Context-Relative Syntactic Categories and the Formalization of Mathematical Text

Aarne Ranta

Department of Philosophy, P.O.Box 24
00014 University of Helsinki, Finland.

The format of grammar presented by Montague (1974) is widely applied in the study of the logical structure of informal language. Thus it is natural to try how it works in the analysis of language in which logical content is uncontroversially a prominent part of meaning, the language of mathematics. In Ranta 1995, we presented a system of syntactic categories obtained from categories of Montague style by adding domain parameters; semantically, this corresponds to the transition from simple type theory to a type theory with multiple domains of individuals and dependent function types. In this paper, we shall generalize the system of categories further, by adding context parameters. In this way, it becomes possible to express arbitrarily complex quantificational structures in English. We go on by defining a category of proof texts, which are interpreted as type-theoretical proof objects. A system of introduction and elimination rules will be presented for the logical constants, as well as some mathematical proof rules.

The rules presented in this paper are a fragment of a grammar implemented in the proof editor ALF. A similar grammar has been implemented for French, too. The implementation can be used as an interactive proof text editor, so that the user chooses proof steps, wordings, and grammatical structures, and the system checks that the text is both mathematically and grammatically correct.

1 Introduction

A mathematical text can be considered from two points of view, linguistic and logical. From the linguistic point of view, a mathematical text is made up from sentences, which, in turn, are made up from nouns and verbs and adjectives, according to the same rules of grammar as any other text. From the logical point of view, a mathematical text is an expression of a theorem or of a proof, made up by applications of logical constants and proof constructors, and by bindings of variables.

Let us consider an example:

> *If f is a differentiable function, then if f is increasing, $f'(x) > 0$ for every x.*

Linguistically, this is an English sentence with some symbols used as constituents, in a way that is normal in mathematical writing. It is a conditional whose antecedent is formed from a noun phrase and a verb phrase; the noun phrase is a single symbol f and the verb phrase is the copula *is* attached to an indefinite noun phrase. In this way, we can enter deeper and deeper in the grammatical structure of the sentence, eventually to find the way in which it is made up from words and suffixes and symbols, and conclude that it is a well-formed English sentence.

Logically, the sentence expresses the proposition

$$(\forall f : R \to R)(\Pi z : (\forall x : R)\mathrm{Diff}(f, x)) \\ (\mathrm{Incr}(f) \supset (\forall x : R)(D(f, x, \mathrm{ap}(z, x)) > 0)).$$

This formalization is in constructive type theory, where the propositions-as-types principle is applied to express the dependence of the term $f'(x)$ on the proof that f is differentiable at x. Thus it is somewhat more accurate than a formalization in predicate calculus. What it is intended to show is the mathematical content of the sentence—the logical role it plays in a theory in which it occurs.

These two analyses of the sentence are very different and, in a sense, complementary: they are both essential for a full understanding of the sentence as a piece of mathematical text. A mathematical text is not correct if it does not conform to both grammatical and logical rules. A complete system of rules for building mathematical texts should thus incorporate both kinds of information. Both linguistic and logical analysis of language ignore some relevant distinctions.

That logical analysis ignores linguistic distinctions is shown clearly by the possibility of expressing a given proposition by hundreds or thousands of different sentences: that the proposition is the same for all of them means that the linguistic differences are suppressed in the proposition. (That there can be so many linguistic expressions is easy to apprehend: if a formula is made up from ten operators and each operator is expressible in two ways, there are 1024 expressions for the proposition.)

That linguistic analysis, on the other hand, usually ignores logical distinctions is perhaps still better known to mathematicians. Two aspects can be pointed out in the example above. First, the formation of sentences from nouns and verbs and adjectives is often presented as much more liberal than what is mathematically meaningful. Both of the sentences

the function f is increasing,
the number x is increasing,

are grammatically well-formed, but only the former one makes mathematical sense. This is because the predicate Incr is only defined for functions, not for numbers. Second, different statuses of symbols are ignored. In our example, the first occurrence of the symbol f is in a *declaration*, corresponding to its use in the quantifier prefix $(\forall f : R \to R)$: this declaration creates a context in which the symbol can be used as a referring expression. The other two occurrences of f are normal uses of the symbol in the context in which it has been declared.

These logical phenomena, although sometimes ignored, are certainly often discussed in modern linguistic theory, under the titles of *selectional restrictions* and *discourse representation*, respectively. In Ranta 1995, we presented a way of making selectional restrictions—logically, domain distinctions—explicit in grammar, by relativizing grammatical categories to domains. For instance, the category PN of proper names was replaced by the family of categories $\mathrm{PN}(A)$, where $A : \mathrm{dom}$. The expressions of each category are *syntax trees*, type-theoretical objects that can be effectively both *interpreted* as ordinary mathematical objects and *sugared* into ordinary English expressions—that is, into strings of characters.

In this paper, we shall present a further relativization of grammatical categories to contexts, to be able to give a full analysis of mathematical texts.

2 Context-Relative Categories

The general form of a grammatical rule is

$$\frac{a_1 : \alpha_1 \quad \cdots \quad a_n : \alpha_n}{\begin{array}{l} C(a_1, \ldots, a_n) : \alpha \\ C(a_1, \ldots, a_n)^\star = F(a_1^\star, \ldots, a_n^\star) \\ C(a_1, \ldots, a_n)^\circ = G(a_1^\circ, \ldots, a_n^\circ) \end{array}}$$

Some of the types $\alpha_1, \ldots, \alpha_n$ are grammatical categories, some of them ordinary type-theoretical categories. The corresponding arguments $a_1, \ldots, a_n$ of C are called syntactic and semantic arguments, respectively. In sugaring, the semantic arguments are erased, but they are indispensable for interpretation, and, indeed, for the very formation of the syntax tree. If the grammatical categories are $\alpha_{i_1}, \ldots, \alpha_{i_k}$, the rule above corresponds to the phrase structure rule

$$\alpha \to \alpha_{i_1} \ \ldots \ \alpha_{i_k}.$$

The semantic arguments of a grammatical construction include domain and context information. For domains, we shall use the ordinary notation for sets of lower-level type theory, as presented in Martin-Löf 1984. For contexts, we shall use a notation stemming from Per Martin-Löf's lectures on substitution calculus in 1992. (Tasistro 1993 uses a slightly different notation, but gives a presentation of the same structure.) In the ALF implementation, the types cont and dom are encoded as inductively defined universes. (So is, consequently, prop, which is the same as dom.)

