MATHÉMATIQUES ET SCIENCES HUMAINES

AARNE RANTA

Structures grammaticales dans le français mathématique : II - (suite et fin)

Mathématiques et sciences humaines, tome 139 (1997), p. 5-36

<http://www.numdam.org/item?id=MSH_1997__139__5_0>

Math. Inf. Sci. hum., (35e année, n°139, *1997*, pp. 5-36)

STRUCTURES GRAMMATICALES DANS LE FRANÇAIS MATHÉMATIQUE : II - (suite et fin)[1]

Aarne RANTA[2]

RÉSUMÉ — *Un système de règles grammaticales est présenté pour analyser un fragment du français permettant l'expression de théorèmes et de preuves mathématiques. Pour cet objectif, on développe une version de la grammaire de Montague, avec des catégories syntaxiques relatives au contexte et aux domaines d'individus. Ce système peut être interprété dans la théorie constructive des types de Martin-Löf. Il est appliqué, d'abord, au français sans symboles mathématiques, avec une attention spéciale aux restrictions de sélection et aux dépendances par rapport à un contexte. Le fragment comprend des verbes et des adjectifs, des formes plurielles, des propositions relatives, et des syntagmes coordonnés. Ensuite, la grammaire est étendue au symbolisme mathématique et à son usage dans le texte français. Le fragment comprend des formules arithmétiques, la notation décimale, les conventions de parenthèses, les variables explicites, des énoncés de théorèmes et des structures textuelles de preuves. On finit par étudier quelques applications de la grammaire, basées sur l'implémentation déclarative de la grammaire dans ALF, un éditeur de preuves.*

SUMMARY — Grammatical Structure in Mathematical French: II.
A system of grammatical rules is presented to analyse a fragment of French that permits the expression of mathematical theorems and proofs. To this end, a version of Montague grammar is developed, with syntactic categories relativized to a context and to domains of individuals. This system can be interpreted in the constructive type theory of Martin-Löf. It is first applied to French without mathematical symbols, paying special attention to selectional restrictions and to dependencies on context. The fragment includes verbs and adjectives, plurals, relative clauses, and coordinated phrases of different categories. Second, the grammar is extended to mathematical symbolism and its embedding in French text. The fragment comprises arithmetical formulae, decimal notation, parenthesis conventions, explicit variables, statements of theorems, and textual structures of proofs. Finally, some applications of the grammar are studied, based on a declarative implementation in the proof editor ALF.

---

[1] La première partie de cet article a paru dans le n° 138 de cette revue, pages 5 à 56, sous le titre «Structures grammaticales dans le français mathématique : I».

[2] Département de Philosophie, B.P. 24, 00014 Université d'Helsinki, Finlande.

Cet article est une synthèse du travail dont j'ai eu l'occasion de présenter des extraits à Paris et à Nancy en février 1996. Je suis reconnaissant aux publics de ces conférences pour leurs remarques et, en particulier, à M. Bourdeau et J.-P. Desclés, de l'Université de Paris-Sorbonne, à D. Lacombe, de l'Université de Paris VII, et à C. Retoré, de l'INRIA Lorraine, pour les entretiens privés sur les thèmes de l'article. P. Martin-Löf, de l'Université de Stockholm, m'a aidé pendant tout le travail par ses critiques et conseils. U. Egli, de l'Université de Constance, A. Lecomte, de l'Université de Grenoble et P. Mäenpää, de l'Université d'Helsinki ont lu la première version de l'article et fait des remarques pertinentes sur le contenu. En outre, M. Bourdeau et A. Lecomte ont corrigé le français des versions préliminaires. Les fautes qui restent dans la version finale sont miennes.

3 LE FRANÇAIS AVEC DES SYMBOLES MATHÉMATIQUES

Une ligne dans l'évolution des mathématiques est l'introduction du symbolisme, pour permettre l'expression concise, exacte et facilement lisible d'idées compliquées. La prose mathématique est ainsi un mélange de mots et de symboles, mais un mélange bien ordonné. Dans l'enseignement des mathématiques, le symbolisme est introduit par des règles d'usage assez précises, et il y a des règles explicites de style qui concernent l'usage du symbolisme dans le texte mathématique. (La partie du texte appartenant à la langue ordinaire est, d'habitude, présupposée, comme si elle était évidente à tous ceux qui parlent la langue dans la vie quotidienne. Or, cette partie «informelle» est aussi importante pour la validité du texte que la partie symbolique.)

Dans les sections précédentes, nous avons étudié la langue générale avec l'objectif de l'analyser avec une précision qui suffise à juger la correction des textes mathématiques. Dans les sections qui suivent, nous introduisons le symbolisme et l'interface entre le symbolisme et la langue générale, ainsi que des structures qui mélangent les deux.

3.1 *Le symbolisme mathématique*

La grammaire des notations symboliques est bien connue dans l'informatique. Cependant, la notation préférée en informatique n'est pas toujours la même que la notation traditionnelle des mathématiques : celle-ci n'est pas toujours formelle dans le sens strict. Sa description grammaticale dans le format que nous avons employé pour le français n'est toutefois pas problématique.

Nous étudions le système suivant de catégories :

catégorie	symbole	où	interprétation	exemple
formule	$\mathrm{Fml}(\Gamma)$	Γ : cont	prop/Γ	$x < 2$
terme	$\mathrm{Trm}(\Gamma, A)$	Γ : cont, A : set/Γ	A/Γ	$\sin(x+1)$
constante	$\mathrm{Const}(A)$	A : set	A	4
variable	$\mathrm{Var}(\Gamma, A)$	Γ : cont, A : set/Γ	A/Γ	x
prédicat à 2 pl.	$\mathrm{Prd}(A, B)$	A, B : set	$(A)(B)\mathrm{prop}$	$<$
préfixe	$\mathrm{Préf}(A, B)$	A, B : set	$(A)B$	sin
postfixe	$\mathrm{Postf}(A, B)$	A, B : set	$(A)B$	!
infixe	$\mathrm{Inf}(A, B, C)$	A, B, C : set	$(A)(B)C$	$+$

Les règles du symbolisme sont obtenues à partir des règles syntagmatiques familières en y incorporant les restrictions de sélection et les dépendances par rapport à un contexte. En outre, nous introduisons une version grammaticale de la technique des **nombres de priorité** pour exprimer les conditions de l'omission des parenthèses. Un tel nombre, désigné par $\mathrm{prior}(c)$, est défini pour tous les termes et pour tous les infixes.[73]

$\mathrm{Fml} \rightarrow \mathrm{Trm}\ \mathrm{Prd}\ \mathrm{Trm}$

$$\frac{\Gamma : \mathrm{cont} \quad A, B : \mathrm{set} \quad a : \mathrm{Trm}(\Gamma, A) \quad F : \mathrm{Prd}(A, B) \quad b : \mathrm{Trm}(\Gamma, B)}{\begin{array}{l}\mathrm{PrédPrd}(\Gamma, A, B, a, F, b) : \mathrm{Fml}(\Gamma) \\ \mathrm{PrédPrd}(\Gamma, A, B, a, F, b)^\star = F^\star(a^\star, b^\star) \\ \mathrm{PrédPrd}(\Gamma, A, B, a, F, b)^\circ = a^\circ\ F^\circ\ b^\circ\end{array}}$$

[73] Voir Paulson (1991) pour les nombres de priorité (en anglais, *precedence*) dans le langage ML. Ils sont aussi utilisés dans le langage Prolog.

Trm $\rightarrow$ Const

$$\frac{\Gamma : \text{cont} \quad A : \text{set} \quad c : \text{Const}(A)}{\begin{array}{l}\text{UseConst}(\Gamma, A, c) : \text{Trm}(\Gamma, A) \\ \text{UseConst}(\Gamma, A, c)^\star = c^\star \\ \text{UseConst}(\Gamma, A, c)^\circ = c^\circ \\ \text{prior}(\text{UseConst}(\Gamma, A, c)) = 9\end{array}}$$

Trm $\rightarrow$ Var

$$\frac{\Gamma : \text{cont} \quad A : \text{set}/\Gamma \quad v : \text{Var}(\Gamma, A)}{\begin{array}{l}\text{UseVar}(\Gamma, A, v) : \text{Trm}(\Gamma, A) \\ \text{UseVar}(\Gamma, A, v)^\star = v^\star \\ \text{UseVar}(\Gamma, A, v)^\circ = v^\circ \\ \text{prior}(\text{UseVar}(\Gamma, A, v)) = 9\end{array}}$$

Trm $\rightarrow$ Préf Trm[74]

$$\frac{\Gamma : \text{cont} \quad A, B : \text{set} \quad f : \text{Préf}(A, B) \quad a : \text{Trm}(\Gamma, A)}{\begin{array}{l}\text{ApplPréf}(\Gamma, A, B, f, a) : \text{Trm}(\Gamma, B) \\ \text{ApplPréf}(\Gamma, A, B, f, a)^\star = f^\star(a^\star) \\ \text{ApplPréf}(\Gamma, A, B, f, a)^\circ = f^\circ\ \text{cas}(\text{prior}(a) < 7, (a^\circ), a^\circ) \\ \text{prior}(\text{ApplPréf}(\Gamma, A, B, f, a)) = 7\end{array}}$$

Trm $\rightarrow$ Trm Postf

$$\frac{\Gamma : \text{cont} \quad A, B : \text{set} \quad a : \text{Trm}(\Gamma, A) \quad f : \text{Postf}(A, B)}{\begin{array}{l}\text{ApplPostf}(\Gamma, A, B, a, f) : \text{Trm}(\Gamma, B) \\ \text{ApplPostf}(\Gamma, A, B, a, f)^\star = f^\star(a^\star) \\ \text{ApplPostf}(\Gamma, A, B, a, f)^\circ = \text{cas}(\text{prior}(a) < 8, (a^\circ), a^\circ)\ f^\circ \\ \text{prior}(\text{ApplPostf}(\Gamma, A, B, a, f)) = 8\end{array}}$$

Trm $\rightarrow$ Trm Inf Trm[75]

$$\frac{\Gamma : \text{cont} \quad A, B, C : \text{set} \quad a : \text{Trm}(\Gamma, A) \quad f : \text{Inf}(A, B, C) \quad b : \text{Trm}(\Gamma, B)}{\begin{array}{l}\text{ApplInf}(\Gamma, A, B, C, a, f, b) : \text{Trm}(\Gamma, C) \\ \text{ApplInf}(\Gamma, A, B, C, a, f, b)^\star = f^\star(a^\star, b^\star) \\ \text{ApplInf}(\Gamma, A, B, C, a, f, b)^\circ = \\ \qquad \text{cas}(\text{prior}(a) < \text{prior}(f), (a^\circ), a^\circ)\ f^\circ\ \text{cas}(\text{prior}(b) \leq \text{prior}(f), (b^\circ), b^\circ) \\ \text{prior}(\text{ApplInf}(\Gamma, A, B, C, a, f, b)) = \text{prior}(f)\end{array}}$$

En outre, il y a des notations spéciales, telles que la notation pour la dérivée d'une fonction (cf. section 2.8) :

[74] L'opérateur booléen $\text{cas}(c, a, b)$ décidant ici de l'usage des parenthèses retourne la valeur a si c est vrai, et b si c est faux.

[75] La règle ApplInf définit la catégorie Inf comme la catégorie des infixes associatifs à gauche. Puisque $+$ et $-$ lui appartiennent, et qu'ils ont la même priorité (v. section 3.3), le terme $7 - 6 + 5$ est analysé $(7 - 6) + 5$.

Trm $\rightarrow$ Trm′(Trm)

$$\frac{\Gamma : \text{cont} \quad f : \text{Trm}(\Gamma, R \rightarrow R) \quad x : \text{Trm}(\Gamma, R) \quad c : \text{Dér}(f^\star, x^\star)/\Gamma}{\begin{array}{l}\text{Dérivée}(\Gamma, f, x, c) : \text{Trm}(\Gamma, R) \\ \text{Dérivée}(\Gamma, f, x, c)^\star = D(f^\star, x^\star, c) \\ \text{Dérivée}(\Gamma, f, x, c)^\circ = f^{\circ\prime}(x^\circ)\end{array}}$$

3.2 *Les variables explicites*

Les extensions du contexte dans la langue ordinaire n'introduisent que des **variables implicites** c.-à-d. des variables visibles dans la représentation formelle mais non dans les chaînes obtenues par linéarisation. Dans la prose mathématique, il y a aussi des **variables explicites**, mais toutes les variables ne le sont pas. Pour ne pas détruire cette distinction, nous introduisons un nouveau type d'extension d'un contexte, où un symbole est fourni pour être utilisé dans le texte linéarisé :[76] [77]

$$\frac{\Gamma : \text{cont} \quad A : \text{set}/\Gamma \quad t : \text{symb}}{(\Gamma,\ x_t : A) : \text{cont}}.$$

Il n'y a aucune différence sémantique entre $(\Gamma,\ x_t : A)$ et $(\Gamma,\ x : A)$. Le jugement

$$x : A/(\Gamma,\ x_t : A)$$

est valide, ce qui signifie qu'une extension explicite a aussi la force d'une extension implicite.

Une extension explicite introduit une **nouvelle variable** dans le contexte, inaugurée dans la grammaire par l'opérateur Nvar. Après chaque extension, soit implicite soit explicite, on peut récupérer les **variables anciennes** par l'opérateur Avar.

