

# MathLang Translation to Isabelle Syntax

Robert Lamar, Fairouz Kamareddine, and J. B. Wells

ULTRA Group, Heriot-Watt University  
<http://www.macs.hw.ac.uk/ultra/>

**Abstract.** Converting mathematical documents from a human-friendly natural language to a form that can be readily processed by computers is often a tedious, manual task. Translating between varied computerised forms is also a difficult problem. MathLang, a system of methods and representations for computerising mathematics, tries to make these tasks more tractable by breaking the translation down into manageable portions. This paper presents a method for creating rules to translate documents from MathLang's internal representation of mathematics to documents in the language of the Isabelle proof assistant. It includes a set of example rules applicable for a particular document. The resulting documents are not completely verifiable by Isabelle, but they represent a point to which a mathematician may take a document without the involvement of an Isabelle expert.

## 1 Introduction

When a mathematician describes a piece of mathematics in written form, it may be of interest to use computers to process this text in a variety of ways. They may use editing software to write the text as they are developing the ideas. Similarly, it is very common for such a text to be typeset so it is pleasing to the eye. It may also be useful to identify the semantics, for presentation to those with disabilities, or there may exist in the document partial calculations which the computer should process for the benefit of the original author or other readers. These are a few ways in which a computer may be used to directly benefit a human who wants to understand the mathematics.

In addition to providing output for humans, a computer may be useful for verifying the correctness of a document, in a variety of ways. These include checking of spelling and grammar in the natural language; syntax checking for mathematical notation; checking soundness of the interrelations between definitions, theorems, and proofs; and formally verifying the logical structure of the document, at different levels of rigour. These are ways in which the computer may provide evaluation and feedback on the document.

For mathematicians, systems falling at the formal end of that spectrum are currently of limited use, as they require considerable investment to gain proficiency. It is possible for a mathematician to author a paper and then pass it on to an expert in some formal system, but this requires the expert to completely comprehend each natural-language document received so as to ensure a faithful

translation. The current methods of translation are almost entirely manual. This leaves a vast chasm between original and formalised documents, in which may be introduced semantic discrepancies. Furthermore, if the original document is changed, there is some risk that necessary changes in the formalised counterpart will be overlooked when the time comes for revision.

## 1.1 Contributions

This paper describes developments in MathLang (a system for computerising mathematics, described in Section 2) which may help close the gap between natural language and formalised documents. MathLang is a system which tries to give as much flexibility to the user as possible, trying to process any style that a person may use to express their mathematics. The existing parts of MathLang, which are assumed to be the starting point for the developments in this paper, are overviewed in Section 2.1.

This paper offers a *process for arriving at rules for translation*. Our goal is to produce, from the already-computerised MathLang document, a text in the language of a formal system. Our chosen target language is Isabelle. To do this, rules are created which operate recursively on a document. The nature of these rules is described in Section 3.1, and is illustrated by a detailed example.

## 1.2 Related Work

In the context of the MathLang project, this work provides tools which provide output which is closer to existing formal systems than ever before. Specifically, it uses the facilities provided by [1, 2] to translate the bulk of a text to the syntax of the proof assistant Isabelle. This is in parallel with [3], which was translating the same kind of document to Mizar [4]. However, in that work the focus was on identifying relevant theories from the Mizar Mathematical Library to include in the environment of a new Mizar document, more so than translating the main text of a MathLang document to Mizar.

In the larger field, there are a number of projects which are attempting to bridge the gap between human-friendly mathematical texts and easily-processed and -verified computer documents. Most focus primarily on one side or another. On the formal end of the spectrum are projects like Mizar [4], Theorema [5] and Isar [6, 7]. These three computer proof systems are designed with syntax that is constructed to be similar to the way that mathematicians write in natural English. A similar approach is being taken in the work of Muhammad Humayoun on MathNat [8], in which he tries to express both mathematical proofs and natural language. In the cases of Mizar and Isar, the language is *like* natural English, but does not provide the author with much flexibility. MathNat and Theorema allow much greater flexibility – in MathNat’s case due to its incorporation of GF [9] – but still force the user to employ a controlled language, which may often be a subset of what an author would normally employ. MathLang endeavours to accommodate any writing style through the use of flexible annotation,

accommodating documents that were never intended to be computerised, such as Euclid’s *Elements* [10] and Landau’s *Grundlagen der Analysis* [11].

On the other side of the natural–formal gap, Aarne Ranta’s system GF [9] has a flexible system for defining grammars, and provides an API for interfacing with other programs, but is not, itself, designed to process documents to further formal states. We wish to process such documents in other interesting ways.

Finally, there are systems such as Isabelle [6] and Coq [12], which are systems for computer proof, along with Logiweb [13], which is a system for document processing that interfaces with arbitrary systems. Each of these allow natural language text to be interleaved with formal expressions in a kind of literate proof document in the manner of CWEB [14]. However, care during revision is necessary to ensure that natural language and formalism remain consistent.

### 1.3 Conclusion

The rules presented in this document are very limited, but they represent a pattern which may be used to develop a library of rules which may be useful in translating a variety of documents to Isabelle syntax. The documents that result from translation are very incomplete, but they may be a useful middle ground between a mathematician who has little knowledge of proof systems, and an Isabelle expert who is trying to formalise the mathematics that the original author has written. The system needs to be extended and tested extensively, but the current work shows a valuable proof-of-concept which merits further investigation.

