Narrative Structure of Mathematical Texts

Fairouz Kamareddine, Manuel Maarek, Krzysztof Retel, and J. B. Wells

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

Abstract. There are many styles for the narrative structure of a mathematical document. Each mathematician has its own conventions and traditions about labeling portions of texts (e.g., *chapter*, *section*, *theorem* or *proof*) and identifying statements according to their logical importance (e.g., *theorem* is more important than *lemma*). Such narrative/structuring labels guide the reader's navigation of the text and form the key components in the reasoning structure of the theory reflected in the text.
We present in this paper a method to computerise the narrative structure of a text which includes the relationships between labeled text entities. These labels and relations are input by the user on top of their natural language text. This narrative structure is then automatically analysed to check its consistency. This automatic analysis consists of two phases: (1) checking the good-usage of labels and relations (i.e., that a "proof" justifies a "theorem" but cannot justify an "axiom") and (2) checking that the logical precedences in the document are self-consistent.
The development of this method was driven by the experience of computerising a number of mathematical documents (covering different authoring styles). We illustrate how such computerised narrative structure could be used for further manipulations, i.e. to build a skeleton of a formal document in a formal system like Mizar, Coq or Isabelle.

1 Introduction

The past forty years have seen a sharp increase in the use of the computer by the mathematician for his work purposes. Such use covers communication, authoring, processing, and checking/verifying mathematical knowledge. There exists already a number of flexible computer tools that allow to produce aesthetic presentations of a mathematical document. This presentation, among others, comprises of a clear structure of a document, and usage of a 'fancy' and easy to read fonts and symbols. However, the presentation of a document and its structure also depends on the style of the mathematician and is usually expressed in terms of structural components (e.g., chapter or section) and mathematical components (e.g., lemma or proof). Moreover, a clear appearance of such components as well as explicitly specified hyperlinks between such components enhances the readability of the document and makes the navigation of a text more enjoyable.

Different styles of writing mathematics. The presentation of a mathematical document is a matter of writing style and involves among other things, a

narrative structure of the document. This narrative structure plays an important narration role throughout the theory presented. Clearly expressed hyperlinks between mathematical components show logical dependencies which help the reader recognize the theory structure of a paper before reading the details. The reader could find his way while reading the document depending on how the structure and dependencies are expressed. One could produce a clear structure of a document by specifying explicitly where the important parts (e.g., sections, definitions, etc.) start and end and also where the dependencies are clearly expressed. In such case, the reader has a clear view of the theory in the document (see Figure 8). Otherwise, if the mathematician writing style is newspaper-like, the reader will have a difficult task finding his way in the document (see for instance example above).

> We prove that *two congruences can be added or subtracted from each other provided both have the same modulus.*
>
> Let
> $$a \equiv b \pmod{m} \text{ and } c \equiv d \pmod{m}. \quad (2)$$
> In order to prove that $a + c \equiv b + d \pmod{m}$ and $a - c \equiv b - d \pmod{m}$ it is sufficient to apply the identities
> $$a + c - (b + d) = (a - b) + (c - d)$$
> and $(a - c) - (b - d) = (a - b) - (c - d)$.
> Similarly, using the identity
> $$ac - bd = (a - b)c + (c - d)b,$$
> we prove that congruences (2) imply the congruence $ac \equiv bd \pmod{m}$.
>
> Consequently, we see that *two congruences having the same modulus can be multiplied by each other.* [...]
>
> It follows from the theorem on the multiplication of congruences that *a congruence can always be multiplied throughout by any integer and that each side of a congruence can be raised to the same natural power.* [...]
>
> W.Sierpiński [12, Chapter V, §1]

Motivations. In this paper we concentrate on the computerisation of the narrative structure of a document. Our main motivations are as follow:

1. *To handle the structure of a mathematical document as it appears on paper and at the same time allowing further computerisation and analysis.* Our proposed annotation system can deal with different styles of writing mathematics.
2. *To allow the presentation of a text with different layouts.* Currently the presentation of the structure of a documents is rather linear. For instance, it may not be clear which parts (chunks of text) of a mathematical document depend on which (which theorem depends on which lemma or definition etc.). Ideally the presentation of a document should be flexible, and allow to manipulate the view of the document to produce fully automatically different views on the structure of a document: *dependency graph*, graph of logical precedences, skeleton of the document in a chosen formal system, etc.
3. *To allow further formalisation.* Capturing the narrative structure of a document is not only for computerisation purposes, but also for further formalisation. The automatically generated views of the narrative structure of a text are very important to generate further forms of the text including a more formalised version (in a chosen formal system) as we illustrate in this article.

Contributions. Our contributions can be summarised as follows:

1. *A Document Rhetorical aspect (DRa) ontology and a related annotation system.* We present an ontology and an associated markup system, that offer a way to explicit the traditional components of a mathematical text (such as chapters, sections, proofs) and the dependencies between them. The ontology is very easy for the mathematician to use and requires no extra skills.

2. *Automatic processing of the narrative structure of a text.* Automated programs take the mathematician's DRa annotated mathematical text and build a number of internal representations and screen views of the narrative structure of the text. This includes: a *dependency graph* that represents relations between annotated parts of text, and its graph of logical precedences. The internal representations are used for further consistency analysis and formalisation while the screen views show the reader the narrative structure.
3. *Reuse of the narrative structure of a document.* We show how the automatically generated representations of the narrative structure of a document lead to a skeleton of a formal document in the Mizar Language. Similar steps lead to formal skeletons in other formal systems.

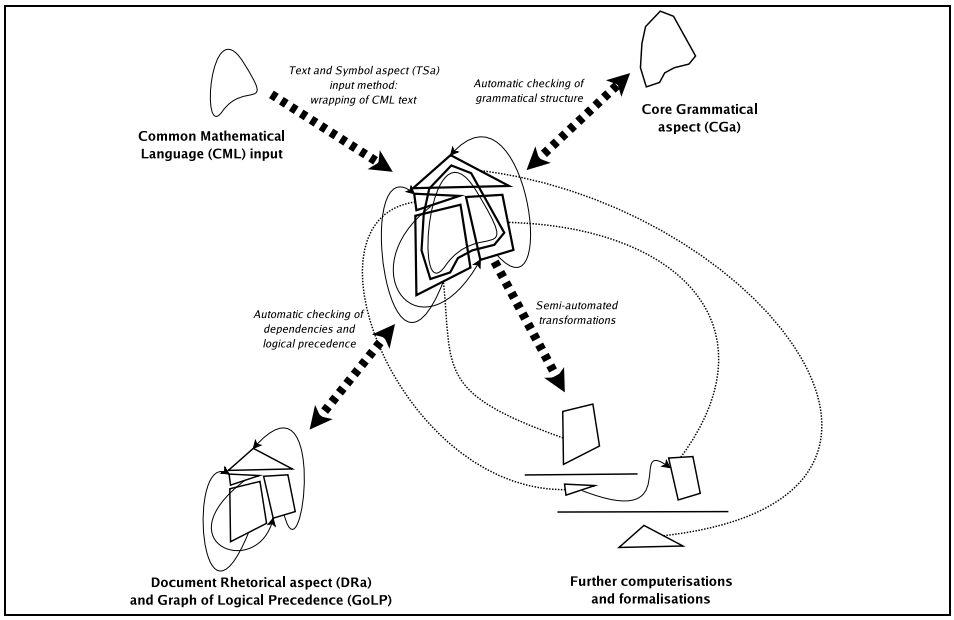


Fig. 1. MathLang project roadmap.
This diagram illustrates both the MathLang decomposition by means of knowledge aspects and the MathLang authoring process. The central piece represents a MathLang document while the corner pieces are elements of the authoring process.