Var $\rightarrow$ symb[78]

$$\frac{\Gamma : \text{cont} \quad A : \text{set}/\Gamma \quad t : \text{symb}}{\begin{array}{l}\text{Nvar}(\Gamma, A, t) : \text{Var}((\Gamma,\ x_t : A), A) \\ \text{Nvar}(\Gamma, A, t)^\star = x \\ \text{Nvar}(\Gamma, A, t)^\circ = t\end{array}}$$

Var $\rightarrow$ Var

$$\frac{\Gamma : \text{cont} \quad A : \text{set}/\Gamma \quad B : \text{set}/\Gamma \quad v : \text{Var}(\Gamma, A)}{\begin{array}{l}\text{Avar}(\Gamma, A, B, v) : \text{Var}((\Gamma,\ x : B), A) \\ \text{Avar}(\Gamma, A, B, v)^\star = v^\star \\ \text{Avar}(\Gamma, A, B, v)^\circ = v^\circ\end{array}}$$

Par exemple, dans le contexte

$$\Gamma = (y_f : R \rightarrow R,\ z : (\forall x : R)\text{Dér}(y, x),\ u : \text{Cr}(y),\ v_x : R),$$

il y a deux variables explicites, les arbres

[76] Cf. la section sur les contextes dans l'introduction.

[77] Dans l'extension du contexte, le variable x doit être fraîche, c.-à-d. pas encore déclarée dans Γ. Nous ne posons pas la condition que le symbole t soit frais, parce qu'un auteur soigneux peut surcharger ses symboles intentionnellement. Il n'y a aucun risque de confusion dans l'interprétation, parce que celle-ci est déterminée par x, et le symbole t n'y joue aucun rôle.

[78] La règle de linéarisation présuppose qu'un symbole peut s'utiliser comme chaîne.

$\mathrm{Avar}(\mathrm{Avar}(\mathrm{Avar}(\mathrm{Nvar}(f)))) : \mathrm{Var}(\Gamma, R \to R)$,

$\mathrm{Nvar}(x) : \mathrm{Var}(\Gamma, x)$,

représentant les variables-chaînes f et x. Dans un contexte obtenu à partir de Γ en y ajoutant une hypothèse, ces représentations sont modifiées par une nouvelle application de Avar.

Donc la représentation grammaticale d'une variable explicite, par un arbre de la catégorie $\mathrm{Var}(\Gamma, A)$, varie selon le contexte, quoique la linéarisation reste la même. Les arbres pour les variables sont, en effet, une version formalisée des **indices** de de Bruijn (1972), indiquant la **profondeur de référence** (*reference depth*). Chez de Bruijn, les indices sont des chiffres : $0, 1, 2, \ldots$ Leur définition formelle dans la grammaire nécessite les arguments de contexte et de domaine. Mais on peut introduire un système d'abréviations, qui permet l'usage des chiffres :

$0(t) = \mathrm{Nvar}(t)$,

$1(t) = \mathrm{Avar}(\mathrm{Nvar}(t))$,

$2(t) = \mathrm{Avar}(\mathrm{Avar}(\mathrm{Nvar}(t)))$,

etc.

Nous avons supprimé les arguments sémantiques mais retenu le symbole t qui est utilisé dans la linéarisation.

3.3 *La notation décimale des nombres entiers*

Pour donner un exemple concret d'un système de notation, considérons les chiffres du système décimal désignant les nombres naturels :

$$0, 1, 2, 3, \ldots, 19063, 19064, \ldots$$

La grammaire de la notation décimale est, pour ainsi dire, une miniature de toute la grammaire, consistant en trois sortes d'objets — les arbres, les chaînes, et les objets mathématiques — et les opérations compositionnelles d'interprétation et de linéarisation. L'objectif a été de trouver les règles qui engendrent tous les chiffres, rien que les chiffres, et chaque chiffre seulement une fois (puisque les chiffres ne sont jamais ambigus). La grammaire suivante est la plus simple que nous ayons trouvée.

Il y a trois catégories, toutes interprétées comme N,

Num de tous les nombres entiers,

Pos des nombres positifs,

Dig des digits $1, 2, \ldots, 9$.

Dans les règles, nous employons des symboles distincts pour les digits-chaînes, les digits-arbres, et les digits-nombres, pour mettre en évidence la structure tripartite de la grammaire. Ainsi nous représentons les chaînes par $0, 1, 2\ldots$, les arbres par $0', 1', 2', \ldots$ et les nombres par $\mathbf{0}, \mathbf{1}, \mathbf{2}, \ldots$

$\mathrm{Num} \to 0 \mid \mathrm{Pos}$

$$\begin{array}{l} 0' : \mathrm{Num} \\ 0'^{\star} = \mathbf{0} \\ 0'^{\circ} = 0 \end{array} \qquad \frac{n : \mathrm{Pos}}{\begin{array}{l} \mathrm{UsePos}(n) : \mathrm{Num} \\ \mathrm{UsePos}(n)^{\star} = n^{\star} \\ \mathrm{UsePos}(n)^{\circ} = n^{\circ} \end{array}}$$

Pos $\rightarrow$ Dig | Pos 0 | Pos Dig

k : Dig	n : Pos	n : Pos k : Dig
UseDig(k) : Pos	AddZ(n) : Pos	AddDig(n, k) : Pos
UseDig(k)* = $k^\star$	AddZ(n)* = $n^\star * \mathbf{10}$	AddDig(n, k)* = $n^\star * \mathbf{10} + k^\star$
UseDig(k)$^\circ$ = k°	AddZ(n)$^\circ$ = $n^\circ 0$	AddDig(n, k)$^\circ$ = $n^\circ k^\circ$

Dig $\rightarrow$ 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

$1'$: Dig	$2'$: Dig		$9'$: Dig
$1'^\star$ = **1**	$2'^\star$ = **2**	$\cdots$	$9'^\star$ = **9**
$1'^\circ$ = 1	$2'^\circ$ = 2		$9'^\circ$ = 9

Par exemple, le chiffre 19063 est représenté par l'arbre

$$\text{AddDig}(\text{AddDig}(\text{AddZ}(\text{AddDig}(\text{UseDig}(1'), 9')), 6'), 3') : \text{Pos}.$$

L'interface entre les chiffres et le symbolisme général est l'opérateur UseNum, qui construit une constante du type Const(N) à partir d'un chiffre,

Const $\rightarrow$ Num

n : Num
UseNum(n) : Const(N)
UseNum(n)* = $n^\star$
UseNum(n)$^\circ$ = n°

Nous concluons cette section par quelques opérateurs du symbolisme arithmétique.

Inf $\rightarrow$ + | − | $*$

$+_N$: Inf(N, N, N)	$-_N$: Inf(N, N, N)	$*_N$: Inf(N, N, N)
$+^\star_N$ = $+_N$	$-^\star_N$ = $-_N$	$*^\star_N$ = $*_N$
$+^\circ_N$ = $+$	$-^\circ_N$ = $-$	$*^\circ_N$ = $*$
prior($+_N$) = **3**	prior($-_N$) = **3**	prior($*_N$) = **6**

Prd $\rightarrow = | < | >$

$=_N$: Prd(N, N)	$<_N$: Prd(N, N)	$>_N$: Prd(N, N)
$=^\star_N$ = $I(N)$	$<^\star_N$ = inf	$>^\star_N$ = sup
$=^\circ_N$ = =	$<^\circ_N$ = <	$>^\circ_N$ = >

3.4 *L'interface entre le symbolisme et le français*

Le style mathématique classique n'emploie le symbolisme que pour les termes singuliers et pour quelques formules atomiques. Les opérateurs logiques sont exprimés en langue naturelle. Puisque le symbolisme que nous avons défini ne contient pas d'opérateurs logiques, notre grammaire est compatible avec ce style classique. L'adjonction des connecteurs et des quantificateurs ne produirait aucune difficulté, mais du point de vue linguistique, il est plus intéressant de voir comment le style classique peut exprimer toutes les structures logiques.

Le principe général de l'usage du symbolisme dans le texte est qu'une phrase peut inclure des parties symboliques, mais qu'une formule ne peut pas inclure des parties non-symboliques. Ainsi un terme (Trm) peut être utilisé comme nom propre, et une formule (Fml) comme phrase, mais l'autre direction n'est pas possible :

la somme de n et de n + 1

est acceptable mais

n + le successeur de n

ne l'est pas. Les deux règles d'interface sont ainsi[79]

PN → Trm

$$\frac{\Gamma : \text{cont} \quad A : \text{set}/\Gamma \quad a : \text{Trm}(\Gamma, A)}{\begin{array}{l}\text{UseTrm}(\Gamma, A, a) : \text{PN}(\Gamma, A) \\ \text{UseTrm}(\Gamma, A, a)^\star = a^\star \\ \text{UseTrm}(\Gamma, A, a)^\circ(r) = r\, a^\circ \\ \text{gen}(\text{UseTrm}(\Gamma, A, a)) = \text{natgenre}(A) \\ \text{num}(\text{UseTrm}(\Gamma, A, a)) = \text{sg}\end{array}}$$

P → Fml

$$\frac{\Gamma : \text{cont} \quad A : \text{Fml}(\Gamma)}{\begin{array}{l}\text{UseFml}(\Gamma, A) : \text{P}(\Gamma) \\ \text{UseFml}(\Gamma, A)^\star = A^\star \\ \text{UseFml}(\Gamma, A)^\circ(m) = A^\circ\end{array}}$$

L'usage d'un terme comme nom propre pose le problème de la définition du genre : est-ce qu'on dit

a et b sont distincts

ou bien

a et b sont distinctes ?

Il y a une autre construction d'interface, qui demande un nom commun pour produire un nom propre à partir d'un terme,

PN → *le* CN Trm

$$\frac{\Gamma : \text{cont} \quad A : \text{CN}(\Gamma) \quad a : \text{Trm}(\Gamma, A^\star)}{\begin{array}{l}\text{leTrm}(\Gamma, A, a) : \text{PN}(\Gamma, A^\star) \\ \text{leTrm}(\Gamma, A, a)^\star = a^\star \\ \text{leTrm}(\Gamma, A, a)^\circ(r) = \mathit{le}(r, \text{gen}(A), \text{sg}, A^\circ(\text{sg}))\, a^\circ \\ \text{gen}(\text{leTrm}(\Gamma, A, a)) = \text{gen}(A) \\ \text{num}(\text{leTrm}(\Gamma, A, a)) = \text{sg}\end{array}}$$

Ainsi on dit, évidemment,

le nombre a et le nombre b sont distincts,

la racine a et la racine b sont distinctes,

[79] Si les chaînes sont représentées par le code LaTeX, c'est dans ces règles-ci qu'on introduit les signe de « dollar » qui marquent le changement entre le mode normal et le mode mathématique de caractères : par exemple, $\text{UseFml}(\Gamma, A)^\circ(m) = \$A^\circ\$$.

même si les termes a et b sont identiques dans les deux phrases — grâce au nom commun qui les introduit. Une solution serait de remplacer l'argument A : set/Γ par A : CN(Γ), mais cela contrarierait le principe que les arguments syntaxiques ne sont jamais effacés (v. section 1.7). Nous définissons ainsi un **genre naturel** des domaines d'objets mathématiques,

$$\text{natgenre} : (\text{set})\text{genre}.$$

Par exemple, le genre naturel des nombres est le masculin, le genre des droites est le féminin.[80]

Maintenant, nous pouvons, enfin, construire l'arbre syntaxique d'un digit utilisé comme syntagme nominal : par exemple, le digit 2 est représenté par l'arbre

$$\text{Mont}(\text{UseTrm}(\text{UseConst}(\text{UseNum}(\text{UsePos}(\text{UseDig}(2')))))).$$

Nous espérons avoir mis en évidence pourquoi chacune de ces six étapes est nécessaire.[81]

Un syntagme nominal de la forme d'une conjonction de termes, telle que

$$1,\ 2\ \textit{et}\ 3,$$

distributive aussi bien que collective, peut être formé par la règle UseTrm en combinaison avec la montée et la coordination. Mais le syntagme

$$\textit{les nombres}\ 1,\ 2\ \textit{et}\ 3,$$

ne peut pas encore être formé. Il nous faut encore trois règles :

SN $\rightarrow$ *les* CN Trm ... *et* Trm

$$\frac{\Gamma : \text{cont} \quad A : \text{CN}(\Gamma) \quad l : \text{list}_2(\text{Trm}(\Gamma, A^\star))}{\begin{array}{l}\text{lesTrm}(\Gamma, A, l) : \text{SN}(\Gamma, A^\star) \\ \text{lesTrm}(\Gamma, A, l)^\star = (Y)\text{listconn}(\&, \text{map}_2((x)Y(x^\star), l)) \\ \text{lesTrm}(\Gamma, A, l)^\circ(r) = \textit{le}(r, \text{gen}(A), \text{pl}, A^\circ(\text{pl}))\ \text{conjform}(\textit{et}, \text{map}_2((x)x^\circ, l)) \\ \text{gen}(\text{lesTrm}(\Gamma, A, l)) = \text{gen}(A) \\ \text{num}(\text{lesTrm}(\Gamma, A, l)) = \text{pl}\end{array}}$$

PNC $\rightarrow$ *les* CN Trm *et* Trm

$$\frac{\Gamma : \text{cont} \quad A : \text{CN}(\Gamma) \quad a : \text{Trm}(\Gamma, A^\star) \quad b : \text{Trm}(\Gamma, A^\star)}{\begin{array}{l}\text{CoupleTrm}(\Gamma, A, a, b) : \text{PNC}(\Gamma, A^\star) \\ \text{CoupleTrm}(\Gamma, A, a, b)^\star = (x^\star, y^\star) \\ \text{CoupleTrm}(\Gamma, A, a, b)^\circ(r) = \textit{le}(r, \text{gen}(A), \text{pl}, A^\circ(\text{pl}))\ a^\circ\ \textit{et}\ b^\circ \\ \text{gen}(\text{CoupleTrm}(\Gamma, A, a, b)) = \text{gen}(A)\end{array}}$$

[80] En français, le genre est considéré comme un genre grammatical, c.-à-d. comme une propriété du nom ; cf. Grevisse (§ 454). Mais l'usage des pronoms et, dans la langue mathématique, des symboles, force parfois la détermination du genre sans l'aide d'un nom. Elle doit alors être effectuée sur la base du type sémantique de l'objet désigné. Bien qu'on pense, souvent, que cela n'est possible que pour les objets animés (Grevisse § 457), il y a une intuition assez forte des genres de quelques objets mathématiques. Par exemple, même si un nombre a est d'abord introduit par la phrase *soit a une racine du polynome* $P(x)$, le symbole a est bientôt employé comme un nom masculin dans le même texte.