## 2 Relevant Systems

This section describes systems and theories that the current work relies upon, but which the reader may not be familiar with. First it describes the system called MathLang. It first provides an overview, then a detailed description of the relevant portions of operational theory. A second section gives an example Isabelle document, drawing the reader’s attention to relevant features of the language.

### 2.1 System Aspects

The current MathLang system [16] is designed to computerise mathematical texts like that seen in Fig. 1. By *computerising*, we mean operating on documents which are easily accessed and modified by computers. We also mean processing documents so they may be easily accessed and modified. Currently MathLang provides several ways to achieve this. They are classified into domains called *aspects* of MathLang. The current aspects are the Text and Symbol aspect [1, 17], the Core Grammatical aspect [2, 17], and the Document Rhetorical aspect [3]. For the current paper we restrict our focus to the first two.

**The Text and Symbol aspect** The Text and Symbol aspect (TSa) is the aspect of MathLang which is directly concerned with the ways in which documents present mathematics to a reader or author. A particular focus of the aspect is the way in which mathematical concepts are abbreviated, such as combining a pair of equations  $a = b$  and  $b = c$  into the single string  $a = b = c$ .

It also provides the details for sensible presentation of the fruits of other aspects of MathLang: these others augment the text with information, but TSa governs how this extra information is *symbolised* in relation to the original *text*. The document in Fig. 1, for instance, has been enhanced in Fig. 2 to show the extra information provided with the Core Grammatical aspect, below.

**Table 1.** Elements of  $\mathcal{C}$  (primitive grammatical categories), with associated colour (see Definition 4) and ontological meaning.

Cat.	Colour	Description
<b>term</b>	blue	common mathematical objects
<b>set</b>	red	collections of <b>terms</b>
<b>noun</b>	orange	families of <b>terms</b>
<b>adj</b>	yellow	adjectives used to construct new <b>nouns</b> from old
<b>stat</b>	green	statements which have some truth value
<b>decl</b>	dk. gray	declarations of new <b>terms</b> , <b>sets</b> , <b>nouns</b> , <b>adjs</b> , or <b>stats</b>
<b>defn</b>	lt. gray	definitions of new symbols
<b>step</b>	salmon	groups of mathematical assertions
<b>cont</b>	purple	contexts containing preliminaries prior to a step

**The Core Grammatical aspect** The Core Grammatical aspect (CGa) provides analysis for the sentence level of the document. It provides a type system for objects, definitions, and assertions and a means for checking the document for type correctness. The types of CGa are summarised in Table 1.

For instance, the variable  $x$  and number 1 could be **declared** as having the type **term**. The operation  $+$  might have type **term**  $\rightarrow$  **term**  $\rightarrow$  **term**. Thus,  $x+1$  would also be considered a **term**. However, this expression can also perhaps be considered as an instance of the **noun** polynomial. Furthermore, the **adjective** linear can be used to modify polynomial to create a new **noun**, linear polynomial, which also classifies  $x+1$ . The manner in which these types are shown to a reader (as in Fig. 2) is defined in Section 2.2.

## 2.2 Operational System

In this section we define the operational system of MathLang. This definition is covered more extensively in an earlier paper [1]. Portions are reproduced here for the benefit of the reader. Some readers may find it beneficial to keep in mind the definition of the XML XPath data model [18], as there exist strong conceptual parallels with the following definition.

Let  $\mathbb{N}$  denote the natural numbers, use  $(-; -)$  to denote ordered pairs, and let functions be sets  $\varphi$  of ordered pairs. Every function has a domain  $\text{dom}(\varphi) = \{a \mid \exists b \ni (a; b) \in \varphi\}$  and a range  $\text{ran}(\varphi) = \{b \mid \exists a \ni (a; b) \in \varphi\}$ . A sequence is a function  $\sigma$  for which  $\text{dom}(\sigma) = \{n \mid 0 \leq n < k\}$  for some  $k \in \mathbb{N}$ . We write  $\square$  for the empty sequence and  $[x_0, x_1, \dots, x_n]$  for the sequence  $\sigma$  such that  $\sigma(i) = x_i$  for each  $i \in \text{dom}(\sigma) = \{0, \dots, n\}$ . Upon that sequence is defined the metric  $|\sigma| = n + 1$ . We define  $\sigma_1, \sigma_2$  to concatenate  $\sigma_1$  and  $\sigma_2$  as the new sequence  $\sigma$  such that  $\text{dom}(\sigma_1, \sigma_2) = \{0, \dots, |\sigma_1| + |\sigma_2| - 1\}$  where  $\sigma(i) = \sigma_1(i)$  for  $i \in \text{dom}(\sigma_1)$  and  $\sigma(i) = \sigma_2(i)$  for  $i - |\sigma_1| \in \text{dom}(\sigma_2)$ . Concatenation is associative. Moreover,  $\square, \sigma = \sigma$  and  $\sigma, \square = \sigma$ . For any set  $S$ , say  $[S]$  denotes  $\{\sigma \mid \text{ran}(\sigma) \subseteq S\}$ .