CML (top-left corner) is the starting point of any MathLang authoring. The blobby shape indicates that CML is highly automation-unfriendly. **TSa**, presented in [4,3], offers facilities to associate portions of text with their meaning in terms of computerised data.	**CGa** (top-right corner) makes explicit the grammatical role played by the elements of the texts. Its shapes mimic those of CML but it has a more precise structure. An automatic checking validates this grammatical aspect [5].
DRa (bottom-left corner) gives the narrative structure of a text which includes the relationships (pictured as arrows) between labeled text entities (pictured as polygons) (Section 2). **GoLP** is an interpretation of these relationships in terms of logical precedence (Section 3).	Starting from a MathLang document we experimented with **further computerisations/formalisations** (Section 5 and [2]). We illustrate this point (bottom-right corner) with a proof-tree like interpretation which is a possible starting point for building a formalised proof.

Thick arrows draw the MathLang authoring process which starts with CML and continues to further formalisations. CGa and DRa steps could be made simultaneously or separately.

Outline. In Section 1.1 we present the MathLang roadmap. Section 2 describes our approach to annotating the structure of mathematical documents and gives the DRa ontology used in the annotation system. Section 3 presents automatic transformations of the document's narrative structure into different views. We also present a formal mathematical model describing those automatically generated views. In Section 4 we present the analysis process of the dependency graph generated from the annotation. In Section 5 we express how the structure and its different views are used to build a skeleton of a part of Mizar article. Finally, in Section 6 we describe related work, conclude and discuss future work.

1.1 The MathLang project roadmap

Since 2001, the MathLang project, has developed a number of prototypes for computerising mathematics. MathLang aims to give *alternative* techniques for capturing the mathematical knowledge of a mathematical text in a way that permits the transformation of this knowledge into new computerised and/or formalised versions while accommodating different degrees of formalisation, different mathematical editing/checking tools and different proof checkers. We started from de Bruijn's Mathematical Vernacular [1] (MV), and Nederpelt's Weak Type Theory (WTT) whose proof theory was developed by Kamareddine [6] and were faced with the huge challenge of how to really create a path from original mathematical texts into fully formalised ones and how would this path differ for different choices of texts, text editors, logical frameworks, and proof checkers.

Extensive computerisations of different mathematical texts (some taken fully from natural language to different levels of computerisation and finally to full Mizar), continue to shape the MathLang language. Its expressiveness has been increased in comparison with MV and WTT. Moreover, MathLang adopted to decompose the computerisation process by means of knowledge components called **aspects**. In the current development of MathLang we have formalised and implemented three aspects: CGa and TSa (see below), and DRa which is the subject of this article. Figure 1 illustrates how the MathLang aspects are combined.

The Core Grammatical aspect (CGa) is a formal language derived from MV and WTT which explicits the grammatical role played by the elements of a mathematical text. CGa has a finite set of grammatical categories: **Terms**, **sets**, **nouns**, **adjectives**, **phrases**, **statements**, **declarations** and **contexts/local-scoping**, **definitions**, **steps**, **blocks**. The MathLang automated type system [5] checks whether the reasoning parts of a document are coherently built.

The Text and Symbol aspect (TSa) builds the bridge between a mathematical text and its grammatical interpretation and adjoins to each CGa expression a string of words and/or symbols which aims to act as its representation. We added information on how each CGa element should be printed on paper or on screen. This makes MathLang's encoding of mathematical texts faithful to traditional mathematical authoring [4]. TSa adds on top of a mathematical text a new dimension to the document where colored boxes represent the grammatical categories of the CGa. We implemented TSa in a *plugin* for the scientific text editor TEXmacs (`http://www.texmacs.org/`).

2 Annotating the narrative structure of a document

This section gives our approach to annotate mathematical documents. The mathematical text on the left hand column of Figure 8 is used as our main example.

2.1 What does the mathematician have to do?

To annotate a mathematical text, the mathematician follows three easy steps:

1. He wraps chunks of text with unique boxes and names each box. Unicity allows avoiding problems when relating between some boxes. For our example of Figure 8, the names are: $S2, D1, D2, T1, PT1, T2, L1, PL1, PT2$.
2. He assigns to each (name of a) box, structural or/and mathematical rhetorical roles which this box may play. The user can either use the structural/mathematical roles listed in Table 1, or specify his own. For our example of Figure 8, we assigned the roles stated in the left hand column in the table below.
3. He explicits the relations between wrapped chunks of texts using the relation names of Table 1. For our example of Figure 8, the relations are presented in the right hand side of the table below.

Assigned rhetorical roles	Relations
($S2$, hasStructuralRhetoricalRole, section)	
($D1$, hasMathematicalRhetoricalRole, definition)	($PT1$, justifies, $T1$)
($D2$, hasMathematicalRhetoricalRole, definition)	($PT2$, justifies, $T2$)
($T1$, hasMathematicalRhetoricalRole, theorem)	($PL1$, justifies, $L1$)
($PT1$, hasMathematicalRhetoricalRole, proof)	($PT1$, uses, $D1$)
($T2$, hasMathematicalRhetoricalRole, theorem)	($PT2$, uses, $L1$)
($L1$, hasMathematicalRhetoricalRole, lemma)	($PL1$, uses, $T1$)
($PL1$, hasMathematicalRhetoricalRole, proof)	($PL1$, uses, $D1$)
($PT2$, hasMathematicalRhetoricalRole, proof)	

We use the RDF triples [16] to represents a statement of a relationship between the things denoted by the names of boxes annotated by mathematician that it links (see triples in the table). Each triple is expressed by *subject-predicate-object* triple, where a *predicate* (i.e., a property) denotes a relationship. The order in a triple between *subject* and *object* is significant, and when transformed into a *dependency graph* the direction of the arc the triple makes, always points toward the *object*.

2.2 The annotation system ontology

Looking at different styles of mathematical knowledge representation we can distinguish two kinds of document structural units: *division elements* and *mathematical units*. *Division elements* express a textual structure (e.g., chapter or section) of a mathematical text. *Mathematical units*, are usually expressed in mathematical textbooks and papers in terms of theorem, lemma or remark. Some *mathematical units*, for instance "proof", are more or less hinted by the authors' style of writing (see the example in the introduction). The human reader is able to recognise and infer them only by looking carefully at the original text.