[81] Mathématiquement, chacune des opérations UseDig, UsePos, UseNum, UseConst et UseTrm est une injection canonique, qui établit la catégorie de départ comme une sous-catégorie de la catégorie d'arrivée. Mont l'est aussi, mais dans un sens modifié, parce que l'interprétation de SN diffère de celle de PN ; cf. section 2.7.

SNC $\rightarrow$ *les* CN Trm, ... Trm *et* Trm

$$\frac{\Gamma : \text{cont} \quad A : \text{CN}(\Gamma) \quad l : \text{list}_3(\text{Trm}(\Gamma, A^\star))}{\begin{array}{l}\text{CollTrm}(\Gamma, A, l) : \text{SNC}(\Gamma, A^\star) \\ \text{CollTrm}(\Gamma, A, l)^\star = (Y)(\forall x, y \in l)Y(x^\star, y^\star) \\ \text{CollTrm}(\Gamma, A, l)^\circ(r) = le(r, \text{gen}(A), \text{pl}, A^\circ(\text{pl}))\ \text{conjform}(et, \text{map}_3((x)x^\circ, l)) \\ \text{gen}(\text{CollTrm}(\Gamma, A, l)) = \text{gen}(A)\end{array}}$$

3.5 *Les déclarations et les listes de déclarations*

Nous n'avons pas encore défini de constructions grammaticales qui étendent le contexte par des variables explicites. L'une des plus fréquentes est la **déclaration de variables**, par exemple,

soient m et n des nombres.

Le contexte créé par cette déclaration est le contexte originel étendu par les deux hypothèses

$$x_m : N,\ y_n : N.$$

Ainsi les symboles m et n sont utilisables comme des variables explicites du type N dans le contexte créé — par exemple, dans l'énoncé d'un théorème,

Soient m et n des nombres. Alors $m + n = n + m$.

Le même théorème peut aussi être exprimé par une implication,

si m et n sont des nombres, alors $m + n = n + m$,

où la déclaration reçoit une forme un peu différente: la copule est mise à l'indicatif, et n'apparaît qu'après les variables. Mais l'interprétation est la même. Enfin, dans l'énoncé

pour tous les nombres m et n, $m + n = n + m$,

la copule manque totalement.

Une déclaration peut introduire des variables avec une propriété, par exemple,

soit n un nombre pair,

qui étend le contexte par deux hypothèses, une explicite et une implicite,[82]

$$x_n : N,\ y : \text{Pair}(x).$$

La propriété est exprimée soit par un adjectif, soit par une proposition relative,

soit n un nombre qui est pair.

Elle peut aussi être une propriété collective des variables, exprimée par un adjectif collectif,

soient a, b, c, d et e des nombres distincts,

[82]Cette déclaration ne peut pas être conçue comme une déclaration d'une variable du type modifié $(\Sigma z : N)\text{Pair}(z)$, si le symbole n est ensuite employé comme une variable du type N. Ainsi *nombre pair* n'est pas analysé comme un nom commun modifié dans cette construction, mais comme deux constituants distincts, le nom *nombre* et l'adjectif *pair*.

ou par une construction avec *tel que*,

$$\textit{soient } a,\ b,\ c,\ d \textit{ et } e \textit{ des nombres tels que } a < b < c < d < e.$$

Dans tous ces cas, le contexte est étendu par quelques variables explicites et par quelques preuves implicites de la propriété.

Nous introduisons la catégorie des *déclarations*,

$$\text{Décl}(\Gamma),$$

où Γ : cont, interprétée

$$\text{cont}/\Gamma,$$

c.-à-d. comme une séquence d'hypothèses qui peuvent dépendre de Γ. La linéarisation a un paramètre, du type

$$\text{déclforme} = \{\mathrm{d}_1, \mathrm{d}_2, \mathrm{d}_3\},$$

qui choisit une des trois formes différentes par rapport à la copule.[83] Une déclaration a le genre et le nombre inhérents ; comme on le verra, ils sont déterminés par le genre du nom commun et par le nombre des variables.[84]

$$\text{Décl} \rightarrow \left\{ \begin{array}{l} \textit{soient} \text{ Var } \ldots \textit{ et} \text{ Var } \textit{des} \text{ CN} \\ \text{Var } \ldots \textit{ et} \text{ Var } \textit{sont des} \text{ CN} \\ \text{CN Var } \ldots \textit{ et} \text{ Var} \end{array} \right\}$$

$$\frac{\Gamma : \text{cont} \quad l : \text{list}_1(\text{symb}) \quad A : \text{CN}(\Gamma)}{\begin{array}{l} \text{DéclCN}(\Gamma, l, A) : \text{Décl}(\Gamma) \\ \text{DéclCN}(\Gamma, l, A)^\star = \overline{x}_l : A^\star \\ \text{DéclCN}(\Gamma, l, A)^\circ(\mathrm{d}_1) = \\ \quad \textit{être}(\text{num}(l), \text{subj})\ \text{etform}(l)\ \text{indéf}(-, \text{gen}(A), \text{num}(l))\ A^\circ(\text{num}(l)) \\ \text{DéclCN}(\Gamma, l, A)^\circ(\mathrm{d}_2) = \\ \quad \text{etform}(l)\ \textit{être}(\text{num}(l), \text{ind})\ \text{indéf}(-, \text{gen}(A), \text{num}(l))\ A^\circ(\text{num}(l)) \\ \text{DéclCN}(\Gamma, l, A)^\circ(\mathrm{d}_3) = A^\circ(\text{num}(l))\ \text{etform}(l) \\ \text{gen}(\text{DéclCN}(\Gamma, l, A)) = \text{gen}(A) \\ \text{num}(\text{DéclCN}(\Gamma, l, A)) = \text{num}(l) \end{array}}$$

$$\text{Décl} \rightarrow \left\{ \begin{array}{l} \textit{soient} \text{ Var } \ldots \textit{ et} \text{ Var } \textit{des} \text{ CN A1} \\ \text{Var } \ldots \textit{ et} \text{ Var } \textit{sont des} \text{ CN A1} \\ \text{CN A1 Var } \ldots \textit{ et} \text{ Var} \end{array} \right\}$$

$$\frac{\Gamma : \text{cont} \quad l : \text{list}_1(\text{symb}) \quad A : \text{CN}(\Gamma) \quad F : \text{A1}(\Gamma, A^\star)}{\begin{array}{l} \text{DéclA1}(\Gamma, l, A, F) : \text{Décl}(\Gamma) \\ \text{DéclA1}(\Gamma, l, A, F)^\star = \overline{x}_l : A^\star,\ y : (\forall x \in \overline{x}) F^\star(x) \\ \text{DéclA1}(\Gamma, l, A, F)^\circ(\mathrm{d}_1) = \textit{être}(\text{num}(l), \text{subj})\ \text{etform}(l) \\ \quad \text{indéf}(-, \text{gen}(A), \text{num}(l))\ A^\circ(\text{num}(l))\ F^\circ(\text{gen}(A), \text{num}(l)) \\ \text{DéclA1}(\Gamma, l, A, F)^\circ(\mathrm{d}_2) = \text{etform}(l)\ \textit{être}(\text{num}(l), \text{ind}) \\ \quad \text{indéf}(-, \text{gen}(A), \text{num}(l))\ A^\circ(\text{num}(l))\ F^\circ(\text{gen}(A), \text{num}(l)) \\ \text{DéclA1}(\Gamma, l, A, F)^\circ(\mathrm{d}_3) = A^\circ(\text{num}(l))\ F^\circ(\text{gen}(A), \text{num}(l))\ \text{etform}(l) \\ \text{gen}(\text{DéclA1}(\Gamma, l, A, F)) = \text{gen}(A) \\ \text{num}(\text{DéclA1}(\Gamma, l, A, F)) = \text{num}(l) \end{array}}$$

[83] Ainsi chacune des règles de construction des déclarations correspond à trois règles syntagmatiques. Une alternative serait un système avec trois catégories distinctes, et trois fois les règles que nous présentons.

[84] Nous supposons le nombre num(l) défini pour les listes de telle façon qu'il est sg si l a un élément, pl s'il en a plusieurs.

$$\text{Décl} \rightarrow \left\{ \begin{array}{l} \textit{soient} \text{ Var } \ldots \textit{ et} \text{ Var } \textit{des} \text{ CN PR} \\ \text{Var } \ldots \textit{ et} \text{ Var } \textit{sont des} \text{ CN PR} \\ \text{CN Var } \ldots \textit{ et} \text{ PR} \end{array} \right\}$$

$$\frac{\Gamma : \text{cont} \quad l : \text{list}_1(\text{symb}) \quad A : \text{CN}(\Gamma) \quad F : \text{PR}(\Gamma, A^\star)}{\begin{array}{l} \text{DéclPR}(\Gamma, l, A, F) : \text{Décl}(\Gamma) \\ \text{DéclPR}(\Gamma, l, A, F)^\star = \overline{x}_l : A^\star,\ y : (\forall x \in \overline{x}) F^\star(x) \\ \text{DéclPR}(\Gamma, l, A, F)^\circ(\text{d}_1) = \textit{être}(\text{num}(l), \text{subj})\ \text{etform}(l) \\ \quad \text{indéf}(-, \text{gen}(A), \text{num}(l))\ A^\circ(\text{num}(l))\ F^\circ(\text{gen}(A), \text{num}(l), \text{ind}) \\ \text{DéclPR}(\Gamma, l, A, F)^\circ(\text{d}_2) = \text{etform}(l)\ \textit{être}(\text{num}(l), \text{ind}) \\ \quad \text{indéf}(-, \text{gen}(A), \text{num}(l))\ A^\circ(\text{num}(l))\ F^\circ(\text{gen}(A), \text{num}(l), \text{ind}) \\ \text{DéclPR}(\Gamma, l, A, F)^\circ(\text{d}_3) = A^\circ(\text{num}(l))\ \text{etform}(l)\ F^\circ(\text{gen}(A), \text{num}(l), \text{ind}) \\ \text{gen}(\text{DéclPR}(\Gamma, l, A, F)) = \text{gen}(A) \\ \text{num}(\text{DéclPR}(\Gamma, l, A, F)) = \text{num}(l) \end{array}}$$

$$\text{Décl} \rightarrow \left\{ \begin{array}{l} \textit{soient} \text{ Var } \ldots \textit{ et} \text{ Var } \textit{des} \text{ CN AC} \\ \text{Var } \ldots \textit{ et} \text{ Var } \textit{sont des} \text{ CN AC} \\ \text{CN AC Var } \ldots \textit{ et} \text{ Var} \end{array} \right\}$$

$$\frac{\Gamma : \text{cont} \quad l : \text{list}_2(\text{symb}) \quad A : \text{CN}(\Gamma) \quad F : \text{AC}(\Gamma, A^\star, A^\star)}{\begin{array}{l} \text{DéclAC}(\Gamma, l, A, F) : \text{Décl}(\Gamma) \\ \text{DéclAC}(\Gamma, l, A, F)^\star = \overline{x}_l : A^\star,\ z : (\forall x, y \in \overline{x}) F^\star(x, y) \\ \text{DéclAC}(\Gamma, l, A, F)^\circ(\text{d}_1) = \textit{être}(\text{pl}, \text{subj})\ \text{etform}(l) \\ \quad \text{indéf}(-, \text{gen}(A), \text{pl})\ A^\circ(\text{pl})\ F^\circ(\text{gen}(A)) \\ \text{DéclAC}(\Gamma, l, A, F)^\circ(\text{d}_2) = \text{etform}(l)\ \textit{être}(\text{pl}, \text{ind}) \\ \quad \text{indéf}(-, \text{gen}(A), \text{pl})\ A^\circ(\text{pl})\ F^\circ(\text{gen}(A)) \\ \text{DéclAC}(\Gamma, l, A, F)^\circ(\text{d}_3) = A^\circ(\text{pl})\ F^\circ(\text{gen}(A))\ \text{etform}(l) \\ \text{gen}(\text{DéclAC}(\Gamma, l, A, F)) = \text{gen}(A) \\ \text{num}(\text{DéclAC}(\Gamma, l, A, F)) = \text{pl} \end{array}}$$