Let  $\mathcal{L} = \mathcal{F} \cup \mathcal{G} \cup \mathcal{S}$  be a set of labels such that elements of  $\mathcal{F}$ ,  $\mathcal{G}$  and  $\mathcal{S}$  are formatting, grammatical, and souring labels, respectively. The set  $\mathcal{F}$ , of *formatting instructions*, varies according to rendering system. We define  $\mathcal{G} = \mathcal{C} \times \mathcal{I}$ , where  $\mathcal{C} = \{\text{term, set, noun, adj, stat, decl, defn, step, cont}\}$ , and contains identifiers for the primitive grammatical categories of Table 1. The set  $\mathcal{I}$  consists of strings used for identifying abstract interpretations. We let  $\ell, f, g, c$  and  $i$  range over  $\mathcal{L}, \mathcal{F}, \mathcal{G}, \mathcal{C}$  and  $\mathcal{I}$ , respectively.

We let  $s$  range over  $\mathcal{S} = \mathcal{S}_u \cup \mathcal{S}_i$  where  $\mathcal{S}_u$  contains *souring identifiers* to be employed directly by the user while  $\mathcal{S}_i$  holds several identifiers used internally for rewriting.  $\mathcal{S}_u$  and  $\mathcal{S}_i$  are disjoint, defined as follows:

$$\begin{aligned} \mathcal{S}_u &= \{\text{fold-left, fold-right, map, base, list, hook, loop, shared}\} \cup (\{\text{position}\} \times \mathbb{N}) \\ \mathcal{S}_i &= \{\text{hook-travel, head, tail, daeh, liat, right-travel, left-travel}\} \cup (\{\text{cursor}\} \times \mathbb{N}) \end{aligned}$$

**Definition 1 (Document).** *Let  $\mathcal{D}$  be the smallest set such that:*

1.  $\square \in \mathcal{D}$ ,
2. if  $d \in \mathcal{D}$  and  $\ell \in \mathcal{L}$  then  $[(\ell; d)] \in \mathcal{D}$ , and
3. if both  $d_1$  and  $d_2$  are elements of  $\mathcal{D}$  then  $(d_1, d_2) \in \mathcal{D}$ .

A *MathLang document* is an element of the set  $\mathcal{D}$ . In addition, we denote by  $\mathcal{D}_{\mathcal{F}}$ ,  $\mathcal{D}_{\mathcal{G}}$ ,  $\mathcal{D}_{\mathcal{F} \cup \mathcal{G}}$ ,  $\mathcal{D}_{\mathcal{G} \cup \mathcal{S}}$  and  $\mathcal{D}_{\mathcal{F} \cup \mathcal{G} \cup \mathcal{S}}$  the sets of documents whose labels are restricted to the respective subscripted set. The variables  $d, d_n$  (where  $n \in \mathbb{N}$ ) denote members of  $\mathcal{D}$ , unless otherwise noted.

*Remark 2 (Notational convention).* We use  $\ell \langle d \rangle$  to denote  $[(\ell; d)]$ . When not ambiguous,  $\ell$  denotes  $\ell \langle \square \rangle$ . A box with black border and **coloured background**,  $\boxed{i d}$ , is used to represent  $(c; i) \langle d \rangle$  (a document with grammatical label), where the background colour of the box corresponds to  $c$  as shown in Table 1 (See Example 6, below). Similarly, a box with thick pink border and white background,  $\boxed{s d}$ , is used to represent  $s \langle d \rangle$ , documents with souring labels.

It is worth noting that these notations have been developed for ease of reading, and particularly interactive annotation of texts. One of Mathlang's biggest motivations is for humans to be able to type a mathematical text in a natural

way on the computer, and then add the grammatical and souring information with ease. The prototype based on  $\text{\TeX}$ macs is described in detail in [17].

Formatting systems are treated as a set of formatting instructions  $\mathcal{F}$ , a blank formatting instruction  $\varepsilon$ , a concatenation operator  $\bullet$ , and a hole-filling function  $\text{fill} : \mathcal{F} \times [\mathcal{F}] \rightarrow \mathcal{F}$ , which takes two arguments, a formatting instruction  $f$  and a sequence of instructions  $\sigma$ . Instruction  $f$  may have holes, denoted  $\overline{[n]}$ , where  $0 \leq n < |\sigma|$ . The instruction  $f$  is rewritten so that each  $\overline{[n]}$  is replaced by  $\sigma(n)$ .

**Definition 3 (Souring).** *Souring is a rewriting process that was described in [1]. The particulars of the procedure are not important to this paper. We may regard the souring function as a black box function  $\text{sour} : \mathcal{D} \rightarrow \mathcal{D}_{\mathcal{F} \cup \mathcal{G}}$ .*

The motivation for souring is as follows: as syntactic sugar is added to a formal document to make it easier to read for humans, syntax souring is added to natural-language documents to make them easier for a computer to process. Before processing a document with the rules in Section 3.1, we typically *sour* the document by applying this function to the document, then further processing the result. An example of a typical souring operation would be to convert  $a = b = c$  to  $a = b, b = c$ .

**Definition 4 (Rendering functions).** *Let  $r : \mathcal{D} \rightarrow \mathcal{F}$  be defined as*

$$r(\square) = \varepsilon \quad (\text{REN1})$$

$$r(f\langle d \rangle) = \text{fill}(f, [r(d(0)), \dots, r(d(|d|-1))]) \quad (\text{REN2})$$

$$r((c; i)\langle d \rangle) = \boxed{i} r(d) \quad (\text{REN3})$$

$$r(s\langle d \rangle) = \boxed{s} r(d) \quad (\text{REN4})$$

$$r(d_1, d_2) = r(d_1) \bullet r(d_2) \quad (\text{REN5})$$

where the background colour of the box given by (REN3) is the colour from Table 1 (i.e.,  $r(\mathbf{term}; i)\langle d \rangle = \boxed{i} r(d)$ ,  $r(\mathbf{set}; i)\langle d \rangle = \boxed{i} r(d)$ , etc.)