We express and tag these structural units (*division elements* and *mathematical units*) explicitly. By explicit annotations of structure units we refine the content of the already captured original text, and at the same time we give a wider possibility for (semi)automatic text manipulation (see Sections 3 and 5).

Ontology. The literature contains many definitions of an ontology. Roughly speaking, an ontology [17] is a representation of terms with their relationships

in a specific domain. An ontology in computer science describes: (1) individuals/instances of a class; the basic objects of ontology (e.g., "Bach" is an instance of class "Person" (`http://www.foaf-project.org/`); (2) classes/abstract groups, sets, or collections of objects (e.g., "Person"); (3) relations/properties, ways that objects can be related to one another, e.g., relation *childOf* (`http://vocab.org/relationship/`), could be used to present a statement in terms of RDF triple: ("Sebastian Bach",*childOf*,"Ambrosius Bach").

DRa ontology in a nutshell. To model our DRa ontology we used the OWL-DL Web Ontology Language, which is the OWL sub-language so-named due to its correspondence with *description logics* [14]. An OWL ontology may include description of classes, instances of them and properties between their elements. The information presentation using OWL is very powerful in the way that it is suitable for exchanging information and processing by other software applications.

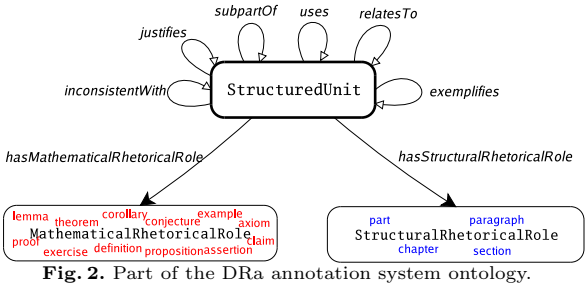


Fig. 2. Part of the DRa annotation system ontology.

Following OWL, our DRa ontology explicits the formal description of classes (whose names start with capital letter, e.g., StructuredUnit), individuals (e.g., section) and properties/relations (whose names start with small letter, e.g., justifies or hasMathematicalRhetoricalRole) in a domain of DRa.

The DRa concepts are given as three disjoint OWL *classes* [15] (see Figure 2):

1. StructuredUnit.
2. MathematicalRhetoricalRole whose instances are lemma, proof, etc.
3. StructuralRhetoricalRole whose instances are chapter, section, etc.

Relations between various instances are given as OWL *object properties* [15]:

1. the ownership relation between structural units and the roles played in a text, i.e. hasMathematicalRhetoricalRole and hasStructuralRhetoricalRole.
   E.g., in Figure 8, $(D1, \text{hasMathematicalRhetoricalRole}, \text{definition})$.
2. The relations between instances of the class StructuredUnit:
   (a) relatesTo, justifies, subpartOf, uses, exemplifies, inconsistentWith.

The relations of the first kind (item 1) are modeled as *object properties* (i.e., link individuals of one class to individuals of another class). The relations presented in item (2) are modeled as *subproperties* of a generic *object property* – specifies, i.e. $(A, \text{specifies}, B)$, where A, B are instances of class StructuredUnit.

Relations between instances of the classes MathematicalRhetoricalRole or StructuralRhetoricalRole and the XML schema datatype (xsd:string) are given as OWL *datatype properties* [15] (i.e., they link individuals of a class to the XML Schema datatypes [18]): hasOtherMathematicalRhetoricalRole and hasOtherStructuralRhetoricalRole. The existence of these relations gives the freedom if one wants to provide a new label not appearing in Table 1, this is possible through the usage of a variant property called hasOtherStructuralRhetoricalRole for *division elements* and hasOtherMathematicalRhetoricalRole for *mathematical units*. The range of

values of such properties is restricted to the XML Schema datatype "string". $(A, \mathsf{hasOtherMathematicalRhetoricalRole}, discussion)$.

Description
Instances for the hasStructuralRhetoricalRole *property:* preamble, part, chapter, section, paragraph, *etc.*
Instances for the hasMathematicalRhetoricalRole *property:* lemma, corollary, theorem, conjecture, definition, axiom, claim, proposition, assertion, proof, exercise, example, problem, solution, *etc.*
Relation
Types of relations: relatesTo, justifies, subpartOf, uses, exemplifies, inconsistentWith

Table 1. DRa annotations.

Since both *division elements* and *mathematical units* express the boundaries of chunks of text, we included them into one class (StructuredUnit) within the DRa ontology. The two disjoint classes: StructuralRhetoricalRole and MathematicalRhetoricalRole allow to represent the different roles played by *division elements* and *mathematical units*. Instances of the first class are conventional names for *division elements* which might at the same time express the hierarchical level of a document structure, i.e., chapter, section, etc. Whereas, all instances of the class MathematicalRhetoricalRole are the common labels and names for *mathematical units*, i.e., theorem, corollary, etc. All instances of the classes StructuralRhetoricalRole and MathematicalRhetoricalRole, are fixed conventional labels used to annotate mathematical documents.

The DRa ontology allows to relate particular instance of the class StructuredUnit with any instance of StructuralRhetoricalRole and MathematicalRhetoricalRole via the properties hasStructuralRhetoricalRole and hasMathematicalRhetoricalRole respectively. We allow the use of both properties when relating to an instance of a class StructuredUnit. This enables to specify, for instance, that a chunk of text plays the structural role "section" and concurrently plays the mathematical role "theorem". By stating two properties simultaneously in a document annotation we allow to encode different styles of writing mathematics.

While annotating the narrative feature of a document, we make explicit correlations between distinct recognised chunks of text. For this, within the DRa ontology, we introduced other properties which describe relations between instances of the class StructuredUnit. These properties are mainly oriented to represent dependencies between *mathematical units* and/or *division elements*. Our DRa ontology clarifies the important relationships in a text. The properties used to represent relations between chunks of text, have human readable names: relatesTo, justifies, subpartOf, uses, inconsistentWith, exemplifies. In a formal system, some of these properties have formal meanings:

1. $(v_1, \mathsf{justifies}, v_2)$ – v_1 describes a proof object that proves the formula v_2.
2. $(v_1, \mathsf{uses}, v_2)$ – (1) All/some variables under the general quantifiers that have been applied in a formula v_2 have been instantiated in formula v_1 which could be proved via simple reasoning where v_2 appears among references needed to prove v_1. (2) The formula v_2 has been unfolded or folded in the formula v_1.
3. $(v_1, \mathsf{subpartOf}, v_2)$ – (1) if v_2 is a formula, then v_1 is an inseparable part of that formula; (2) if v_2 is a proof object, then v_1 is part of that proof object.
4. $(v_1, \mathsf{inconsistentWith}, v_2)$ – if v_1 and v_2 are proof objects of one formula, then the environment in which those proof objects were achieved is inconsistent.

3 Automatic transformation of a DRa annotated text

In this section we show how to use the DRa annotated text to automatically create a number of views of the text including the *dependency graph* (that represents relations between annotated parts of text) and the graph of logical precedences. These views and graphs are formally described in this section.

