JACOBS
UNIVERSITY

Master Thesis

Mapping Mathematics with Theory Graphs
A Case Study and a Prototype

submitted by

Bastian Laubner

August 29, 2007

Jacobs University Bremen
School of Engineering and Science

Supervisor: Prof. Dr. Michael Kohlhase
Second Reviewer: Prof. Dr. Dierk Schleicher

Declaration

This thesis is submitted in partial fulfilment of the requirements for receiving a Master of Science degree in Mathematics at Jacobs University Bremen. The research contained in this work was conducted under the supervision of Prof. Michael Kohlhase at Jacobs University. All work presented in this Master Thesis is my own, unless stated otherwise.

I hereby declare that this submission is my own work and that it does not contain any material previously published or written by another person. References to and contributions from other sources are explicitly marked and acknowledged in this thesis.

Bastian Laubner
Bremen, 28^{th} August 2007

Acknowledgements

First and foremost, I would like to thank my thesis advisor, Michael Kohlhase, for the opportunity to work on this thesis. The time I have invested into this project has been full of insights and discoveries. His contagious enthusiasm and vision have guided me throughout the production of this thesis. My work on the thesis and I personally benefitted greatly from our frequent discussions.

I am grateful to the Ph.D. students in the KWARC group, especially Normen Müller, Florian Rabe, Immanuel Normann, and Christoph Lange for practical and moral support throughout the creation of this thesis, and for many fruitful discussions.

I would also like to thank the members of the Bourbaki II group, Matthias Bröcheler, Kristina Sojakova, and Bogdan Minzu, for some early insights on the mechanics of semantic annotation.

I would like to warmly thank my girlfriend Vivian Ebert and my friend Jürgen Zeitz for proofreading and making valuable comments on early drafts of the thesis.

Special thanks go to my parents. Without their love and support the creation of this thesis would not have been possible.

I am indebted to Dierk Schleicher who made it possible for me to enter the mathematics program at an exceptional point in time and who paved the way for me to write this thesis on the borderline between mathematics and computer science.

Abstract

In this paper we present a case study and a prototype that demonstrate the usefulness of theory graphs for the management of mathematical knowledge. For our case study, we develop heuristics that enable us to carry out transformations of mathematical texts into theory graphs in a standardized manner. We transform parts of algebra texts by Bourbaki and by Hungerford into theory graphs and assess the translation results using a set of criteria that we develop. Our Bourbaker prototype visualizes these theory graphs and offers advanced search functionality on them. Based on the results of the study and the capabilities of the prototype we give a promising outlook on the role of theory graphs in mathematics.

Contents

Declaration i

Acknowledgements ii

Abstract iii

Table of contents vi

1 Introduction 1
- 1.1 The theory-graphical approach . . . 2
- 1.2 The scope of this thesis . . . 3

2 Basics 5
- 2.1 The axiomatic method . . . 5
- 2.2 Theory morphisms in logic . . . 7
  - 2.2.1 Working with theory morphisms . . . 9
- 2.3 Mathematical knowledge management . . . 11
  - 2.3.1 XML . . . 12
  - 2.3.2 OMDoc . . . 13
- 2.4 Structural theory morphisms . . . 15
  - 2.4.1 Definitional theory morphisms . . . 15
  - 2.4.2 Postulated theory morphisms . . . 16
  - 2.4.3 Flattening theory graphs . . . 17
  - 2.4.4 Identification of symbols . . . 17
  - 2.4.5 Toolbox theories . . . 19
  - 2.4.6 Automatically translating natural language statements by theory morphisms . . . 20
- 2.5 The value of theory graphs in contemporary mathematics . . . 22

3 Existing Systems and Theory Graphs 24
- 3.1 Digital libraries . . . 25
- 3.2 Formal libraries . . . 29
  - 3.2.1 Mizar . . . 30
  - 3.2.2 IMPS . . . 32
- 3.3 Algebraic specification languages . . . 35
- 3.4 Interactive learning systems . . . 37

3.5 Semantic libraries . . . 38

4 The Bourbaki case study **41**
4.1 Why Bourbaki? . . . 41
4.1.1 The adequacy of digital libraries . . . 41
4.1.2 The adequacy of foundational encyclopedias . . . 42
4.1.3 The adequacy of Bourbaki Algebra I . . . 43
4.1.4 A historical account of Bourbaki . . . 43
4.2 The transformation of Bourbaki into linked theories . . . 45
4.2.1 Heuristics for the transformation . . . 46
4.2.2 Criteria for assessment . . . 49
4.3 Evaluation of Bourbaki's transformation . . . 52
4.3.1 Clarity assessment . . . 53
4.3.2 Quality assessment . . . 57
4.3.3 Compatibility assessment . . . 59
4.4 A comparison to Hungerford's "Algebra" . . . 60
4.4.1 Clarity assessment . . . 60
4.4.2 Quality assessment . . . 63
4.4.3 Compatibility Assessment . . . 65
4.4.4 Combining Bourbaki and Hungerford . . . 66
4.5 Conclusion of the case study . . . 67

5 Theory graph tools **69**
5.1 Visualization of theory graphs . . . 70
5.1.1 Principles in theory graph visualization . . . 70
5.1.2 Using the graph visualization and tweaking the graph . . . 72
5.1.3 Developing an interactive theory graph model . . . 73
5.2 Searching in the presence of theory morphisms . . . 74
5.2.1 Semantic search for mathematical formulas . . . 75
5.2.2 Using the search . . . 76
5.2.3 The tweaked search paradigm . . . 77
5.2.4 Presentation of search results . . . 78
5.2.5 Towards shallow proof assistance . . . 80
5.3 Editing theory graphs . . . 81

6 The prototype **83**
6.1 Development . . . 83
6.2 The main classes . . . 83
6.3 The memory representation of the theory graph . . . 84
6.4 The search functionality . . . 85
6.5 The theory editor . . . 87
6.6 The OrganicGraph applet . . . 87

7 Future Work **88**
7.1 Immediate steps . . . 88
7.1.1 OMBase . . . 89
7.1.2 JOMDoc . . . 89
7.1.3 OMDoc 1.8 . . . 90
7.2 Beyond the prototype . . . 90
7.2.1 Building a large theory graph . . . 91
7.2.2 The next generation of digital libraries . . . 92
7.2.3 Changing the way mathematics works . . . 93

8 Conclusion **95**

Appendix **96**

A Graph data **96**

B The classes of the Bourbaker prototype **98**

Bibliography **103**

Chapter 1

Introduction

Despite the rapid advent of information technology, the field of mathematics has profitted little from the computer as a tool. Fields have split off that are occupied with the arcane development of mechanized theorem provers or the application of algebraic and analytic algorithms in the realm of high-performance computing. The benefits for the classical fields of mathematics are still limited to typesetting programs and the fast retrieval of journal articles from online databases.

This is all the more surprising given that mathematics is an inherently structured field whose underlying organization is believed to be well understood. In the past decade, the scientific community has realized this and reacted with the establishment of the new field of mathematical knowledge management. First advancements included the creation of formats that enable the computer to make use of the semantic information that is contained in mathematical formulas and documents. The long-run aim is to offer advanced tools for managing and using mathematical knowledge to aid the scientific process.

