pour $n = 2^\alpha + 1$, et réalisé dans tous les autres cas.

Cas 4 $k \pm 1 \perp n$, $n \pm 1 \not\perp k$
Il est réalisé pour toute forme de n.

B.4 $Supp(n) \cap Supp(k) = \emptyset$

B.4.1 n pair

Si n est pair, alors étant donné la disjonction des supports, k est nécessairement impair, et donc $k \pm 1$ est pair, et ne peut être premier avec n: seuls les cas 13, 14, 15, et 16 sont possibles. Ces quatre cas sont de plus réalisés.

B.4.2 n impair

- Les cas toujours réalisés sont les cas: 5, 8, 9 et 12;
- Les cas réalisés sauf lorsque $n \equiv 0[3]$: 1, 2, 3 et 4;
- Les cas réalisés sauf lorsque $n = 2^\alpha - 1$: 2, 6, 10 et 14;
- Les cas réalisés sauf lorsque $n = 2^\alpha + 1$: 3, 7, 11 et 15;
- Les cas réalisés sauf lorsque n est primaire: 13, 14, 15, et 16.

Et dans les cas non réalisés, on peut montrer qu'ils sont impossibles.

En définitive, tous les cas ont été passés en revue et aucune indécision ne demeure.

B.5 $Supp(n) \backslash Supp(k) \neq \emptyset$, $Supp(k) \backslash Supp(n) \neq \emptyset$ et $n \not\perp k$

B.5.1 n pair

- Les cas toujours réalisés sont les cas: 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16;
- Les cas réalisés si $Supp(n) \neq \{2, 3\}$ (et dans le cas contraire ils sont impossibles): 1, 2, 3, 4.

B.5.2 n impair

- Les cas toujours réalisés sont les cas: 1, 4, 5, 8, 9, 12, 13, et 16;
- Les cas réalisés sauf lorsque $n = 2^\alpha - 1$ (et dans ce cas ils sont impossibles): 2, 6, 10, et 14;
- Les cas réalisés sauf lorsque $n = 2^\alpha + 1$ (et dans ce cas ils sont impossibles): 3, 7, 11 et 15.

Là encore, tout est déterminé.

B.6 $Supp(k) \subsetneq Supp(n)$

Comme $p \mid k \Rightarrow p \mid n$ donc on a nécessairement $n \pm 1 \perp k$, ce qui nous laisse seulement 4 possibilités: les cas 1, 5, 9, 13.

B.6.1 n pair

- Les cas toujours réalisés sont les cas: 9, 13;
- Le cas réalisé sauf si $Supp(n) = \{2, 3\}$ (et dans ce cas il est impossible): 1

Il reste le cas 5, dont on sait qu'il est:

- Réalisé pour $n \equiv 0[6]$ ou $n = 2(2^\alpha + 1)$;
- Impossible pour $n = 2(2^\alpha - 1)$ tel que $|Supp(n)| = 2$;

Pour les autres formes de n il y a donc une indétermination: on ne sait pas si pour une forme donnée tous les n vérifie le cas, le rende impossible, ou si certains n vérifie et d'autres non (ce qui introduirait un sous-cas). On a donc au plus 2 destinées possibles pour chacun de ces cas indéterminé.

B.6.2 n impair

- Les cas toujours réalisés sont les cas: 1, 9;

Pour le cas 13, on sait qu'il est impossible si $|Supp(n) = 2|$, sinon on ne sait rien. Ce cas correspond aux éléments impairs de support de cardinal ≥ 3 de l'ensemble X.
Pour le cas 5, on ne sait pas non plus. Ce cas correspond aux éléments impairs de l'ensemble Y.
On sait cependant en étudiant les ensembles X et Y, que si $n \in X$, alors $n \in Y$. Il y a donc au plus trois destinées possibles (pour un élément k dans $Y \backslash X$, un dans X, un en dehors de Y) pour ces cas indeterminés.

B.7 Conclusion sur le nombre de destinées

On est donc en mesure, parmi les 37 formes de n distinguées précédemment, de dire lesquelles ont au plus une destinée, au plus 2, au plus 3. Le nombre maximal de destinées par forme de n est indiqué dans le tableau B.2

Si on additionne tout, on trouve 57 destinées au plus, à quoi il faut ajouter les destinées de 0, 1, 2, 3, 4 et 6 qui ont été écartées au cours de la discussion générale: cela fait un total d'au plus 63 destinées.