$G = (V, A, E)$
$V = \{v \mid v = nodeId\}$ – set of vertices
$A = \{a \mid a = (v, r) \ \wedge \ r \in MR \cup SR \ \wedge \ MR \cap SR = \emptyset\}$ – set of vertices attributes
$E = \{e \mid e = (v_{src}, \alpha, v_{anch}) \ \wedge \ v_{src}, v_{anch} \in V \ \wedge \ \alpha \in L_d\}$ – set of edges

where
L_d = {relatesTo, justifies, subpartOf, uses, inconsistentWith, exemplifies} – the set of allowed labels in a dependency graph
MR – the set of MathematicalRhetoricalRoles, cf. Table 1
SR – the set of StructuralRhetoricalRoles, cf. Table 1
$nodeId$ – a unique name/identifier given by the user while wrapping the text with boxes

Fig. 3. Formal presentation of a *dependency graph*.

3.1 The automatically generated dependency graph of a document

A document's dependency graph is a directed labeled graph with attributes assigned to the vertices (see Figure 3). The vertices (resp. attributes resp. edges) of such graph are the names of boxes (resp. mathematical or structural rhetorical roles resp. relations) specified by the user during the first (resp. second resp. third) step of the annotation of the document described Section 2.1.

Figure 4 (and the right hand side of Figure 8) presents the dependency graph of our particular example. This graph consists of (1) relations between parts of the text which are represented by visible arrows, and (2) graph nodes which have specified (but not visible) mathematical or/and structural rhetorical roles. Dependencies between the annotated chunks of text play an important role in mathematical knowledge representation. Thanks to those dependencies, the reader finds his own way while reading the text without the need to understand all its subtleties. Moreover, we will show in the next sections that these dependencies give the ability to present other views on a document, and structuring the skeleton of a document in a formal language Mizar. Dependencies graph (view as in Figure 4) are found automatically from the mathematicians' input in Section 2.1.

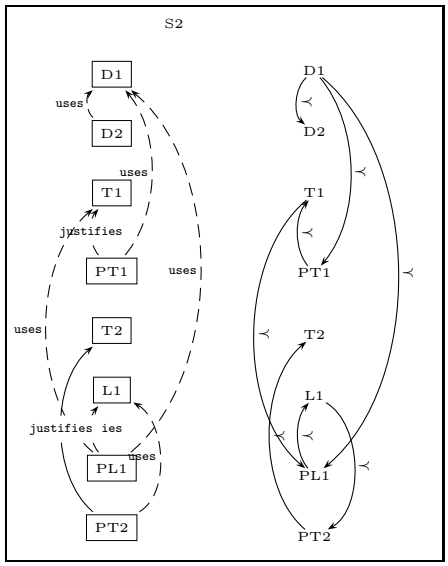


Fig. 4. *Dependency graph* and *GoLP*
On the left hand side we have the automatically generated presentation of the *dependency graph* constructed from the input of the mathematician in Section 2.1 for our main example of Figure 8. The right hand side of the figure presents automatically generated *GoLP* from the *dependency graph*.

3.2 Logical precedences of mathematical relations

The annotation identifies and makes explicit different parts of the text, stores either the mathematical or structural or both roles of each chunk of text, and annotates the relations between recognised chunks of text (see Section 2.1). The usage of the DRa system allows us to express relations explicitly in the computerised version of the original text. This explicit representation of relations allows to build a graph of logical precedences between different chunks of the text.

The ***logical precedence*** between two chunks of text indicates the relative positions of the chunks in a sequence of reasoning steps. These steps, together with other steps in the document, contribute to the analysis of the logical correctness of the original text. *Logical precedence* is independent of the sequential appearance of the chunks of text in a document. For instance, in Figure 8, the "Proof" (node $PT1$) is stated after "Theorem 1.19" (node $T1$). However, the *logical precedence* between node $PT1$ and node $T1$ is the other way (see the direction of the arrow established between both nodes shown in Figure 4). In such case we say that $PT1$ logically precede $T1$.

We assume the following kinds of *logical precedences*, namely: *strong logical precedence*, *weak logical precedence* and *not-specified logical precedence* , annotated in the paper as: $\prec$, $\preceq$ and $\backsimeq$ respectively.

In Section 2.2 we gave a DRa ontology which allows to explicit the relations between recognised StructuredUnits in a document. Each such stated relation expresses its own *logical precedence* (see Table 2).

Relation	*Annotation triple*	*Logic precedence pattern*
relatesTo	$(A, \mathsf{relatesTo}, B)$	$A \backsimeq B$
subpartOf	$(A, \mathsf{subpartOf}, B)$	$A \preceq B$
justifies	$(A, \mathsf{justifies}, B)$	$A \prec B$
inconsistentWith	$(A, \mathsf{inconsistentWith}, B)$	$B \prec A$
uses	(A, uses, B)	$B \prec A$
exemplifies	$(A, \mathsf{exemplifies}, B)$	$B \prec A$

Table 2. *Logical precedencies* of specific relations

3.3 The automatically generated graph of logical precedences:GoLP

Using the *logical precedence* of each relation (see Table 2), one can automatically build for a mathematical text, a *graph of logical precedences* (GoLP). Figure 4 gives the automatically generated GoLP for our main example. GoLP is a directed graph with labeled edges, achieved by automatic transformation of the dependency graph using the transformation function *Trans* (see Figure 5). In a GoLP, the direction of an edge together with a label of that edge

Graph transformation

$Trans : G_{DG} \rightarrow G'_{GoLP}$

$Trans((v, a, e)) = (v', e')$

(where $v' = Trans_V(v)$ and $e' = Trans_E(e)$)

Vertex transformation

$Trans_V : V_{DG} \rightarrow V'_{GoLP}$

$Trans_V(v) = v$

Edge transformation

$Trans_E : E_{DG} \rightarrow E'_{GoLP}$

$Trans_E((v_{src}, \mathsf{relatesTo}, v_{anch})) = (v'_{src}, \backsimeq, v'_{anch})$

$Trans_E((v_{src}, \mathsf{justifies}, v_{anch})) = (v'_{src}, \prec, v'_{anch})$

$Trans_E((v_{src}, \mathsf{subpartOf}, v_{anch})) = (v'_{src}, \preceq, v'_{anch})$

$Trans_E((v_{src}, \mathsf{uses}, v_{anch})) = (v'_{anch}, \prec, v'_{src})$

$Trans_E((v_{src}, \mathsf{inconsistentWith}, v_{anch})) = (v'_{anch}, \prec, v'_{src})$

$Trans_E((v_{src}, \mathsf{exemplifies}, v_{anch})) = (v'_{anch}, \prec, v'_{src})$

(where $v'_{src} = Trans_V(v_{src})$ and $v'_{anch} = Trans_V(v_{anch})$)

Fig. 5. Dependency graph transformation function.

expresses the *logical precedence* corresponding to the relation in a dependency graph from which the edge (in GoLP) was achieved. Figure 6 gives the formal definition of a *graph of logical precedences*. Assume that G is the *dependency graph* (DG) shown in Figure 4 and G' is the *graph of logical precedences* ($GoLP$) shown in Figure 4. Using the transformation function *Trans* shown in Figure 5, we can automatically transform G into G'.

