

## 1 Introduction: Syntax versus Semantics

Let us consider a typical definition of what it means to be a ‘second order’ statement:

The language of second-order logic extends the language of first-order logic by allowing quantification of predicate symbols and function symbols. (*Stanford Encyclopaedia of Philosophy*)

There is a subtlety implicit in the notion of ‘quantification over predicate symbols and function symbols’. This quantification can occur either at the syntactic level or at the semantic level; that is, one can take as the definition of second-order-ness either

1. That displayed notation contains a ‘slot’ which may take an arbitrary predicate/function, or
2. The underlying semantic representation contains a ‘slot’ which may take an arbitrary predicate/function.

Surprising as it may seem, there are examples where the syntax is second-order but the underlying semantics can be expressed in a first-order way. Somewhat arbitrarily, we will consider ‘second order’ and ‘higher order’ to be syntactically determined properties; higher-order statements which necessarily have higher-order semantics will be said to be *intrinsically* higher-order statements, and those which may take first-order semantics will be said to be *superficially* higher-order statements.<sup>1</sup>

## 2 Syntax

### 2.1 Intrinsically Higher-Order Statements

As a rule, mathematicians tend to avoid the explicit use of intrinsically higher-order statements. The general convention in mainstream mathematics is to set up Zermelo-Fraenkel set theory with certain schematic/higher-order axioms,<sup>2</sup> and then to build up all other mathematics by reference to these axioms. So, for example, instead of presenting an explicitly higher-order version of the axiom of induction, one may state a transparently first-order variant by reference to set theory, as in

If  $A$  is a set of natural numbers which contains 0 and contains  $S n$  for  $n \in A$ , then  $A$  contains all natural numbers.

In this way one confines the explicitly schematic/higher-order material to a very small portion of the mathematical text.

---

<sup>1</sup>I am more than willing to change this terminology if presented with better alternatives.

<sup>2</sup>I do not propose to try to draw sharp distinction between axiom schemata and higher-order axioms. All of the following material, and correspondingly all of the semantic material produced by the compiler, should be interpretable in either sense; the actual distinction should only be drawn by the module checking the logic, i.e. by the theorem prover to which we are directing the logical output.

Because of this convention, we have very little material on which to base a general ‘theory of intrinsically higher-order constructs’; indeed, the only two data points which we can rely on are the axiom/axiom schema of separation and the axiom/axiom schema of replacement. Given this, we should certainly start by stating those axioms. The following versions are taken from Hamilton’s *Numbers, Sets and Axioms* (p. 118):

(separation)

Given any well-formed formula  $\mathcal{A}(y)$  and any set  $x$ , there is a set  $\{y \in x | \mathcal{A}(y)\}$ .

(replacement)

Given any well-formed formula  $\mathcal{F}(x, y)$  which determines a function and any set  $u$ , there is a set  $v$  consisting of all objects  $y$  for which there is  $x \in u$  such that  $\mathcal{F}(x, y)$  holds.

These are the clearest versions that I have been able to find in the literature, but they are still not quite correct. (Because mathematicians are so rarely required to treat intrinsically higher-order statements, they tend to become rather imprecise when forced to do so.)

In the case of the axiom of separation, the above statement does not in fact pin down what ‘ $\{y \in x | \mathcal{A}(y)\}$ ’ actually *is*; it relies on the fuzzy analogical reasoning power of the reader to infer this from the shape of the notation. In order to make the statement precise, we should emend it to e.g.

**Axiom of Specification** Given any well-formed formula  $\mathcal{A}(y)$  and any set  $x$ , there is a set  $u$  such that for all objects  $y$ ,  $y \in u$  if and only if  $y \in x$  and  $\mathcal{A}(y)$ . This set  $u$  is denoted by  $\{y \in x | \mathcal{A}(y)\}$ .