Appendix C

Programme de vérification: code source en Caml

Pour tout renseignement sur le langage Caml, on peut se reporter à [4] ou à [8].

C.1 L'analyse syntaxique et sémantique

On a utilisé les outils fournis par Caml: `ocamlyacc` et `ocamllex`, qui engendrent automatiquement l'analyseur syntaxique et l'analyseur sémantique à partir d'un fichier `formule_lexer.mll` fournissant la syntaxe des éléments à analyser, et d'un fichier `formule_parser.mly` indiquant la grammaire.

```
(* Fichier formule_lexer.mll *)
{
open Formule_parser
}

rule token = parse
    [' ' '\t' '\n']         {token lexbuf}
  | '('                     {Lparen}
  | ')'                     {Rparen}
  | ','                     {Koma}
  | "Forall"                {Lforall}
  | "Exists"                {Lexists}
  | "Et"                    {Let}
  | "Ou"                    {Lou}
  | "Non"                   {Lnon}
  | "->"                    {Limplique}
  | "<->"                   {Lequivalent}
  | "Coprem"                {Lcopremier}
```

```
  | "Succ"                    {Lsuccesseur}
  | ['a'-'z']                 {Lvariable (Lexing.lexeme lexbuf)}

(* Fichier formule_parser.mli *)

open Formule_types

Lsuccesseur

ma_formule :
    Lvariable
      { Variable $1}
  | Lsuccesseur Lparen Lvariable Koma  Lvariable Rparen
      { Successeur ((Variable $3) , (Variable $5))}
  | Lcopremier Lparen Lvariable Koma Lvariable  Rparen
      { Copremier  ((Variable $3) , (Variable $5))}
  | Lnon  Lparen ma_formule  Rparen
      { Non $3 }
  | Lparen ma_formule Rparen Let  Lparen ma_formule Rparen
      { Et ($2 , $6) }
  | Lparen ma_formule Rparen Lou Lparen ma_formule  Rparen
      { Ou ($2 , $6) }
  | Lparen ma_formule Rparen Limplique Lparen ma_formule Rparen
      { Implique ($2 , $6) }
  | Lparen ma_formule Rparen Lequivalent Lparen ma_formule Rparen
      { Equivalent ($2 , $6) }
  | Lforall Lvariable Lparen ma_formule Rparen
      { Forall ((Variable $2) , $4) }
  | Lexists Lvariable Lparen ma_formule  Rparen
      { Exists ((Variable $2) , $4) }
;
```

Cette phase permet de transformer une chaîne de caractères tapée par l'utilisateur en un objet du type `formule`, structuré et manipulable par le programme. On se donne en outre, une fois engendrés les fichiers `formule_parser.ml` et `formule_lexer.ml`, un outil pour parser les fichiers ou les chaînes de caractères en ligne de commande:

```
(* Fichier parse_formule.ml *)

(* Fonction qui parse à partir d'un fichier *)
let parse_formule_file f =
  let lexbuf=Lexing.from_channel (open_in f)
  in
```

```
  ((ma_formule token lexbuf):Formule_types.formule)
;;
(* fonction qui parse à partir d'une chaîne de caractères *)
let parse_formule_string s =
    let lexbuf=Lexing.from_string s
    in
    ((ma_formule token lexbuf):Formule_types.formule)
;;
```