$G' = (V', E')$
$V' = \{v' \mid v' = nodeId\}$ – set of vertices
$E' = \{e' \mid e' = (v'_{src}, \alpha', v'_{anch}) \ \wedge \ v'_{src}, v'_{anch} \in V' \ \wedge \ \alpha' \in L_p\}$ – set of edges

where $L_p = \{\trianglerighteq, \trianglerighteq, \prec\}$ – the set of *logical precedences* in GoLP

Fig. 6. Formal presentation of a *graph of logical precedences* (GoLP).

4 Automatic analysis of the dependency graph and GoLP

This section explains the checking of the DRa annotation done in two phases:

1. Checking the annotation of distinct roles of recognised fragments of text and the good-usage of labels and relations.
2. Checking that the GoLP logical precedences are self-contained.

Pre-analysis of the dependency graph The first phase of checking catches some inconsistencies while representing different roles of recognised chunks of text and stated dependencies between them. E.g., if two chunks of text were annotated as "proof" resp. "axiom", and if a relation justifies is stated between them (i.e. $(proof, \mathsf{justifies}, axiom)$), the first stage validation gives a warning to the user while analyzing such stated relation. It returns two warnings: one on the relation type – which might/should be different, and another on the role specified for each chunk of text – which was mistakenly specified.

This checking captures other cases. Assume that one has specified simultaneously two MathematicalRhetoricalRole for a chunk of text, for instance "axiom" and "proposition". In such a case the analysis will point a warning stating that "axiom" cannot be provable, whereas "proposition" can be proved. Similarly, if one simultaneously states two StructuralRhetoricalRole for one chunk of text, the analysis will point the warning, if they extensively differ, for instance "chapter" and "subsection". Where the difference between a "chapter" and a "subsection" is that the background knowledge of a "chapter" is something like an external library, whereas for "subsection" the context is more specific and composed of small chunks of text from the previous sections or chapters.

Checking the consistency of labels in a GoLP To allow the analysis of a GoLP we have identified a number of common relational properties for *logical precedences* (see Table 3). These properties are reusable while checking the labeling consistency in a GoLP – see following section.

We give an extended version of *transitive closure* of a directed graph, which we call *extended transitive closure*. A *transitive closure* of a directed graph (built using for example Roy-Warshall's algorithm [10,11]) differs from our *extended transitive closure* of the graph in the data stored in the latter. We give a formal definition of the *transitive/extended transitive closure* of a directed graph.

Relational properties	*Weak logical precedence*	*Strong logical precedence*
reflexive	$C \preceq C$	
irreflexive		$\neg(C \prec C)$
asymmetric		$C \prec C' \Longrightarrow \neg\ (C' \prec C)$
antisymmetric	$C \preceq C' \wedge C' \preceq C \Longrightarrow C = C'$	
transitive	$A \prec B \wedge B \preceq C \Longrightarrow A \preceq C$	$A \prec B \wedge B \prec C \Longrightarrow A \prec C$
	$(A \prec B \wedge B \preceq C) \vee (A \preceq B \wedge B \prec C) \Longrightarrow A \prec C$	

Table 3. Relational properties of *logical precedences*

Let us take a directed graph $G = (V, E)$ where V is a set of vertices and E is a set of directed edges denoted as (v, w), where $v, w \in V$. We denote by $\pi_{(v,w)} = \{v, v_0, v_1, \ldots, v_k, w\}$ a path from the initial vertex v to the terminal vertex w in G, which goes through the vertices $v_0, v_1, \ldots, v_k$, where $k \in \mathbb{N}$.
A *transitive closure* of graph G is a graph $G^+ = (V, E^+)$ such that E^+ contains an edge (v, w) if and only if G contains a path $\pi_{(v,w)}$. The *extended transitive closure* of graph G is a graph $G^* = (V, E^+, \Pi^*)$ such that $\Pi^* = \{[\pi^1_{(v,w)}; \pi^2_{(v,w)}; \ldots; \pi^k_{(v,w)}] \mid v, w \in V \ \wedge\ \pi^i_{(v,w)} \neq \pi^j_{(v,w)}\ for\ i, j \in \mathbb{N},\ i \neq j\}$ if and only if $\pi_{(v,w)} \in E^+$. We denote by $[\pi^1_{(v,w)}; \pi^2_{(v,w)}; \ldots; \pi^k_{(v,w)}]$ the list of all possible paths from vertex v to vertex w in the graph G^+.

The *extended transitive closure* of a GoLP contains: (1) the vertices of the GoLP (2) the edges (which we get while building a normal transitive closure using Roy-Warshall's algorithm) and (3) the list of all possible paths that create each edge in the original transitive closure of a graph. From such stored data we can generate a list of labels for each edge in the *extended transitive closure* by reusing the relational properties of *logical precedences* (see Table 3). Once we build the *extended transitive closure* of the GoLP, we check for each edge if the paths corresponding to that edge are consistently labeled.

We illustrate the analysis of consistent labeling on the GoLP based on our example. Take the nodes $D1$ and $PL1$, and the edge $(D1, PL1)$ (see Figure 4). In the extended transitive closure of our GoLP we have two paths that form an edge $(D1, PL1)$: (1) a direct path denoted $\pi^d_{(D1,PL1)} = \{D1, PL1\}$, (2) an indirect path denoted $\pi^{ind}_{(D1,PL1)} = \{D1, PT1, T1, PL1\}$. The direct path is labeled with a *strong logical precedence* symbol $\prec$, denoted as $\pi^{d,\prec}_{(D1,PL1)}$. When evaluating the label of the second indirect path $\pi^{ind}_{(D1,PL1)}$, we have to take into account the relational properties of the *logical precedences* shown in Table 3. In our case, we use the transitivity of *strong logical precedence* (symbol: $\prec$) between the three edges: $(D1, PT1)$, $(PT1, T1)$, $(T1, PL1)$. From this, we obtain a label $\prec$ of an indirect path $\pi^{ind}_{(D1,PL1)}$, denoted $\pi^{ind,\prec}_{(D1,PL1)}$, which has the same label as a direct path $\pi^{d,\prec}_{(D1,PL1)}$. Finally, we can say that the edge $(D1, PL1)$ in the graph of logical precedences (GoLP) is labeled consistently.

Labeling consistency validation is performed on each existing edge in the *extended transitive closure* of GoLP built from a dependency graph of the original document. Once we go through the whole checking of the graph we can say that the GoLP is valid according to the consistent labeling.