$$\text{Décl} \rightarrow \left\{ \begin{array}{l} \textit{soient} \text{ Var } \ldots \textit{ et} \text{ Var } \textit{des} \text{ CN } \textit{tels que} \text{ P} \\ \text{Var } \ldots \textit{ et} \text{ Var } \textit{sont des} \text{ CN} \textit{tels que} \text{ P} \\ \text{CN Var } \ldots \textit{ et} \text{ Var } \textit{tels que} \text{ P} \end{array} \right\}$$

$$\frac{\Gamma : \text{cont} \quad l : \text{list}_1(\text{symb}) \quad A : \text{CN}(\Gamma) \quad B : \text{P}(\Gamma + \overline{x}_l : A^\star)}{\begin{array}{l} \text{DéclP}(\Gamma, l, A, B) : \text{Décl}(\Gamma) \\ \text{DéclP}(\Gamma, l, A, B)^\star = \overline{x}_l : A^\star,\ y : B^\star \\ \text{DéclP}(\Gamma, l, A, B)^\circ(\text{d}_1) = \textit{être}(\text{num}(l), \text{subj})\ \text{etform}(l) \\ \quad \text{indéf}(-, \text{gen}(A), \text{num}(l))\ A^\circ(\text{num}(l))\ \textit{tel}(\text{gen}(A), \text{num}(l))\ \textit{que}(B^\circ(\text{ind})) \\ \text{DéclP}(\Gamma, l, A, B)^\circ(\text{d}_2) = \text{etform}(l)\ \textit{être}(\text{num}(l), \text{ind}) \\ \quad \text{indéf}(-, \text{gen}(A), \text{num}(l))\ A^\circ(\text{num}(l))\ \textit{tel}(\text{gen}(A), \text{num}(l))\ \textit{que}(B^\circ(\text{ind})) \\ \text{DéclP}(\Gamma, l, A, B)^\circ(\text{d}_3) = A^\circ(\text{num}(l))\ \text{etform}(l)\ \textit{tel}(\text{gen}(A), \text{num}(l))\ \textit{que}(B^\circ(\text{ind})) \\ \text{gen}(\text{DéclP}(\Gamma, l, A, B)) = \text{gen}(A) \\ \text{num}(\text{DéclP}(\Gamma, l, A, B)) = \text{num}(l) \end{array}}$$

Plusieurs déclarations peuvent être combinées dans une **liste de déclarations**,

$$\text{LD}(\Gamma),$$

interprétée cont/Γ. Dans l'interprétation, les nouvelles hypothèses sont ajoutées aux anciennes. À côté des déclarations, une telle liste peut contenir des phrases :

Soient m et n des nombres. Supposons que $m < n$. *Alors* $m + 1 < n + 1$.

Si m et n sont des nombres et $m < n$, *alors* $m + 1 < n + 1$.

De telles combinaisons ne sont pas possibles pour la forme *pour tous les nombres m et n* — cette forme n'accepte que les déclarations singulières, pas de listes. Donc nous ne définissons la linéarisation des listes de déclarations que pour les paramètres d_1 et d_2. Pour la même raison, nous n'avons besoin de définir ni le genre ni le nombre des listes de déclarations. Le produit de la linéarisation de LD est une liste de chaînes, qui peut être transformée en une chaîne de la forme d'un texte, comme dans le premier exemple ci-dessus, ou bien d'une conjonction, comme dans le second.

Il y a quatre règles pour la catégorie LD, deux règles de base et deux d'extension.[85]

LD → Décl

$$\frac{\Gamma : \mathrm{cont} \quad D : \mathrm{Décl}(\Gamma)}{\begin{array}{l}\mathrm{PremDécl}(\Gamma, D) : \mathrm{LD}(\Gamma) \\ \mathrm{PremDécl}(\Gamma, D)^\star = D^\star \\ \mathrm{PremDécl}(\Gamma, D)^\circ(d) = \mathrm{bs}_1(D^\circ(d))\end{array}}$$

LD → P

$$\frac{\Gamma : \mathrm{cont} \quad A : \mathrm{P}(\Gamma)}{\begin{array}{l}\mathrm{PremP}(\Gamma, A) : \mathrm{LD}(\Gamma) \\ \mathrm{PremP}(\Gamma, A)^\star = x : A^\star \\ \mathrm{PremP}(\Gamma, A)^\circ(\mathrm{d}_1) = \mathrm{bs}_1(\textit{supposons que}(A^\circ(\mathrm{ind}))) \\ \mathrm{PremP}(\Gamma, A)^\circ(\mathrm{d}_2) = \mathrm{bs}_1(A^\circ(\mathrm{ind}))\end{array}}$$

LD → LD Décl

$$\frac{\Gamma : \mathrm{cont} \quad L : \mathrm{LD}(\Gamma) \quad D : \mathrm{Décl}(\Gamma + L^\star)}{\begin{array}{l}\mathrm{AddDécl}(\Gamma, L, D) : \mathrm{LD}(\Gamma) \\ \mathrm{AddDécl}(\Gamma, L, D)^\star = L^\star + D^\star \\ \mathrm{AddDécl}(\Gamma, L, D)^\circ(d) = \mathrm{cons}_1(L^\circ(d), D^\circ(d))\end{array}}$$

LD → LD P

$$\frac{\Gamma : \mathrm{cont} \quad L : \mathrm{LD}(\Gamma) \quad A : \mathrm{P}(\Gamma + L^\star)}{\begin{array}{l}\mathrm{AddP}(\Gamma, L, A) : \mathrm{LD}(\Gamma) \\ \mathrm{AddP}(\Gamma, L, A)^\star = L^\star, x : A^\star \\ \mathrm{AddP}(\Gamma, L, A)^\circ(\mathrm{d}_1) = \mathrm{cons}_1(L^\circ(\mathrm{d}_1), \textit{supposons que}(A^\circ(\mathrm{ind}))) \\ \mathrm{AddP}(\Gamma, L, A)^\circ(\mathrm{d}_2) = \mathrm{cons}_1(L^\circ(\mathrm{d}_2), A^\circ(\mathrm{ind}))\end{array}}$$

3.6 *Les formes de théorème*

Les trois formes de déclaration sont utilisées dans trois formes de théorème, illustrées par les énoncés

[85] Si on admettait la liste vide, on n'aurait besoin que de trois règles. Mais il serait bizarre de trouver une liste vide de déclarations devant n'importe quelle phrase.

Soient m et n des nombres. Alors $m + n = n + m$.

Si m et n sont des nombres, alors $m + n = n + m$.

Pour tous les nombres m et n, $m + n = n + m$.

La première forme est celle d'un texte, qui consiste en plusieurs phrases: chaque déclaration et la conclusion commencent par une majuscule et se terminent par un point. Donc cet énoncé ne peut pas être employé comme partie d'une phrase, et il nous faut introduire une nouvelle catégorie,

$$\mathrm{Thm}(\Gamma),$$

des **textes de théorème** dans le contexte Γ, interprétée

$$\mathrm{prop}/\Gamma.$$

Le mode d'un texte de théorème est toujours l'indicatif, donc il n'y a pas de paramètres de linéarisation. Un texte de théorème est formé d'une liste de déclarations et d'une phrase dans le contexte étendu par ces déclarations. Il est interprété par la quantification universelle sur toutes les hypothèses de l'extension, ce qui résulte dans une proposition dans le contexte originel. La linéarisation choisit la première alternative de la linéarisation de la liste de déclarations et produit un texte.

Thm $\rightarrow$ LD. P.

$$\frac{\Gamma : \mathrm{cont} \quad L : \mathrm{LD}(\Gamma) \quad A : \mathrm{P}(\Gamma + L^\star)}{\begin{array}{l}\mathrm{TextThm}(\Gamma, L, A) : \mathrm{Thm}(\Gamma) \\ \mathrm{TextThm}(\Gamma, L, A)^\star = \forall(L^\star, A^\star) \\ \mathrm{TextThm}(\Gamma, L, A)^\circ = \mathrm{textform}(L^\circ(\mathrm{d}_1))\ \textit{Alors}\ A^\circ(\mathrm{ind}).\end{array}}$$

Notre exemple résulte donc de l'arbre

$$\begin{array}{l}\mathrm{TextThm}(\mathrm{PremDecl}(\mathrm{DeclCN}([m,n], \mathrm{nombre})), \\ \qquad \mathrm{UseFml}(\mathrm{PredPrd}(\mathrm{ApplInf}(\mathrm{UseVar}(1(m)), +_N, \mathrm{UseVar}(0(n))), =_N, \\ \qquad\qquad \mathrm{ApplInf}(\mathrm{UseVar}(0(n)), +_N, \mathrm{UseVar}(1(m))))))\end{array}$$

dont l'interprétation est

$$(\forall x : N)(\forall y : N)(x + y = y + x).$$

La seconde forme de théorème est une phrase, utilisable comme partie d'autres phrases. Elle est une implication, où la liste de déclarations s'est linéarisée en une suite de phrases séparées par des virgules, la dernière paire par la conjonction *et*. Son interprétation est la même que celle d'un texte de théorème.

P $\rightarrow$ *si* LD, *alors* P

$$\frac{\Gamma : \mathrm{cont} \quad L : \mathrm{LD}(\Gamma) \quad A : \mathrm{P}(\Gamma + L^\star)}{\begin{array}{l}\text{si-alors}(\Gamma, L, A) : \mathrm{P}(\Gamma) \\ \text{si-alors}(\Gamma, L, A)^\star = \forall(L^\star, A^\star) \\ \text{si-alors}(\Gamma, L, A)^\circ(m) = \textit{si}(\mathrm{etform}(L^\circ(\mathrm{d}_2))),\ \textit{alors}\ A^\circ(m)\end{array}}$$

Ainsi notre exemple s'obtient comme la linéarisation, à l'indicatif, de l'arbre

si-alors(PremDecl(DeclCN([m, n], nombre)),
UseFml(PredPrd(ApplInf(UseVar(1(m)), $+_N$, UseVar(0(n))), $=_N$,
ApplInf(UseVar(0(n)), $+_N$, UseVar(1(m))))))

dont l'interprétation est la même que celle du premier exemple.

La troisième forme de théorème est la **quantification explicite**, ou les variables d'une déclaration sont liées par un déterminant. Notre exemple est formé par la règle

P $\rightarrow$ *pour tous les* Décl, P[86]

$$\frac{\Gamma : \text{cont} \quad D : \text{Décl}(\Gamma) \quad A : \text{P}(\Gamma + D^\star)}{\begin{array}{l}\text{pour-tout}(\Gamma, D, A) : \text{P}(\Gamma) \\ \text{pour-tout}(\Gamma, D, A)^\star = \forall(D^\star, A^\star) \\ \text{pour-tout}(\Gamma, D, A)^\circ(m) = \\ \quad \textit{pour tout}(\text{gen}(D), \text{num}(D))\ \text{cas}(\text{num}(D) = \text{sg}, \textit{, les})\ D^\circ(\text{d}_3),\ A^\circ(m)\end{array}}$$

comme l'arbre

pour-tout(DeclCN([m, n], nombre),
UseFml(PredPrd(ApplInf(UseVar(1(m)), $+_N$, UseVar(0(n))), $=_N$,
ApplInf(UseVar(0(n)), $+_N$, UseVar(1(m))))))

qui a la même interprétation que les exemples ci-dessus. Une autre expression typique pour la quantification universelle est

P $\rightarrow$ *quels que soient les* Décl, P

$$\frac{\Gamma : \text{cont} \quad D : \text{Décl}(\Gamma) \quad A : \text{P}(\Gamma + D^\star)}{\begin{array}{l}\text{quel-que-soit}(\Gamma, D, A) : \text{P}(\Gamma) \\ \text{quel-que-soit}(\Gamma, D, A)^\star = \forall(D^\star, A^\star) \\ \text{quel-que-soit}(\Gamma, D, A)^\circ(m) = \textit{quel}(\text{gen}(D), \text{num}(D)) \\ \quad \textit{que}(\textit{être}(\text{num}(D), \text{subj}))\ \textit{le}(-, \text{gen}(D), \text{num}(D), D^\circ(\text{d}_3)),\ A^\circ(m)\end{array}}$$

Le mode de la phrase qui suit le quantificateur peut dépendre de celui-ci. Donc un théorème existentiel peut être exprimé par la forme

P $\rightarrow$ *il existe des* Décl *tels que* P

$$\frac{\Gamma : \text{cont} \quad D : \text{Décl}(\Gamma) \quad A : \text{P}(\Gamma + D^\star)}{\begin{array}{l}\text{il-existe}(\Gamma, D, A) : \text{P}(\Gamma) \\ \text{il-existe}(\Gamma, D, A)^\star = \exists(D^\star, A^\star) \\ \text{il-existe}(\Gamma, D, A)^\circ(m) = \\ \quad \textit{il existe}\ \text{indéf}(-, \text{gen}(D), \text{num}(D))\ D^\circ(\text{d}_3)\ \textit{tel}(\text{gen}(D), \text{num}(D))\ \textit{que}(A^\circ(\text{subj}))\end{array}}$$

qui demande le subjonctif,[87] tandis que la forme universelle correspondante,

P $\rightarrow$ *tous les* Décl *sont tels que* P,

[86] L'article défini est omis dans la forme singulière. Donc on engendre *pour tout nombre n*, *pour tous les nombres m et n*. Pour produire aussi *pour tous les nombres n*, il faudrait modifier les règles de linéarisation des déclarations, parce que, maintenant, elles imposent le singulier du nom commun si la liste de variables n'a qu'un élément.