C.2 Les outils de manipulation de destinées et de formules

```
(*
La fonction verification_formule_close permet de vérifier
si la formule est close; pour cela elle réalise une liste
de variables liées et vérifie grâce à la fonction
test_appartient_variable que chaque variable des termes est
dans la liste en question.
*)

let rec test_appartient_variable x l =
   match l with
        []   -> false
      | u::m -> if x=u then true
                       else test_appartient_variable x m
;;

let verification_formule_close f =
   let rec variable l k =
      match k with
           Vrai                   -> true
         | Faux                   -> true
         | Non g                  -> variable l g
         | Et (g,h)               -> (variable l g) && (variable l h)
         | Ou (g,h)               -> (variable l g) && (variable l h)
         | Implique (g,h)         -> (variable l g) && (variable l h)
         | Equivalent (g,h)       -> (variable l g) && (variable l h)
         | Forall (Variable x , g) -> variable (x::l) g
         | Exists (Variable x , g) -> variable (x::l) g
         | Copremier (Variable x, Variable y)    ->
            if ((test_appartient_variable x l)
             && (test_appartient_variable x l))
              then true
```

```
             else false
         | Successeur (Variable x, Variable y)   ->
            if ((test_appartient_variable x l)
             && (test_appartient_variable x l))
             then true
             else false
         | Variable x             -> false
         | _                      ->
            failwith "Verification formule close impossible"
  in variable [] f
;;

(*
La fonction rang_quantification compte le rang de
quantification d'une formule
*)

let rec rang_quantification f =
   match f with
        Vrai                                -> 0
      | Faux                                -> 0
      | Non g                               -> rang_quantification g
      | Et (g,h)                            ->
          max (rang_quantification g) (rang_quantification h)
      | Ou (g,h)                            ->
          max (rang_quantification g) (rang_quantification h)
      | Implique (g,h)                      ->
          max (rang_quantification g) (rang_quantification h)
      | Equivalent (g,h)                    ->
          max (rang_quantification g) (rang_quantification h)
      | Forall (Variable x , g )            -> (rang_quantification g) +1
      | Exists (Variable x , g )            -> (rang_quantification g) +1
      | Copremier (Variable x, Variable y)    -> 0
      | Successeur (Variable x, Variable y)   -> 0
      | Variable x                          -> 0
      | _                                   ->
          failwith "Le rang de quantification est impossible a calculer"
;;

(*
La formule hauteur_destinee calcule la hauteur d'une
liste de destinées
*)
```

```
let rec hauteur_destinee dlist =
   match dlist with
        [Empty]             -> 0
      | [Node (_,_,_,l)]    -> (hauteur_destinee l)+1
      | (Node(_,_,_,l))::m  ->
          max ((hauteur_destinee l)+1) (hauteur_destinee m)
      | _                   ->
failwith "Impossible de calculer la hauteur de la liste de destinées"
;;

(*
La fonction verif_dest_formule vérifie que le rang de
quantification de la formule est inférieur à la hauteur
de la liste de destinées
*)

let verif_dest_formule d f =
   if ((hauteur_destinee d)>= (rang_quantification f))
      then true
      else false
;;

(*
La fonction examine intervient lorsque l'on veut regarder
si un prédicat est vérifié dans la destinée: cette fonction
prend en paramètre un entier qui indique à quel endroit dans
la liste de propriétés il faut regarder et renvoie l'élément
en question
*)

let examine n l =
   let rec aux k m =
      if (k<n) then
                  match m with
                       []       ->
        failwith "Pas de nième propriété dans la liste de propriétés"
                     | x::s     -> aux (k+1) s
      else if (k=n) then
                       match m with
                              []       ->
        failwith "Pas de nième propriété dans la liste de propriétés"
                            | x::s     -> x
                    else
        failwith "Impossible d'examiner la propriété demandée"
```

```
   in aux 1 l
;;

(*
La fonction test_appartient_variables_nivelees teste si
une variable appartient à une liste de variables pondérées
par leur niveau, et elle renvoie un couple booleen * niveau
*)

let rec test_appartient_variables_nivelees x l =
   match l with
        []          -> (false, -1)
      | (u,r)::m    -> if (u=x) then (true , r)
                                 else
          test_appartient_variables_nivelees x m
;;

(*
La fonction ajoute_liste ajoute, s'il elle n'y est pas déjà,
une variable à la liste des variables nivelées
*)

let ajoute_liste x l =
   if (fst (test_appartient_variables_nivelees x l) = false)
      then match l with
                []            -> [(x,1)]
              | ((u,r))::m    -> ((x,r+1))::l
      else l
;;
```

