

## Chapter 8

# Extensions

There are a large number of minor topics in the language of mathematics which do not have space to describe in depth. §8.1 and §8.2 give very brief outlines of our approach to these miscellaneous topics in textual and symbolic mathematics respectively.<sup>1</sup> §8.3 discusses a final major topic which we intend to present fully in a separate paper.

### 8.1 Textual Extensions

#### *Anaphora*

We note that the typed parsing algorithm of Chapter 6 is fully capable of using type information to determine anaphoric antecedents. Notwithstanding this, we present evidence suggesting that examples where this is needed (as, for example, are given in (Ranta, 1994, p. 8)) are rare in real texts.

#### *Definite Descriptions*

We follow the DRT-specific approach of van der Sandt (1992) closely. In particular, we treat the presuppositions associated with definite descriptions by introducing generalised anaphors which can have presuppositional material attached to them. We note that the distinction between binding and accommodation is misleading when applied to mathematics; in all cases, the appropriate way in which to treat presuppositions attached to definite descriptions is to verify that an object with the appropriate properties exist.

---

<sup>1</sup>These may be expanded on in the version of this thesis submitted for a PhD, depending on the availability of a word limit extension.

If there is an explicit antecedent to which one may bind, this is trivially the appropriate object; if not, the object is not explicitly available and a non-trivial proof may be required to verify that it exists.

### *Prepositional Phrases*

We present an extension of our combined syntactic-semantic rules to allow slots which are tagged with prepositions as well as slots that are tagged with numbers, and show how this may be used to handle prepositional phrases. We recall that the typed parsing algorithm algorithm can resolve PP attachment ambiguity using type information, and note that because our analysis of prepositional phrases does not overgenerate, this resolves all cases of PP attachment ambiguity that we have encountered in texts. Finally, we discuss the way in which prepositional phrases interact with the mechanism of adaptivity.

A

We note that the meaning of the ubiquitous determiner ‘a’ can vary in meaning depending on its position in the sentence, as in

**A** [every] domain is simply connected.

0 is **a** [some/one] natural number.

We relate this to genericity, and present an extension of our compositional framework to handle this.

## 8.2 *Symbolic Extensions*

### *Higher-Order Constructs*

We note that mathematicians very rarely make explicitly higher-order remarks, preferring to rephrase said remarks in terms of Zermelo-Fraenkel set theory. We note that the most significant instances in which such rephrasing is not possible are the ZF Axioms of Separation and Replacement themselves; we then consider the presentation of those axioms in a range of textbooks and note that these are all informal or incorrect to some degree. We introduce and illustrate a formal mechanism for describing higher-order constructs, designed to fit seamlessly into real mathematical texts.

We then note that certain notation is invariably defined for specific mathematical objects but subsequently applied to arbitrary terms; for example,

$\lim_{x \rightarrow \infty} \bullet$  is formally defined on functions, but subsequently may be applied to arbitrary terms. We relate this ‘superficially higher-order’ notation to the formal mechanisms previously introduced.

Finally, we discuss two options for the semantic translation of higher-order material and conclude that a representation involving bound variables is preferable to one involving  $\lambda$ -terms on several distinct grounds.

### *Ellipsis*

We analyse expressions like ‘ $1^2 + 2^2 + \dots + n^2$ ’. We present examples showing that this notation cannot be extensional. We also show that both the dependence of the ‘ $\dots$ ’ notation on the natural numbers and the existence of finite and infinite variants of ‘ $\dots$ ’ mean that the notation must be analysed as being introduced by a definition, rather than being built in to our analysis of mathematical language itself. We show that the same mechanism used to handle higher-order constructs can be used to introduce ellipsis via definitions, and present concrete definitions to introduce finite and infinite variants of ellipsis.

### *Chained Infix Formal Relations*

We note that ‘overlapping’ mathematical formulae may be compressed into a single long formula, as in the use of:

$$a_0 < a_1 < a_2 < a_3$$

for

$$a_0 < a_1 \text{ and } a_1 < a_2 \text{ and } a_2 < a_3.$$

We use the illegal example ‘ $*a < b > c$ ’ to demonstrate that this usage is restricted, and present conditions that license it. We indicate how the theoretical framework is expanded to handle this construct, and how it interacts with the type system.