This is the direction that this work is going into. For this, we present *theory graphs* as a framework in which to organize mathematics. Theory graphs are essentially nothing new but rather just an explication of the structure that is inherent to mathematics. Figure 1.1 shows an example of a theory graph.

At first sight, the graph expresses basic relations between theories. For example, we see that every tree is a connected graph and that in turn every connected graph is a graph. The vertices of the graph are *theories* and the edges between them are *theory morphisms*.

Theory morphisms express more complicated relations than simple subsumption of theories. They have to because otherwise Figure 1.1 would not make sense. A plane graph is a graph, but a plane graph is not a real plane. We say that theory morphisms are *qualified links* between theories.

Technically, all the theories in our example have their own collection of special symbols and an axiomatization that fixes the meaning of these symbols. For example, the theory *graph* has a set of vertices V and a set of edges E together with axioms saying that the object under consideration is a graph (e.g., $E \subseteq V \times V$). The theory of a *plane graph* has, among others, a symbol for the real plane as its base set that embeds the graph. So the link between the *real plane* and *plane graph*

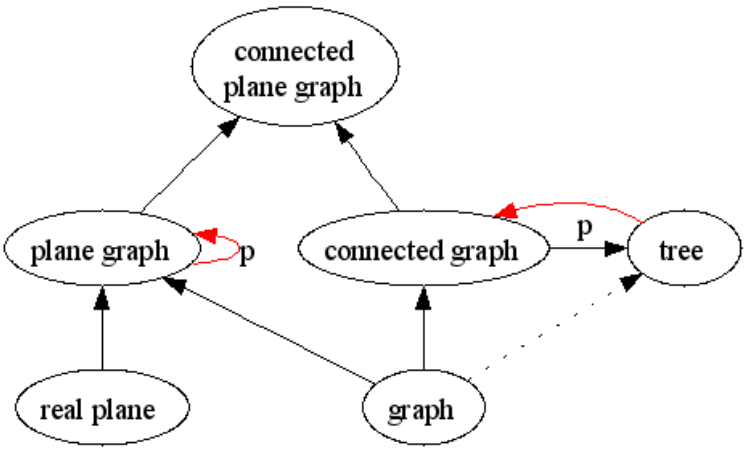

Figure 1.1: A sample theory graph

merely maps the real plane to this base set symbol.

Linking theories in this way does not only produce pretty pictures. Every statement that is true for general graphs is also true for connected plane graphs, appropriately translated using the properties of the theory morphisms in between. In our small sample graph, this connection is rather trivial, and no mathematician would consider employing heavy machinery to obtain such an insight. However, this automatic propagation of axioms, theorems, propositions, and corollaries in theory graphs opens the prospect of automated bookkeeping of these connections. Thus, theory graphs provide an effective model for a form of automated mathematical knowledge management. If the theory graph in figure 1.1 was much larger, such a tool has the power to provide us with insights about theories that we would not have thought about off the top of our heads.

1.1 The theory-graphical approach

As suggested by the sample graph in Figure 1.1, using theory graphs for structuring mathematical knowledge makes the structure of mathematical theories explicit and tangible. We can orient ourselves on such a graph, pin down points of interest and find directions that guide us in our mathematical work. This is why we speak of building a *map of mathematics*.

Yet, this is not the only property that makes theory graphs suitable for structuring mathematical knowledge. In this paper, we want to show that theory graphs are *modular*, which means that they can be dynamically extended and shrunk over time without destroying the information that the graph contains. Being modular is an important property that enables the use of theory graphs for dynamic knowledge management over time.

Furthermore, we argue that theory graphs can act as a structured *bulletin board*. Essentially, the structural hull of a theory graph can be filled with content from different sources and of diverse nature as long as we can be reasonably sure

that the meaning of the annotations are equivalent. For instance, we can express axioms and theorems in a theory both as natural language statements and in a higher order logical formalism. While the natural language statement is meant for the human reader, formal languages would be suitable for mechanized tools to process semantic information.

1.2 The scope of this thesis

In this paper we want to demonstrate that theory graphs are a really useful tool for mathematical knowledge management. We approach this problem from two angles before we tie the ends together: our case study assesses the *feasibility* of transforming standard mathematical textbook content into linked theories and our prototype implements a number of functions on theory graphs that demonstrate their aptitude for added-value services.

In the case study in Chapter 4, we compare parts of algebra texts by Bourbaki and Hungerford. For this, we translate them into a theory-graphical structure that is formalized using the *OMDoc* markup language. The translations alone take up more pages than this thesis. Then we develop heuristics that enable us to translate these texts in a fairly standardized manner. Additionally, we introduce a detailed set of criteria which we use to assess the result of the translation procedure and compare the two translations.

Starting in Chapter 5, we discuss the functions of our *Bourbaker*[1] *prototype*. The prototype handles theory graphs marked up in OMDoc. It uses the information in the OMDoc document to build a theory graph representation. Bourbaker *visualizes* theory graphs in a dynamic way and thus makes these graphs *tangible*. As an example of its management capabilities of theory graphs, it implements search functionality with which we can retrieve mathematical formulas in a semantically meaningful way. Finally, our Bourbaker suite includes a small editor that makes it possible to create theory graphs in a structured way.

Chapter 2 introduces the foundations of the concepts that we have introduced up until now. It discusses the logical and structural foundations of theory morphisms, the use of the axiomatic method, and the mathematical knowledge management that we employ in the case study and the prototype. Chapter 3 provides an overview over existing attempts to capture mathematics at large in order to offer added-value services of various kinds.

Chapter 4 contains our case study of the transformation of Bourbaki's and Hungerford's texts into theory graphs. The heuristics and criteria that we mentioned above are presented in Section 4.2 before the chapter moves on to discuss the results of the transformations.

In Chapter 5, we present the guiding ideas and principles behind the functions that are implemented within the Bourbaker suite prototype. Chapter 6 then gives details on the prototype's implementation.

[1]The phonetic resemblance to "Brubaker," a 1980 action movie with Robert Redford, is completely coincidental.

Finally, we discuss future work and visions in Chapter 7, before we conclude this paper in Chapter 8.

Chapter 2

Basics

This section introduces a few basic concepts that this thesis depends on. The Bourbaki case study in Chapter 4 is an attempt to explicate the finely granulated theory structure in Bourbaki's Algebra I. The method of finely granulating mathematical theories we call the *little theories approach*. Section 2.1 gives a general introduction to the use of the axiomatic method in mathematics and the little theories approach. The goal in constructing little theories is the construction of a theory graph whose edges represent theory morphisms. Section 2.2 introduces theory morphisms in logic along with a practical example of the usage of theory graph structure. Finally, both the case study and the Bourbaker prototype use the OMDoc markup format for the formalization of mathematics. The OMDoc format is introduced in Section 2.3. The formats implications for theory graphs are discussed in Section 2.4. Section 2.5 offers an account how making theory graphs explicit has many advantages in contemporary mathematics.