[87] Cette règle n'empêche pas qu'on puisse trouver des exemples à l'indicatif: ce qui justifie cette règle grammaticale c'est que le subjonctif est toujours possible.

demande l'indicatif. L'introduction d'une catégorie spéciale des quantificateurs explicites forcerait une définition du mode inhérent des déterminants. Au lieu de faire cela, nous avons choisi d'analyser les quantificateurs explicites simplement comme des formes de théorème.

3.7 *Les textes de preuve : les règles de la déduction naturelle*

Il est bien évident pour le linguiste qu'une phrase n'est pas simplement une séquence de mots, mais une structure ordonnée, un arbre. L'étude grammaticale des textes montre, de la même façon, qu'un texte n'est pas simplement une séquence de phrases, mais une structure ordonnée. Nous avons déjà considéré un exemple, le texte de théorème, qui est une structure correspondant à la quantification dans la théorie des types, où chaque phrase joue un rôle spécial, soit celui d'une hypothèse, soit celui du conséquent. En outre, chaque phrase d'un texte de théorème peut dépendre des phrases antérieures.

Un type de textes où la structure arborescente est très évidente est celui des **textes de preuve**, ou de **démonstration**, qui correspondent à des arbres de déduction. Chaque phrase occupe un nœud dans cet arbre, et fonctionne comme prémisse ou comme conclusion, ou bien dans les deux rôles.

La catégorie des textes de preuve,

$$\mathrm{Preuve}(\Gamma, A),$$

où Γ : cont, A : prop/Γ, est interprétée

$$A/\Gamma.$$

En d'autres termes, un texte de preuve est interprété comme un objet-preuve, dans le sens de la théorie constructive des types.

Nous présentons un système de règles où chacune des règles de la **déduction naturelle** de Gentzen (1934) est modifiée en une règle textuelle, qui combine des textes prouvant les prémisses pour produire une preuve de la conclusion. Toutes les règles permettent la variation d'expression. Par exemple, dans l'introduction de la conjonction, la conclusion a la forme

$$A^{\circ}(\mathrm{ind}) \quad \text{et} \quad B^{\circ}(\mathrm{ind}).$$

Comme prémisses, la règle demande une preuve de la proposition $A^{\star}$ et une preuve de la proposition $B^{\star}$, mais rien ne demande que ces propositions soient exprimées par les mêmes phrases A et B. Donc le passage

> ... 2 est premier... 3 est premier. En somme, 2 est premier et le successeur de 2 est premier.

est une instance valide de l'introduction de la conjonction. Cette possibilité est le fondement de la variation stylistique dans un texte de preuve.

Les linéarisations de nos arbres de preuve sont inspirées par le programme de Coscoy *et al.* (1995), qui traduit les λ-termes en texte.[88] Bien entendu, les règles engendrent aussi la ponctuation du texte, selon le principe que le point n'est mis que quand la preuve est employée comme partie d'une structure plus grande. Le soulignement indique les passages commençant par une majuscule. Les caractères romains sont utilisés au lieu des italiques.

[88] La structure de leur programme est minimaliste, au sens de la section 1.1, en tant qu'il ne connaît que les chaînes et les λ-termes — il n'y a pas d'arbres syntaxiques.

L'introduction de la conjonction. $\dfrac{A \quad B}{A \,\&\, B}$

$$\frac{\Gamma : \text{cont} \quad A, B : \text{P}(\Gamma) \quad a : \text{Preuve}(\Gamma, A^\star) \quad b : \text{Preuve}(\Gamma, B^\star)}{\begin{array}{l}\text{ConjI}(\Gamma, A, B, a, b) : \text{Preuve}(\Gamma, A^\star \& B^\star) \\ \text{ConjI}(\Gamma, A, B, a, b)^\star = (a^\star, b^\star) \\ \text{ConjI}(\Gamma, A, B, a, b)^\circ = a^\circ. \quad \underline{b^\circ}. \quad \text{En somme, } A^\circ(\text{ind}) \text{ et } B^\circ(\text{ind})\end{array}}$$

L'élimination de la conjonction.[89] $\dfrac{A \,\&\, B}{A} \qquad \dfrac{A \,\&\, B}{B}$

$$\frac{\Gamma : \text{cont} \quad A : \text{P}(\Gamma) \quad B : \text{prop}/\Gamma \quad c : \text{Preuve}(\Gamma, A^\star \& B)}{\begin{array}{l}\text{ConjEg}(\Gamma, A, B, c) : \text{Preuve}(\Gamma, A^\star) \\ \text{ConjEg}(\Gamma, A, B, c)^\star = p(c^\star) \\ \text{ConjEg}(\Gamma, A, B, c)^\circ = c^\circ. \text{ A fortiori, } A^\circ(\text{ind})\end{array}}$$

$$\frac{\Gamma : \text{cont} \quad A : \text{prop}/\Gamma \quad B : \text{P}(\Gamma) \quad c : \text{Preuve}(\Gamma, A \& B^\star)}{\begin{array}{l}\text{ConjEd}(\Gamma, A, B, c) : \text{Preuve}(\Gamma, B^\star) \\ \text{ConjEd}(\Gamma, A, B, c)^\star = q(c^\star) \\ \text{ConjEd}(\Gamma, A, B, c)^\circ = c^\circ. \text{ A fortiori, } B^\circ(\text{ind})\end{array}}$$

L'introduction de l'implication progressive.[90] $\dfrac{\begin{array}{c}(x : A) \\ B(x)\end{array}}{(\Pi x : A)B(x)}$

$$\frac{\Gamma : \text{cont} \quad A : \text{P}(\Gamma) \quad B : \text{prop}/(\Gamma,\ x : A^\star) \quad t : \text{symb} \quad b : \text{Preuve}((\Gamma,\ x_t : A^\star), B)}{\begin{array}{l}\text{ImplI}(\Gamma, A, B, t, b) : \text{Preuve}(\Gamma, (\Pi x : A^\star)B) \\ \text{ImplI}(\Gamma, A, B, t, b)^\star = (\lambda x)b^\star \\ \text{ImplI}(\Gamma, A, B, t, b)^\circ = \text{supposons que} \\ \qquad\qquad (t)\ A^\circ(\text{ind}). \\ \qquad \underline{b^\circ}\end{array}}$$

L'élimination de l'implication progressive. $\dfrac{(\Pi x : A)B(x) \quad a : A}{B(a)}$

$$\frac{\begin{array}{l}\Gamma : \text{cont} \quad A : \text{prop}/\Gamma \quad B : \text{P}((\Gamma,\ x : A)) \\ c : \text{Preuve}(\Gamma, (\Pi x : A)B^\star) \quad a : \text{Preuve}(\Gamma, A)\end{array}}{\begin{array}{l}\text{ImplE}(\Gamma, A, B, c, a) : \text{Preuve}(\Gamma, B^\star(x = a^\star)) \\ \text{ImplE}(\Gamma, A, B, c, a)^\star = \text{ap}(c^\star, a^\star) \\ \text{ImplE}(\Gamma, A, B, c, a)^\circ = c^\circ. \quad \underline{a^\circ}. \quad \text{Nous concluons} \\ \qquad\qquad\qquad\qquad que(B^\circ(\text{ind}))\end{array}}$$

[89] Le terme de la conjonction qui n'est pas visible dans la linéarisation est typé comme une proposition et non comme une phrase, en conséquence du principe sur les arguments cachés ; v. section 1.7.

[90] L'extension du contexte est explicite : l'hypothèse est marquée par une étiquette, qui peut être employée plus tard pour faire une référence à l'hypothèse (cf. la section suivante). Logiquement, une telle étiquette est un objet-preuve variable. Puisque la conclusion n'est pas énoncée, le conséquent de l'implication est exprimé par un argument sémantique.

L'introduction de la disjonction. $\dfrac{A}{A \vee B} \qquad \dfrac{B}{A \vee B}$

Γ : cont A, B : P(Γ) a : Preuve($\Gamma, A^\star$)

DisjIg(Γ, A, B, a) : Preuve($\Gamma, A^\star \vee B^\star$)
DisjIg(Γ, A, B, a)* = $i(a^\star)$
DisjIg(Γ, A, B, a)$^\circ$ = a°. A fortiori, A°(ind) ou B°(ind)

Γ : cont A, B : P(Γ) b : Preuve($\Gamma, B^\star$)

DisjIr(Γ, A, B, b) : Preuve($\Gamma, A^\star \vee B^\star$)
DisjIr(Γ, A, B, b)* = $j(b^\star)$
DisjIr(Γ, A, B, b)$^\circ$ = b°. A fortiori, A°(ind) ou B°(ind)

L'élimination de la disjonction. $\dfrac{A \vee B \quad \begin{matrix}(A)\\ C\end{matrix} \quad \begin{matrix}(B)\\ C\end{matrix}}{C}$

Γ : cont A, B : P(Γ) C : P(Γ)
c : Preuve($\Gamma, A^\star \vee B^\star$)
t : symb d : Preuve($(\Gamma,\ x_t : A^\star), C^\star$)
u : symb e : Preuve($(\Gamma,\ y_u : B^\star), C^\star$)

DisjE($\Gamma, A, B, C, c, t, d, u, e$) : Preuve($\Gamma, C^\star$)
DisjE($\Gamma, A, B, C, c, t, d, u, e$)* = $D(c^\star, (x)d^\star, (y)e^\star)$
DisjE($\Gamma, A, B, C, c, t, d, u, e$)$^\circ$ = c°. Montrons *que*(C°(ind)), en distinguant deux cas. Dans le premier cas, supposons que

(t) A°(ind).

$\underline{d^\circ}$. Dans le second cas, supposons que

(u) B°(ind).

$\underline{\epsilon^\circ}$

L'élimination de l'absurdité. $\dfrac{\bot}{C}$

Γ : cont C : P(Γ) c : Preuve($\Gamma, \bot$)

AbsE(Γ, C, c) : Preuve($\Gamma, C^\star$)
AbsE(Γ, C, c)* = $R_0(c^\star)$
AbsE(Γ, C, c)$^\circ$ = c°. Donc C°(ind)

L'introduction du quantificateur universel. $\dfrac{\begin{matrix}(x : A)\\ B(x)\end{matrix}}{(\forall x : A)B(x)}$

$$\frac{\begin{array}{l}\Gamma : \text{cont} \quad A : \text{CN}(\Gamma) \quad t : \text{symb} \quad B : \text{prop}/(\Gamma,\ x : A^\star) \\ b : \text{Preuve}((\Gamma,\ x_t : A^\star), B)\end{array}}{\begin{array}{l}\text{UnivI}(\Gamma, A, t, B, b) : \text{Preuve}(\Gamma, (\forall x : A^\star)B) \\ \text{UnivI}(\Gamma, A, t, B, b)^\star \ = \ (\lambda x)b^\star \\ \text{UnivI}(\Gamma, A, t, B, b)^\circ = \ \text{soit } t\ \mathit{un}(\text{gen}(A))\ A^\circ(\text{sg}).\ \underline{b^\circ}\end{array}}$$

L'élimination du quantificateur universel.[91] [92]

$$\frac{(\forall x : A)B(x) \quad a : A}{B(a)}$$

$$\frac{\begin{array}{l}\Gamma : \text{cont} \quad A : \text{set}/\Gamma \quad t : \text{symb} \\ B : \text{P}((\Gamma,\ x_t : A)) \quad c : \text{Preuve}(\Gamma, (\forall x : A)B^\star) \\ a : \text{Trm}(\Gamma, A)\end{array}}{\begin{array}{l}\text{UnivE}(\Gamma, A, t, B, c, a) : \text{Preuve}(\Gamma, B^\star(x = a^\star)) \\ \text{UnivE}(\Gamma, A, t, B, c, a)^\star \ = \ \text{ap}(c^\star, a^\star) \\ \text{UnivE}(\Gamma, A, t, B, c, a)^\circ = \ c^\circ.\ \text{En particulier, } B[a/x]^\circ(\text{ind})\end{array}}$$

L'introduction du quantificateur existentiel.

$$\frac{a : A \quad B(a)}{(\exists x : A)B(x)}$$

$$\frac{\begin{array}{l}\Gamma : \text{cont} \quad A : \text{CN}(\Gamma) \quad t : \text{symb} \quad B : \text{P}((\Gamma,\ x_t : A^\star)) \\ a : A^\star/\Gamma \quad b : \text{Preuve}(\Gamma, B^\star(x = a))\end{array}}{\begin{array}{l}\text{ExistI}(\Gamma, A, t, B, a, b) : \text{Preuve}(\Gamma, (\exists x : A^\star)B^\star) \\ \text{ExistI}(\Gamma, A, t, B, a, b)^\star \ = \ (a, b^\star) \\ \text{ExistI}(\Gamma, A, t, B, a, b)^\circ = \ b^\circ.\ \text{Donc il existe } \mathit{un}(\text{gen}(A))\ A^\circ(\text{sg}) \\ \qquad\qquad t\ \mathit{tel}(\text{gen}(A))\ \mathit{que}(B^\circ(\text{subj}))\end{array}}$$

L'élimination du quantificateur existentiel.

$$\frac{(\exists x : A)B(x) \qquad \begin{array}{c}(x : A, B(x)) \\ C\end{array}}{C}$$

$$\frac{\begin{array}{l}\Gamma : \text{cont} \quad A : \text{CN}(\Gamma) \quad t : \text{symb} \quad B : \text{P}((\Gamma,\ x_t : A^\star)) \\ C : \text{P}(\Gamma) \quad c : \text{Preuve}(\Gamma, (\exists x : A^\star)B^\star) \\ u : \text{symb} \quad d : \text{Preuve}(((\Gamma,\ x_t : A^\star),\ y_u : B^\star), C^\star)\end{array}}{\begin{array}{l}\text{ExistE}(\Gamma, A, t, B, C, c, u, d) : \text{Preuve}(\Gamma, C^\star) \\ \text{ExistE}(\Gamma, A, t, B, C, c, u, d)^\star \ = \ E(c^\star, (x, y)d^\star) \\ \text{ExistE}(\Gamma, A, t, B, C, c, u, d)^\circ = \ c^\circ.\ \text{Soit } t\ \mathit{un}(\text{gen}(A))\ A^\circ(\text{ind}) \\ \qquad\qquad \mathit{tel}(\text{gen}(A))\ \text{que} \\ \qquad\qquad\qquad (u)\ B^\circ(\text{ind}). \\ \qquad\qquad \underline{d^\circ}.\ \text{Donc } C^\circ(\text{ind})\end{array}}$$

[91]Nous n'avons pas défini la substitution dans les arbres syntaxiques, désignée par $[a/x]$. La définition est inductive sur la structure des arbres.