5 From the document narrative structure to the formal document skeleton in formal systems

So far, the mathematician's DRa annotations of his text in Section 2.1 have been used to automatically produce the dependency graph and the GoLP of the text which explicit the narrative, structural and logical features of the text. In this section, we explain how the automatically generated dependency graph and GoLP are used for further processing and formalisation of the text into formal proof checkers. In particular, we express how the dependency graph together with the GoLP are reused to build a skeleton of a part of a Mizar article – *Text-Proper*. We do not go into the technical details. Instead, we present roughly the transformation hints based on our main example resulting in the Mizar *Text-Proper* skeleton of Figure 7. For extensive details on the passage from the dependency graph and the GoLP into Mizar *Text-Proper*, Mizar formal proof sketch FPS and full Mizar, see [2]

In Section 2.1, the mathematician specified that a big box named $S2$ is an entire section in the document. In Mizar the *Text-Proper* part of a document could be divided into a sequence of *Sections*, where each *Section* starts with `begin` and consists of a sequence of theorems and definitions together with their proofs. The division of the *Text-Proper* into *Sections* has no impact on the correctness of the Mizar document. Hence, the whole box is indicated to be a section by specifying explicitly `begin` at the very top of the right hand side of Figure 7. It also consists of two lines `::Section` and `::Title ...` which are treated as Mizar comments, and are solely oriented for the Mizar user consumption, or the reader of the Mizar file. Inside `::Title ...` it is a good practice (in the Mizar community) to specify the title of this *Section* of the Mizar document.

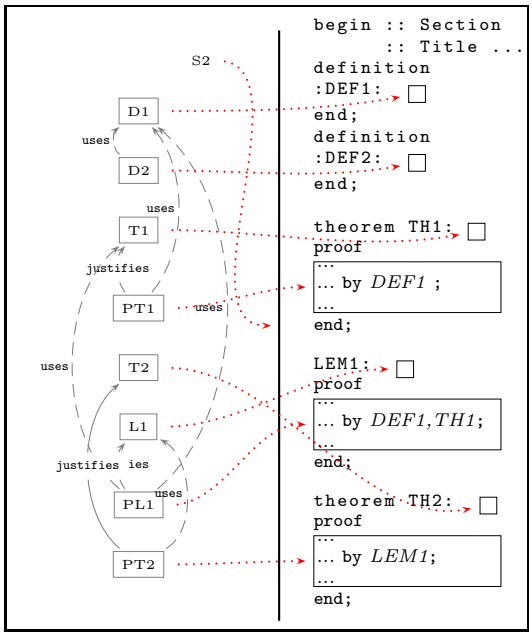


Fig. 7. Transformation into Mizar skeleton. The left hand side reproduces the dependency graph of our example (Figure 8). On the right hand side we show the Mizar *Text-Proper* skeleton of the same example. The arrows from left to right show how the dependency graph is used to build the Mizar *Text-Proper* skeleton. □ stands for holes (incomplete proofs).

Since the mathematician specified for the box $\boxed{D1}$ the MathematicalRhetoricalRole definition, then it is transformed into Mizar syntax as: `definition :DEF1:` $\boxed{D1}$ `end;` (see Figure 7). In Mizar we introduce the label `DEF1` for this definition to be able to refer to it in further reasoning steps.

Since the mathematician specified for the box $\boxed{T1}$ the MathematicalRhetoricalRole theorem, then it is transformed into Mizar syntax as: `theorem` $\boxed{T1}$. Moreover, since the box $\boxed{PT1}$ has the MathematicalRhetoricalRole proof, then we transform it into: `proof` $\boxed{1}$ `end;`. Moreover, since a block of steps having the mathematical role proof is in the relation justifies with a single statement, we can say that this is a particular *Justification* in Mizar, which is transformed into a specific form. See the corresponding transformation arrows in Figure 7.

In the dependency graph of our main example we also specified that some blocks of text use other blocks. For instance a block of text named $\boxed{PT2}$ uses statement $\boxed{L1}$. Here, we transform $\boxed{PT2}$ into a specific Mizar *Proof* block, which contains an expression with *Straightforward-Justification* to statement $\boxed{L1}$, where in Mizar it is reused by referring to a label (i.e., `LEM1`) that was assigned to a statement $\boxed{L1}$ during the transformation into the Mizar syntax.

During the transformation of the dependency graph, we use the GoLP of our main example to be able to put annotated and named chunks of text into a proper Mizar order inside the Mizar skeleton.

The above transformation process leads to a part of a Mizar *Text-Proper* skeleton of a Mizar document (given in Figure 7 for our main example).

The grammatical information of the original text, which is captured by the CGa aspect of MathLang and stored in the MathLang document, can be then used to fill more details in the current skeleton of the Mizar document. This better filled document could be transformed later into a proper Mizar document. The work describing these transformation and usage of the MathLang document for the migration process into the Mizar language, has been described in [2]

6 Related Work, Conclusions and Future Work

Many studies have been carried on the structure of documents. For example, the Text Encoding Initiative Guidelines are international standards that enable the representation of a variety of literary and linguistic texts (`http://www.tei-c.org/`). DocBook (`http://www.docbook.org`), provides a system for writing a structured document using XML. Another tool is OMDoc ([13] - Open Mathematical Documents, see below). All these mentioned systems have the possibility to separate and divide a document into a number of different structural components (sections or mathematical assertions) and allow annotate them in the computerised version of a document. However, our proposed markup system is simpler and is concentrated only on the annotation of the narrative structure of mathematical documents, whereas others are more oriented for capturing also documents subtleties. We believe that separating the concerns and the steps of computerisations can play a very helpful role in developing computer tools that can aid the various levels of computerisation and formalisation.

OMDoc vs DRa – a short comparison. OMDoc presents mathematical knowledge on three levels: the object and formula level, the statement level, and the theory level. What is made explicit by the DRa markup, is similar to the statement level and partly to the theory level in the OMDoc system. The OMDoc markup distinguishes the knowledge elements of a theory into constitutive

ones like symbols, axioms, and definitions (which present the essence of the annotated theory) and non-constitutive ones such as assertions, their proofs, examples (which illustrate properties and attributes of mathematical objects determined by the constitutive statements). This shows a different approach to annotating the same knowledge. The aim of introducing the DRa is to be able to catch and store the narrative structure of the text, and simultaneously allow to stay as close as possible to the original document and the style it was written in. Therefore, on the DRa markup we do not distinguish constitutive or non-constitutive statements. We recognize only one class of elements, called StructuredUnit, and we distinguish the roles they play in mathematical knowledge representation.

Therefore, the purposes/aims of OMDoc and DRa are different. As already mentioned, all instances of the class MathematicalRhetoricalRole in the DRa ontology, are presented as separated disjoint classes in the OMDoc ontology [19]. This means that "axiom" is an ontology class in the OMDoc ontology, where in the DRa it is expressed as an instance (individual) of a class MathematicalRhetoricalRole. This particular name "axiom" expresses a role of the text labeled by that name, and hence in the DRa ontology we annotate it by stating the property hasMathematicalRhetoricalRole whose range value is an appropriate instance (i.e., "axiom") of a class MathematicalRhetoricalRole.