### *Term Lists*

We observe that comma-separated lists of terms may be used to abbreviate similar formulae or sentences, as when ‘ $a, b \in \mathbb{N}$ ’ is a shorthand for ‘ $a \in \mathbb{N}$  and  $b \in \mathbb{N}$ ’. We use illegal examples such as ‘ $x \in a, b$ ’ to suggest that this may only be done at the start of a formula. We present a simple way of handling term lists within the framework of Chapter 3, without extending it, by introducing a single grammatical category.

*Exposed Formulae*

*Exposed formulae* are formulae that are used as if they were terms, as in:

For every real number  $\epsilon > 0$ , there exists a real number  $\delta > 0$   
such that ...

We note that these always occur at the textual/symbolic boundary: the exposed formula must be used as a noun phrase, not a term. (Cf. ‘\*Then  $\epsilon \in \mathbb{R} > 0$ ’.) We also show that exposed formulae must start with a variable, like  $\epsilon$ ,  $\delta$  above. (Cf. ‘\*For every  $0 < \epsilon$ , ... ’.) We show that the same grammatical category used to handle term lists also enables us to handle exposed formulae without altering the theoretical framework.

*Concrete Numbers*

We handle concrete numbers such as ‘114’ and ‘3.1415’ by introducing an invisible marker ‘ $\oplus_d$ ’ between adjacent digits. Syntactic and semantic interpretation of  $\oplus_d$  may then be introduced for specific domains via domains. We demonstrate that this allows material after the decimal point to be interpreted right-to-left, as in ‘ $.1(415) = .1 + (.415)/10$ ’ and material before the decimal point be parsed left-to-right, as in ‘ $(114)2 = 114 \times 10 + 2$ ’. We also observe that this allows tensors, such as  $g$  in, say, general relativity, to support the notation  $g_{11}$ , where ‘11’ does not indicate the number eleven but two distinct subscripts, ‘1’ and ‘1’.

*Symbols and Symbol Conventions*

We note that in individual mathematical texts, authors may explicitly state that particular *symbols* are conventionally used to indicate specific kinds of object, as when e.g. ‘ $p$ ’ is used to denote primes. We note that such conventions carry across to *derived symbols* such as ‘ $p'$ ’ and ‘ $p_0$ ’. We conclude that some formal analysis of the structure of symbols is necessary, and present such an analysis. Finally, we present formal mechanisms in the projected language for handling symbols and conventions, chosen to resemble the language used elsewhere in mathematical texts.

### 8.3 Further Work: Covert Arguments

This section sketches material that we intend to present in a separate paper; it places particular emphasis on showing that the problem it discusses is separable from the material discussed in the main part of the thesis.

*Motivation*

As we noted in §5.3.3, it is not necessary to have dependent types in order to understand the linguistic structure of a piece of mathematics. That is, we never need types like `ELEMENT OF THE GROUP  $G$`  or  `$n \times n$  MATRIX OF NUMBERS` in order to determine the syntax of a piece of mathematics, or to resolve the overloading of notation. Nevertheless, there are cases where it is desirable to encode in the semantic representation the kind of information that might be carried by dependent types (such as the group in which an element  $g$  lies, or the size of a matrix). The classic example is when one encounters an expression

$$gg'$$

where both  $g$  and  $g'$  are elements of some group  $G$ . Here one would like to obtain as semantic representation not simply

$$\text{group\_product}(\mathbf{g}, \mathbf{g}')$$

but rather

$$\text{group\_product}(\mathbf{g}, \mathbf{g}', \mathbf{G}).$$

That is, one would like to encode as part of the semantic representation the group in which the product is evaluated.

We know of a clear (and, to our knowledge, novel) diagnostic characterising precisely those cases in which one wishes to infer information of this kind. The diagnostic is as follows:

Suppose that one encounters a definition with the essential form (§5.4.1):

LHS is defined to be RHS.

or

We say that LHS if RHS.

If the RHS refers to a variable available from the context (§2.5, §3.5) and that variable does not appear on the LHS, then the variable corresponds to an invisible *covert argument* in the semantic function or predicate being defined.