[92]Dans cette règle, on attendrait peut-être que a soit du type PN, un nom propre quelconque. Mais puisque la variable t peut se trouver dans une formule, la substitution produirait des résultats inacceptables. Dans une règle plus générale, avec a : PN(Γ, A), la conclusion devrait être exprimée par une substitution explicite : *En particulier,* B°*(ind)* a°(pour).

Il y a aussi des règles spécifiquement mathématiques, telles que la règle de l'induction sur les nombres naturels :

$$\frac{C(0) \qquad \begin{matrix}(x : N, C(x)) \\ C(s(x))\end{matrix}}{(\forall x : N)C(x)}$$

$$\frac{\begin{matrix}\Gamma : \text{cont} \quad t : \text{symb} \quad C : \text{P}((\Gamma,\ x_t : N)) \\ d : \text{Preuve}(\Gamma, C^\star(x = 0)) \\ u : \text{symb} \quad e : \text{Preuve}(((\Gamma,\ x_t : N),\ y_u : C^\star), C^\star(x = s(x)))\end{matrix}}{\begin{matrix}\text{Ind}N(\Gamma, t, C, d, u, e) : \text{Preuve}(\Gamma, (\forall x : N)C^\star) \\ \text{Ind}N(\Gamma, t, C, d, u, e)^\star = (\lambda x)R(x, d^\star, (x, y)e^\star) \\ \text{Ind}N(\Gamma, t, C, d, u, e)^\circ = \text{montrons que pour tout nombre } t,\ C^\circ(\text{ind}),\ \text{par induction. Pour la base, rappelons } que(d^\circ).\ \text{Pour l'hypothèse d'induction, considérons un nombre } t \text{ tel que} \\ \qquad\qquad (u)\ C^\circ(\text{ind}). \\ \underline{e^\circ}\end{matrix}}$$

3.8 *Les textes de preuve : règles générales*

Le système de règles textuelles de la déduction naturelle que nous venons de présenter est une démonstration constructive de la complétude de notre fragment du français, montrant que chaque preuve dans le calcul des prédicats peut être exprimée par un texte.[93] Mais ces règles excluent encore beaucoup de formulations qu'on rencontre dans les textes mathématiques. Sans doute, peu de textes réels suivent la déduction naturelle pas à pas : on fait, plutôt, des sauts assez longs, en comprimant plusieurs pas en un, ou même en supprimant des pas.

La suppression est grammaticalement très simple : quand l'auteur dit qu'il y a une preuve « évidente » ou « triviale », il connaît une preuve, ne la montre pas, mais continue comme s'il l'avait donnée. La preuve doit donc être représentée par un argument sémantique.

$$\frac{\Gamma : \text{cont} \quad A : \text{P}(\Gamma) \quad c : A^\star/\Gamma}{\begin{matrix}\text{évidemment}(\Gamma, A, c) : \text{Preuve}(\Gamma, A^\star) \\ \text{évidemment}(\Gamma, A, c)^\star = c \\ \text{évidemment}(\Gamma, A, c)^\circ = \text{évidemment}, A^\circ(\text{ind})\end{matrix}}$$

Cette règle pose une difficulté à l'analyse automatique des textes : puisqu'il n'y a pas de limite de complexité de la preuve c supprimée, le problème de trouver c peut être aussi difficile que le problème de prouver un théorème mathématique quelconque.

En général, ce qui empêche les démonstrations de croître énormément quand les théorèmes prouvés deviennent plus compliqués est l'usage de résultats déjà prouvés. Dans une démonstration, on dit simplement,

par le théorème 4,

[93] La complétude est relative à la possibilité d'exprimer toutes les propositions par des phrases françaises. Cette question est posée dans la section 4.4.

en s'appuyant sur un passage qui figure plus haut dans le texte,

> Théorème 4. Soit f une fonction croissante. Supposons que pour tout point x, f est dérivable en x. Alors, pour tout point x, $f'(x) \geq 0$.
>
> Démonstration. Soit f une fonction réelle...

Pour formuler une règle qui permet l'usage des résultats donnés, il faut d'abord une nouvelle catégorie, celle des **étiquettes de preuve**,

$$\text{Étiq}(\Gamma, A),$$

avec la même interprétation que Preuve(Γ, A). Une telle étiquette est un terme singulier qui désigne un objet-preuve: par exemple, l'expression *le théorème 4* peut fonctionner comme une étiquette de la preuve donnée sous le titre *le théorème 4*. Ce qui est appliqué quand on «applique le théorème 4 » n'est pas la proposition mais sa preuve. La règle d'usage d'une étiquette est

$$\frac{\Gamma : \text{cont} \quad A : \text{P}(\Gamma) \quad c : \text{Étiq}(\Gamma, A^\star)}{\begin{array}{l}\text{UseÉtiq}(\Gamma, A, c) : \text{Preuve}(\Gamma, A^\star) \\ \text{UseÉtiq}(\Gamma, A, c)^\star = c^\star \\ \text{UseÉtiq}(\Gamma, A, c)^\circ = \text{par } c^\circ,\ A^\circ(\text{ind})\end{array}}$$

Un exemple d'étiquette, autre que le nom d'un théorème, est l'hypothèse marquée par une variable, de la façon que nous avons vue dans la règle d'introduction de l'implication dans la section précédente. Si une preuve est donnée par une variable explicite dans le contexte, cette variable peut être employée comme étiquette:

$$\frac{\Gamma : \text{cont} \quad A : \text{prop}/\Gamma \quad v : \text{Var}(\Gamma, A)}{\begin{array}{l}\text{hypothèse}(\Gamma, A, v) : \text{Étiq}(\Gamma, A) \\ \text{hypothèse}(\Gamma, A, v)^\star = v^\star \\ \text{hypothèse}(\Gamma, A, v)^\circ = \text{l'hypothèse } v^\circ\end{array}}$$

Une forme fréquente d'une règle comprimée est l'application simultanée d'un théorème à toutes les prémisses qu'il demande. En suivant nos règles pas à pas, on pourrait produire le texte

> Par le théorème 4, si f est une fonction croissante et pour tout point x, f est dérivable en x, alors, pour tout point x, $f'(x) \geq 0$. En particulier, si g est croissante et pour tout point x, g est dérivable en x, alors, pour tout point x, $g'(x) \geq 0$. Par la proposition 12, g est croissante. Nous concluons que si pour tout point x, g est dérivable en x, alors, pour tout point x, $g'(x) \geq 0$. Par l'hypothèse h, si x est un nombre réel, g est dérivable en x. Nous concluons que pour tout nombre x, $g'(x) \geq 0$. En particulier, $g'(2) \geq 0$.

Ce texte est construit par des applications successives de ImplE et UnivE. Mais ce qu'on écrit plus souvent est, simplement,

> Par la proposition 12, g est croissante. Par l'hypothèse h, si x est un nombre réel, g est dérivable en x. Par le théorème 4, $f'(2) \geq 0$.

Le pas d'application consiste en ce qu'on fournit toutes les quatre prémisses du théorème simultanément. Pour ainsi dire, on emploie le théorème lui-même comme règle d'inférence, sans la médiation des règles logiques.

Un tel pas d'application est aussi utilisé couramment dans la théorie des types du niveau supérieur. Sa forme générale est

$$\frac{c : (x_1 : A_1)\cdots(x_n : A_n)A \quad a_1 : A_1 \quad \cdots \quad a_n : A_n(x_1 = a_1, \ldots, x_{n-1} = a_{n-1})}{c(a_1, \ldots, a_n) : A(x_1 = a_1, \ldots, x_n = a_n)}$$

La version textuelle de l'application générale est une sorte d'itération arbitraire des règles ImplE et UnivE. Le théorème appliqué est une proposition universelle sur une séquence Δ d'hypothèses ajoutées à un contexte Γ (cf. la fin de la section sur les contextes dans l'introduction). Si Δ a la forme

$$x_1 : A_1,\ x_2 : A_2,\ \ldots,\ x_n : A_n$$

l'application du théorème demande n prémisses de la forme

$$\mathrm{Cat}_i(\Gamma, A_i\gamma_i),$$

où la catégorie Cat_i est Trm si l'hypothèse correspondante est explicite et Preuve si elle est implicite ; γ_i est la substitution des interprétations des prémisses antérieures pour les variables. Dans la règle de linéarisation, la séquence

$$a_{i_1}, \ldots, a_{i_k}$$

comprend ceux des objets dont le type est Preuve ; ils sont cités dans le texte avant la conclusion. Les arguments $a_1, \ldots, a_n$ sont substitués aux variables correspondantes dans la conclusion :

$$\begin{array}{rcl}
\Gamma & : & \mathrm{cont} \\
\Delta & : & \mathrm{cont}/\Gamma \\
A & : & \mathrm{P}(\Gamma + \Delta) \\
c & : & \acute{\mathrm{E}}\mathrm{tiq}(\Gamma, \forall(\Delta, A^\star)) \\
a_1 & : & \mathrm{Cat}_1(\Gamma, A_1) \\
a_2 & : & \mathrm{Cat}_2(\Gamma, A_2(x_1 = a_1^\star)) \\
\vdots & & \\
a_n & : & \mathrm{Cat}_n(\Gamma, A_n(x_1 = a_1^\star, \ldots, x_{n-1} = a_{n-1}^\star)) \\
\hline
\mathrm{Appl}\acute{\mathrm{E}}\mathrm{tiq}(\Gamma, \Delta, A, c, a_1, \ldots, a_n) & : & \mathrm{Preuve}(\Gamma, A^\star(x_1 = a_1^\star, \ldots, x_n = a_n^\star)) \\
\mathrm{Appl}\acute{\mathrm{E}}\mathrm{tiq}(\Gamma, \Delta, A, c, a_1, \ldots, a_n)^\star & = & \mathrm{ap}(\ldots \mathrm{ap}(\mathrm{ap}(c^\star, a_1^\star), a_2^\star), \ldots a_n^\star) \\
\mathrm{Appl}\acute{\mathrm{E}}\mathrm{tiq}(\Gamma, \Delta, A, c, a_1, \ldots, a_n)^\circ & = & a_{i_1}^\circ.\ \underline{a_{i_2}^\circ}.\ \ldots \underline{a_{i_k}^\circ}. \\
 & & \mathrm{Par}\ \underline{c^\circ}, \\
 & & A[a_1/x_1, \ldots, a_n/x_n]^\circ(\mathrm{ind})
\end{array}$$

4 APPLICATIONS ET EXEMPLES

4.1 *L'implémentation sur ordinateur*

La théorie constructive des types a été implémentée sur ordinateur dans la forme d'un **éditeur de preuves.** C'est un logiciel qui effectue le **contrôle de typage** des termes de la théorie des types, ainsi que quelques constructions automatiques par des **tactiques**

de preuve. Puisque les preuves mathématiques sont représentables comme des termes, dont les types sont les propositions prouvées, le contrôle de typage est en même temps le contrôle de la correction des preuves. Les tactiques de preuve y ajoutent un élément de la démonstration automatique, mais en général, la démonstration est interactive: l'utilisateur du système doit lui donner les pas décisifs.

Par exemple, dans le système ALF,[94] les nombres naturels, leur égalité, et le principe d'induction sont définis

```
N : Set [] C
 0 : N [] C
 s : (N)N [] C

EqN : (N;N)Set [] C
 eqax1 :  EqN(0,0) [] C
 eqax2 : (a,b:N; EqN(a,b))EqN(s(a),s(b)) [] C

rec : (C:(N)Set; c:N; d:C(0); f:(x:N; C(x))C(s(x)))C(c) [] I
  rec(C,0,d,e) = d
  rec(C,s(n),d,e) = e(n,rec(C,n,d,e))
```

On se met à prouver un théorème par l'instruction qui définit une nouvelle constante comme le point d'interrogation. Ensuite, on donne des instructions de **raffinement**, c.-à-d. on propose une forme pour une constante inconnue. Le système répond par une liste de nouveaux problèmes, de contraintes, et de raffinements dérivés par la tactique d'unification. Par exemple, pour prouver la réflexivité de l'égalité, on introduit une constante inconnue et lui donne un premier raffinement:

```
-> define thm1 = ? : (n:N)EqN(n,n) ;;
-> refine thm1 with [n]rec ;;
```

Le système répond

```
New definitions :

C = ? : (N)Set      [n:N]
c = ? : N      [n:N]
d = ? : C(0)      [n:N]
f = ? : (x:N;C(x))C(s(x))      [n:N]

Constraints :
C(c) = EqN(n,n) : Set    [n:N]
```

La grammaire est implémentée de la même façon qu'une théorie mathématique. Par exemple, la règle de prédication du verbe intransitif est

```
PredV1 : (G:Cont; A:(TC(G))Dom; Q:SN(G,A); F:V1(G,A))P(G) [] C
 IP(G,PredV1(_,A,Q,F),g) = ISN(G,A,Q,g,[x]IV1(G,A,F,g,x))
  SP(G,PredV1(_,A,Q,F),m) =
   mm(SSN(G,A,Q,nul),SV1(G,A,F,GenSN(G,A,Q),NumSN(G,A,Q),m))
```

[94] « Another Logical Framework », v. Magnusson (1994).