2.1 The axiomatic method

Over 2300 years ago, Euclid's *Elements* [Euc56] established the field of (Euclidean) geometry based on a small set of basic principles. His book remained the most important textbook in Western mathematics for the following 2000 years. It was not until the 19^{th} century that his basic principles were questioned and mathematicians began to ask whether other systems of axioms were worthwhile for investigation as well. The idea to found mathematics on a small set of first principles, however, remained prominent thereafter. In 1879, Gottlob Frege published his famous "Begriffsschrift" [Fre07], which laid proper foundations of axiomatic predicate logic. Alfred Whitehead and Bertrand Russell's celebrated "Principia Mathematica" [WR10], which was published in 1910, widely drew on Frege's foundational work.

The advent of the axiomatic method as an alternative to the discursive mathematics at the time, brought along an intensive research on the foundations of mathematics. Most notably, the Zermelo-Fraenkel set of axioms (ZF) emerged, which is today the most accepted axiomatization of set theory at the heart of mathematics. ZF and also ZFC (Zermelo-Fraenkel axioms + axiom of choice) are

examples of what William Farmer calls the "big-theory" approach to mathematics [Far93]. This approach aims at finding a strong set of axioms that allows to carry out mostly all of what a mathematician might want to prove. The "Principia Mathematica" had the same goal with the idea that every theorem and every proof in mathematics should use just a basic set of axioms as tools. After formalizing set theory, ordinal and cardinal numbers, and real arithmetic in three large volumes, mathematicians were mostly convinced that such an axiomatic formalization of mathematics is possible in general, but that such a project would be disproportionately tedious.

The big-theory approach had a counterpart that was not aiming at finding a single set of axioms for everything. David Hilbert's eminent "Grundlagen der Geometrie" [HB99] is an illustration of this "little-theories" approach. It presents an axiomatization of geometry that was novel at the time, grouped into five different sets of axioms. Hilbert calls these groups by the names axioms of incidence, order, congruence, parallels, and continuity. These axioms are developed in individual chapters, and at every step Hilbert shows what kinds of theorems can be derived from the axioms presented thus far. He is careful to show what axioms are used for the proof of theorems, and diverse theories such as Euclidean and non-Euclidean geometry naturally flow out of his presentation.

The big-theory and the little-theories approaches are not mutually exclusive. In fact, virtually all mathematical treatises assume some version of set theory as the underlying logical body, but their development usually proceeds by the implicit development of small theories on which investigation is focused. Theorems in one theory are then transported to other theories by theory morphisms[1], which are often handled implicitly as well. Farmer, Guttman, and Thayer observe the common usage of theory morphisms in mathematics in [FGT92] and they propose the mechanism of theory morphisms to be used in automated deduction systems. Much like in mathematics, little theories give mechanized systems a structure that limits the frame of reference to what is of immediate interest, thus reducing the size of the search space for theorem proving and proof checking. Farmer et al. propose their own mechanized reasoning system IMPS, which allows theories to be defined in higher-order logic (see Section 3.2.2).

In this work, we take the little theories approach one step further by allowing for the abstract formalization of theories in dissociation with any fixed logic. We claim that such formalization is natural in the realm of mathematics, much like mathematicians do usually not make reference to a specific logical formalism that could be used to cast their derivations in.

[1]The terms "theory interpretation" or "syntactic interpretation" are commonly used instead of "theory morphism." "Theory inclusion" is also used; we reserve this term for postulated theory morphisms.

2.2 Theory morphisms in logic

Section 2.1 has introduced the little theories approach to mathematics as the conceptual idea that we are building on. This section presents the basic definitions and properties for theory morphisms. Standard expositions on theory interpretations are for example in [End02], [Far93], and [EFT94].

By a theory T in a language $\mathcal{L}_T$ we understand a set of sentences in $\mathcal{L}_T$ which is closed under logical implication, i.e., whenever $T \models \varphi$, then $\varphi \in T$. We index $\mathcal{L}$ with T in order to emphasize that different theories will usually have different languages. We are mainly concerned with the non-logical symbols of a language $\mathcal{L}$, which we will denote by $\mathcal{L}^n$. Similarly, we denote the set of logical symbols in $\mathcal{L}$ by $\mathcal{L}^l$. This is enough to define our (somewhat restricted) notion of a theory morphism here:

Definition 2.2.1 (Theory morphism) *Let theories S and T be given with languages $\mathcal{L}_S$ and $\mathcal{L}_T$ respectively. Then a* theory translation *π from S to T is a mapping $\mathcal{L}_S \to \mathcal{L}_T$ such that:*

1. *$\pi = id_{\mathcal{L}^l_S}$ on $\mathcal{L}^l_S$;*
2. *for every m-ary relation $R(v_1, \ldots, v_m) \in \mathcal{L}^n_S$, $\pi(R)$ is a formula in $\mathcal{L}_T$ in which only $v_1, \ldots, v_m$ may occur free;*
3. *for every m-ary function $f(v_1, \ldots, v_m) \in \mathcal{L}^n_S$, $\pi(f)$ is a term in $\mathcal{L}_T$ in which only $v_1, \ldots, v_m$ may occur free.*

Let us denote by π^* the homomorphism between $\mathcal{L}^*_S \to \mathcal{L}^*_T$ induced by π. Then π^* is called a theory morphism *or* theory interpretation *from S to T if for every $\mathcal{L}_S$-formula φ we have $S \models \varphi \Rightarrow T \models \pi(\varphi)$.*

If π^* is a theory morphism, then we also call π a theory morphism or simply morphism, and we identify π with π^* when no confusion can arise. We call S the source theory *and T the* target theory *of π and write $\pi : S \to T$.*

Notice that this definition of a theory morphism is not as strong as in the classical literature. Usually, theory morphisms map universally quantified formulas to universal formulas with restrictions, so that the valid sentences in S only need to hold on a subset of the universe of a model for T. A translation τ may then do the following:

$$\tau(\forall x.\varphi(x)) = \forall x.U(x) \to \tau(\varphi(x)),$$

where $U(x)$ is an arbitrary formula with x as its only free variable. Our definition of theory morphisms is included in the general kind by setting $U(x) = \mathrm{T}$ for the vacuously true condition T. General theory morphisms allow to define a larger variety of links between theories, but we lose the property that they are homomorphisms which is important for mechanically handling statements in natural language. Consider the assertion in Quotation 1.

"For all $x \in E$ we have that $x \circ e = x$."

Quotation 1: A simple assertion from *monoid* in mathematical vernacular

When the theory *monoid* is translated to another theory, then we would like to be able to transport this statement as well in an automated manner. However, the universal quantifier over x is spelt out in natural language, and is therefore not accessible to translation. For this reason, we limit ourselves to theory morphisms that do not restrict the target universe.

For our purpose, the most interesting property of theory morphisms is the propagation of theorems along theory morphisms. By definition 2.2.1, every axiom, proposition, or theorem in the source theory is also valid in the target theory after translation. Consider the theory graph in Figure 2.1.

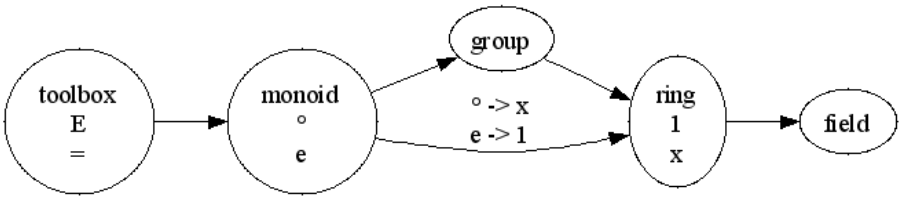

Figure 2.1: A standard example of a theory graph

The statement from Quotation 1 is one of the axioms of the *monoid*, and by the displayed theory morphisms it also holds in *group*, *ring*, and *field* in the same way. Additionally, there is a theory morphism from *monoid* into the multiplicative structure of the *ring*, for which this statement also holds. The translated statement then simply reads as in Quotation 2, and similarly this assertion holds in *field*, which imports *group*.