C.3 La fonction d'évaluation

```
(*
Corps de la fonction d'évaluation
Elle prend en entrée une destinée (et non une liste de destinées)
et une formule à évaluer
*)
let evalue d f =
   let rec aux l d f =
      match f with
           Vrai                   ->  true
         | Faux                   ->  false
         | Non g                  ->  if (aux l d g)
```

```
                                          then false
                                          else true
      | Et (g,h)                -> ((aux l d g)&&(aux l d h))
      | Ou (g,h)                -> ((aux l d g)||(aux l d h))
      | Implique (g,h)          -> aux l d (Ou (Non g , h))
      | Equivalent (g,h)        ->
          aux l d (Et (Implique (g,h),Implique (h,g)))
      | Forall (Variable x ,h)  -> if (l = (ajoute_liste x l))
          then
failwith "La variable associée à ce quantificateur est déja liée"
          else
(match d with
      Empty           -> failwith "Destinée vide"
    | Node (k,u,m,s)  -> let rec aux_forall t =
        match t with
             [Empty]       -> failwith "Destinée trop courte"
           | [z]           -> aux (ajoute_liste x l) z h
           | z::m          -> ((aux (ajoute_liste x l) z h)
                                &&(aux_forall m))
           | _             -> failwith "Destinée mal foutue"
                          in aux_forall s )
      | Exists (Variable x ,h)  -> if (l = (ajoute_liste x l))
          then
failwith "La variable associée à ce quantificateur est déja liée"
          else
(match d with
      Empty           -> failwith "Destinée vide"
    | Node (k,u,m,s)  -> let rec aux_exists t =
        match t with
             [Empty]       -> failwith "Destinée trop courte"
           | [z]           -> aux (ajoute_liste x l) z h
           | z::m          -> ((aux (ajoute_liste x l) z h)
                                ||(aux_exists m))
           | _             -> failwith "Destinée mal foutue"
                         in aux_exists s )
      | Copremier (Variable x, Variable y)   ->
          if ((fst(test_appartient_variables_nivelees x l))
            &&(fst(test_appartient_variables_nivelees y l)))
            then
let aux_copremier a b = if (a=b)
   then
     match d with
         Empty         -> failwith "Destinée vide"
       | Node(k,u,m,r) -> if ((max a b) <k)
```

```
               then
failwith "Formule mal construite: imbriquez mieux les quantificateurs"
               else  examine 1 m
      else
        match d with
             Empty         -> failwith "Destinée vide"
           | Node(k,u,m,r) -> if ((max a b) <k)
               then
failwith "Formule mal construite: imbriquez mieux les quantificateurs"
               else examine (3*((max a b)-(min a b)) -1) m
   in aux_copremier (snd(test_appartient_variables_nivelees x l))
                    (snd(test_appartient_variables_nivelees y l))
               else failwith "Les variables ne sont pas liées"
         | Successeur (Variable v, Variable w)  ->
             if ((fst(test_appartient_variables_nivelees v l))
               &&(fst(test_appartient_variables_nivelees w l)))
               then
   let aux_successeur a b = if (b>a)
      then
        match d with
             Empty         ->  failwith "Destinée vide"
           | Node(k,u,m,r) ->   if ((max a b) <k)
               then
failwith "Formule mal construite: imbriquez mieux les quantificateurs"
               else  examine (3*(b-a)) m
      else if (a>b)
        then
          match d with
               Empty         ->  failwith "Destinée vide"
             | Node(k,u,m,r) ->  if ((max a b) <k)
                 then
failwith "Formule mal construite: imbriquez mieux les quantificateurs"
                 else  examine (3*(b-a)+1) m
        else failwith "Les variables ne peuvent pas être au même niveau"
   in aux_successeur (snd(test_appartient_variables_nivelees v l))
                     (snd(test_appartient_variables_nivelees w l))
               else failwith "Les variables n'ont pas été liées"
         | _                        ->
   failwith "La formule n'a pu être evaluée"
   in aux [] d f
;;
```