For example, consider one possible definition of the product of elements of a group:

If  $g$  and  $g'$  are elements of a group  $G$ , then we will write  $gg'$  for  $m(g, g')$ , where  $m$  is the multiplication function of  $G$ .

Here the right hand side refers to a variable  $G$  from the context, but left-hand side ( $'gg'$ ) does not. Thus  $G$  corresponds to a covert argument in *group-product*.

Given the above diagnostic, it is easy to find further examples of covert arguments. For example, the trace of a matrix may be defined as follows:

Let  $A$  be a  $n \times n$  square matrix  $A$ . The trace of  $A$  is defined to be

$$\text{Tr}(A) = \sum_{i=1}^n A_{ii}$$

Here both the new N' 'trace [of  $\bullet$ ]' and the new piece of notation ' $\text{Tr}(A)$ ' rely on  $n$  for their meaning but do not have it as part of their syntactic realisation. Thus both term and notation refer to a covert argument. Similarly, when defining the product of a  $n \times k$  matrix and a  $k \times m$  matrix one finds that all of  $n, k, m$  appear as covert arguments.

### Framework

In the typed parsing process, we simultaneously derived and used a large amount of semantic information because it was necessary to do so; as we showed in detail in Chapter 4 and elsewhere, this information was necessary to derive the syntactic structure of a piece of mathematics. As we have noted, information about covert arguments are *not* needed to deduce syntactic structure; it is never the case that a piece of mathematics will have a different structure depending on, say, which group a product is evaluated in, or the size of matrices being multiplied. Even when one forms illegal expressions — for example by trying to multiply object from different groups or by trying to multiply matrices of incompatible sizes — this can simply be analysed as presupposition failure; we know of no case where illegal operations of this kind should be analysed as having a fundamentally different reading. Note in this respect that it is crucial that *only* objects which have no fundamental type may have a relational type (§5.4.3). This means that, for example, the product of a vector  $\underline{x}$  and a matrix  $A$  cannot be analysed as a product of ELEMENTS OF SOME GROUP because  $\underline{x}$  and  $A$  have fundamental types.

As information about covert arguments is never needed to determine syntactic structure, we do not deduce it as part of the typed parsing process. This is partly for parsimony, but it also ensures that expressions which have the same meaning are always treated in the same way by our approach to covert arguments, regardless of their syntactic form; this is an attractive property.

Our first step in handling covert arguments is to introduce extra slots in the semantic functions and predicates. The typed parsing process will simply fill such slots with placeholders which we will denote ‘-’. So, for example, ‘ $gg$ ’ will be translated into

$$\text{group\_product}(\mathbf{g}, \mathbf{g}', -).$$

We then deduce the identity of the covert arguments purely by examining the semantic representations, using means sketched out below.

Note that the use of placeholders is a form of underspecification; the semantic representations produced by the main theory are underspecified with respect to covert arguments. The key point here is that this use of underspecification *separates* the problem of determining covert arguments from the material discussed in the body of this thesis: each can be considered entirely independently of the other.

### *Subtleties*

Now that we have shown that the problem of determining covert arguments is separable from the material discussed in this thesis, we will sketch the remaining points more rapidly. We will concentrate on the example involving products in a group. First, some quick points:

**Presentation.** For exactly the same reasons given in §5.3.4, i.e. because one can erect a group structure on any set, we are interested not in whether an object is an element of a group, but in whether it has been *explicitly presented as* an element of a group.

**Reasoning.** Consider the case where we have subgroups  $H$  and  $K$  of a group  $G$ . If we see the expression ‘ $hk$ ’, where  $h \in H$  and  $k \in K$ , we need to interpret this as a product in the group  $G$ , not a product in either  $H$  or  $K$ . In particular, we cannot only rely on the way in which  $h$  and  $k$  have been directly presented as elements of  $H$  and  $K$  respectively; we also need to utilise the fact that  $H$  and  $K$  have been explicitly presented as subgroups of  $G$ . In other words, some degree of *reasoning* is required.

**Ambiguity.** If  $H$  is a subgroup of a group  $G$ , and we see a reference to ‘ $hh'$ ’, where  $h, h' \in H$ , then the product may be interpreted either in  $H$  or in  $G$ . i.e. we have ambiguity with regard to what the covert argument should be. In this case the ambiguity is spurious: ‘ $hh'$ ’ has the same meaning regardless of which candidate is adopted as the covert argument.