"For all $x \in E$ we have that $x \times 1 = x$."

Quotation 2: The translation of Quotation 1 into *group*

This particular example shows how an axiom from the theory *monoid* is translated to the valid assertions in the theories *group*, *ring*, and *field*. As it happens, the translated axiom is equivalent to one of the defining axioms in each of the target theories. Hence, in this case, the translation along the theory morphism replaces the need for defining this particular axiom a second time because it is already present via translation. The next section is going to investigate such "definitional" theory morphisms in detail. Actually, this kind of axiom re-use is abundant in mathematics. A connected graph is nothing but a graph with one more axiom specifying connectedness, an action is just a specific kind of function, and so on. Whenever we meet a definition that specifies an existing concept further, then we usually have a theory morphism from the less specific theory to the more specific one.

2.2.1 Working with theory morphisms

This section presents a practical example on how theory morphisms play out in a practical proof situation. First, let us define an important notion that has already been introduced in Chapter 1 and which appears throughout this work. Let us consider theories to be vertices in a graph, and for every theory morphism we draw a directed edge from the source theory to the target theory. We will call such a graph a *theory graph*[2]. The concept of theory graphs is analogous to that of development graphs (c.f. [AHMS02, MAH06]).

The theorem whose proof we want to sketch is Euler's famous polyhedron theorem in Quotation 3.

> "THEOREM: For any connected plane graph we have
>
> $$n - m + f = 2,$$
>
> where n is the number of vertices, m the number of edges, and f the number of faces of the graph."

Quotation 3: Euler's polyhedron formula

The proof is due to Coxeter. We only sketch it here utilizing the theories from the theory graph displayed in Figure 2.2. Euler's theorem itself is part of the theory of a *connected plane graph*.

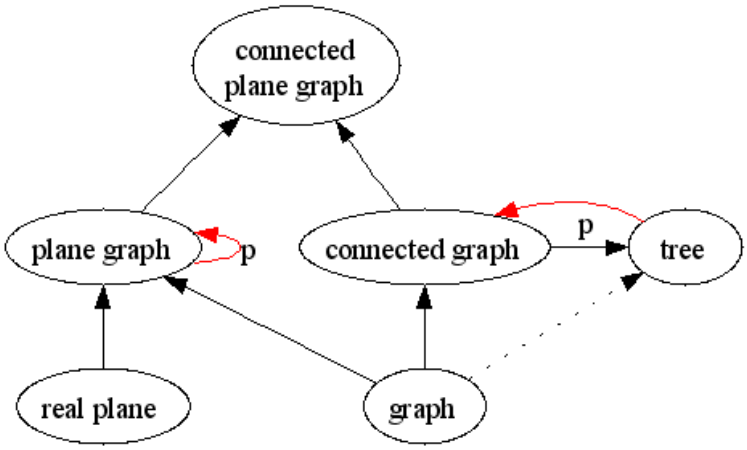

Figure 2.2: A theory graph for the proof of Euler's formula

For a given plane graph $G = (V, E)$ with nodes V and edges E, consider its dual graph $G^* = (V^*, E^*)$ which is constructed by taking as vertices the set of faces of the graph, and we connect two faces with an edge if and only if they are adjacent. The dual graph is again a plane graph, which is marked by a theory

[2]Strictly speaking, we are dealing with multigraphs as we allow for multiple edges between theories for different theory morphisms.

inclusion from *plane graph* into itself. Figure 2.3 illustrates a plane graph with its dual.

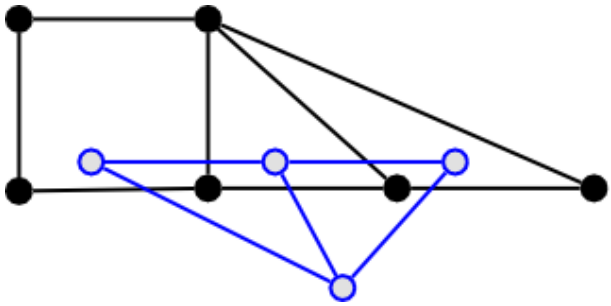

Figure 2.3: A plane graph and its dual

We define a mapping Φ from E to E^* by associating with $e \in E$ the edge in E^* between the two faces that e is bounded by. Now let $T \subset E$ be a spanning tree of G. Every connected graph has a spanning tree, which can be expressed by a morphism from *tree* into *connected graph*[3]. We claim that $T^* = \Phi(E \setminus T)$ is a spanning tree of G^*. Spanning trees of G and G^* are illustrated in Figure 2.4.

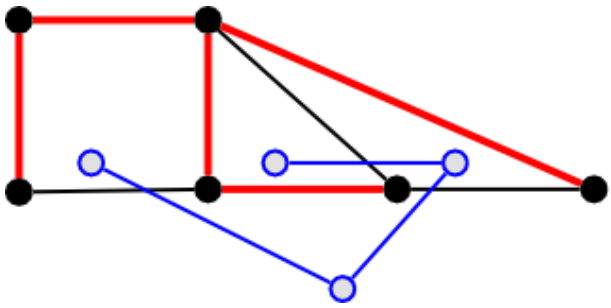

Figure 2.4: Spanning trees of the plane graph in Figure 2.3 and its dual

This part is actually hard to prove as it requires the Jordan curve theorem from $\mathbb{R}^2$. Another argument using the Jordan curve theorem shows that Φ is injective on $E \setminus T$. Finally, the number of edges in any tree is one less than the number of vertices. This is a theorem in *tree*. We obtain

$$m = |T| + |E \setminus T| = |T| + |\Phi(E \setminus T)| = n - 1 + f - 1,$$

which finishes the proof.

The proof of Euler's formula shows how theorems both about the real plane and about graphs are combined into a new theorem. It is important to stress that we carried out this proof without any limitations from the theory graph's structure. Rather, we were able to proceed with the proof in much the same way that it was originally stated. Also, we did not have to assume that all the theorems we used were available in a formal language. In fact, our proof was informal itself.

[3]this requires the notion of a subset

Nevertheless, we could use the full content of the theory graph. The theory graph is the explication of the underlying mathematical structure that mathematicians use implicitly on a daily basis.

2.3 Mathematical knowledge management

The field of mathematical knowledge management (MKM) has emerged in the last decade with the goal of formalizing mathematical knowledge in ways so that automated systems can administer it and make it readily available to the human user. The natural domain of these systems is the world wide web as it promises equal and immediate access to the provided data from anywhere in the world.