By writing

$$\Gamma : \mathrm{cont},$$

we mean that Γ is a sequence of hypotheses, of the form

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

where $n = 0, 1, 2, \ldots$ and each A_k is a set depending on the hypotheses up to $n = k - 1$. We write

$$J/\Gamma$$

for a judgement J, of any of the four forms of Martin-Löf (1984), when made in the context Γ. We think of contexts as being defined inductively by the rules

$$() : \mathrm{cont}, \qquad \frac{\Gamma : \mathrm{cont} \quad A : \mathrm{dom}/\Gamma}{(\Gamma, \ x : A) : \mathrm{cont}},$$

where x is a fresh variable.

We can now generalize the category S of sentences to the family of categories

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

where $\Gamma : \mathrm{cont}$, of sentences in the context Γ. Expressions of $\mathrm{S}(\Gamma)$ are interpreted as propositional functions

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

The variables of Γ may thus occur free in A. (The old category S is now obtained as the special case $\mathrm{S}(())$, the category of sentences in the empty context.)

A typical example of a sentence in a non-empty context is the succedent of a conditional. The ordinary phrase structure rule,

$$\mathrm{S} \rightarrow \textit{if}\ \mathrm{S}\ \textit{then}\ \mathrm{S},$$

presents the conditional as a mere concatenation of two sentences. But the sentence

if a number is even then it is divisible by 2

cannot be understood as the mere concatenation of the sentences *a number is even* and *it is divisible by* 2, because the second sentence depends on the first sentence: the pronoun *it* is interpreted as the number given in the context of the first sentence. The precise grammatical rule for the conditional, which makes this dependence explicit, is

$$\frac{\Gamma : \mathrm{cont} \quad A : \mathrm{S}(\Gamma) \quad B : \mathrm{S}((\Gamma,\ x : A^\star))}{\begin{array}{l} \mathrm{if}(\Gamma, A, B) : \mathrm{S}(\Gamma) \\ \mathrm{if}(\Gamma, A, B)^\star = (\Pi x : A^\star)B^\star \\ \mathrm{if}(\Gamma, A, B)^\circ = \textit{if}\ A^\circ\ \textit{then}\ B^\circ \end{array}}.$$

The category PN of proper names, which was first generalized to the family $\mathrm{PN}(A)$ of proper names of type A, is now generalized to the family

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

where $\Gamma : \mathrm{cont}$ and $A : \mathrm{dom}/\Gamma$, of proper names of type A in the context Γ. Expressions of $\mathrm{PN}(\Gamma, A)$ are interpreted as elements of A depending on Γ, that is, as objects

$$a : A/\Gamma.$$

A typical example is the anaphoric pronoun *it*,

$$\frac{\Gamma : \mathrm{cont} \quad A : \mathrm{dom}/\Gamma \quad a : A/\Gamma}{\begin{array}{l} \mathrm{Pron}(\Gamma, A, a) : \mathrm{PN}(\Gamma, A) \\ \mathrm{Pron}(\Gamma, A, a)^\star = a \\ \mathrm{Pron}(\Gamma, A, a)^\circ = \textit{it} \end{array}}$$

The corresponding phrase structure rule

$$\mathrm{PN} \rightarrow \textit{it}$$

is particularly unrevealing as for the conditions of use of the pronoun:

The anaphoric pronoun *it* may be used as a name referring to any individual of any type given in the context.

(Cf. Ranta 1994, chapter 4, for more discussion about the use of anaphoric pronouns. Their use in mathematical text is often avoided if the uniqueness of interpretation is in danger.)

More complicated examples of a proper names in a context are

the inverse of the function f,
the derivative of f at x,
the intersection point of l and m.

They are, superficially, similar to the proper names

the square of the number x,
the value of the polynomial P at x,
the greatest common divisor of 51 *and* 85.

But logically, the examples are different. Each of the former ones hides a *presupposition*, a condition whose proof must be given in the context. In the first example, f must be bijective; in the second one, f must be differentiable at x; and in the third one, the lines l and m must intersect. The proofs are provided by semantical arguments of the grammatical constructions. For instance, the third example is produced by the rule

$$\frac{\begin{array}{l}\Gamma : \mathrm{cont} \quad A, B : \mathrm{dom}/\Gamma \quad C : (A)(B)\mathrm{prop}/\Gamma \quad D : \mathrm{dom}/\Gamma \\ f : \mathrm{F4}(\Gamma, A, B, C, D) \quad a : \mathrm{PN}(\Gamma, A) \quad b : \mathrm{PN}(\Gamma, B) \quad c : C(a^\star, b^\star)/\Gamma\end{array}}{\begin{array}{l}\mathrm{DRAW}(\Gamma, A, B, C, D, f, a, b, c) : \mathrm{PN}(\Gamma, D) \\ \mathrm{DRAW}(\Gamma, A, B, C, D, f, a, b, c)^\star = f^\star(a^\star, b^\star, c) \\ \mathrm{DRAW}(\Gamma, A, B, C, D, f, a, b, c)^\circ = \textit{the } f^\circ \textit{ of } a^\circ \textit{ and } b^\circ.\end{array}}$$

(Cf. Ranta 1995, section 8, where the same rule is given without dependence on a context, thus requiring a constant proof of the presupposition.)

The generalization of most of the categories and rules of Ranta 1995 is straightforwardly done pointwise. The following table shows the context-relative versions of some of the main categories.

category	of	where	is interpreted
$\mathrm{S}(\Gamma)$	sentences	Γ : cont	prop/Γ
$\mathrm{CN}(\Gamma)$	common nouns	Γ : cont	dom/Γ
$\mathrm{PN}(\Gamma, A)$	proper names	Γ : cont, A : dom$/\Gamma$	A/Γ
$\mathrm{NP}(\Gamma, A)$	noun phrases	Γ : cont, A : dom$/\Gamma$	$((A)\mathrm{prop})\mathrm{prop}/\Gamma$
$\mathrm{V1}(\Gamma, A)$	intransitive verbs	Γ : cont, A : dom$/\Gamma$	$(A)\mathrm{prop}/\Gamma$
$\mathrm{V2}(\Gamma, A, B)$	transitive verbs	Γ : cont, A, B : dom$/\Gamma$	$(A)(B)\mathrm{prop}/\Gamma$