In the case of the axiom of replacement, the statement of the axiom is rather roundabout. In order to interpret a typical usage of the axiom, such as

$$\{1/n | n \in \mathbb{N}\}$$

one needs to introduce a ‘dummy’ well formed formula like

$$\mathcal{F}(x, y) \Leftrightarrow y = 1/x$$

Having verified that the formula ‘determines a function’, i.e. that for every  $x$  there is at most one  $y$  such that  $\mathcal{F}(x, y)$ , one needs to substitute this well formed formula into the axiom of replacement to obtain:

There is a set  $v$  consisting of all objects  $y$  for which there is  $n \in \mathbb{N}$  such that  $y = 1/n$  holds.

From a strictly mathematical point of view, this formulation is perfectly adequate; but for our purposes, it becomes extremely difficult to actually define the notation associate it with the axiom, i.e. to set things up so that ‘ $\{1/n | n \in \mathbb{N}\}$ ’ is in fact a term. The key point here is that we need some formalisation of precisely what kind of thing may be legitimately substituted into the context ‘ $\{\bullet | n \in \mathbb{N}\}$ ’.

In order to resolve this issue, we will need a formal notion of ‘well-formed term’ analogous to ‘well-formed formula’; using such a notion, we can write:

**Axiom of Replacement**

Given any well-formed term  $\mathcal{T}(x)$  and any set  $U$ , there is a set  $V$  such that for all objects  $y$ ,  $y \in V$  if and only if  $y = \mathcal{T}(u)$  for some  $u \in U$ .  $V$  is denoted by  $\{\mathcal{T}(u) | u \in U\}$ .

**2.1.1 Digression: Filter and Map**

It will be apparent that the axioms of separation and replacement correspond to the notions of *filter* and *map* in functional programming languages. So, an example of replacement like

$$\{\mathbf{A} \in M_3(\mathbb{R}) | \mathbf{A}\mathbf{A}^T = \mathbf{I}\}$$

corresponds to

$$(\text{filter } (\lambda \mathbf{A}. (\mathbf{A}\mathbf{A}^T =? \mathbf{I})) M_3(\mathbb{R}))$$

in a functional programming language;<sup>3</sup> similarly, an example of separation like

$$\{1/n | n \in \mathbb{N}\}$$

corresponds to

$$(\text{map } (\lambda n. (1/n)) \mathbb{N}).$$

(I would argue that the mathematical equivalents are significantly easier to take in — although this judgment may be affected by my relative familiarity with the two versions. On the other hand, the popularity of the list comprehension construct in Haskell may lend some weight to this claim.)

Having found analogues for *map* and *filter*, the natural progression is towards finding an analogue for *fold*. Since *fold* only makes sense for finite, ordered collections, it is clear that this analogue will not be a notion that is applicable to general sets. In fact, the analogue for fold in mathematics is the use of ellipsis, which will be our primary example in the next section.

**2.2 Nonextensional Constructs**

As noted above, the two axioms we have just discussed are essentially the *only* examples of intrinsically higher-order statements in mathematics. As such, we have very little material on which to base an extension to the core language aimed at supporting higher-order statements. Fortunately, we will find that some very small changes to our system will allow us to support some important sugary constructs using the same machinery that supports higher-order statements. The constructs in question are called nonextensional constructs; the reason for this name will become apparent when we consider the following example of a nonextensional construct:

<sup>3</sup>I am of course blurring the distinction between sets and lists.

**Example: Finite Sums Using Ellipsis**

Mathematicians often define n-ary folded or ‘iterated’ versions of common associative binary operators; so corresponding to the + operator one defines the iterated version  $\Sigma$ , corresponding to the  $\times$  operator one defines the iterated version  $\Pi$ , and so on.<sup>4</sup> However, in much of mathematics, these operators turn out to be relatively infrequent; the reason for this is that mathematicians tend to use ellipses in place of these iterated operators. So rather than writing

$$\sum_{i=1}^n i^2$$

one typically writes

$$1^2 + 2^2 + \cdots + n^2$$

as this latter variant is much easier to take in. (Indeed, excessive use of  $\Sigma$ , etc., instead of ellipsis arguably signifies a particular stage of mathematical immaturity in a certain, very precise, kind of mathematician.)