Both annotation systems OMDoc and DRa allow to markup dependencies between statements. In the OMDoc file format they are implemented by means of the `for` attribute to OMDoc's elements (e.g., `<proof for="#id-of-assertion">`). A possible encoding of a part of our main example shown in Figure 8 in OMDoc is sketched[1] on the right hand side. Within the DRa system we annotate the relation as RDF triple, and it might be expressed in the MathLang internal file using any kind of XML-RDF recommendations.

```
<definition xml:id="node-D1.def">
  <CMP>A subset A ⊂ R is inductive if [...]
<assertion xml:id="thm-T1" type="theorem">
  <CMP>Let J be a subset of Z+ [...]
<proof xml:id="proof-PT1" for="#thm-T1">
  <CMP>J is inductive so J contains [...]
```

The other and main advantage of DRa over OMDoc is a possible analysis of the dependency graph and the GoLP, which are automatically built from the performed annotation. This analysis allows to check the annotation of the narrative/rhetorical aspect of a document (see Section 4). Although OMDoc gives a lot of elements and constructions that can be used to structure mathematical documents, these allow the user some software compatibility but no validation yet. The DRa annotation system gives the user a validation tool making it possible to analyse the well formation/encoding of the rhetorical aspect of a document.

Other work [8,9] present a method to express the logical structure of a document and hyperlinks between chunks of mathematical text that enhance the readability of a document and the navigation throughout the text. Proposed methods detect the logical structure of a text and several types of hyperlinks from printed mathematical documents. Our approach differs in the sense that we propose an annotation system that allows to express such logical structure and

[1] For readability and brevity, we show only the opening tag of each XML element for most elements; we use indentation to express nesting.

hyperlinks/relations while authoring a document. Moreover, we use the dependency graph achieved from the annotation to build a formal document skeleton (as we have done in Mizar and can be done in other systems).

Conclusions. We have presented in this paper our approach to computerise the narrative aspect of mathematical texts. We built a DRa ontology which described formally the domain of narrative/structural representations of mathematical knowledge in a document. The ontology allows to share a common understanding of the structure of the represented knowledge among other people and software agents. The ontology separates a domain knowledge (DRa) from the operational knowledge – the actual annotation. By using the ontology we annotated/marked up our main example shown in Figure 8.

We presented the meaning behind the DRa annotation and gave automated tools which generate different representations of the document structure. We showed how the encoded Document Rhetorical aspect annotation could be validated for checking the well formation of the annotation. We also expressed which mistakes made during annotation we are able to automatically catch. Finally we demonstrated how the dependency graph and the graph of logical precedences are used to build the skeleton of Mizar *Text-Proper*.

Future work. MathLang is an ongoing project. The DRa encoding system is a part of the MathLang project. As future work, we need to concentrate on the evaluation and improvement of the DRa system. We need to finish the implementation of the DRa validation rules and test them on a bigger examples. We have to work further on the DRa ontology. In particular we have to refine the instances of a class StructuralRhetoricalRole. Namely, we need to separate the depth level of structural units labels from the actual meaning of a unit. For instance "section" and "subsection", for the representation purposes, differ only on the embedded relation. Therefore we have to investigate how the depth level can be incorporated within the DRa ontology.

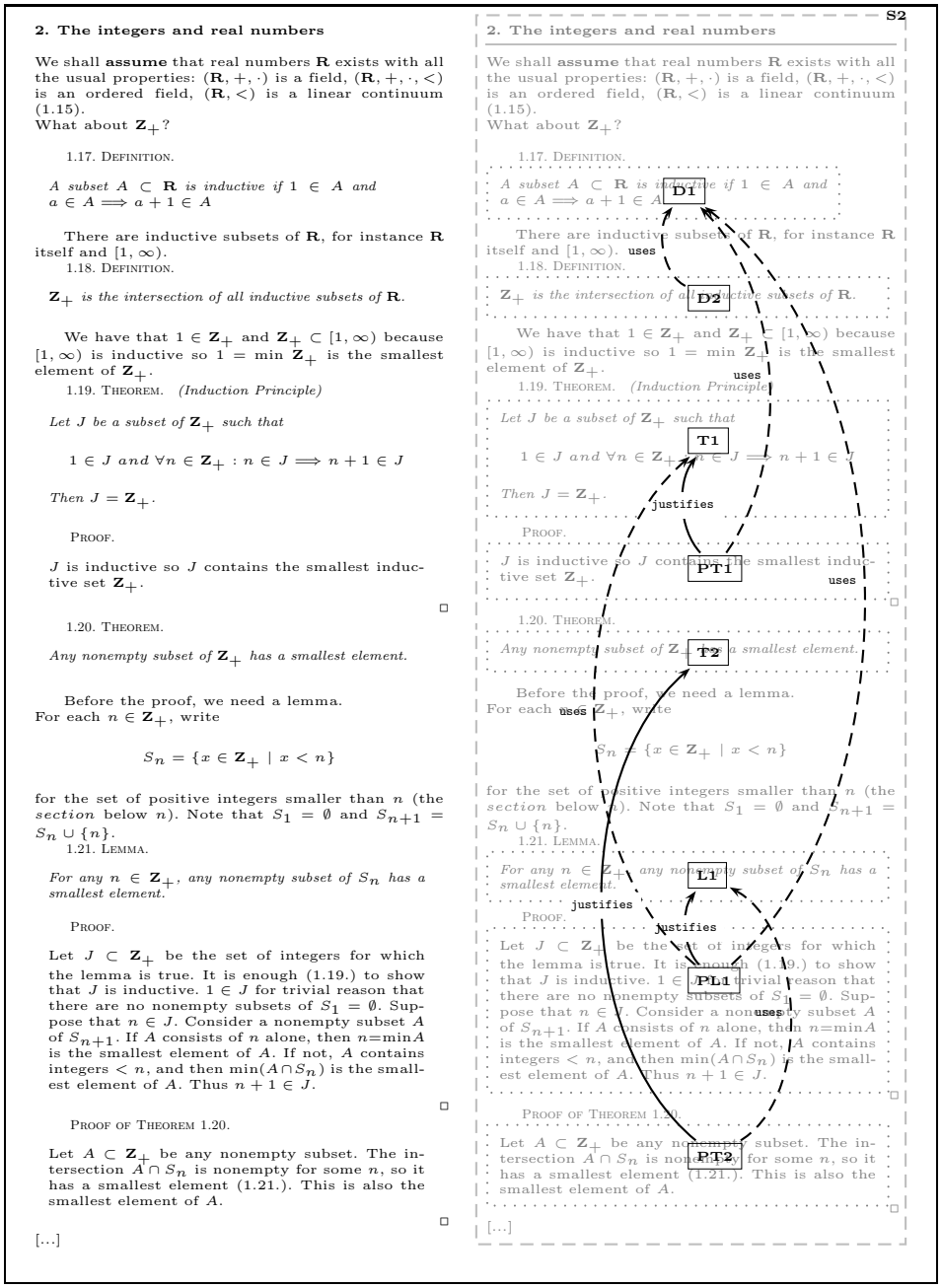
We also need to investigate how a mathematician could add his own intended relation to the DRa system. For instance, he might want to add the explanationOf relation which could be used to express that $(example, \mathsf{explanationOf}, definition)$. We have to incorporate this kind of possibilities within the DRa markup system.