The next point is crucial. Consider the following example:

Assume that we have a sequence of groups  $G_1, G_2, \dots$ , such that  $G_n$  is a subgroup of  $G_m$  if and only if  $n$  divides  $m$ . (An example of this kind genuinely occurs when one is giving a concrete construction of the algebraic closure of the  $p$ -adic numbers.) Now, suppose that  $g_k \in G_k$  for each  $k$ , and we see a reference to the product ' $g_n g_m$ '. In what group(s) does the multiplication take place? The answer is that the multiplication may be interpreted in  $G_k$  where  $k$  is any multiple of the highest common factor of  $n$  and  $m$ .

The key point of this example is that although in *most* cases determining covert arguments requires very little reasoning, it is difficult to put a hard upper bound on the strength of reasoning that is required. We expect there to be very few cases which require us to compute something more complicated than a highest common factor, but we are not confident in asserting this as a uniform upper bound.

### *Mechanism*

Stepping back, we can see that we have exactly the same situation we encountered in §6.5. In particular:

1. There is no theoretical bound on how difficult examples may become. One can contrive an example where  $G_n$  is a subgroup of  $G_m$  if and only if  $P(n, m)$  holds, where  $P$  is an arbitrary property. (Examples like this will never occur in practice but are completely legitimate from a mathematical point of view.) Interpreting ' $g_n g_m$ ' then involves deducing arbitrary facts about  $P$ . I.e. the general problem is halting equivalent.
2. There is no hard, uniform upper bound on the strength of reasoning that is required to resolve examples that occur *in practice*.
3. On the other hand, the required strength of reasoning tails off rapidly relative to the complexity of examples; one finds fewer and fewer instances of harder and harder examples.
4. From a *computational* perspective, we would like a mechanism that a) resolves the vast majority of practical cases and b) detects extremely hard or contrived cases rather than beginning an extremely expensive computation (i.e. the mechanism should be 'resilient to sabotage').

The mechanism we adopt will tackle this problem in a very similar way to the mechanism of §6.5; the major difference will be that when it detects an excessively hard problem it will not produce an error, but fall back to

an implicit specification of the covert argument using modal logic. The key features of the mechanism are as follows:

1. In examples like the one above, involving  $G_1, G_2, \dots$ , such that  $G_n$ , there are too many mathematical objects for us to keep track of facts about each one individually. Instead of doing so we will keep track of facts about *intensions*, like ' $G_\bullet$ '; each intension will refer to a family of objects.
2. For each intension, and each explicitly presented relationship that might be relevant in determining covert arguments (such as ' $-$  is a member of the group  $-$ ' or ' $-$  is a subgroup of  $-$ '), we keep track of upper and lower bounds for the collection of relationships. (The reference to 'relevance' of relationships is a rough expression of a technical property.) Note that if one is not dealing with sequences of groups, etc., but simply a small number of individual objects like  $G, H$  and  $K$ , then we are keeping track of all relevant relationships between individual objects.
3. We ensure that these relationships remain closed under inference; for example, if we know that ' $h$  is a member of the group  $H$ ' and ' $H$  is a subgroup of  $G$ ', we deduce that  $h$  is a member of the group  $G$ .
4. Suppose that we have conditional inferences, i.e. we have (explicitly presented) facts that look like:

If  $g$  is a member of the group  $G$  and  $H$  is a subgroup of  $G$   
and [*arbitrary fact holds*], then  $g$  is a member of  $H$ .

where the arbitrary fact is *not* a relevant relationship. When we encounter concrete examples where this rule might apply, we will not attempt to determine whether the arbitrary fact holds. Instead we will allow the lower and upper bounds to diverge from each other: we will add ' $g$  is a member of  $H$ ' to the upper bound for the collection of relationships, but *not* to the lower bound. By doing so we encode the fact that ' $g$  *might* be a member of  $H$ '.

5. The separation of upper and lower bounds also arises when one has definite information about a single object that cannot be encoded at the level of the intension; for example, a definite fact about  $G_2$  needs to be encoded as the fact that might be true about  $G_\bullet$ .