Now, consider what happens when we apply this notation to the sequence  $(a_n)$  defined by:

$$a_n = \begin{cases} n & \text{if } n \leq 1 \text{ or } n \text{ is prime;} \\ 0 & \text{otherwise.} \end{cases}$$

We have that

$$\begin{aligned} a_1 &= 1 \\ a_2 &= 2 \\ a_3 &= 3 \\ a_{11} &= 11 \end{aligned}$$

And yet it is categorically not the case that

$$a_1 + a_2 + a_3 + \cdots + a_{11} = 1 + 2 + 3 + \cdots + 11$$

So we are forced to conclude that this ‘finite ellipsis’ notation for sums is *not extensional*.

Now, how do these non-extensional constructs relate to the higher-order constructs dealt with above? The answer is very simple; given that we have introduced a mechanism which let us explicitly quantify over terms (by writing ‘Given any well-formed term  $\mathcal{T}(x)$ ’, etc.), we will reuse this mechanism to introduce non-extensional constructs. So to deal with our ‘finite ellipsis’ example above, we might write something like:

Let  $T(x)$  be a real-valued well-formed term for  $x$  an integer, and let  $n$  be a natural number; then we write  $T(1)+T(2)+T(3)+\dots+T(n)$  for  $\sum_{i=1}^n T(i)$ .

---

<sup>4</sup>I will assume familiarity with at least  $\Sigma$  rather than giving the rather tedious inductive definition.

### 2.3 Superficially Higher-Order Statements

Last and not least, we will turn to the examples which have higher-order syntax but, potentially, first-order semantics. We would expect that any mechanism which handled the intrinsically higher-order examples should have no problem with these superficially higher-order examples; and, indeed, this turns out to be the case — but these examples still have some interesting features of their own which we will discuss.

The easiest examples to understand are the individual instances of notation licensed by the axioms of specification and replacement. In any given example, the content underlying such notation can always be re-expressed in a purely first-order way. So, for example, if we see

$\{1/n | n \in \mathbb{N}\}$  has a limit point.

we could convert this assertion to

By the axiom of replacement, there is a set  $Y$  such that:

- i) If  $n \in \mathbb{N}$ , then  $1/n \in Y$ ;
- ii) If  $y \in Y$ , then  $y = 1/n$  for some  $n \in \mathbb{N}$ .

This set  $Y$  has a limit point.

To emphasise what is happening here: The axiom of specification is a higher-order axiom; its content cannot be stated in a first-order way.<sup>5</sup> But the notation derived from the axiom can be handled using only first-order content. (Whether or not we should perform this transformation is an issue we will discuss below.)

---

<sup>5</sup>It has been pointed out to me that this statement is in fact false. One may construct a first-order datatype which exactly mirrors the structure of formulae, so that the instances of this data type precisely correspond to all the predicates in the language, and use this to state the axiom. I don't feel that I genuinely understand this yet, but it seems like a general mechanism for forcing higher-order logic into first order terms; clearly, given results by Godel, etc., there must be a catch here. (Possibly there is an ambiguity between the countable collection of predicates that one may write down, and the uncountable collection of 'possible predicates'?) Regardless, morally speaking, a single-statement expression of the axiom of specification quantifies over predicates and so is a higher-order statement.

There are, however, many more examples of ‘higher-order sugar’ than those arising from set-theoretic examples. Perhaps the most representative example is given by the notation for limits of sequences. A typical definition of a limit is:<sup>6</sup>

DEFINITION 1.2.2.

The sequence  $(s_n)$  converges to (the real number)  $l$ , if given any (real number)  $\epsilon > 0$ , there exists (an integer)  $N$  such that  $|s_n - l| < \epsilon$  for all  $n \geq N$ .

This is usually shortened by omitting the parts in parentheses.