Another advantage is that we do not provide yet another concrete syntax for mathematical encoding. Instead, we incorporate the markup of the narrative aspect of a mathematical text into the existing encoded document. We believe that a clear separation between different aspects of mathematical knowledge and their markup brings a clear guidance for non expert authors. This guidance mainly helps to extract from the original text, different aspects of mathematical knowledge at different phases of its computerisation.

References

1. N.G. de Bruijn. The mathematical vernacular, a language for mathematics with typed sets. In *Workshop on Programming Logic*, Sweden, 1987.
2. Fairouz Kamareddine, M. Maarek, K. Retel and J.B. Wells. Gradual computerisation/formalisation of mathematical texts into mizar. `http://www.macs.hw.ac.uk/~retel/`, 2007.
3. Fairouz Kamareddine, R. Lamar, M. Maarek and J.B. Wells. Restoring Natural Language as a Computerised Mathematics Input Method. `http://www.macs.hw.ac.uk/~mm20/`, 2007.
4. Fairouz Kamareddine, M. Maarek, and J. B. Wells. Flexible encoding of mathematics on the computer. In *Mathematical Knowledge Management, 3rd Int'l Conf., Proceedings*, volume 3119 of *LNCS*, pages 160–174. Springer, 2004.
5. Fairouz Kamareddine, M. Maarek, and J. B. Wells. Toward an object-oriented structure for mathematical text. *In Mathematical Knowledge Management, 4th Int'l Conf.*, volume 3863 of *LNAI*. Springer, 2006. Pages 217–233.
6. Fairouz Kamareddine and R. Nederpelt. A refinement of de Bruijn's formal language of mathematics. *J. Logic Lang. Inform.*, 13(3):287–340, 2004.
7. Jesper M. Moller. General topology. *Authors' notes*, Available at `http://www.math.ku.dk/~moller/e03/3gt/notes/gtnotes.pdf`, last visit 2007-02-25.
8. K. Nakagawa and M. Suzuki. Mathematical knowledge browser with automatic hyperlink detection. *In Mathematical Knowledge Management, 4th Int'l Conf.*, volume 3863 of *LNAI*. Springer, 2006.
9. K. Nakagawa, A. Nomura and M. Suzuki. Extraction of Logical Structure from Articles in Mathematics *In Mathematical Knowledge Management, 4th Int'l Conf.*, volume 3863 of *LNAI*. Springer, 2006.
10. B. Roy. Transitivité et connexité. *C. R. Acad. Sci. Paris*, 249:216–218, 1959.
11. S. Warshall. A theorem on boolean matrices. *J. ACM*, 9(1):11–12, 1962.
12. W. Sierpiński. Elementary Theory of Numbers. *PWN*, Warszawa, 1964.
13. Michael Kohlhase. OMDoc: An Open Markup Format for Mathematical Documents (Version 1.2). *Volume 4180 of LNAI*, Springer Verlag, 2006.
14. D. L. McGuinness and F. van Harmelen. *OWL Web Ontology Language Overview.* W3C Recommendation, 2004.
15. M. K. Smith, Ch. Welty and D. L. McGuinness. OWL Web Ontology Language Guide. *W3C Recommendation*, 2004.
16. O. Lassila and R. R. Swick Resource Description Framework (RDF) Model and Syntax Specification. *W3C Recommendation*, 1999.
17. T. Gruber. What is an Ontology? `http://www-ksl.stanford.edu/kst/what-is-an-ontology.html`, last visit 2007-02-25.
18. P. V. Biron and A. Malhotra. XML Schema Part 2: Datatypes. *W3C Recommendation*, 2001.
19. Ch. Lange. SWiM – A Semantic Wiki for Mathematical Knowledge Management. *Technical Report*, December, 2006.

A Original and DRa-annotated text of our example



2. The integers and real numbers

We shall **assume** that real numbers $\mathbf{R}$ exists with all the usual properties: $(\mathbf{R}, +, \cdot)$ is a field, $(\mathbf{R}, +, \cdot, <)$ is an ordered field, $(\mathbf{R}, <)$ is a linear continuum (1.15).
What about $\mathbf{Z}_+$?

1.17. Definition.

A subset $A \subset \mathbf{R}$ is inductive if $1 \in A$ and $a \in A \Longrightarrow a + 1 \in A$

There are inductive subsets of $\mathbf{R}$, for instance $\mathbf{R}$ itself and $[1, \infty)$.

1.18. Definition.

$\mathbf{Z}_+$ is the intersection of all inductive subsets of $\mathbf{R}$.

We have that $1 \in \mathbf{Z}_+$ and $\mathbf{Z}_+ \subset [1, \infty)$ because $[1, \infty)$ is inductive so $1 = \min \mathbf{Z}_+$ is the smallest element of $\mathbf{Z}_+$.

1.19. Theorem. *(Induction Principle)*

Let J be a subset of $\mathbf{Z}_+$ such that

$$1 \in J \text{ and } \forall n \in \mathbf{Z}_+ : n \in J \Longrightarrow n + 1 \in J$$

Then $J = \mathbf{Z}_+$.

Proof.

J is inductive so J contains the smallest inductive set $\mathbf{Z}_+$. □

1.20. Theorem.

Any nonempty subset of $\mathbf{Z}_+$ has a smallest element.

Before the proof, we need a lemma.
For each $n \in \mathbf{Z}_+$, write

$$S_n = \{x \in \mathbf{Z}_+ \mid x < n\}$$

for the set of positive integers smaller than n (the *section* below n). Note that $S_1 = \emptyset$ and $S_{n+1} = S_n \cup \{n\}$.

1.21. Lemma.

For any $n \in \mathbf{Z}_+$, any nonempty subset of S_n has a smallest element.

Proof.

Let $J \subset \mathbf{Z}_+$ be the set of integers for which the lemma is true. It is enough (1.19.) to show that J is inductive. $1 \in J$ for trivial reason that there are no nonempty subsets of $S_1 = \emptyset$. Suppose that $n \in J$. Consider a nonempty subset A of S_{n+1}. If A consists of n alone, then n=min$A$ is the smallest element of A. If not, A contains integers $< n$, and then $\min(A \cap S_n)$ is the smallest element of A. Thus $n + 1 \in J$. □

Proof of Theorem 1.20.

Let $A \subset \mathbf{Z}_+$ be any nonempty subset. The intersection $A \cap S_n$ is nonempty for some n, so it has a smallest element (1.21.). This is also the smallest element of A. □

[...]

Fig. 8. Fragment of text without and with dependency graph.
The original text [7, Chapter III, §2] of presented example is taken from J.M. Moller's notes [7] regarding general topology and is reproduced on the left hand side of the figure. The right hand side of the figure shows the automatically generated dependency graph for the text where relations between parts of the text are represented by visible arrows and graph nodes have specified (but not visible) mathematical or structural rhetorical roles.