The first rule of predication,

$$\mathrm{S} \rightarrow \mathrm{NP}\ \mathrm{V1}$$

becomes

$$\frac{\Gamma : \text{cont} \quad A : \text{dom}/\Gamma \quad Q : \text{NP}(\Gamma, A) \quad F : \text{V1}(\Gamma, A)}{\begin{array}{l}\text{SUBJ}(\Gamma, A, Q, F) : \text{S}(\Gamma) \\ \text{SUBJ}(\Gamma, A, Q, F)^{\star} = Q^{\star}(F^{\star}) \\ \text{SUBJ}(\Gamma, A, Q, F)^{\circ} = Q^{\circ}\ F^{\circ}.\end{array}}$$

Basic expressions, that is, lexical items, are generalized, accordingly, to expressions usable in any context. Their interpretations do not effectively depend on the context. For instance,

$$\frac{\Gamma : \text{cont}}{\begin{array}{l}\text{zero}(\Gamma) : \text{PN}(\Gamma, N) \\ \text{zero}(\Gamma)^{\star} = 0 \\ \text{zero}(\Gamma)^{\circ} = 0\end{array}}$$

(One might first think of basic expressions as expressions in the empty context, but this would not accord with the pointwise relativization of other forms of expression. Moreover, basic expressions would then not be readily usable in nonempty contexts.)

The really significant use of context-relative categories is made in the description of progressive structures, in rules that extend the context. We have already mentioned the progressive implication; the conjunction is analogous, interpreted in terms of Σ rather than just &. Yet another device of extending a context is an assumption made in the course of a proof text. As we shall see in Section 6, an assumption extends the context until it is discharged.

3 Explicit Variables

What we have considered so far might be called the implicit context of an expression: the variables appear in the syntax tree and in the logical interpretation, but they are not visible in the English string itself, after sugaring. In the language of mathematics, explicit variables are used as well. In the statement

if n is a number, then $n + 1$ is divisible by itself,

the antecedent introduces the explicit variable n, of type N, which is then used in the succedent. In the statement

> *if l and m are intersecting lines, then l contains the intersection point of l and m,*

there are two explicit variables, l and m of type Ln, and one implicit variable, a proof that l intersects m. All these variables are used in the succedent.

In the grammatical analysis of a mathematical text, one must keep track not only of the context, but of what part of the context is explicit. It is clear that explicit context can also be used in the same way as implicit context: in the statement *if n is a number, then the successor of it is divisible by itself,*

the reference to the number n is made by using the pronoun *it*. But implicit variables can only be used as hidden, semantical arguments.

To formalize the distinction between explicit and implicit variables, we extend the inductive definition of contexts, which was given in the previous section, by the clause

$$\frac{\Gamma : \mathrm{cont} \quad A : \mathrm{dom}/\Gamma \quad t : \mathrm{Str}}{(\Gamma,\ x_t : A) : \mathrm{cont}},$$

where x is a fresh variable. The string t is the variable name to be used in the text; the difference to the implicit variable declaration is that such a text name is provided, not only a theory-internal name. (To guarantee the freshness of t with respect to the text names introduced earlier, we may either demand t to be fresh, or add a sufficient number of primes to it in sugaring.)

There is no semantic difference between $(\Gamma,\ x_t : A)$ and $(\Gamma,\ x : A)$. That is, they are satisfied by exactly the same substitutions, namely

$$(\gamma, x = a),$$

where γ satisfies Γ and $a : A\gamma$. The distinction is used in the definition of the syntactic category

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

of explicit variables of type A in the context Γ. The interpretation of such an explicit variable is an element of A in Γ,

$$a : A/\Gamma.$$

(In fact, it will always be one of the variables declared in Γ.)

The first rule of formation of explicit variables says that whenever a context Γ is extended to $(\Gamma,\ x_t : A)$, we may form an expression of the category $\mathrm{VAR}((\Gamma,\ x_t : A), A)$, a *new variable*, interpreted as x and sugared to t,

$$\frac{\Gamma : \mathrm{cont} \quad A : \mathrm{dom}/\Gamma \quad t : \mathrm{Str}}{\begin{array}{l} \mathrm{NVAR}(\Gamma, A, t) : \mathrm{VAR}((\Gamma,\ x_t : A), A) \\ \mathrm{NVAR}(\Gamma, A, t)^{\star} = x \\ \mathrm{NVAR}(\Gamma, A, t)^{\circ} = t. \end{array}}$$

The explicit variables of a context are inherited by all extensions of the context. This is expressed by the grammatical rule for *ancient variables*,

$$\frac{\Gamma : \mathrm{cont} \quad A, B : \mathrm{dom}/\Gamma \quad v : \mathrm{VAR}(\Gamma, A)}{\begin{array}{l} \mathrm{AVAR}(\Gamma, A, B, v) : \mathrm{VAR}((\Gamma,\ x : B), A) \\ \mathrm{AVAR}(\Gamma, A, B, v)^{\star} = v^{\star} \\ \mathrm{AVAR}(\Gamma, A, B, v)^{\circ} = v^{\circ} \end{array}}$$

This rule is also valid for the extension of Γ by an explicit variable. (In the notation that we are using, there is a slight difficulty in the interpretation of

NVAR(Γ, A, t): it is defined to be x, which is not shown in the syntax tree itself, but only in the type of the tree. In the metamathematical definition of contexts used in the ALF implementation, an extended context is a code of a set ot the form $(\Sigma x : G)A(x)$, and the interpretation of the last variable of the context is the function $(z)q(z)$ defined on this set. The interpretation of an ancient variable AVAR(Γ, A, B, v) is $(z)v^\star(p(z))$.)

The grammatical representation of an English explicit variable, a syntax tree of type VAR(Γ, A), changes as the context grows, although the sugaring does not change. The syntax trees of explicit variables are, in fact, a version of the *indices* of de Bruijn (1972), indicating the *reference depth* of variables: the last, the one before the last, etc.