**Definition 5 (Extract original document).** *Usually, some  $d \in \mathcal{D}$  consists of a “typical” mathematical text plus some information which is stored in the labels from  $\mathcal{G} \cup \mathcal{S}$ . For any document which has this property, it may be useful to filter  $d$  with the function  $\text{od} : \mathcal{D} \rightarrow \mathcal{D}_{\mathcal{F}}$ , defined as*

$$\text{od}(\square) = \square \quad (\text{OD1})$$

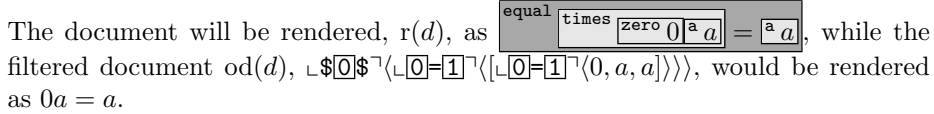
$$\text{od}(\ell\langle d \rangle) = \begin{cases} \ell\langle \text{od}(d) \rangle & \text{if } \ell \in \mathcal{F} \\ d & \text{otherwise} \end{cases} \quad (\text{OD2})$$

$$\text{od}(d_1, d_2) = \text{od}(d_1), \text{od}(d_2). \quad (\text{OD3})$$

*It is then possible to obtain the mathematician’s original text as  $r(\text{od}(d))$ .*

*Example 6.* In this example, formatting instructions are taken to be from the  $\text{\LaTeX}$  typesetting system. Consider the document  $d$  given as

$$\llcorner \$\text{\O}\$^\rceil \langle (\text{\texttt{stat}; equal}) \langle \llcorner \text{\O}=\text{\texttt{1}}^\rceil \langle [(\text{\texttt{term}; times}) \langle [(\text{\texttt{term}; zero}) \langle 0 \rangle, (\text{\texttt{term}; a}) \langle a \rangle \rangle], (\text{\texttt{term}; a}) \langle a \rangle \rangle \rangle \rangle \rangle.$$

The document will be rendered,  $r(d)$ , as 

### 2.3 Overview of Isabelle

For the benefit of the reader, the following provides a brief overview of pertinent parts of the proof assistant Isabelle. Isabelle/HOL was chosen as a target for translation from MathLang because it is a popular, mature system with extensive documentation. The authors of the current paper do not consider themselves to be Isabelle experts, but it was straightforward to learn the basics of the system in order to begin making basic proof documents. Isabelle [6] allows a user to express and record formulae and reasoning steps. It is designed to work with a variety of logical foundations, the most popular being HOL. Isar [7] enhances the language of Isabelle for a more declarative proof style.

What follows is a summary of certain Isabelle features which may be useful in understanding the remainder of this document, referencing Figure 3.

In Isabelle, formal developments are organised into *theories*, which are given unique names and stored in separate text files. The theory in Figure 3, for instance, is stored in `Groebner_Basis.thy` and begins with the indicated line 7 (The previous lines in the file are all comments). Line 8 of this example shows that the current theory may need to use results formalised in the Isabelle theory `NatBin`. This `imports` directive allows access to definitions and results of the other theory. The rest of the file, which will develop new formalisations, is enclosed with the keywords `begin` (line 14) and `end` (line 440).

A `locale` is an Isabelle construct which defines a local scope in which assumptions and symbols are declared. Theorems may then depend on locales for the premises on which they are proved. Line 259 starts the declaration of a locale, which in this case has the name `gb_field` and inherits the properties of locale `gb_ring`. It starts by declaring (`fixing`) a pair of constants with type signatures and stating two axioms for the locale. This is followed by a lemma and proof.

## 3 Rules for Translating Documents

Section 2.2 described the operational representation of MathLang documents. The documents are stored as an assembly of labels, each of which has a particular role (formatting, grammatical, or souring). It may be of interest to reformat the document into another form. In this section, we give a set of translation

rules which could be recursively applied to a MathLang document, and easily extended to be applied to other documents. This set of rules converts some of the information of the document to Isabelle syntax.

### 3.1 Example Rules for Translating to Isabelle

In this section, we describe a set of rules which are sufficient to translate the document in Figure 2 to the language of Isabelle. These are given to show how rules can be created to cover different cases in MathLang documents. Suppose that  $d$  is a document which has been soured (see Definition 3). Then we apply mutually recursive translation rules  $\mathcal{T} : \mathcal{D}_G \rightarrow \mathcal{D}_F$  as partial translations of  $d$  into the syntax of Isabelle. These are defined from the top down: each rule may rely on other rules which are defined later in the section. Figure 4, in Section 3.2, shows the translation given by the rules.