Other ways of writing ‘ $(s_n)$  converges to  $l$ ’ are ‘ $s_n \rightarrow l$  as  $n \rightarrow \infty$ ’ and ‘ $\lim_{n \rightarrow \infty} s_n = l$ ’.

(*Sutherland, Introduction to Metric and Topological Spaces, p. 7.*)

A typical usage of the notation introduced by this definition is:

Deduce that

$$\lim_{n \rightarrow \infty} (a_1^n + a_2^n + \cdots + a_r^n)^{1/n} = a$$

(*Sutherland, Introduction to Metric and Topological Spaces, p. 16.*)

The surprising feature of this example is the mismatch between the definition of a limit and its usage; the definition is stated in a way that only applies to sequences, but the notation is subsequently applied to arbitrary terms. This mismatch raises some significant technical issues, and there is a clear case that it is, strictly speaking, incorrect — but the underlying mathematics is in fact sound. The reason for this comes in several parts:

1. The notation is extensional; that is, replacing the term by an extensionally equivalent term leaves the denotation of the limit unchanged.
2. Given any term  $\mathcal{T}(x)$  which may legitimately be used in the limit notation, there is a sequence which is extensionally equivalent. That is, there is  $f$  such that  $f(x) = \mathcal{T}(x)$  for all  $x$  in  $M$ .
3. (Working in a fixed metric space  $M$ ) The set of all the sequences arising in 2. is itself a mathematical object.

Together, these facts tell us that the *mathematical content* of an apparently higher-order assertion about the existence of general limits like

Let  $\mathcal{T}(n)$  be a real-valued well-formed term for  $n$  a natural numberan integer; if  $\mathcal{T}(n+1) > \mathcal{T}(n)$  for all  $n \in \mathbb{N}$ , and there exists  $c \in \mathbb{R}$  such that  $\mathcal{T}(n) < c$  for all  $n \in \mathbb{N}$ , then  $\mathcal{T}(n)$  converges to some real number.

<sup>6</sup>This example raises a number of issues that are significant but not relevant to the current discussion. To highlight these for future discussion: 1. Automatic typing of variables (‘ $\epsilon$ ’ conventionally denotes a real number, ‘ $p$ ’ conventionally denotes a prime number, etc.); 2. Grammatical errors (the first comma is incorrect); 3. Supermathematical phraseology that cannot reasonably be captured (‘Other ways of writing ...’); 4. Syncategorematic usages of ‘ $\equiv$ ’ and reanalysis (as in  $\lim_{n \rightarrow \infty} s_n = l$ ); 5. Compound Objects (like ‘ $(s_n)$ ’); 6. Flagging variables (like  $n$  in ‘ $(s_n)$ ’ or  $x$  in ‘ $f(x)$ ’).

can be stated using a transparently first-order assertion about the existence of limits for sequences, namely

Every increasing sequence of real numbers which is bounded above converges to some real number.

Also, as before, individual instances of the limit notation can clearly be rewritten in a first-order way. So we find that, morally speaking, the ‘notion of a limit’ is not in any sense a higher-order concept; it is simply that it turns out to be convenient to use higher-order syntax with this notion.

Unlike our intrinsically higher-order assertions, there are many superficially (i.e. purely syntactically) higher-order assertions similar to this example with limits. As far as I am aware, all such examples use first-order definitions; the reason for this seems to be mathematicians are very uncomfortable making second-order assertions.

### 2.3.1 Handling Superficially Higher-Order Material

We need to make a choice as to how to represent in the syntax i) superficially higher-order definitions and ii) superficially higher-order assertions. In the case of definitions, I am unable to fully describe a principle which distinguishes first-order definitions tacitly introducing higher-order syntax, like, say, the definition of a limit for functions,<sup>7</sup> from other definitions which mention similar objects without licensing higher-order syntax, such as the definitions of function inverses or function composites.<sup>8</sup> So in this case, we will require a (syntactically) higher-order version of the definition to be explicitly presented. Note that one may choose also to include the first-order version (i.e., the version referring to sequences or functions or etc.); the principles relating to reanalysis and the “is unambiguous” directive will together prevent the two versions from clashing.