In de Bruijn's original work, the indices are bare numerals. Their formal definition in type theory forces them to have context and domain arguments, and the string argument permits their sugaring to the symbols actually used in the text. But we can introduce a system of abbreviations reminiscent of de Bruijn's notation, in which AVAR operators are conceived as successors, and the context and domain arguments are hidden:

$$
\begin{aligned}
0(t) &= \mathrm{NVAR}(t), \\
1(t) &= \mathrm{AVAR}(\mathrm{NVAR}(t)), \\
2(t) &= \mathrm{AVAR}(\mathrm{AVAR}(\mathrm{NVAR}(t))), \\
&\text{etc.}
\end{aligned}
$$

4 Mathematical Symbolism in English Text

The variant of English that we are describing could be called, following de Bruijn, the Mathematical Vernacular, "the very precise mixture of words and formulas used by mathematicians in their better moments" (de Bruijn 1994, p. 865). It is possible to use formulae as sentences and terms as proper names, as in many of the examples already given. But it is against the rules of mathematical writing to use English proper names as parts of terms, or sentences as parts of formulae. The mathematical symbolism has its own internal grammar and an interface into the English grammar.

The small fragment of symbolism that we need in arithmetic and geometry comprises formulae (FML) built from terms (TRM) by infix predicates (PRD) such as $=, <, >$,

$$\mathrm{FML} \rightarrow \mathrm{TRM}\ \mathrm{PRD}\ \mathrm{TRM}$$

$$
\frac{\Gamma : \mathrm{cont} \quad A, B : \mathrm{dom} \quad a : \mathrm{TRM}(\Gamma, A) \quad F : \mathrm{PRD}(A, B) \quad b : \mathrm{TRM}(\Gamma, B)}{\begin{array}{l}\mathrm{PRED}(\Gamma, A, B, a, F, b) : \mathrm{FML}(\Gamma) \\ \mathrm{PRED}(\Gamma, A, B, a, F, b)^\star = F^\star(a^\star, b^\star) \\ \mathrm{PRED}(\Gamma, A, B, a, F, b)^\circ = a^\circ\ F^\circ\ b^\circ\end{array}}
$$

Terms are either variables or constants, or built from given terms by various well-known operators, such as prefixes, postfixes, and infixes, or by more specific ones such as the derivative:

TRM $\rightarrow$ TRM INF TRM

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

The interface of the symbolism to English is defined by two rules:

S $\rightarrow$ FML.

$$\frac{\Gamma : \text{cont} \quad A : \text{FML}(\Gamma)}{\begin{array}{l}\text{FORMULA}(\Gamma, A) : \text{S}(\Gamma) \\ \text{FORMULA}(\Gamma, A)^\star = A^\star \\ \text{FORMULA}(\Gamma, A)^\circ = \$A^\circ\$\end{array}}$$

PN $\rightarrow$ TRM.

$$\frac{\Gamma : \text{cont} \quad A : \text{dom}/\Gamma \quad a : \text{TRM}(\Gamma, A)}{\begin{array}{l}\text{TERM}(\Gamma, A, a) : \text{PN}(\Gamma, A) \\ \text{TERM}(\Gamma, A, a)^\star = a^\star \\ \text{TERM}(\Gamma, A, a)^\circ = \$a^\circ\$\end{array}}$$

Sugaring in these rules produces LaTeX code, in which dollar signs are used for delimiting passages typeset in a special "math font".

Some authors prefer to embed symbolic expressions in English text by means of some extra words, writing A *holds* instead of bare A and *the* A a instead of bare a, but we shall here stay content with plain symbolic sentences and proper names. Notice that, in order for the sugaring *the* A° a° of the term proper name to make sense, the premise $A : \text{dom}/\Gamma$ would have to be strengthened to $A : \text{CN}(\Gamma)$.

5 Explicit quantifiers

A common structure in which the context is extended in a mathematical text is an explicit quantifier, either universal or existential:

S $\rightarrow$ *for all* CN VAR, S

$$\frac{\Gamma : \text{cont} \quad A : \text{CN}(\Gamma) \quad t : \text{Str} \quad B : \text{S}((\Gamma,\ x_t : A^\star))}{\begin{array}{l}\text{forall}(\Gamma, A, t, B) : \text{S}(\Gamma) \\ \text{forall}(\Gamma, A, t, B)^\star = (\forall x : A^\star)B^\star \\ \text{forall}(\Gamma, A, t, B)^\circ = \textit{for all}\ A^\circ(\text{pl})\ t, B^\circ\end{array}}$$

S $\rightarrow$ *there exists a* CN VAR *such that* S

$$\frac{\Gamma : \text{cont} \quad A : \text{CN}(\Gamma) \quad t : \text{Str} \quad B : \text{S}((\Gamma,\ x_t : A^\star))}{\begin{array}{l}\text{exists}(\Gamma, A, t, B) : \text{S}(\Gamma) \\ \text{exists}(\Gamma, A, t, B)^\star = (\exists x : A^\star)B^\star \\ \text{exists}(\Gamma, A, t, B)^\circ = \textit{there exists}\ \text{INDEF}(A^\circ)\ t\ \textit{such that}\ B^\circ\end{array}}$$

(In the former rule, the parameter pl of the sugaring of A produces the plural. In the latter rule, the operation INDEF produces the article *a* or *an*, depending on A°.)

Equipped with the two rules of explicit quantification, as well as the rules of progressive implication and conjunction (see Section 2), our fragment of mathematical English can now express any proposition of the form

$$(Q_1 x_1 : A_1)(Q_2 x_2 : A_2) \cdots (Q_n x_n : A_n)A,$$

where each Q_i is either universal or existential, each A_i is expressible by a sentence or by a common noun, and A is expressible by a sentence.

Of course, there are other forms of explicit quantification in natural language, for instance, ones by which whole lists of variables can be declared at the same time. But we can already find an expression for the proposition stated in Section 1,

> *for all functions f, if f is differentiable, then if f is increasing, then for all points x, $f'(x) > 0$.*

Assuming it unproblematic how to define an operator VARPREDA1 with the effect

$$\mathrm{S} \rightarrow \mathrm{VAR}\ \mathit{is}\ \mathrm{AI}$$

(cf. Ranta 1995, sections 7 and 11, for predication rules), we build the syntax tree

$\mathrm{forall}(\mathrm{function}, f, \mathrm{if}(\mathrm{VARPREDA1}(0(f), \mathrm{differentiable}),$
$\mathrm{if}(\mathrm{VARPREDA1}(1(f), \mathrm{increasing}), \mathrm{forall}(\mathrm{point}, x, \mathrm{FORMULA}(\mathrm{PRED}(\Gamma,$
$\mathrm{DER}(\mathrm{VARIABLE}(3(f)), \mathrm{VARIABLE}(0(x)), \mathrm{ap}(z, v)), >, \mathrm{zero}_R))))))$

with the desired interpretation and sugaring. We have hidden the semantic arguments of the tree, except the context of the last sentence, which is decisive for the well-formation of the term $f'(x)$, the four-hypothesis context

$$\Gamma = (y_f : R \rightarrow R,\ z : (\forall x : R)\mathrm{Diff}(y, x),\ u : \mathrm{Incr}(y),\ v_x : R).$$

(Observe, furthermore, the operator VARIABLE, which takes a variable into a term, the commoun noun *point* interpreted as R, the 0 of the reals zero_R, and the interpretation of the adjective differentiable as the property of being everywhere differentiable.)