In the first rule, `(*name*)` is an Isabelle comment which should be replaced with a name for the theory. Similarly, `(*theories*)` is a list of other theories which contain required prior knowledge. Constructing this list of theories is outside the scope of this paper. For the current work, we leave this task to an Isabelle expert, to fill in the blanks. The root document tree may be translated by the following rule.

$$\mathcal{T}_{\text{root}}(d) = \text{fill}(\ulcorner \text{theory } (*name*) \text{ imports } (*theories*) \\ \text{begin } \boxed{1} \text{ end} \urcorner, [\mathcal{T}_{\text{main}}(d)]) \quad (\text{ROOT1})$$

This inserts the main frame for the theory and then invokes  $\mathcal{T}_{\text{main}}$ , as defined below. Note that the aforementioned `(*theories*)` list would likely depend on the contents of any preface, but that is outside of the scope of this paper, so (MAIN1) returns an empty string, ignoring its contents.

$$\mathcal{T}_{\text{main}}(\text{preface } d) = \ulcorner \urcorner \quad (\text{MAIN1})$$

$$\mathcal{T}_{\text{main}}(\text{definition } d) = \mathcal{T}_{\text{def}}(d) \quad (\text{MAIN2})$$

$$\mathcal{T}_{\text{main}}(\text{theorem } \boxed{i \quad i'}, d) = \mathcal{T}_{\text{thm}}(i', d) \quad (\text{MAIN3})$$

$$\mathcal{T}_{\text{main}}(\text{proof } d) = \mathcal{T}_{\text{pf}}(d) \quad (\text{MAIN4})$$

$$\mathcal{T}_{\text{main}}(d_1, d_2) = \mathcal{T}_{\text{main}}(d_1) \bullet \mathcal{T}_{\text{main}}(d_2) \quad (\text{MAIN5})$$

When the main text contains a **definition** annotation surrounding **nouns**, this kind of annotation may be translated with the following rule.



$$\mathcal{T}_{\text{def}} \left( \boxed{\boxed{i} \mid \boxed{i'} \mid \text{props } d} \right) = \text{fill} (\llcorner \text{locale } \boxed{0} = \boxed{1} \urcorner, [i, \mathcal{T}_{\text{def}}(d)]) \quad (\text{D1})$$

$$\mathcal{T}_{\text{def}} \left( \boxed{\boxed{i} \mid \boxed{i'} \mid} \right) = \text{fill} (\llcorner \text{fixes } \boxed{0} :: \text{'r'} \text{ assumes } \llcorner \boxed{0} : \boxed{1} \urcorner, [i, i']) \quad (\text{D2})$$

$$\mathcal{T}_{\text{def}} \left( \boxed{\boxed{i} \mid d} \right) = \text{fill} (\llcorner \text{fixes } \boxed{0} :: \llcorner \boxed{1} \urcorner, [i, \mathcal{T}_{\text{ty}}(\boxed{i} \mid d)]) \quad (\text{D3})$$

$$\mathcal{T}_{\text{def}} \left( \boxed{\boxed{i} \mid} \right) = \text{fill} (\llcorner \text{fixes } \boxed{0} :: \text{'r set'} \urcorner, [i]) \quad (\text{D4})$$

$$\mathcal{T}_{\text{def}} \left( \boxed{i} \mid d \right) = \text{fill} (\llcorner \text{assumes } \llcorner \boxed{1} \urcorner, [\mathcal{T}_{\text{pfx}}(\boxed{i} \mid d)]) \quad (\text{D5})$$

$$\mathcal{T}_{\text{def}}(d_1, d_2) = \mathcal{T}_{\text{def}}(d_1) \bullet \mathcal{T}_{\text{def}}(d_2) \quad (\text{D6})$$

For  $\mathcal{T}_{\text{ty}}$ , for  $i \in \mathcal{I}, d \in \mathcal{D}$  we have  $\boxed{i} \mid d \in \{\boxed{i} \mid d, \boxed{i} \mid d, \boxed{i} \mid d\}$  (term, set, or statement). This rule extracts the type signature for the given expression.

$$\mathcal{T}_{\text{ty}}(\boxed{i} \mid d) = \text{fill} (\llcorner \boxed{0} \Rightarrow \boxed{1} \urcorner, [\mathcal{T}_{\text{ty}}(d), i]) \quad (\text{TY2})$$

$$\mathcal{T}_{\text{ty}}(\boxed{i} \mid d, d') = \text{fill} (\llcorner \boxed{0} \Rightarrow \boxed{1} \urcorner, [\mathcal{T}_{\text{ty}}(d'), \mathcal{T}_{\text{ty}}(\boxed{i} \mid d)]) \quad (\text{TY3})$$

$$\mathcal{T}_{\text{ty}}(\boxed{i} \mid) = \llcorner \text{'r'} \urcorner \quad \mathcal{T}_{\text{ty}}(\boxed{i} \mid) = \llcorner \text{'r set'} \urcorner \quad \mathcal{T}_{\text{ty}}(\boxed{i} \mid) = \llcorner \text{bool} \urcorner \quad (\text{TY1})$$

*Example 7.* When (D3) is applied to the annotated expression

addition (denoted by  $\boxed{\text{plus } \boxed{\# a} + \boxed{\# b}}$ )

the result of the translation is

`s    fixes    plus    ::    'r => 'r => 'r`

where all three of the symbols in the type signature are 'r because the three inner boxes were all **terms**.