In the case of individual assertions about e.g. limits, the situation is less clear. Insofar as the syntactically higher-order and syntactically first-order versions are equivalent in terms of mathematical content, the choice is not one that affects the system itself but one that affects the nature of the output to the logic component (i.e. to a theorem prover). Mathematical aesthetics will generally require that the first-order variant (stated in terms of sequences or functions or etc.) be present, but there is no obstacle to having the higher-order variant also present, if this turns out to be necessary.

---

<sup>7</sup>To wit:

Suppose [...] that we have a function  $f : \mathbf{R} \rightarrow \mathbf{R}$ . Let  $a \in \mathbf{R}$ .

Definition 1.3.1. We say that  $f(x)$  tends to the limit  $l$  as  $x$  tends to  $a$  ( $\lim_{x \rightarrow a} f(x) = l$ ) if given any (real number)  $\epsilon > 0$ , there exists a (real number)  $\delta > 0$  such that  $|f(x) - l| < \epsilon$  for all  $x$  satisfying  $0 < |x - a| < \delta$ .  
(*Sutherland, Introduction to Metric and Topological Spaces, p. 11.*)

<sup>8</sup>This is not to say that no such mechanism is possible; indeed, there are telling features in the above example, such as the fact that  $x$  in ‘ $f(x)$ ’ is a bound variable. But turning this into a reliable principle would require a nontrivial amount of work, and at the moment we are more concerned with building a initial, minimal version of the system than adding features. Although it will not be apparent from these documents, which only present one side of the story, we have been very brutal with respect to shelving and delaying features.

### 3 Semantics

There are two semantic questions which we need to resolve. First, how should we augment our semantic representation (based on DRT) to handle higher-order statements? Second, should we translate superficially higher-order statements to a first-order form in the semantic representation?

#### 3.1 Semantic Representation

It is clear that we will need to adjust our atomic terms and atomic predicates to allow other, higher-order, terms/predicates as arguments. (We will of course type the slots of the atomic terms/predicates to make sure that one cannot substitute a higher-order term where a first-order term is expected, etc.)

The only remaining question is as to what form the representations of the higher-order terms/predicates should take. There are two sensible answers to this question. The first option is to use  $\lambda$ -abstraction, so that the higher-order terms/predicates are represented by  $\lambda$ -terms in categories like  $e \rightarrow t$  and  $t \rightarrow t$ . The second option is to use  $\lambda$ -free terms from categories like  $e$  and  $t$  as representations, using extra ‘dummy’ variables to represent the  $\lambda$ ; under this method, we would add extra slots to the higher-order terms/predicates to keep track of the variables. The first method is standard in computer science; the second method is more mathematical. Let us illustrate the two variants with a concrete example, namely the set

$$\{x \in A \mid x > 0\}.$$

Under the first proposal, this set would be represented by something like

$$\text{separated\_set}(A, \lambda x.(x > 0)).$$

Under the second proposal, the set would be represented by something like

$$\text{separated\_set}(A, x, x > 0).$$

The differences between these two variants are subtle, and I have spent some time trying to deduce which variant is appropriate. I eventually settled on the second variant, but I find the deeper reasons difficult to articulate.<sup>9</sup>

#### 3.2 Translation

Given that nearly all higher-order statements are superficially higher-order, and that first-order theorem proving technology is relatively powerful, it seems desirable to transform assertions to first-order versions wherever possible. However: this reductive translation does not need to be done by the main compiler module; and, given that that module is already more than complex enough, we will defer the translation to a separate program which operates only on the semantic representation output by the compiler.

<sup>9</sup>One significant shallow point is that  $\lambda$ -is used internally during the *construction* of the semantic representation, so that the absence of lambdas is an effective test for whether the representation has been completely constructed; reusing the same  $\lambda$ s would invalidate that test, and introducing a separate level of  $\lambda$ s would merely be confusing.