6 Proof Texts

The proof editor Coq has an interface facility of rewriting formal proofs as English texts (see Coscoy *et al.* 1995). The rewriting procedure is a direct translation of a λ term. As such, it resembles the sugaring procedure presented in Ranta 1994, which takes type-theoretical formulae directly into strings of words. As we now prefer an indirect approach based on syntax trees, we introduce the new syntactic category

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

where Γ : cont and A : prop/Γ, of proofs of the proposition A in the context Γ. A syntax tree of type PROOF(Γ, A) is interpreted as a proof object

$$a : A/\Gamma.$$

The category PROOF(Γ, A) is thus semantically equal to the category PN(Γ, A) of proper names, even though the superficial difference is big: proper names are typically the smallest units of language, whereas proof texts are very large units.

There are lots of rules for building proof texts. What we shall show here are rules corresponding to Gentzen's natural deduction rules for predicate calculus. (Mathematicians, of course, usually prefer much more compressed rules.) We shall mainly follow the wordings of Coscoy *et al.* 1995. Sometimes we shall have to change them because we also sugar the expressions of propositions: Coscoy *et al.* only sugar the proof steps, leaving the propositions formal. The interpretations of the logical constants, as well as the set-theoretical notation, are from Martin-Löf 1984.

All punctuation is defined by the sugaring rules, and so is the use of capitals: passages starting with capital letters are underlined. The font used for English expressions is changed from italic to roman, to conform with the usual practice of mathematical writing.

Conjunction introduction. $\dfrac{A \quad B}{A \& B}$

$$\frac{\Gamma : \mathrm{cont} \quad A, B : \mathrm{S}(\Gamma) \quad a : \mathrm{PROOF}(\Gamma, A^{\star}) \quad b : \mathrm{PROOF}(\Gamma, B^{\star})}{\begin{array}{l}\mathrm{ConjI}(\Gamma, A, B, a, b) : \mathrm{PROOF}(\Gamma, A^{\star} \& B^{\star}) \\ \mathrm{ConjI}(\Gamma, A, B, a, b)^{\star} = (a^{\star}, b^{\star}) \\ \mathrm{ConjI}(\Gamma, A, B, a, b)^{\circ} = a^{\circ}.\ \underline{b^{\circ}}.\ \text{Altogether, } A^{\circ} \text{ and } B^{\circ}\end{array}}$$

Conjunction elimination. $\dfrac{A \& B}{A} \qquad \dfrac{A \& B}{B}$

$$\frac{\Gamma : \mathrm{cont} \quad A : \mathrm{S}(\Gamma) \quad B : \mathrm{prop}/\Gamma \quad c : \mathrm{PROOF}(\Gamma, A^{\star} \& B)}{\begin{array}{l}\mathrm{ConjEl}(\Gamma, A, B, c) : \mathrm{PROOF}(\Gamma, A^{\star}) \\ \mathrm{ConjEl}(\Gamma, A, B, c)^{\star} = p(c^{\star}) \\ \mathrm{ConjEl}(\Gamma, A, B, c)^{\circ} = c^{\circ}.\ \text{A fortiori, } A^{\circ}\end{array}}$$

$$\frac{\Gamma : \mathrm{cont} \quad A : \mathrm{prop}/\Gamma \quad B : \mathrm{S}(\Gamma) \quad c : \mathrm{PROOF}(\Gamma, A \& B^{\star})}{\begin{array}{l}\mathrm{ConjEr}(\Gamma, A, B, c) : \mathrm{PROOF}(\Gamma, B^{\star}) \\ \mathrm{ConjEr}(\Gamma, A, B, c)^{\star} = q(c^{\star}) \\ \mathrm{ConjEr}(\Gamma, A, B, c)^{\circ} = c^{\circ}.\ \text{A fortiori, } B^{\circ}\end{array}}$$

The grammatical rules for conjunction elimination are perhaps stronger than expected: in the left rule, the second premise is $B : \mathrm{prop}/\Gamma$ instead of $B : \mathrm{S}(\Gamma)$, and in the right rule, the first premise is weakened correspondigly. Thus the conjunct that is not the conclusion of the rule can be any proposition in Γ, not necessarily one expressible by a sentence. The reason is that only the conclusion is used in the sugaring rule. Following the principle that syntactic arguments are not deleted in sugaring (Ranta 1995, section 6), we treat the other conjunct as a semantic argument of the rule.

Implication introduction. $$\frac{\begin{matrix}(A)\\ B\end{matrix}}{A \supset B}$$

$$\frac{\Gamma : \mathrm{cont} \quad A, B : \mathrm{S}(\Gamma) \quad t : \mathrm{Str} \quad b : \mathrm{PROOF}((\Gamma,\ x_t : A^\star), B^\star)}{\begin{array}{l}\mathrm{ImplI}(\Gamma, A, B, t, b) : \mathrm{PROOF}(\Gamma, A^\star \supset B^\star)\\ \mathrm{ImplI}(\Gamma, A, B, t, b)^\star = (\lambda x)b^\star\\ \mathrm{ImplI}(\Gamma, A, B, t, b)^\circ = \text{assume}\end{array}}$$

$$A^\circ.\ (t)$$

$$\underline{b^\circ}.\ \text{Hence, if } A^\circ\text{, then } B^\circ$$

Several things are to be observed about implication introduction. First, the hypothesis is marked by an explicit variable, which makes it possible to refer to the assumption later in the course of the proof. Second, the rule is easy to strengthen to having the progressive implication

$$(\Pi x : A^\star)B^\star$$

in its conclusion; the third premise is then

$$B : \mathrm{S}((\Gamma,\ x : A^\star)).$$

(An analogous generalization is possible for the conjunction rules.) Third, Coscoy *et al.* 1995 suggest, for stylistic reasons, the omission of the conclusion from the sugaring. The argument B is then not used in sugaring, and it can be weakened to a semantic argument.