If, on the other hand, we want to convert several boxes – again, for  $i \in \mathcal{I}, d \in \mathcal{D}$  we have  $\boxed{i} \mid d \in \{\boxed{i} \mid d, \boxed{i} \mid d, \boxed{i} \mid d\}$  (term, set, or statement) – the following rules turn the boxes into a prefix notation that is Isabelle-friendly, although it is not perfect (See Note 9, below).

$$\mathcal{T}_{\text{pfx}}(\boxed{i} \mid) = i \quad (\text{PFX1})$$

$$\mathcal{T}_{\text{pfx}}(\boxed{i} \mid d) = \text{fill} (\llcorner \boxed{0} \ \boxed{1} \urcorner, [i, \mathcal{T}_{\text{pfx-inner}}(d)]) \quad (\text{PFX2})$$

$$\mathcal{T}_{\text{pfx-inner}}(\boxed{i} \mid, d') = \text{fill} (\llcorner \boxed{0} \ \boxed{1} \urcorner, [i, \mathcal{T}_{\text{pfx-inner}}(d)]) \quad (\text{PFX3})$$

$$\mathcal{T}_{\text{pfx-inner}}(\boxed{i} \mid d, d') = \text{fill} (\llcorner (\boxed{0} \ \boxed{1}) \ \boxed{2} \urcorner, [i, \mathcal{T}_{\text{pfx-inner}}(d), \mathcal{T}_{\text{pfx-inner}}(d')]) \quad (\text{PFX4})$$

Example 8. We see that the annotated expression

```
not set-equal R a non emptyset empty set
```

which may then be manipulated to

```
7 assumes "not (set-equal R emptyset)"
```

Note 9. It is possible, on a case-by-case basis, to translate expressions such as `emptyset` to the more Isabelle-friendly `{}`, or even `equals zero (times a zero)` to `zero = a * zero`, but this kind of automated translation may not be useful or even desirable for the user. We leave it, for the moment, to future work.

Example 10. To illustrate the way that  $\mathcal{T}_{\text{def}}$ ,  $\mathcal{T}_{\text{ty}}$ , and  $\mathcal{T}_{\text{pfx}}$  work together, note

```
ring ring
Definition 1. A R ring R R is
carriernonempty not set-equal R a non emptyset empty set with two
binary operations, plus addition (denoted by plus # a + # b) ...
```

would be translated into

```
5 locale ring =
6   fixes R :: "'r set"
7   assumes "not (set-equal R emptyset)"
8   fixes plus :: "'r => 'r => 'r"
```

$$\mathcal{T}_{\text{thm}}\left(p, \boxed{\boxed{i} \mid \boxed{i} d}\right) = \text{fill}(\perp \text{theorem (in } \boxed{0} \boxed{1}: \text{"} \boxed{2} \text{"} \perp, [p, i, \mathcal{T}_{\text{pfx}}(\boxed{i} d)]) \quad (\text{THM1})$$

$$\mathcal{T}_{\text{thm}}(p, (d_1, d_2)) = \mathcal{T}_{\text{thm}}(p, d_1) \bullet \mathcal{T}_{\text{thm}}(p, d_2) \quad (\text{THM2})$$

Example 11. If the above rule is applied to the theorem in Figure 2,

```
mrule1 mrule1
and equal times a a zero 0 = times zero 0 a a
equal times zero 0 a a = zero 0
```

it would result in the output

```

28 theorem (in ring) mrule1:
29 shows "and (equal (times a zero) (times zero a))
30           (equal (times zero a) zero)"

```

The final rules are filled in as follows. We note that in Isabelle, theorems pass their locale information on to their associated proof. Thus, although we see the declaration of a ring as context for both theorems and definitions (as denoted  $\boxed{\boxed{r} \text{ ring}}$  in Figure 2), and this is necessary for MathLang’s internal type checking, we do not need this information in the translation. Thus, (PF1) returns an empty string.

$$\mathcal{T}_{\text{pf}}(\boxed{d}) = \ulcorner \urcorner \quad (\text{PF1})$$

$$\mathcal{T}_{\text{pf}}(\boxed{i d}) = \text{fill} \left( \ulcorner \text{have } \boxed{0} \urcorner, \left[ \mathcal{T}_{\text{pfx}}(\boxed{i d}) \right] \right) \quad (\text{PF2})$$

*Example 12.* This will translate  $\boxed{\text{equal } \boxed{\text{zero } 0} = \boxed{\text{times } a \boxed{d} \boxed{\text{zero } 0}}$  to the code

```

40 have "equal zero (times a zero)"

```

### 3.2 Resulting Code

With the aid of the rules from Section 3, the Isabelle code in Figure 4 may be constructed (again based on the annotations of the small ring theory in Figure 2). The rules described in this section are sufficient to translate the document given in Figure 2 to Isabelle syntax, and even to get the user very close to a formal proof sketch, but the rules as defined are only sufficient for an extremely small subset of examples. It is not difficult to find a new document for which the translation rules give us an Isabelle-like text which is an insufficient representation of the original mathematics.

The translation shown in Figure 4 shows several specific drawbacks: First, the document does not successfully pass through the Isabelle system for several reasons. There are some trivial things, like the theory name on Line 1, which are simple to add but are not easily provided by an intelligent system. Providing the list of imported theories, also, is difficult for a person who does not know the existing libraries nor how to search them for relevant information. The main failure of the resulting locale definition is the form of expressions such as equality. On a case-by-case basis, such things could be converted (in the case of `equal` and `set-equal`, an infix ‘=’ would satisfy Isabelle nicely), but it is hard to say that such transformations would be generally useful without being highly context-sensitive.

In addition to these shortcomings, the relationship between theorems and proofs is not ordered well. In the original text, it makes perfect sense for the author to write what are essentially two theorems, then prove them in the same order. However, Isabelle prefers proofs to directly follow their assertions, and the fact that lines 36–41 should be moved just before line 35 is not addressed well.