C'est un codage assez direct de la règle présentée dans la section 2.3, avec la modification que les contextes et les domaines sont représentés comme des éléments des ensembles `Cont` et `Dom`. Les catégories syntaxiques sont des ensembles. Chaque catégorie a ses propres fonctions d'interprétation et de linéarisation ; dans cette règle-ci, on utilise `IP`, `IV1`, `ISN` et `SP`, `SV1`, `SSN`. Les chaînes sont représentées comme des listes de caractères ASCII ; `mm` est la concaténation qui ajoute un blanc entre les chaînes concaténées.

Pour construire un arbre syntaxique, on agit exactement comme pour la construction d'une preuve. Par exemple, pour analyser l'énoncé 3 *est impair*, on définit un arbre inconnu puis on le raffine par le constructeur `PredV1` :

```
-> define arbre1 = ? : P(Vac) ;;

-> refine arbre1 with PredV1 ;;
New definitions :

A = ? : (TC(Vac))Dom      []
Q = ? : SN(Vac,A)      []
F = ? : V1(Vac,A)      []
Derived refinements :
G = Vac

There are no constraints.
```

Finalement, on aura construit l'arbre

```
PredV1(Vac,[g]N',Mont(Vac,[g]N',
   UseTrm(Vac,N',UseConst(Vac,N',UseNum(UsePos(UseDig(3')))))),
 PredA1(Vac,[g]N',Etre(verum),UseA1(Vac,N',impair)))
```

dont l'interprétation et la linéarisation peuvent être calculées comme **formes normales** des termes où les opérateurs en question sont appliqués à l'arbre,

```
-> nf IP(Vac,arbre1,tt) ;;

normal form : Od'(s(s(s(0))))
-> nf SP(Vac,arbre1,ind) ;;

normal form : cc(36,cc(51,cc(36,cc(32,cv(101,cc(115,cc(116,
cc(32,cv(105,cc(109,cc(112,cc(97,cc(105,cc(114,cce))))))))))))))
```

De ce résultat de linéarisation, on obtient la chaîne

```
$3$ est impair.
```

Cette chaîne est le code LaTeX pour le texte

3 est impair.

Comme langage de programmation, ALF est une sorte de ML[95] renforcé par les types dépendants. Son interface d'utilisation est encore assez rudimentaire : il ne connaît que la notation purement fonctionnelle, et la transformation des listes de codes ASCII en chaînes doit être effectuée dans un autre langage. Mais ce qui est essentiel, c'est qu'il est capable d'exprimer les théories mathématiques, dont les règles de notre grammaire, d'une manière purement déclarative. Cela serait impossible sans les types dépendants.

[95] Un langage fonctionnel typé ; v. Paulson (1991).

4.2 *L'analyse syntaxique*

L'implémentation déclarative de la grammaire ne donne pas encore d'algorithme d'analyse: cet algorithme n'appartient pas à la définition de la langue. Comme nous avons vu dans la section 1.2, l'analyse est un problème de recherche, le problème de trouver un arbre dont la linéarisation est la chaîne donnée. Dans la section précédente, nous avons montré un procédé interactif d'analyse. Évidemment, la part de travail de l'utilisateur dépend des tactiques incorporées dans le système.

Une tactique très utile est l'analyse automatique de la structure syntagmatique, ce qui est une tâche routinière pour les règles hors-contexte. Une telle routine trouve les arguments syntaxiques des arbres; ce qui reste à faire est de remplir les places des arguments sémantiques. Ceux-ci sont pour une bonne part des arguments de typage, qui sont trouvés par l'unification. Donc l'exemple de la section précédente a une analyse complètement automatique:

```
-> refine arbre1 with PredV1(_,_,Mont(_,_,
      UseTrm(_,_,UseConst(_,_,UseNum(UsePos(UseDig(3'))))))),
   PredA1(_,_,Etre(verum),UseA1(_,_,impair))) ;;
Derived refinements :
A3 = N' G3 = G2 A2 = N' G2 = G1 A1 = [g]N' G1 = G A = [g]N'
A5 = N' G5 = G4 A4 = [z]N' G4 = G G = Vac
There are no constraints.
```

L'automatisation complète de l'analyse syntaxique est certainement impossible, parce qu'un argument sémantique peut aussi être, par exemple, une preuve cachée d'une complexité arbitraire. Dans un tel cas, le résultat du raffinement n'est pas un arbre complet, mais un arbre avec des constituants inconnus, qui peuvent être trouvés ensuite par la méthode interactive.

4.3 *L'éditeur des textes de preuve*

L'énonciation automatique d'une proposition donnée n'appartient pas à la définition de notre grammaire, et c'est encore le procédé interactif qu'il faut considérer comme fondamental. Les tactiques convenables ne sont pas aussi bien connues que dans le cas de l'analyse. Il y a un programme assez effectif et général, celui de Coscoy *et al.* (1994), qui traduit les termes de preuve du système Coq en français et en anglais, mais il ne concerne que les expressions appartenant à la structure des preuves: les propositions dans les preuves sont toujours exprimées par des formules.[96]

Le procédé interactif d'énonciation est utile aussi pour la raison qu'il y a tant de variations possibles: déjà dans la grammaire que nous avons présentée, une proposition reçoit facilement des milliers d'énoncés.[97] Une énonciation automatique devrait toujours choisir une des alternatives, ce qui résulterait dans un texte monotone, ou bien fonctionner d'une façon aléatoire, ce qui serait sans doute encore plus bizarre. On désire de la variation dans le texte, mais de la variation contrôlée.

Donc une tâche computationnelle que nous trouvons très prometteuse est l'**éditeur des textes de preuve**, un logiciel qui aide à la construction d'un texte mathématique.

[96] Dans Ranta (1994, chapitre 9), nous avons essayé de trouver un algorithme qui transforme les formules en anglais, mais c'est en partie la difficulté énorme de cette tâche que nous a mené à l'introduction de la grammaire tripartite avec les arbres syntaxiques.

[97] Supposons qu'une formule est construite par 12 opérateurs, dont chacun peut être exprimé de deux manières. Alors la formule peut être exprimée de 4096 manières.

L'utilisateur fait les pas décisifs, le système remplit les détails routiniers et vérifie que tout est correct. Étant donné le contrôle de typage, la partie essentielle d'un tel système est la grammaire elle-même, qui définit et la construction grammaticale et la correction mathématique.

4.4 *La traduction d'une théorie formelle en français*

La question se pose de savoir si notre grammaire et **complète**, c.-à-d. capable d'exprimer toutes les propositions mathématiques. Cette question peut être précisée en une question mathématique si on choisit un langage formel L, tel que celui de l'arithmétique de Peano : le problème

d'exprimer toute formule close de L par une expression de type P(())

est un problème mathématique dont la solution est un algorithme qui associe un arbre du type P(()) à toute formule de L. Si on veut un tel algorithme pour permettre une traduction automatique des expressions produites par une machine, on veut peut-être renforcer le problème en y ajoutant la condition d'univocité :

d'exprimer toute formule close de L par une expression univoque de type P(()).

La première tâche devient de plus en plus facile si la grammaire est étendue par de nouvelles règles, tandis que la seconde tâche peut aussi devenir moins facile.

Notre objectif primaire n'a pas été de définir des traductions entre les formalismes mathématiques et le français, mais de trouver des règles de celui-ci, permettant la formulation rigoureuse des problèmes de traduction. Toutefois, les structures grammaticales que nous avons définies permettent l'expression de chaque forme propositionnelle du calcul des prédicats :[98]

proposition	forme syntaxique	forme linéarisée
$A \,\&\, B$	ConjrP(et-et, $[A, B]$)	*et A et B*
$A \vee B$	ConjrP(ou-ou, $[A, B]$)	*ou A ou B*
$A \supset B$	si-alors(PremP(A), B)	*si A, alors B*
$\sim A$	pasvrai(A)	*il n'est pas vrai que A*
$(\forall x : A)B$	pour-tout(DéclCN($[x], A$), B)	*pour tout A x, B*
$(\exists x : A)B$	il-existe(DéclCN($[x], A$), B)	*il existe un A x tel que B*

Les structures montrées dans ce tableau pourraient servir de base pour un algorithme de traduction, parce qu'elles peuvent être combinées et itérées d'une façon arbitraire ; bien sûr, le français produit par un tel algorithme est assez monotone.

Pour étendre l'algorithme de traduction aux formules atomiques d'une théorie particulière, il faut aussi associer des phrases à celles-ci. Une méthode systématique de faire cela est d'associer un verbe ou un adjectif à chaque prédicat de la théorie. Par exemple, quelques prédicats arithmétiques peuvent être exprimés par les mots suivants :

prédicat	expression française	phrase linéarisée
$a = b$	égal : A2($\grave{a}$)	*a est égal à b*
$a > b$	supérieur : A2($\grave{a}$)	*a est supérieur à b*
Div(a, b)	divisible : A2(par)	*a est divisible par b*
Pr(a)	premier : A1	*a est premier*

[98] Les connecteurs sont exprimés par des conjonctions répétées pour garantir l'univocité.

Les formules atomiques sont ensuite traduites en utilisant des **cadres de prédication**, c.-à-d. des macros qui correspondent à l'application des prédicats logiques. Un cadre de prédication s'applique à un verbe ou à un adjectif et à un ou deux noms propres, pour produire une phrase.

P $\rightarrow$ PN V1

$$\frac{\Gamma : \text{cont} \quad A : \text{set}/\Gamma \quad F : \text{V1}(\Gamma, A) \quad a : \text{PN}(\Gamma, A)}{\text{CadreV1}(\Gamma, A, F, a) = \text{PrédV1}(\Gamma, A, \text{Mont}(\Gamma, A, a), F) : \text{P}(\Gamma)}$$

P $\rightarrow$ PN V2 PN

$$\frac{\Gamma : \text{cont} \quad A, B : \text{set}/\Gamma \quad r : \text{préfixe} \quad F : \text{V2}(\Gamma, A, B, r) \quad a : \text{PN}(\Gamma, A) \quad b : \text{PN}(\Gamma, B)}{\begin{array}{l}\text{CadreV2}(\Gamma, A, B, r, F, a, b) = \\ \quad \text{CadreV1}(\Gamma, A, \text{ComplV2}(\Gamma, A, B, r, F, \text{Mont}(\Gamma, B, b)), a) : \text{P}(\Gamma)\end{array}}$$

P $\rightarrow$ PN *est* A1

$$\frac{\Gamma : \text{cont} \quad A : \text{set}/\Gamma \quad F : \text{A1}(\Gamma, A) \quad a : \text{PN}(\Gamma, A)}{\text{CadreA1}(\Gamma, A, F, a) = \text{CadreV1}(\Gamma, A, \text{PrédA1}(\Gamma, A, \text{être(vrai)}, F), a) : \text{P}(\Gamma)}$$

P $\rightarrow$ PN *est* A2 PN

$$\frac{\Gamma : \text{cont} \quad A, B : \text{set}/\Gamma \quad r : \text{préfixe} \quad F : \text{A2}(\Gamma, A, B, r) \quad a : \text{PN}(\Gamma, A) \quad b : \text{PN}(\Gamma, B)}{\begin{array}{l}\text{CadreA2}(\Gamma, A, B, r, F, a, b) = \\ \quad \text{CadreV2}(\Gamma, A, B, r, \text{PrédA2}(\Gamma, A, B, r, \text{être(vrai)}, F), a, b) : \text{P}(\Gamma)\end{array}}$$

P $\rightarrow$ PN *et* PN VC

$$\frac{\Gamma : \text{cont} \quad A : \text{set}/\Gamma \quad B : \text{set}/\Gamma \quad F : \text{VC}(\Gamma, A, B) \quad a : \text{PN}(\Gamma, A) \quad b : \text{PN}(\Gamma, B)}{\begin{array}{l}\text{CadreVC}(\Gamma, A, B, F, a, b) = \\ \quad \text{PrédVC}(\Gamma, A, B, \text{MontPNC}(\Gamma, A, B, \text{CouplePN}(\Gamma, A, B, a, b)), F) : \text{P}(\Gamma)\end{array}}$$

P $\rightarrow$ PN *et* PN *sont* AC

$$\frac{\Gamma : \text{cont} \quad A : \text{set}/\Gamma \quad B : \text{set}/\Gamma \quad F : \text{AC}(\Gamma, A, B) \quad a : \text{PN}(\Gamma, A) \quad b : \text{PN}(\Gamma, B)}{\begin{array}{l}\text{CadreAC}(\Gamma, A, B, F, a, b) = \\ \quad \text{CadreVC}(\Gamma, A, B, \text{PrédAC}(\Gamma, A, B, \text{être(vrai)}, F), a, b) : \text{P}(\Gamma)\end{array}}$$

Si la traduction des termes singuliers est définie comme l'opération triviale qui consiste à utiliser les termes eux-mêmes dans le texte, par l'opérateur UseTrm (section 3.4), la méthode de traduction de l'arithmétique est complète. (Le terme de domaine N est exprimé par le nom commun *nombre*.) Par exemple, la traduction de la formule

$$(\forall x : N)(\forall y : N)(x + y = y + x)$$

est la phrase

pour tout nombre x, pour tout nombre y, $x + y$ est égal à $y + x$.[99]

[99] D'autres expressions pour la même proposition sont données dans la section 3.6.