Implication elimination. $$\frac{A \supset B \quad A}{B}$$

$$\frac{\begin{array}{l}\Gamma : \mathrm{cont} \quad A : \mathrm{prop}/\Gamma \quad B : \mathrm{S}(\Gamma)\\ c : \mathrm{PROOF}(\Gamma, A \supset B^\star) \quad a : \mathrm{PROOF}(\Gamma, A)\end{array}}{\begin{array}{l}\mathrm{ImplE}(\Gamma, A, B, c, a) : \mathrm{PROOF}(\Gamma, B^\star)\\ \mathrm{ImplE}(\Gamma, A, B, c, a)^\star = \mathrm{ap}(c^\star, a^\star)\\ \mathrm{ImplE}(\Gamma, A, B, c, a)^\circ = c^\circ.\ \underline{a^\circ}.\ \text{We deduce that } B^\circ\end{array}}$$

Observe, again, that A is a semantic argument not used in sugaring.

Disjunction introduction. $\dfrac{A}{A \vee B} \qquad \dfrac{B}{A \vee B}$

$$\frac{\Gamma : \mathrm{cont} \quad A, B : \mathrm{S}(\Gamma) \quad a : \mathrm{PROOF}(\Gamma, A^\star)}{\begin{array}{l}\mathrm{DisjIl}(\Gamma, A, B, a) : \mathrm{PROOF}(\Gamma, A^\star \vee B^\star) \\ \mathrm{DisjIl}(\Gamma, A, B, a)^\star = i(a^\star) \\ \mathrm{DisjIl}(\Gamma, A, B, a)^\circ = a^\circ.\ \text{A fortiori, } A^\circ \text{ or } B^\circ\end{array}}$$

$$\frac{\Gamma : \mathrm{cont} \quad A, B : \mathrm{S}(\Gamma) \quad b : \mathrm{PROOF}(\Gamma, B^\star)}{\begin{array}{l}\mathrm{DisjIr}(\Gamma, A, B, b) : \mathrm{PROOF}(\Gamma, A^\star \vee B^\star) \\ \mathrm{DisjIr}(\Gamma, A, B, b)^\star = j(b^\star) \\ \mathrm{DisjIr}(\Gamma, A, B, b)^\circ = b^\circ.\ \text{A fortiori, } A^\circ \text{ or } B^\circ\end{array}}$$

Disjunction elimination. $\dfrac{A \vee B \quad \begin{array}{c}(A)\\ C\end{array} \quad \begin{array}{c}(B)\\ C\end{array}}{C}$

$$\frac{\begin{array}{l}\Gamma : \mathrm{cont} \quad A, B : \mathrm{S}(\Gamma) \quad C : \mathrm{S}(\Gamma) \\ c : \mathrm{PROOF}(\Gamma, A^\star \vee B^\star) \\ t : \mathrm{Str} \quad d : \mathrm{PROOF}((\Gamma,\ x_t : A^\star), C^\star) \\ u : \mathrm{Str} \quad e : \mathrm{PROOF}((\Gamma,\ y_u : B^\star), C^\star)\end{array}}{\begin{array}{l}\mathrm{DisjE}(\Gamma, A, B, C, c, t, d, u, e) : \mathrm{PROOF}(\Gamma, C^\star) \\ \mathrm{DisjE}(\Gamma, A, B, C, c, t, d, u, e)^\star = D(c^\star, (x)d^\star, (y)e^\star) \\ \mathrm{DisjE}(\Gamma, A, B, C, c, t, d, u, e)^\circ = c^\circ.\ \text{Hence we have two cases. First, assume}\end{array}}$$

A°. (t)

$\underline{d^\circ}$. Second, assume

B°. (u)

$\underline{e^\circ}$. Thus C°, in both cases

Absurdity elimination. $\dfrac{\perp}{C}$

$$\frac{\Gamma : \mathrm{cont} \quad C : \mathrm{S}(\Gamma) \quad c : \mathrm{PROOF}(\Gamma, \perp)}{\begin{array}{l}\mathrm{AbsE}(\Gamma, C, c) : \mathrm{PROOF}(\Gamma, C^\star) \\ \mathrm{AbsE}(\Gamma, C, c)^\star = R_0(c^\star) \\ \mathrm{AbsE}(\Gamma, C, c)^\circ = c^\circ.\ \text{Hence } C^\circ\end{array}}$$

Universal introduction. $\dfrac{\begin{array}{c}(x : A) \\ B(x)\end{array}}{(\forall x : A)B(x)}$

$$\frac{\Gamma : \text{cont} \quad A : \text{CN}(\Gamma) \quad t : \text{Str} \quad B : \text{S}((\Gamma,\ x_t : A^\star)) \quad b : \text{PROOF}((\Gamma,\ x_t : A^\star), B^\star)}{\begin{array}{ll} \text{UnivI}(\Gamma, A, t, B, b) : \text{PROOF}(\Gamma, (\forall x : A^\star)B^\star) & \\ \text{UnivI}(\Gamma, A, t, B, b)^\star = (\lambda x)b^\star & \\ \text{UnivI}(\Gamma, A, t, B, b)^\circ = & \text{consider an arbitrary } A^\circ\ t.\ \underline{b^\circ}.\ \text{We have proved that, for all } t,\ B^\circ, \text{ since } t \text{ is arbitrary} \end{array}}$$

Just as in the case of implication introduction, Coscoy *et al.* 1995 suggest the omission of the statement of the conclusion. This means the weakening of B to a semantic argument.

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

$$\frac{\Gamma : \text{cont} \quad A : \text{dom}/\Gamma \quad t : \text{Str} \quad B : \text{S}((\Gamma,\ x_t : A)) \quad c : \text{PROOF}(\Gamma, (\forall x : A)B^\star) \quad a : \text{PN}(\Gamma, A)}{\begin{array}{l} \text{UnivE}(\Gamma, A, t, B, c, a) : \text{PROOF}(\Gamma, B^\star(x = a^\star)) \\ \text{UnivE}(\Gamma, A, t, B, c, a)^\star = \text{ap}(c^\star, a^\star) \\ \text{UnivE}(\Gamma, A, t, B, c, a)^\circ = c^\circ.\ \text{In particular, } B^\circ \text{ holds for } t \text{ set to } a^\circ \end{array}}$$

Here there is an alternative sugaring, which is even more natural:

c°. In particular, $B^\circ[a^\circ/t]$

That is, t is replaced by a° in B°. But a can then not be just any proper name: it must be a symbolic term, since the variable symbol t may have occurrences inside formulae. The explicit substitution style is thus more general.