In its conception phase, MKM was mainly concerned with developing formats for the semantic representation of mathematical content that can be processed by computers. Existing formats such as the popular typesetting language LaTeX could not provide for exact semantics. Consider the following simple example:

$$\sum_{a \in A} a \sum_{b \in B} b$$

```
\begin{equation*}
\sum_{a \in A} a \sum_{b \in B} b
\end{equation*}
```

For a computer, it is impossible to decide whether the first sum scopes over the second or whether the two sums are multiplied. Depending on A, the first sum might be absolutely convergent, and then these two notions are the same by distributivity, assuming that both sums converge. However, understanding the natural language formula context is beyond the capabilities of a computer. Mathematicians tend to only disambiguate conflicts in the semantics to a human reader, which is not the same as the semantics that the machine can perceive.

Consequently, formal languages were developed that make the semantic structure of mathematical formulas explicit. The most prominent examples today are the Content-MathML format [ABC+03] by the W3C Math working group and the OpenMath standard [BCC+04] by the OpenMath society. Both these formats are based on XML, which will be introduced in Section 2.3.1. In this thesis we focus on OpenMath, which has slightly simpler syntax. One version of the above example would look as in Listing 2.1.

```
<OMOBJ xmlns="http://www.openmath.org/OpenMath" version="2.0"
    cdbase="http://www.openmath.org/cd">
    <OMA>
        <OMS cd="arith1" name="times"/>
        <OMA>
            <OMS cd="arith1" name="sum"/>
            <OMV name="A"/>
            <OMBIND>
                <OMS cd="fns1" name="lambda"/>
                <OMBVAR>
                    <OMV name="a"/>
                </OMBVAR>
                <OMV name="a"/>
            </OMBIND>
        </OMA>

        <OMA>
            <OMS cd="arith1" name="sum"/>
            <OMV name="B"/>
```

```
            <OMBIND>
                <OMS cd="fns1" name="lambda"/>
                <OMBVAR>
                    <OMV name="b"/>
                </OMBVAR>
                <OMV name="b"/>
            </OMBIND>
        </OMA>
    </OMA>
</OMOBJ>
```

Listing 2.1: An OpenMath example

The `<OMA>`-tags encode function application of the element's first child, with the remaining children as ordered arguments. `<OMV>`-tags specify variables and `<OMS>`-tags stand for symbols that are defined in separate content dictionaries. The `<OMBIND>`-tags encode the application of binders. In the above Listing, both binders are λ-binders that specify lambda function-abstraction over the variables `a` and `b` inside the `<OMBVAR>`-tags.

The semantic markup of mathematical formulas is an important first step in the automated management of mathematical knowledge. But formulas are only the most microscopical level in mathematical documents. The OMDoc format [Koh06] was developed to allow for the formalization and proper structuring of whole mathematical documents with semantic annotation. OMDoc will be more closely investigated in Section 2.3.2. OMDoc is itself XML-based and allows for the application of advanced MKM-techniques such as content re-use and the management of evolving documents. In particular, as OMDoc was developed for mathematical documents, it offers specific markup tags such as for `theories` and `definitions`, and it allows for the specification of theory morphisms inside theories. Both the Bourbaki case study in Chapter 4 and the Bourbaker suite prototype described in Chapters 5 and 6 use OMDoc as their representation format for mathematical content.

2.3.1 XML

XML is a recommendation of the World Wide Web Consortium (W3C) for general purpose markup. It is an extensible language that allows the definition of custom tags and can be used for the implementation of application languages by introducing semantic constraints. OMDoc is one such language that specifies its own tags (e.g., `theory`, `axiom`, `proof`), and so are OpenMath and MathML. XML documents are saved as standard text files on the computer's hard disk. XML tags look similar to HTML tags. Listing 2.2 shows the content from a simple XML file.

```
<?xml version="1.0" encoding="UTF-8"?>

<quiz xmlns="http://www.quiztime.org" xml:id="quizfile">
    <question>
        How large is the largest theory graph?
    </question>
    <answer>
        There is no largest theory graph.
    </answer>
    <importance val="relevant"/>
```

```
</quiz>
```

Listing 2.2: A sample XML document

The listing shows the usage of pairs of tags such as <question> and </question> that delimit the content of the question-element. All tag-pairs that are contained in an element are called the child elements of that element. For example, the quiz-element has as children question, answer, and importance. The importance-element has one so-called attribute val which has the value relevant.

XML files can be semantically constrained by using document type definitions or schemas. They can specify, for example, that a quiz-element may only contain a certain type of children. More information on XML can be found in [BPSM97].

2.3.2 OMDoc

As stated in [Koh06], "the OMDoc format is an open markup language for mathematical documents and the knowledge encapsulated in them." The current version 1.2 was chiefly developed by Michael Kohlhase. It grew out of the need for specifying the content of mathematical documents in the Ωmega group at Saarland University and it is used today by various research groups such as the Maya project at the Deutsches Forschungszentrum für künstliche Intelligenz or the Connexions project at Carnegie Mellon University. An older version of the OMDoc format is used in the ActiveMath e-learning platform, which is described in Section 3.4.

For the representation of mathematical knowledge, OMDoc defines three distinct levels of content markup [Koh06, BK07a]:

- Theory level:
  - specification of symbols owned by the parent theory
  - definitional theory imports specifying symbol mappings
  - postulated theory inclusions stipulating proof obligations
- Statement level:
  - specific tags for the statement of `axiom`s, `definition`s, `theorem`s, `proof`s, etc.
  - enables statements in several versions and several different formal languages simultaneously
- Object level:
  - statement of semantically marked up formulas in OpenMath or MathML format
  - semantic links to symbols on the theory level

```
<theory xml:id="ring">
    <symbol name="times"/>
    <symbol name="one"/>

    <imports xml:id="add.import" from="#commutative_group"/>

    <imports xml:id="mult.import" from="#monoid">
        <morphism>
            <requation>
                <OMOBJ> <OMS cd="monoid" name="operation"/> </OMOBJ>
                <OMOBJ> <OMS cd="ring" name="times"/> </OMOBJ>
            </requation>
            <requation>
                <OMOBJ> <OMS cd="monoid" name="neutral_element"/> </OMOBJ>
                <OMOBJ> <OMS cd="ring" name="one"/> </OMOBJ>
            </requation>
        </morphism>
    </imports>

    <axiom xml:id="ring.distribution">
        <CMP>
            <OMOBJ> <OMS cd="monoid" name="operation"/> </OMOBJ>
            distributes over
            <OMOBJ> <OMS cd="ring" name="times"/> </OMOBJ>
        </CMP>
    </axiom>
</theory>
```

Listing 2.3: The theory *ring* in OMDoc

The example in Listing 2.3 exemplifies these levels. It contains the somewhat minimal definition of the theory *ring*.

The theory *ring* owns two symbols: `times` and `one`. The first one of the `import`-elements specifies the structure of additive abelian *group* of the *ring*, while the second one specifies the monoid structure of the *ring*'s multiplication. For the second import, we have to translate the symbols `operation` and `neutral_element` in order to avoid identification with the same symbols in `commutative_group`. Finally, the `axiom` element shows the simple statement of mixed distributivity in a *ring* in natural language. The symbols that are concerned in this `axiom` are marked up; they refer to their home theory as content dictionaries with the attribute `cd`.