It may not even immediately be evident to the human eye that it is these lines, exactly, which should be associated with theorem `mrule1`. There is the smaller matter that these proofs should be surrounded with `proof . . . qed` pairs, but this issue goes hand-in-hand with the aforementioned problem of discerning which proof lines go with which theorem.

The major hurdle, however, is that for Isabelle to find this theory correct, it requires much more information. None of the proof claims (`have "...`) are justified, and there are significant holes in the reasoning. This is largely due to the fact that the original author simply left many holes which would be evident to a human reader, considering them unnecessary. When this theory file is developed to a point at which Isabelle is completely satisfied, it is approximately 10 times longer.

While these problems are significant, we believe that the current end-result has merit. One of the major benefits is that this can be performed by a mathematician who knows little-to-nothing about Isabelle. The (very incomplete) theory in Figure 4 can then be given to an Isabelle expert for development into a robust theory. This way, they have a starting point in Isabelle syntax, which may save them time in understanding the intent of the document.

## References

1. Fairouz Kamareddine, Robert Lamar, Manuel Maarek, and J. B. Wells. Restoring natural language as a computerised mathematics input method. In *MKM '07* [20], pages 280–295.
2. Fairouz Kamareddine, Manuel Maarek, and J. B. Wells. Toward an object-oriented structure for mathematical text. In *Mathematical Knowledge Management, 4th Int'l Conf., Proceedings*, volume 3863 of *Lecture Notes in Artificial Intelligence*, pages 217–233. Springer, 2006.
3. Fairouz Kamareddine, Manuel Maarek, Krzysztof Retel, and J. B. Wells. Narrative structure of mathematical texts. In *MKM '07* [20], pages 296–311.
4. P. Rudnicki. An overview of the Mizar project. In *Proceedings of the 1992 Workshop on Types for Proofs and Programs*, 1992.
5. B. Buchberger, A. Crăciun, T. Jebelean, L. Kovács, T. Kutsia, K. Nakagawa, F. Piroi, N. Popov, J. Robu, M. Rosenkranz, and W. Windsteiger. Theorema: Towards computer-aided mathematical theory exploration. *Journal of Applied Logic*, pages 470–504, 2006.
6. Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer-Verlag, 2002.
7. M. Wenzel. Isar – a generic interpretative approach to readable formal proof documents. In *Theorem Proving in Higher Order Logics: 12th Int'l Conf., Proceedings*, volume 1690 of *Lecture Notes in Computer Science*, pages 167–184. Springer, 1999.
8. Muhammad Humayoun. Software specifications and mathematical proofs in natural languages. Poster in Journées scientifiques du cluster, ISLE Rhône-Alpes, Domaine universitaire, Grenoble, France, 2008.
9. Aarne Ranta. Grammatical framework: A type-theoretical grammar formalism. *J. Funct. Programming*, 14(2):145–189, 2004.

10. Thomas L. Heath. *The 13 Books of Euclid's Elements*. Dover, 1956. In 3 volumes. Sir Thomas Heath originally published this in 1908.
11. Edmund Landau. *Grundlagen der Analysis*. Chelsea, 1930.
12. LogiCal Project, INRIA, Rocquencourt, France. *The Coq Proof Assistant Reference Manual – Version 8.0*, June 2004. Available at <ftp://ftp.inria.fr/INRIA/coq/V8.0/doc/>.
13. Klaus Grue. The layers of logiweb. In MKM '07 [20], pages 250–264.
14. Donald Ervin Knuth and Silvio Levy. *The CWEB System of Structured Documentation: Version 3.0*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1994.
15. Joseph A. Gallian. *Contemporary Abstract Algebra*. Houghton Mifflin Company, 5th edition, 2002.
16. Fairouz Kamareddine and J. B. Wells. Computerizing mathematical text with MathLang. In Mauricio Ayala-Rincon and Heusler, editors, *Proc. Second Workshop on Logical and Semantic Frameworks, with Applications*, pages 5–30, Ouro Preto, Minas Gerais, Brazil, 2008. Elsevier. The LSFA '07 (post-event) proceedings is published as vol. 205 (2008-04-06) of *Elec. Notes in Theoret. Comp. Sci.*
17. Manuel Maarek. *Mathematical Documents Faithfully Computerised: the Grammatical and Text & Symbol Aspects of the MathLang Framework*. PhD thesis, Heriot-Watt University, Edinburgh, Scotland, June 2007.
18. James Clark and Steve DeRose. *XML Path Language (XPath) Version 1.0*. W3C (World Wide Web Consortium), <http://www.w3.org/TR/xpath>, 1999.
19. Amine Chaieb. Semiring normalization and groebner bases. File Groebner\_Basis.thy, in Isabelle2007 distribution, October 2007.
20. *Towards Mechanized Mathematical Assistants (Calculemus 2007 and MKM 2007 Joint Proceedings)*, volume 4573 of *Lecture Notes in Artificial Intelligence*. Springer, 2007.

## Rings

**Definition 1.** A ring  $R$  is a nonempty set with two binary operations, addition (denoted by  $a + b$ ) and multiplication (denoted by  $ab$ ), such that for all  $a, b, c$  in  $R$ :

1.  $a + b = b + a$ .
2.  $(a + b) + c = a + (b + c)$ .
3. There is an additive identity  $0$ . That is, there is an element  $0$  in  $R$  such that  $a + 0 = a$  for all  $a$  in  $R$ .
4. There is an element  $-a$  in  $R$  such that  $a + (-a) = 0$ .
5.  $a(bc) = (ab)c$ .
6.  $a(b + c) = ab + ac$  and  $(b + c)a = ba + ca$ .