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

$$\frac{\Gamma : \text{cont} \quad A : \text{CN}(\Gamma) \quad t : \text{Str} \quad B : \text{S}((\Gamma,\ x_t : A^\star)) \quad a : A^\star/\Gamma \quad b : \text{PROOF}(\Gamma, B^\star(x = a))}{\begin{array}{ll} \text{ExistI}(\Gamma, A, t, B, a, b) : \text{PROOF}(\Gamma, (\exists x : A^\star)B^\star) & \\ \text{ExistI}(\Gamma, A, t, B, a, b)^\star = (a, b^\star) & \\ \text{ExistI}(\Gamma, A, t, B, a, b)^\circ = & b^\circ.\ \text{Thus there exists INDEF}(A^\circ)\ t \text{ such that } b^\circ \end{array}}$$

Observe that a is a semantic argument not used in sugaring.

Existential elimination. $$\frac{(\exists x : A)B(x) \quad \begin{array}{c}(x : A, B(x)) \\ C\end{array}}{C}$$

$$\frac{\Gamma : \text{cont} \quad A : \text{CN}(\Gamma) \quad t : \text{Str} \quad B : \text{S}((\Gamma,\ x_t : A^\star)) \qquad C : \text{S}(\Gamma) \quad c : \text{PROOF}(\Gamma, (\exists x : A)B^\star) \qquad u : \text{Str} \quad d : \text{PROOF}(((\Gamma,\ x_t : A^\star),\ y_u : B^\star), C^\star)}{\begin{array}{l}\text{ExistE}(\Gamma, A, t, B, C, c, u, d) : \text{PROOF}(\Gamma, C^\star) \\ \text{ExistE}(\Gamma, A, t, B, C, c, u, d)^\star \ = \ E(c^\star, (x, y)d^\star) \\ \text{ExistE}(\Gamma, A, t, B, C, c, u, d)^\circ = \ c^\circ.\ \text{Consider an arbitrary } A^\circ\ t \text{ such that} \\ \qquad\qquad\qquad\qquad\qquad\qquad B^\circ.\ (u) \\ \qquad\qquad\qquad\qquad \underline{d^\circ}.\ \text{Thus } C^\circ\text{, independently of } t\end{array}}$$

We have now shown how proof texts are built from smaller proof texts by rules that correspond to the natural deduction rules of predicate calculus. Sometimes a proof required as a constituent is just a proof by assumption, which is provided by the following rule.

Assumption.

$$\frac{\Gamma : \text{cont} \quad A : \text{S}(\Gamma) \quad v : \text{VAR}(\Gamma, A^\star)}{\begin{array}{l}\text{Ass}(\Gamma, A, v) : \text{PROOF}(\Gamma, A^\star) \\ \text{Ass}(\Gamma, A, v)^\star \ = \ v^\star \\ \text{Ass}(\Gamma, A, v)^\circ = \ \text{by the assumption } v^\circ,\ A^\circ\end{array}}$$

It is certainly not common to build proofs with so small steps as the rules of natural deduction. To generate more advanced proofs, we need lots of compressed rules, beginning from double and triple universal introduction, Modus Tollens, de Morgan laws, etc. We also need other rules than those concerning logical constants. As an example, consider the proof rules corresponding to the inductive definition of Even and Odd,

$$\text{Even}(0) \qquad \frac{\text{Even}(a)}{\text{Odd}(s(a))} \qquad \frac{\text{Odd}(a)}{\text{Even}(s(a))}$$

$$\frac{\Gamma : \text{cont}}{\begin{array}{l}\text{evax1}(\Gamma) : \text{PROOF}(\Gamma, \text{Even}(0)) \\ \text{evax1}(\Gamma)^\star \ = \ \text{evz} \\ \text{evax1}(\Gamma)^\circ = \ \text{by the first axiom of evenness, 0 is even}\end{array}}$$

$$\frac{\Gamma : \text{cont} \quad a : \text{PN}(\Gamma, N) \quad b : PROOF(\Gamma, \text{Even}(a^\star))}{\begin{array}{l}\text{evax2}(\Gamma, a, b) : \text{PROOF}(\Gamma, \text{Odd}(0)) \\ \text{evax2}(\Gamma, a, b)^\star \ = \ \text{ods}(a^\star, b^\star) \\ \text{evax2}(\Gamma, a, b)^\circ = \ b^\circ.\ \text{By the second axiom of evenness, the} \\ \qquad\qquad\qquad\qquad \text{successor of } a^\circ \text{ is odd}\end{array}}$$

$$\frac{\Gamma : \mathrm{cont} \quad a : \mathrm{PN}(\Gamma, N) \quad b : PROOF(\Gamma, \mathrm{Odd}(a^\star))}{\begin{array}{l} \mathrm{evax3}(\Gamma, a, b) : \mathrm{PROOF}(\Gamma, \mathrm{Even}(0)) \\ \mathrm{evax3}(\Gamma, a, b)^\star = \mathrm{evs}(a^\star, b^\star) \\ \mathrm{evax3}(\Gamma, a, b)^\circ = b^\circ. \text{ By the third axiom of evenness, the suc-} \\ \qquad\qquad\qquad\quad \text{cessor of } a^\circ \text{ is even} \end{array}}$$

(The three natural deduction rules are formalized by the three constants evz, ods, and evs.)

As the last example, consider the principle of mathematical induction, in the usual form, which compresses together type-theoretical N elimination and universal introduction:

Mathematical induction.
$$\frac{C(0) \qquad \begin{array}{c} (x : N, C(x)) \\ C(s(x)) \end{array}}{(\forall x : N)C(x)}$$

$$\frac{\begin{array}{l} \Gamma : \mathrm{cont} \quad t : \mathrm{Str} \quad C : \mathrm{S}((\Gamma,\ x_t : N)) \\ d : \mathrm{PROOF}(\Gamma, C^\star(x = 0)) \\ u : \mathrm{Str} \quad e : \mathrm{PROOF}(((\Gamma,\ x_t : N),\ y_u : C^\star), C^\star(x = s(x))) \end{array}}{\begin{array}{l} \mathrm{Ind}(\Gamma, t, C, d, u, e) : \mathrm{PROOF}(\Gamma, (\forall x : N)C^\star) \\ \mathrm{Ind}(\Gamma, t, C, d, u, e)^\star = (\lambda x)R(x, d^\star, (x, y)e^\star) \\ \mathrm{Ind}(\Gamma, t, C, d, u, e)^\circ = \text{we proceed by induction. First, } d^\circ. \text{ Second,} \\ \qquad\qquad\qquad\qquad\quad \text{consider an arbitrary } t \text{ such that} \\ \qquad\qquad\qquad\qquad\qquad\qquad\quad C^\circ.\ (u) \\ \qquad\qquad\qquad\qquad\quad \underline{e^\circ}. \text{ So we have proved that, for all } t,\ C^\circ \end{array}}$$