Notice that the markup of theories in OMDoc restricts theories to be finitely axiomatizable in the used logic. For example, the theory of *fields of characteristic 0* is not finitely axiomatizable in first-order logic. Also, there are logical theories for which no decidable set of axioms can be found at all. The Zermelo-Fraenkel axioms for set theory contain axiom-schemes that define a countable number of axioms, and there is no finite axiomatization. Such axiom schemes require special treatment on the side of the structural specification and structural theory morphisms that we do not discuss here. In the Bourbaki study in Chapter 4, theories without finite axiomatization do not play a role. Section 7 will suggest some possible modifications to deal with such theories as well.

2.4 Structural theory morphisms

OMDoc's capability to mark up theory morphisms is going to be a main tool for the theory graphs considered in this work. On top of the propagation of theorems along theory morphisms, we will use morphisms to simplify the definition of theories and to express theorems about connections in the theory graph. At the same time, we do not require the computer to check if the theory morphism is actually valid. Theory morphisms thus become a rather structural tool in building mathematical theories that give an outer structure to the mathematical statements they connect. The burden to keep the theory graph consistent is placed on the user.

2.4.1 Definitional theory morphisms

Definitional theory morphisms are the simplest and most commonly used theory morphisms in mathematics. Their advantage is that they make available all axiomatic definitions from a source theory, so that the mathematician only has to specify the additional axioms. Consider the sample definition in Quotation 4.

> "A topological exponential map is a universal covering map from $\mathbb{R}^2$ to $\mathbb{R}^2 \setminus \{0\}$."

Quotation 4: Definition of a topological exponential map

The Quotation makes essential reference to the concept of a "universal covering map" and no mathematician would seriously consider to fully axiomatize this concept for the sake of defining a topological exponential map. At the heart of this definition are theory morphisms from *universal covering map* and $\mathbb{R}^2$ that instantiate the *topological exponential map* as a special case of a covering map. A graphical representation of the involved theory morphisms is shown in Figure 2.5.

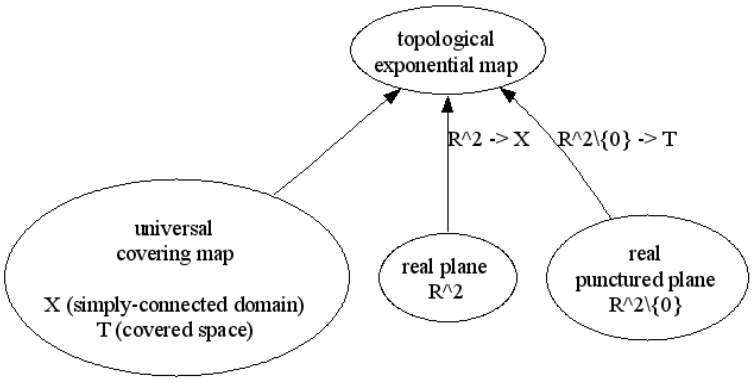

Figure 2.5: Theory morphisms in the definition of a topological exponential map

In OMDoc we can define theories in an analogous way by defining the required axioms partly or wholly by theory morphisms. Morphisms that are used to include

axioms from another theory are called *definitional theory morphisms* or *theory imports*. Listing 2.4 shows the definition of the *topological exponential map* using only theory imports.

```
<theory xml:id="topologicalExponentialMap">

    <imports from="#universalCoveringMap"/>

    <imports from="#realPlane">
        <morphism>
            <requation>
                <OMOBJ> <OMS cd="realPlane" name="ℝ²"/> </OMOBJ>
                <OMOBJ> <OMS cd="universalCoveringMap" name="X"/> </OMOBJ>
            </requation>
        </morphism>
    </imports>

    <imports from="#realPuncturedPlane">
        <morphism>
            <requation>
                <OMOBJ> <OMS cd="realPuncturedPlane" name="ℝ₀²"/> </OMOBJ>
                <OMOBJ> <OMS cd="universalCoveringMap" name="T"/> <OMOBJ>
            </requation>
        </morphism>
    </imports>
</theory>
```

Listing 2.4: OMDoc definition of *topological exponential map*

As we do not have a means of verification that theory imports actually generate sensible, let alone consistent theories, we place this burden on the author. As definitional theory morphisms transport axioms from one theory to another, we require that theory graphs must not contain any circle of theory imports.

2.4.2 Postulated theory morphisms

Theory morphisms that are not definitional are called *postulated theory morphisms*. A postulated theory inclusion π maps the source theory with axioms A_S to strict consequences of the target theory's axioms A_T. The fact that $\pi(A_S)$ strictly follows from the axioms A_T which we have specified usually has to be proven in order for the theory inclusion to exist. For every axiom $a \in A_S$, proving that $T \models \pi(a)$ is called a *proof obligation*. Unlike definitional theory morphisms, we allow postulated morphisms to create circles in the theory graph.

For example, the red arrows in Figure 2.2 indicate postulated theory inclusions. For another example, let us extend the sample theory graph from Figure 2.1. A *division ring* is a *ring* with $0 \neq 1$ in which every element is invertible under multiplication. The corresponding theory graph is shown in Figure 2.6.

Figure 2.6: An extension of our standard example

The centralizer of a *division ring* is the set of elements that commute with every other ring element under multiplication. Now the centralizer is a field together with the induced addition and multiplication operations, which means that we can postulate a theory morphism from *field* to the centralizer of *division ring*. The figure shows this postulated theory inclusion as a red edge. However, the centralizer is not uniquely specified by the property of being a subfield, so the translated *field* axioms do not replace any of the *division ring*'s axioms under this theory inclusion.

In practice, postulated theory morphisms often occur as theorems in mathematics. The fact that the centralizer of a division ring forms a field would be such a theorem, that every connected graph has a spanning tree as a subgraph would be another. Postulated morphisms are particularly interesting when they connect far apart theories. Examples of such theory morphisms include Curry-Howard-Isomorphisms or Stone's representation theorem.[1] EdNote(1)

2.4.3 Flattening theory graphs

In OMDoc, we use theory morphisms explicitly to transport `axiom`s and `assertion`s through the theory graph. Thus, a theory does not only consist of the symbols, axioms, and assertions that are immediately stated as children to the `theory`, but also of those that are included in the theory by recursive imports.

When we want to make explicit all the statements that virtually belong to a theory, then we speak of the *flattened theory graph*. Likewise, we speak of *flattened theories* if we want to specifically all symbols and statements that are recursively included in the theory.

As postulated theory inclusions may create circles, the flattening procedure introduces possibly countable many statements inside a theory. Therefore, the flattened graph is never fully created, but only up to a certain depth from every node. This will play a role in searching the theory graph, which will be discussed in Section 5.2.

2.4.4 Identification of symbols

As we are using theory morphisms to import symbols and axioms, we have to specify how to handle multiple imports. We allow imports that simply import symbols as they are stated in the flattened source theory without any translation. This makes the creation of theory graphs lean because we do not have to restate all symbols every time. However, this can also be a source of confusion. Let us consider the two examples in Figure 2.7.