When we come to try to fill in a covert argument, we take the relevant relationships, list all the relevant candidates, and extract the lower and upper bounds for them. For example, when we encounter  $gg'$  and know that both  $g$  and  $g'$  have the type ELEMENT OF SOME GROUP, we extract the lower and upper bounds for the relationships

$g$  is a member of the group –

and

$g'$  is a member of the group –.

Effectively, we find which groups  $g$  is *certainly* an element of, and which groups  $g$  *might* be an element of, and similarly for  $g'$ .

We then determine from these relationships a collection of objects that *can* or *might* be used as the covert argument. In this case, this involves determining which groups multiplication certainly could take place in and which groups multiplication might take place in. Suppose that the lower and upper bounds match, i.e. that we are sure of exactly what the candidate groups are. In this case, we construct a *presupposition* to the effect that it does not matter which candidate we choose, and take one as the covert argument. (So, for example, in the example given under the heading of ‘Ambiguity’ above, where  $h, h' \in H$  and  $H \leq G$ , we would construct a presupposition to the effect that ‘ $h *_H h' = h *_G h'$ ’.) Where we are dealing with intensions like ‘ $G_\bullet$ ’, we universally quantify over the intension, for example presupposing that we can interpret the product in any  $G_n$  and the choice of  $n$  does not matter.

In the case where the lower and upper bounds do not match, we fall back to a different method involving modal logic. Suppose we have an atomic predicate  $P(x_1, x_2, \dots, x_k, -)$  involving a covert argument, and that we have definite candidates  $d_1, d_2, \dots, d_n$ , and possible candidates  $p_1, p_2, \dots, p_m$  for that covert argument. It turns out that we can construct predicates  $R_i$  such that  $p_i$  is a candidate if and only if  $R(x_1, x_2, \dots, x_k, p_i)$  *necessarily* holds.

For example, suppose that we have a subgroup  $H$  of  $G$  and elements  $g, g' \in G$ .  $G$  is a definite candidate for the group in which the product  $gg'$  is evaluated; suppose also that we know that  $H$  is a possible candidate (due to some conditional inference). Then  $H$  is an actual candidate if and only if we could have proved that both  $g, g'$  lie in  $H$  (using sufficiently strong reasoning). But being able to prove that  $g$  and  $g'$  lie in  $H$  is equivalent to  $g$  and  $g'$  lying in  $H$  in all models;<sup>2</sup> and that in turn can be encoded as ‘ $\Box(g, g' \in H)$ ’, where  $\Box$  is the ‘necessary’ operator from modal logic.

<sup>2</sup>Or at least, this is true up to considerations of undecidability, which turn out to be irrelevant here.

Once we have the  $R_i$ , we may simply replace  $P(x_1, x_2, \dots, x_k, -)$  with  $\exists h.P(x_1, x_2, \dots, x_k, h)$ , where  $h$  carries the presupposition that:

$$\begin{aligned}
P(x_1, x_2, \dots, x_k, h) &\Leftrightarrow P(x_1, x_2, \dots, x_k, d_1) \wedge \\
P(x_1, x_2, \dots, x_k, h) &\Leftrightarrow P(x_1, x_2, \dots, x_k, d_2) \wedge \\
&\vdots \\
P(x_1, x_2, \dots, x_k, h) &\Leftrightarrow P(x_1, x_2, \dots, x_k, d_n) \wedge \\
(\Box R_1(x_1, x_2, \dots, x_k, p_i) \Rightarrow (P(x_1, x_2, \dots, x_k, h) \Leftrightarrow P(x_1, x_2, \dots, x_k, p_1))) &\wedge \\
(\Box R_2(x_1, x_2, \dots, x_k, p_i) \Rightarrow (P(x_1, x_2, \dots, x_k, h) \Leftrightarrow P(x_1, x_2, \dots, x_k, p_2))) &\wedge \\
&\vdots \\
(\Box R_m(x_1, x_2, \dots, x_k, p_i) \Rightarrow (P(x_1, x_2, \dots, x_k, h) \Leftrightarrow P(x_1, x_2, \dots, x_k, p_m))) &
\end{aligned}$$

I.e. we assert that it makes no difference which of the definite candidates is chosen, as all are provably equivalent; and we assert that it makes no difference which of the *actual* potential candidates is chosen, as all are provably equivalent. The key operation we have used is to displace the (potentially arbitrary hard) problem we need to decide into the logical representation, rather than actually attempting to solve that problem inside our linguistic theory.