4.5 *Les optimisations et la variation stylistique*

La monotonie du français produit par une traduction automatique peut être évitée par l'usage de structures alternatives. Il y a des **optimisations** évidentes, applicables aux structures logiques particulières, qu'on peut reconnaître dans les formules. Par exemple, une suite de plusieurs quantificateurs universels peut être comprimée en une déclaration :

> *pour tous les nombres m et n* au lieu de *pour tout nombre m, pour tout nombre n*,
>
> *pour tout nombre pair n* au lieu de *pour tout nombre n, si n est pair.*

La négation d'une proposition atomique peut être exprimée par la négation du verbe,

> 91 *n'est pas premier* au lieu de *il n'est pas vrai que* 91 *soit premier.*

La répétition d'un argument est évitée par l'usage d'une forme réfléchie,

> $n + 1$ *est égal à lui-même* au lieu de $n + 1$ *est égal à* $n + 1$.

La conjonction simple peut remplacer la conjonction répétée s'il n'y a pas de risque d'ambiguïté,

> 2 *est pair et* 3 *est impair* au lieu de *et* 2 *est pair et* 3 *est impair.*

Au lieu de coordonner deux phrases, on peut coordonner les constituants minimaux,

> 12 *est divisible par* 3 *et* 4 plutôt que 12 *est divisible par* 3 *et divisible par* 4 plutôt que 12 *est divisible par* 3 *et* 12 *est divisible par* 4.

De telles optimisations peuvent être automatisées dans l'algorithme de traduction. L'étude des optimisations permises par une grammaire formelle appartient à la stylistique de la langue mathématique.[100]

Même un algorithme de traduction optimisé est déterministe : une formule entière est toujours exprimée par la même phrase.[101] Donc les optimisations ne sont pas le seul facteur de vivacité d'un texte. Un autre facteur sont les choix libres des expressions alternatives, qui ne sont possibles que dans un système interactif de traduction.

4.6 *Exemple : une preuve arithmétique*

Les règles textuelles de la déduction naturelle (v. section 3.7) permettent l'extension de la traduction des formules en traduction des preuves formelles. Comme exemple, construisons une petite preuve inductive. À partir des axiomes

> Axiome 1. 0 est pair.
>
> Axiome 2. Pour tout nombre n, si n est pair, alors $n + 1$ est impair.
>
> Axiome 3. Pour tout nombre n, si n est impair, alors $n + 1$ est pair.

on peut prouver le théorème

[100] Une grande partie de l'article de Coscoy, Kahn et Théry (1995) concerne l'optimisation des textes de preuve.

[101] Cependant, si une formule est utilisée comme sous-formule, la manière dont elle est exprimée dépend du complexe dont elle fait partie. Donc la treduction optimisée n'est pas compositionnelle.

Théorème 1. Tout nombre est pair ou impair.

Les expressions formelles du théorème et de sa preuve sont, dans la notation d'ALF,

```
thm1 = Pi(N,[x]Vel(Ev(x),Od(x))) : Set      []

prf1 = lambda(N,[z]Union(Ev(z),Od(z)),[x]rec([h]Union(Ev(h),
       Od(h)),x,inl(Ev(0),Od(0),evz),[x',h]when(Ev(x'),Od(x'),
       [z]Union(Ev(s(x')),Od(s(x'))),h,[x'']inr(Ev(s(x')),
       Od(s(x')),odtr(x',x'')),[y]inl(Ev(s(x')),Od(s(x')),
       evtr(x',y))))) : Pi(N,[x]Vel(Ev(x),Od(x)))      []
```

Probablement, cette preuve formelle est moins facile à lire que la version française :[102]

Théorème 1. Tout nombre est pair ou impair.

Démonstration. Montrons que pour tout nombre n, n est pair ou impair, par induction. Pour la base, rappelons que par le premier axiome de parité, 0 est pair. *A fortiori*, 0 est pair ou 0 est impair. Pour l'hypothèse d'induction, considérons un nombre n tel que

(a) n est pair ou impair.

Par l'hypothèse a, n est pair ou impair. Montrons que $n + 1$ est pair ou impair, en distinguant deux cas. Dans le premier cas, supposons que

(b) n est pair.

Par l'hypothèse b, n est pair. Par le deuxième axiome de parité, $n + 1$ est impair. *A fortiori*, $n + 1$ est pair ou $n + 1$ est impair. Dans le second cas, supposons que

(c) n est impair.

Par l'hypothèse c, n est impair. Par le troisième axiome de parité, $n + 1$ est pair. *A fortiori*, $n + 1$ est pair ou $n + 1$ est impair.

4.7 *Exemple : les axiomes de la géométrie constructive*

Comme système d'exemples qui démontre les ressources de la grammaire entière, nous présentons les énoncés des axiomes de la géométrie constructive de von Plato (1995). Ils ont été produits interactivement par l'implémentation de la grammaire en ALF.

Il y a des groupes d'axiomes avec la même structure logique, ce qui nous a donné une occasion de démontrer les expressions alternatives pour ces structures.

D'abord, trois axiomes de l'irreflexivité d'une relation :

$(\forall x : A) \sim C(x, x)$

Axiome Ia1. Si a est un point, alors a n'est pas distinct de a.

Axiome Ia2. Toute droite est égale à elle-même.[103]

Axiome Ia3. Quelle que soit la droite l, l ne se coupe pas.

[102] Le texte qui suit a été produit par ALF : un arbre syntaxique a été construit interactivement, et le système a produit sa linéarisation en code LaTeX.

[103] Dans la géométrie constructive, l'égalité est définie comme la négation de la distinction.

Ensuite, trois axiomes d'une propriété des relations dont la contraposition est la transitivité :

$(\forall x, y : A)(C(x,y) \supset (\forall z : A)(C(x,z) \vee C(y,z)))$

Axiome Ib1. Si a et b sont des points distincts, alors pour tout point c, ou a ou b est distinct de c.

Axiome Ib2. Si l et m sont des droites distinctes et n est une droite, alors soit l est distincte de n soit m est distincte de n.

Axiome Ib3. Si l et m sont des droites telles que l coupe m, alors quelle que soit la droite n, n coupe l ou m.

Deux axiomes pour des constructions géométriques c avec des présuppositions :

$(\forall x, y : A)(\Pi z : C(x,y))(\sim D(x, c(x,y,z)) \;\&\; \sim D(y, c(x,y,z)))$

Axiome II.1. Si a et b sont des points distincts, alors a et b sont incidents à la ligne de connexion entre a et b.

Axiome II.3. Si l et m sont des droites convergentes, alors l et m contiennent le point d'intersection des droites l et m.

Les autres axiomes sans commentaires :

Axiome III. Si a et b sont des points distincts et l et m sont des droites distinctes, alors a ou b est extérieur à l ou à m.

Axiome IV.1. Si a est un point et l est une droite à laquelle a est extérieur, alors quel que soit le point b, b est distinct du point a ou extérieur à la droite l.

Axiome IV.2. Si a est un point et l est une droite à laquelle a est extérieur, alors quelle que soit la droite m, les droites l et m sont distinctes ou le point a est extérieur à m.

Axiome IV.3. Si l et m sont des droites convergentes et n est une droite, alors les droites m et n sont distinctes ou l coupe n.

Axiome A1a. Si l est une droite, alors quel que soit le point a, la parallèle à l par a contient le point a.

Axiome A1b. Si l est une droite, alors quel que soit le point a, la parallèle à l par a est parallèle à la droite l.

Axiome A2. Si l et m sont des droites distinctes, alors pour tout point a, a est extérieur à l ou à m ou les droites l et m sont convergentes.

Axiome P1. Si l et m sont des droites distinctes, alors l et m sont convergentes.

Axiome O1. Si l et m sont des droites, alors l et m sont convergentes ou anorthogonales.

Axiome O2. Si l et m sont des droites convergentes et anorthogonales, alors quelle que soit la droite n, l et n sont convergentes et anorthogonales ou m et n sont convergentes et anorthogonales.

Axiome O3a. Si l est une droite et a est un point, alors la perpendiculaire à l par a est perpendiculaire à l.

Axiome O3b. Si l est une droite et a est un point, alors la perpendiculaire à l par a contient a.

Axiome O4. Soient l et m des droites distinctes. Soit a un point. Soit n une droite. Alors le point a est extérieur à l ou à m ou les droites l et n sont anorthogonales ou m et n sont anorthogonales.

BIBLIOGRAPHIE

ABEILLÉ A., *Les nouvelles syntaxes*, Paris, Armand Colin, 1993.

AJDUKIEWICZ K., « Die syntaktische Konnexität », *Studia Philosophica* 1 (1935), pp. 1–27.

BAR-HILLEL Y., « A quasi-arithmetical notation for syntactic description », *Language*, 29 (1953), pp. 47–58.

DE BRUIJN N. G., « Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem », *Indagationes Mathematicae* 34 (1972), pp. 381–392.

DE BRUIJN N. G., « The mathematical vernacular, a language for mathematics with typed sets », NEDERPELT R., éd., *Selected Papers on Automath*, pp. 865–935, Amsterdam, North-Holland, 1994.

CHAMBREUIL M., *Grammaire de Montague : langage, traduction, interprétation*, ADOSA, Clermont-Ferrand, 1989.

CHOMSKY N., *Aspects of the Theory of Syntax*, Cambridge, Ma., The M.I.T. Press, 1965.

COSCOY Y., KAHN G. et THÉRY L., *Extracting text from proofs*, Rapport de recherche n. 2459, Sophia-Antipolis, INRIA, 1995.

FREGE G., *Begriffsschrift*, Halle A/S, Louis Nebert, 1879.

GAZDAR G., KLEIN E., PULLUM G. et SAG I., *Generalized Phrase Structure Grammar*, Oxford, Basil Blackwell, 1985.

GEACH P., *Reference and Generality*, Ithaca, New York, Cornell University Press, 1962.

GENTZEN G., « Untersuchungen über das logische Schliessen », *Mathematische Zeitschrift*, 39 (1934), pp. 176–210 et 405–431.

GREVISSE M., *Le bon usage*, Treizième édition, Paris, Duculot, 1993.

HEYTING A., *Intuitionism*, Amsterdam, North-Holland, 1956.

LAMBEK J., « The mathematics of sentence structure », *American Mathematical Monthly*, 65 (1958), pp. 154–170.

LAMPORT L., *LaTeX. A Document Preparation System*, Reading, Ma., Addison-Wesley Publishing Company, 1986.

LECOMTE A., *Modèles logiques en théorie linguistique*, Synthèse de travaux présentés en vue de l'habilitation à diriger des recherches, Université de Grenoble, 1994.

MAGNUSSON L., *The Implementation of ALF — a Proof Editor Based on Martin-Löf's Monomorphic Type Theory with Explicit Substitutions*, Thèse doctorale, Department of Computing Science, University of Göteborg, 1994.

MARTIN-LÖF P., *Intuitionistic Type Theory*, Naples, Bibliopolis, 1984.

MONTAGUE R., *Formal Philosophy*, Collected papers edited by R. THOMASON, New Haven, Yale University Press, 1974.

MORRILL G., *Type Logical Grammar*, Dordrecht, Kluwer Academic Publishers, 1994.

NORDSTRÖM B., PETERSSON K. et SMITH J., *Programming in Martin-Löf's Type Theory. An Introduction*, Oxford, Clarendon Press, 1990.

PAULSON L., *ML for the Working Programmer*, Cambridge, Cambridge University Press, 1991.

VON PLATO J., «The axioms of constructive geometry», *Annals of Pure and Applied Logic*, 76 (1995), pp. 169–200.

RANTA A., *Type Theoretical Grammar*, Oxford, Oxford University Press, 1994.

RANTA A., «Syntactic categories in the language of mathematics», DYBJER P., NORDSTRÖM B. et SMITH J., éd., *Types for Proofs and Programs*, pp. 162–182, *Lecture Notes in Computer Science* 996, Heidelberg, Springer-Verlag, 1995.

RANTA A., «Context-relative syntactic categories and the formalization of mathematical text», BERARDI S. et COPPO M., éd., *Types for Proofs and Programs*, pp. 231–248, *Lecture Notes in Computer Science* 1158, Heidelberg, Springer-Verlag, 1996.

SHIEBER S., *An Introduction to Unification-Based Approaches to Grammar*, Menlo Park, Ca., CSLI, 1986.

STEEDMAN M., «Combinators and grammars», OEHRLE R., BACH E. et WHEELER D., éd., *Categorial Grammars and Natural Language Structures*, pp. 417–442, Dordrecht, D. Reidel, 1988.

TASISTRO A., *Formulation of Martin-Löf's Monomorphic Theory of Types with Explicit Substitutions*, Thèse de licence, Department of Computing Science, University of Göteborg, 1993.