The graph on the left hand side shows the definition of a monoid. There is really nothing more to it, because a monoid is nothing but an associative unital magma. The theories *unital magma* and *associative magma* are themselves defined by imports and additional axioms and symbols (in the case of *unital magma*). Therefore, these two theories share the symbols from the flattened theory *magma*, which would typically be a binary operation $\circ$ and a base set S. When we now

[1] EdNote: more examples...no.✓

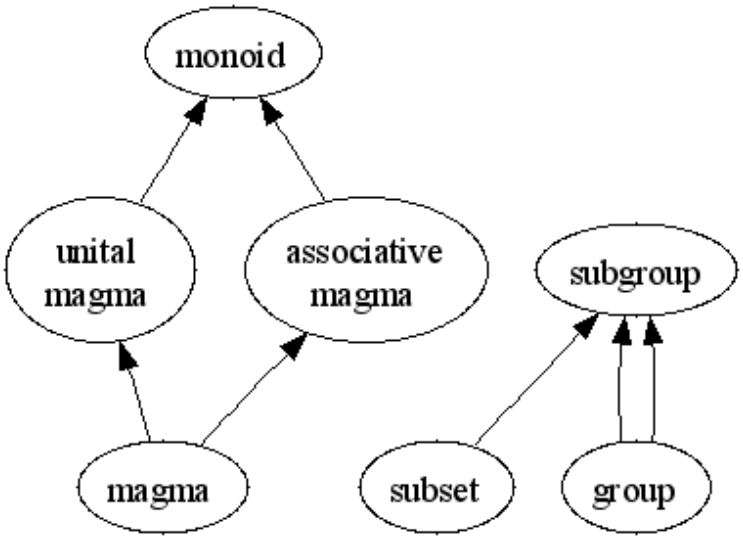

Figure 2.7: Two sample graphs with symbol identification

import *unital magma* and *associative magma* into *monoid*, we implicitly *identify these symbols* again. That means, instead of owning two instances $\circ_{um}$ and $\circ_{am}$ of $\circ$, these symbols are unified into one symbol which is still addressed as the $\circ$ symbol owned by *magma*.

Let s and t be two symbols that are imported without translation into a theory T by theory morphisms $i : S_1 \to T$ and $j : S_2 \to T$, respectively. Then we identify s and t if they are in the intersection of the flattened S_1 and S_2. This is the case if and only if $s = t$ is owned by a theory U and there are morphism paths from U to S_1 and from U to S_2 that both never map $s = t$ to any other expression. Thus, if we want to make sure that a symbol that we are importing is not identified with any other symbol, then we simply have to redefine the symbol in our new theory and use an explicit symbol map.

Obviously, we use identification for reasons of simplicity. We could also postulate that clashes like this have to be resolved manually by designing proper symbol mappings. In that case, we would have to define symbols $\circ_m$ and S_m in *monoid* and map the operation and base set from *unital magma* and *associative magma* to these symbols, respectively. As this is tedious, we like to spare the user this obligation.

This simplification in usage comes at the expense of situations where we have to be careful not to blindly identify symbols. The theory graph snippet on the right hand side in Figure 2.7 shows the structure of the definition of the theory *subgroup*. The theory imports the theory *subset* which contains two symbols A and B as sets and an axiom saying that $A \subset B$. Then, what is left to do is importing the theory *group* twice, once for A and once for B. Obviously, we do not want the base sets of these groups to be identified, so we map them to A and B respectively. We would not mind the identity elements from the two group imports to be identified, as the group's and the subgroup's identity elements are identical.

Now we might think the same thing for the binary operation on the group

because it is doing the same thing. But we have to be careful: if we identify the two binary operations then we also implicitly identify their domains that we have just explicitly separated. This does not make any sense, hence we have to define at least one additional symbol (e.g., denoting the "induced law" on A) that we can map to in order to keep the operations apart.

Our symbol identification approach simplifies many definitions by implicitly unifying symbols that are meant to be the same. However, this does not relieve the user from deciding when symbols may be identified and when this would be ill-defined. In general, it is always safer to redefine symbols.

The issue of symbol identification is one example that shows that studying structural morphisms is interesting in its own right and in separation from their counterparts in logic. Florian Rabe, Michael Kohlhase, and Normen Müller are currently developing a decidable typing calculus describing well-formed theory graphs [RKM07]. However, their system does not allow for symbol identification. We do not use an explicit notion of well-formed theory graphs in our case study and the Bourbaker prototype, but the integration of this notion into our approach will be important in the future development.

2.4.5 Toolbox theories

When building a theory graph, it is often desirable not to start it from scratch but to postulate a number of known concepts beforehand. We need to do that ourselves in the case study in Chapter 4 because the algebra texts we translate refer to notions of set theory that we did not formally formalize.

Symbols and axioms that we need as a basis can then be arranged in what we call a *toolbox theory* that underlies the graph. The toolbox theories in our case study typically include the definitions of symbols for sets, set inclusion, equality, etc.

In fact, we do not even want to fully specify the toolbox theory as this would be just as tedious as building the underlying theory graph. Instead, we leave it open to fill the place of the toolbox with a suitable theory graph that gives meaning to the symbols defined in there. Figure 2.8 illustrates this principle.

In the figure, we underlay the graph with the Zermelo-Fraenkel axioms of set theory and the axiom of choice (ZFC). The toolbox theory acts as an interface here. The theory graph above the toolbox uses the symbols defined in the toolbox, while the imports from the ZFC graph give meaning to those symbols.

We can leave the question of what theory graph to use underneath the toolbox deliberately open. We can fit both shallow definitions of concepts there or large and fine-grained theory graphs that include a lot of statements themselves. In this way, we do not have to decide for a specific foundation of our work before designing the theory graph, but we can just underlay it with whatever axiom system makes sense for us later. If the toolbox theory is carefully designed, we may even fit theory graphs of very different nature in that place.

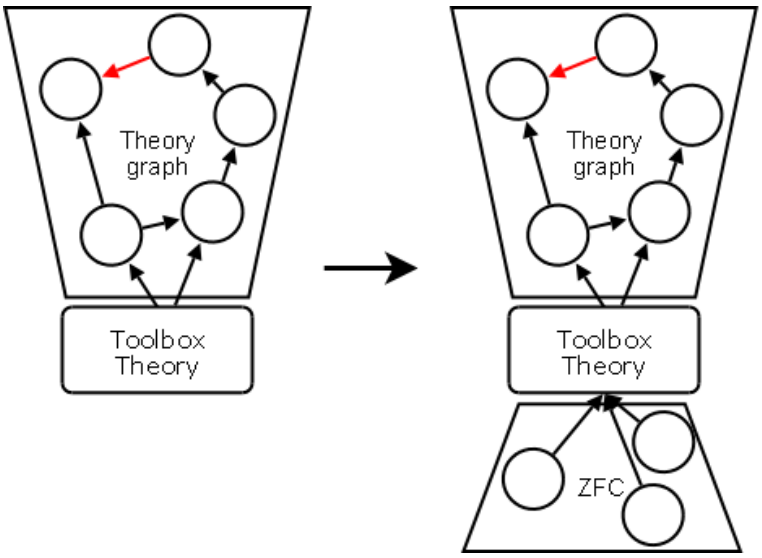

Figure 2.8: The toolbox theory as an interface

2.4.6 Automatically translating natural language statements by theory morphisms

As mentioned before, we use the semantic specifications of theory morphisms to automatically translate marked up formulas in natural language axioms and assertions. Doing this is not safe. There is no guarantee for us that the translated statements will make sense in the target theory because only marked up formulas are translated to the new theories, but not the natural language context. This is still problematic after we have eliminated the theory morphisms that restrict of the universe over universal quantifiers. Consider the example in Quotation 5.