The case where the covert argument appears in a term is essentially identical. First one removes functions from the semantic representation by introducing extra variables (essentially reversing the process of §3.3.3), so that e.g.  $f(g(x)) = y$  is replaced with

$$\exists a.(a = g(x) \wedge \exists b.(b = f(a) \wedge b = y)).$$

Then one handles each expression of the form ‘ $a = f(x_1, x_2, \dots, x_{k-1}, -)$ ’ (involving a  $k$ -ary function  $f$ ) that includes a covert argument exactly as if it were a  $(k + 1)$ -ary predicate ‘ $P_f(a, x_1, x_2, \dots, x_{k-1}, -)$ ’.

### Notes

This approach is extremely resistant to artificially hard problems (i.e. ‘sabotage’) in exactly the same way as our method in §6.5. Suppose, for example, that we set up groups  $H$  and  $K$  such that  $K$  is a subgroup of  $H$  if and only if the Riemann hypothesis is true. This will cause lower and upper bounds to diverge from each other but will not result in any expensive computations or other problematic behaviour. If the divergence of the bounds becomes relevant, i.e. if we have a case in which a multiplication *might* be interpreted in  $K$  depending on whether or not the Riemann hypothesis is true, this will simply be encoded into a piece of modal logic that essentially asserts:

If the Riemann hypothesis is provable (i.e. necessarily true), then multiplication in  $K$  must give the same answer as multiplication in  $H$ .

It is worth emphasising that we do need the operator  $\square$ . To see this, suppose that we are considering not the Riemann hypothesis but some property  $P(H, K)$  involving  $H$  and  $K$ . It may be that  $H$  and  $K$  are entirely unrelated, and so  $P(H, K)$  is not provable, but that  $P(H, K)$  holds *in some models*; for example, the groups  $C_2$  and  $SO_3$  are unrelated, but happen to have elements in common *in some models*. Nevertheless, if  $P(H, K)$  is false, we do not want to check that

If  $P(H, K)$  holds *in some model*, then *in that model* multiplication in  $K$  must give the same answer as multiplication in  $H$ .

Rather, because  $H$  and  $K$  are unrelated,  $K$  is irrelevant; we want to say nothing about it at all. (To continue with our example, just because the base sets of  $C_2$  and  $SO_3$  may overlap in some models, we do not want to force them to have compatible group structures in those models.) Thus we *must* check that the property  $P(H, K)$  is provable, i.e. holds in all models, before licensing  $K$  as a candidate in any model; and to do this, we need the operator  $\square$ .

It is also worth noting that we have simplified the above description slightly regarding the treatment of spurious ambiguity. If we have been explicitly shown facts like

If  $H$  is a subgroup of  $G$ , then  $h *_H h' = h *_G h'$ .

then we can use these facts to avoid constructing presuppositions. The key to doing this is grouping the candidates for the covert argument into equivalence classes with respect to the relevant notion. For example, in our ' $hh'$ ' case, when we need to list the candidates for the group in which the multiplication takes place, we should not state that

$H$  is a definite candidate;  $G$  is a definite candidate,

but rather that

$H$  or  $G$  is a definite candidate (and it makes no difference which we choose).

This approach can also reduce the number of cases in which we need to fall back to modal logic. For example, if we have a state

$G$  is a definite candidate;  $H$  *might* be a candidate,

and we know that  $H$  and  $G$  give the same result, then we can reduce to the state where

$G$  or  $H$  is a definite candidate (and it makes no difference which we choose).

Here fusion of the possible candidate  $H$  with the definite candidate  $G$  has squeezed out the modality: we know *for certain* that ' $G$  or  $H$ ' is a definite candidate, and so modal logic need not be utilised. This last point is important because real in many cases elements of a group  $G$  could turn out to be elements of some subgroup  $H$  by some indirect chain of reasoning, and we prefer not to be forced to invoke modal logic in all such cases.