C.4 L'interface avec l'utilisateur

```
open String;;


(**********************
  INTERFACE UTILISATEUR
 **********************)

(* Premier contact *)

let presentation =
   "   **************************************************
   *                                                *
   *  Bonjour, vous êtes dans l'interface interactive  *
   *  de l'évaluateur de formules du successeur et de  *
   *  la coprimarité à deux quantificateurs.          *
   *                                                *
   **************************************************\n"
;;

(* Liste des aides liées à chaque commande *)


let help_of_eval =
  "eval file nom_de_fichier: lance l'exécution de l'évaluation
      de la formule apparaissant dans ce fichier
eval formule: évalue directement la formule tapée
      après la commande\n"
;;

let help_of_quit_interface =
  "quit: quitte la magnifique interface utilisateur
      et revient au top_level de OCaml\n"
;;


(* Commande help *)

let help_interface s =
   match s with
      "eval"      ->  print_string help_of_eval
     |"quit"      ->  print_string help_of_quit_interface
```

```
     | _          ->  print_string help_of_eval;
                      print_string help_of_quit_interface;
;;

(* Commandes eval *)

let print_result =
   function
         true -> print_string "La formule est un théorème\n"
       | false -> print_string "La formule n'est pas un théorème\n"
;;

let eval s =
try
   let formule =
      if ((sub s 0 4)="file")
         then parse_formule_file (sub s 5 ((length s)-5))
         else parse_formule_string s
   in
     if (verif_dest_formule (dest2) formule)
       then print_result (evalue (Node(0, -1 , [] , dest2)) formule)
       else failwith "Il y a trop de quantificateurs"
with  Failure comment -> print_string (comment^"\n")
     |_               -> print_string ("Formule non valide\n")
;;

(* Interface *)

let aux s =
try
   if (s ="help")
     then (help_interface "";true)
     else
       if (s="quit")
         then ( print_string "Au revoir...";false)
         else
           if (sub s 0 4 = "eval")
             then (eval (sub s 5 ((length s)-5));true)
             else
               if (sub s 0 4 ="help")
                 then (help_interface (sub s 5 ((length s)-5));true)
                 else (print_string "Commande non valide\n";true)
with  Failure s -> (print_string (s^"\n");true)
     |_         -> (print_string "Commande non valide\n";true)
```

```
;;

let rec interface () =
   print_string "Evaluation> ";
   flush stdout;
   let s=read_line()
   in (if (aux s=true) then interface ())
;;


(* exécution *)

let top_level () =
   print_string presentation;
   help_interface "";
   interface ()
;;
```

Bibliography

[1] G.D.Birkhoff, H.S.Vandiver, *On the integral divisors of $a^n - b^n$* , Annals of Mathematics, Serie II, Vol.5, 1904. (p. 173-180)

[2] P.Cegielski, *Quelques contributions à l'étude des arithmétiques faibles*, Thèse de Doctorat d'Etat, Université Paris7, 1990. (p. 115 - 141)

[3] R.D.Carmichael, *On the numerical factors of the arithmetic forms $\alpha^n \pm \beta^n$*, Annals of Mathematics, Serie II, Vol.15, 1913-1914. (p.30-69)

[4] E.Chailloux - P.Manoury - B.Pagano, *Développement d'applications avec Objective Caml*, O'Reilly, 2000.

[5] R.Cori - D.Lascar, *Logique mathématique. Cours et exercices*, Tomes 1 et 2, Masson, 1993.

[6] M.Demazure, *Cours d'algèbre. Primalité, divisilbilité, codes*, Cassini, 1997. (p.43)

[7] S.Grigorieff, D.Richard, *Contribution à l'étude d'une conjecture de théorie des nombres par le codage ZBV*, L'enseignement mathématique, 2^e série, Tome 35, Fascicule 1-2, 1989.

[8] X.Leroy - P.Weis, Le langage Caml, 2$^{\text{ème}}$ édition, Dunod, 1999.

[9] F.Nézondet, p-*destinées et applications à la théorie du successeur et de la coprimarité sur les entiers*, Thèse de doctorat, Université d'Auvergne, juin 1997.

[10] F.Nézondet - D.Richard, *Brief survey and latest results up to 97 on successor and coprimeness*, prépublication du LLAIC n°63, 1997.

[11] D.Richard, *All mathematical sets of powers of primes are first-order definable in terms of the successor function and the coprimness predicate*, Discret Mathematics 53, North Holland, 1985.

[12] D.Richard, *Definability in terms of the successor function and the coprimness predicate in the set of arbitrary integers*, Journal of Symbolic Logic, Vol.54, N°4, 1989.

[13] J.P.SERRE, *Cours d'arithmétique*, Presses Universitaires de France, 4^e édition, 1995.

[14] A.WOODS, *Some problems in logic and number theory and their connections*, Thesis, University of Manchester, 1981.