7 Proof Text Editor

We have defined a system of syntax trees together with sugarings into English and interpretations as mathematical objects. But we have not shown how to perform *parsing*—to take English strings into syntax trees—and *phrasing*—to take mathematical objects into syntax trees. We have already noticed that sugaring and interpretation destroy information, so that parsing and phrasing are, essentially, search procedures. They can have multiple outcomes or none at all. (Phrasing, typically, has multiple outcomes: a given mathematical object can be expressed in several ways. For parsing, the opposite is typical, since there are so many meaningless strings.) But it should be possible to define search algorithms for limited fragments as solutions to the parsing and phrasing problems, which can be stated in type theory as follows:

$$\mathrm{list}((\exists y : \mathrm{S}(()))I(\mathrm{Str}, y^\circ, x)) \ \ (x : \mathrm{Str}),$$
$$\mathrm{list}((\exists y : \mathrm{S}(()))I(\mathrm{prop}, y^\star, x)) \ \ (x : \mathrm{prop}).$$

A translation of a proof term into a text could then be defined as phrasing followed by sugaring.

But there is another way of seeing the task of generating proof text, namely as an interactive process. Users of Coq and ALF are used to building proof terms interactively, rather than striving after automatic theorem provers. Now, if someone wants to get not only a formal proof but a corresponding text, he can try to build a syntax tree of type PROOF interactively. From this, he will obtain both a text and a formal proof. He can make decisions concerning the text, so that he can, for example, choose different expressions for one and the same proposition in different places, to avoid monotony. The proof text editor checks both mathematical and grammatical correctness all the time.

The top level of the proof text editor can be defined as the function

$$\begin{array}{rcl}\mathrm{ThmWithProof} : (A : \mathrm{S}(()))(a : \mathrm{PROOF}((), A^{\star}))\mathrm{Str} & & \\ \mathrm{ThmWithProof}(A, a) & = & \text{Theorem. } \underline{A^{\circ}}. \\ & & \text{Proof. } \underline{a^{\circ}}. \end{array}$$

The user can start editing a proof text by refining an unspecified string with ThmWithProof. He then gets the subgoals

$$A = ? : \mathrm{S}(()),\ a = ? : \mathrm{PROOF}((), A^{\star}).$$

The example shown below has been created by using ALF in this way. The corresponding proof tree is

$$\dfrac{\dfrac{E(0)}{E(0)\vee O(0)}\,\vee Il \qquad \dfrac{\begin{matrix}3.\\ E(x)\vee O(x)\end{matrix} \quad \dfrac{\begin{matrix}1.\\ \dfrac{E(x)}{O(s(x))}\,\mathrm{ods}\end{matrix}}{E(s(x))\vee O(s(x))}\,\vee Ir \quad \dfrac{\begin{matrix}2.\\ \dfrac{O(x)}{E(s(x))}\,\mathrm{evs}\end{matrix}}{E(s(x))\vee O(s(x))}\,\vee Il}{E(s(x))\vee O(s(x))}\,\vee E, 1., 2.}{(\forall x : N)(E(x)\vee O(x))}\,\mathrm{Ind}, 3.$$

Acknowledgements

I am grateful to Yann Coscoy and to Gilles Kahn for the stimulating exchange of ideas.

Example Proof Text Produced by Machine

Theorem. Every number is even or odd.

Proof. We proceed by induction. First, by the first axiom of evenness, 0 is even. A fortiori, 0 is even or 0 is odd. Second, consider an arbitrary number n such that

$$n \text{ is even or odd. } (a)$$

By the assumption a, n is even or odd. So we have two cases. First, assume

$$n \text{ is even. } (b)$$

By the assumption b, n is even. By the second axiom of evenness, the successor of n is odd. A fortiori, the successor of n is even or the successor of n is odd. Second, assume

$$n \text{ is odd. } (c)$$

By the assumption c, n is odd. By the third axiom of evenness, the successor of n is even. A fortiori, the successor of n is even or the successor of n is odd. Hence, the successor of n is even or odd in both cases. So we have proved that, for all n, n is even or odd.

References

N.G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae* 34, pages 381–392, 1972. Reprinted in R. Nederpelt, editor, *Selected Papers on Automath*, pages 375–388. North-Holland, Amsterdam, 1994.

N.G. de Bruijn. The mathematical vernacular, a language for mathematics with typed sets. R. Nederpelt, editor, *Selected Papers on Automath*, pages 865–935. North-Holland, Amsterdam, 1994.

Yann Coscoy, Gilles Kahn and Laurent Théry. Extracting text from proofs. Rapport de recherche n.2459, INRIA, Sophia-Antipolis, 1995.

Lena Magnusson and Bengt Nordström. The ALF Proof Editor and Its Proof Engine. In H. Barendregt and T. Nipkow, editors, *Types for Proofs and Programs*, pages 213–237. *Lecture Notes in Computer Science* 806, Springer-Verlag, Heidelberg, 1994.

Per Martin-Löf. *Intuitionistic Type Theory*. Bibliopolis, Naples, 1984.

Richard Montague. *Formal Philosophy*. Yale University Press, New Haven, 1974. Collected papers edited by Richmond Thomason.

Aarne Ranta. *Type Theoretical Grammar*. Oxford University Press, Oxford, 1994.

Aarne Ranta. Syntactic categories in the language of mathematics. In P. Dybjer, B. Nordström, and J. Smith, editors, *Types for Proofs and Programs*, pages 162-182, *Lecture Notes in Computer Science* 996, Springer-Verlag, Heidelberg, 1995.

Alvaro Tasistro. *Formulation of Martin-Löf's Monomorphic Theory of Types with Explicit Substitutions*, Licenciate's thesis. Department of Computing Science, University of Göteborg, 1993.

This article was processed using the LaTeX macro package with LLNCS style