**Theorem 2.**

1.  $a0 = 0a = 0$ .
2.  $a(-b) = (-a)b = -ab$ .

**Proof.**

Consider rule 1.

Clearly,

$$0 + a0 = a0 = a(0 + 0) = a0 + a0. \quad (1)$$

So, by cancellation,  $0 = a0$ . Similarly,  $0a = 0$ .

To prove rule 2, we observe that  $a(-b) + ab = a(-b + b) = a0 = 0$ .

Adding  $-(-ab)$  to both sides yields  $a(-b) = -ab$ . The remainder of rule 2 is done analogously.  $\square$

**Fig. 1.** Ring theory text as taken from *Contemporary Abstract Algebra* [15].

preface

equal r r set-equal r r in r in and r in not-equal r r not-equal r r emptyset

### Rings

definition

ring

**Definition 1.** A **ring**  $R$  is a **non-empty set** with two binary operations, **addition** (denoted by  $+$ ) and **multiplication** (denoted by  $\cdot$ ), such that for all  $a, b, c, d$  in  $R$ :

- $a + b = b + a$
- $a + (b + c) = (a + b) + c$
- There is an **additive identity**  $0$ . That is, there is an element  $0$  in  $R$  such that  $a + 0 = a$  for all  $a$  in  $R$ .
- There is an element  $-a$  in  $R$  such that  $a + (-a) = 0$ .
- $a \cdot (b \cdot c) = (a \cdot b) \cdot c$
- $a \cdot (b + c) = a \cdot b + a \cdot c$  and  $(a + b) \cdot c = a \cdot c + b \cdot c$

**Theorem 2.**

- $a \cdot 0 = 0 \cdot a = 0$
- $a \cdot (-b) = -(a \cdot b)$  and  $(-a) \cdot b = -(a \cdot b)$

**Proof.**

Consider rule 1.

Clearly,

$$a \cdot 0 = a \cdot (0 + 0) = a \cdot 0 + a \cdot 0$$

$$a \cdot 0 = a \cdot 0 + a \cdot 0$$

So, by cancellation,  $a \cdot 0 = 0$ . Similarly,  $0 \cdot a = 0$ .

To prove rule 2, we observe that

$$a \cdot (-b) + (a \cdot b) = a \cdot (-b + b) = a \cdot 0 = 0$$

Adding  $-(a \cdot b)$  to both sides yields  $a \cdot (-b) = -(a \cdot b)$ . The remainder of rule 2 is done analogously.  $\square$

Fig. 2. Ring theory text (Fig. 1) enhanced with CGa information (See Sec. 2.1).

```

7  theory Groebner_Basis
8  imports NatBin
...
14 begin
...
259 locale gb_field = gb_ring +
260   fixes divide :: "'a \ $\rightarrow$  'a \ $\rightarrow$  'a"
261   and inverse :: "'a \ $\rightarrow$  'a"
262   assumes divide: "divide x y = mul x (inverse y)"
263   and inverse: "inverse x = divide r1 x"
...
338 lemma no_zero_divisors_neq0:
339   assumes az: "(a::'a::no_zero_divisors) \ $\neq$  0"
340   and ab: "a*b = 0" shows "b = 0"
341 proof -
342   { assume bz: "b \ $\neq$  0"
343     from no_zero_divisors [OF az bz] ab have False by blast }
344   thus "b = 0" by blast
345 qed
...
440 end

```

**Fig. 3.** Excerpts of code [19] from Isabelle/HOL distribution.



```

theory (* name *)
imports (* theories *)
begin

5  locale ring =
    fixes R :: "'r set"
    assumes "not (set-equal R emptyset)"
    fixes plus  :: "'r => 'r => 'r"
    fixes times :: "'r => 'r => 'r"
10  fixes a  :: "'r"
    assumes "a : R"
    fixes b  :: "'r"
    assumes "b : R"
    fixes c  :: "'r"
    assumes "c : R"
15  assumes "equal (plus a b) (plus b a)"
    assumes "equal (plus (plus a b) c) (plus a (plus b c))"
    fixes zero :: "'r"
    assumes "zero : R"
20  assumes "equal (plus a zero) a"
    fixes negative :: "'r => 'r"
    assumes "equal (plus a (negative a)) zero"
    assumes "equal (times a (times b c)) (times (times a b) c)"
    assumes "equal (times a (plus b c)) (plus (times a b) (times a c))"
25  assumes "(times (plus b c) a) (plus (times b a) (times c a))"

theorem (in ring) mrule1:
shows "and (equal (times a zero) (times zero a))
30      (equal (times zero a) zero)"

theorem (in ring) mrule2:
shows "and (equal (times a (negative b)) (times (negative a) b))
35      (equal (times (negative a) b) (negative (times a b)))"

have "equal (plus zero (times a zero)) (times a zero)"
have "equal (times a zero) (times a (plus zero zero))"
have "equal (times a (plus zero zero))
40      (plus (times a zero) (times a zero))"
have "equal zero (times a zero)"
have "equal (times zero a) zero"

have "equal (plus (times a (negative b)) (times a b))
      (times a (plus (negative b) b))"
45  have "equal (times a (plus (negative b) b)) (times a zero)"
    have "equal (times a zero) zero"
    have "equal (times a (negative b)) (negative (times a b))"

end

```

Fig. 4. Isabelle code created using rules from Section 3 on annotations in Figure 2.