> "Let M be a monoid with operation $\circ$. If $e \in M$ is the identity element, then $a \circ e = a$ for all elements a in the monoid."

Quotation 5: An assertion that is problematic to translate

This assertion does a few bad things that make it unfit for automatic translation. Consider the automatically translated version of the statement to the multiplicative structure in the real numbers (see Quotation 6).

> "Let $\mathbb{R} \setminus \{0\}$ be a monoid with operation $\cdot$. If $1 \in \mathbb{R} \setminus \{0\}$ is the identity element, then $a \cdot 1 = a$ for all elements a in the monoid."

Quotation 6: The automatic translation of Quotation 5

Clearly, the references to *monoid* are out of place because the statement now refers to the multiplicative structure of the reals. Furthermore, the reference to

"the" identity element is ambiguous now because there are two such identity elements in the reals, 0 and 1. Finally, we would probably like to replace the reference to "elements" by "numbers."

Generally, very confusing or downright false examples may be constructed by automatically translating statements that intersperse natural language and mathematical formulas. We do it anyways for two reasons. Firstly, as long as the natural language around the mathematical elements is just a spruced up version of the corresponding logical formula, we can translate the statement without loss of meaning. Especially when a great portion of the statement is semantically marked up it is usually close to a logical formula. If, in addition, we avoid text references to theories and their symbols, we essentially steer clear of most pitfalls. Inside the theory graph, the above assertion would be stated in the theory *monoid* which already provides the symbols $\circ$, e, and M. Quotation **??** shows the version of Quotation 5 that is appropriate for the theory graph.

"$a \circ e = a$ for all $a \in M$."

Quotation 7: The statement of Quotation 5 in the theory graph

There is clearly no problem in automatically translating that version of the assertion.

Secondly, we would like to perform semantic search on the theory graph, but displaying translated formulas in the flattened theory graph is not telling if these formulas are cut out of their context. Whether we want it or not, we therefore have to show the translated formulas of the statement in their static language context.

In this work, we solve the problem of ambiguities by giving the user a detailed account of the morphism path of the statement so that she can investigate in detail whether the proposed translation makes sense. For the future, it would be good to develop a framework for intermingling formulas and text that also marks up natural language terms and poses constraints that guarantee that automatic translation is possible.

Notice that we only translate `assertions` and `axioms`, but not `definitions`. In logic, definitions are merely a certain kinds of axioms that are guaranteed not to change the content of the theory. As such, there is no problem in translating these formulas via theory morphisms to other theories. That said, we would like to be able to write down statements such as "we call e the identity element of the monoid" somewhere without these statements propagating through the theory graph. Choosing definitions for this purpose might not be the only way of doing this. It does force us to properly formalize theories through axioms, though, which is desirable for the theories to become well specified.

2.5 The value of theory graphs in contemporary mathematics

Theory morphisms and our concept of theory graphs are nothing new to mathematics. Formal logic is studying theory interpretations explicitly since the 1950s while their concept has been in use in mathematics for much longer [FGT93], even though implicitly.

In this section we want to show that making the theory graph structure explicit in mathematics has many advantages for many branches of mathematics. For this, let us dwell on the mathematical sciences for a while, for which our theory graph structure is made.

Mathematics is organized in many different fields and branches. It is unlikely today that any single person has a good overview of the trends and progress in all of these fields, even if we restrict ourselves to the core fields of mathematics. Unfortunately, this makes it very likely that a lot of work is done multiple times just because the involved people had no knowledge of the other's field. Therefore, a central resource that allows the quick browsing of results in mathematical fields is very desirable. Such quick browsing can only work if it is semantically annotated, which is one of the features of our approach.

Generally, such a resource would leverage a lot of productivity as it allows to quickly cross-reference mathematical fields that otherwise would only be accessible by monitoring a large number of specialized mathematical journals. The same advantages exist for the conservations of results in mathematics. Most modern mathematics students would shudder at the perspective of excavating old 1920s Russian journals from their library's shelves. Integrated into a theory graph, these results would remain accessible, organized and classified by the structure of the theories they make statements about.

In some cases, mathematical fields are governed by programmatical work. This is usually targeted at the solution of some important problem, such as the Riemann hypothesis. An excellent example of programmatic work is the classification of the finite simple groups in group theory. The attack on the problem started in the 1950s; the classification theorem was declared proven in the 1980s (c.f. [Sol95, Asc04]). However, the proof was extremely intricate, filling tens of thousands of pages, many of which had not even been published. Until the early 1990s, gaps had to be filled in, and several groups of mathematicians still today are concerned with revamping the classification in order to make it accessible at all.

We claim that huge programs like the classification could profit from the formalization into theory graphs. This starts with the structure that is given to large proofs of several hundred pages. At the very least, a theory graph assures the mathematician that all the dots are connected. On top of that, theory graphs give mathematical projects a tangible outline that is nowhere hinted at either because the structure is too intricate for communicating intuitions about it, or because the plan of proof was not clear in the beginning. A theory graph gives a real-time account of the program's progress and allows for conceptual elements in the form

of conjectured theories and morphisms.

The automated reasoning community is dreaming of a time where mathematicians stop publishing in journals and cast their proofs in formal languages instead, which can then be verified by mechanized procedures. This dream is impressively described in the QED Manifesto from 1995 [QED95]. It has not become reality, though, and it is not conceivable that it will in the near future. While the simple structuring of mathematics does not enable mechanized reasoning by itself, a central and up-to-date theory graph would give a realistic model of mathematics. In our approach, such theory graphs can be partly annotated with formal languages, which allows the automated verification of parts of the graph. Verified proofs and morphisms then show publicly as trustworthy items which gives the application of automated systems a reputation outside the respective journals.

Recently, Grigori Perelman was nominated for the Fields medal for his proof of the long unresolved Poincaré conjecture. He was nominated for the prize for publishing the first manuscript that outlined the proof in detail. His proof was plastered with holes which were filled in by an independent group of mathematicians soon after his publication. However, it was not this group but Perelman who was honored. It seems that in large parts of mathematics, especially in cutting-edge research, devising the right concepts is considered more valuable than publishing complete proofs, let alone formal proofs. OMDoc theory graphs allow for the formalization of mathematics without requiring complete proofs, while at the same time supplying the tools for mapping out conceptual ideas. The authors of the QED Manifesto wanted to change the perception of what constitutes good mathematics. Our theory-graphical method, however, seems to model already what is considered good mathematics today.

Chapter 3

Existing Systems and Theory Graphs

The past 30 years have seen many efforts to use the power of computers for aiding mathematicians. At the time, mathematicians began building up libraries of mathematical content in formal languages that were suitable for machine processing. This section will give an overview over those systems that use structures reminiscent to theory graphs and will compare their approach to the ideas behind Bourbaker.

The first implementations were geared towards automated theorem proving.[2] EdNote(2) Mathematicians and computer scientists alike had hoped that the power of formal logical systems would convince mathematicians to cast their theorems and proofs in such languages, so that their validity could be tested by proof checkers. Much to the community's dismay, not a single such system has fundamentally changed the way mathematicians are working today. The discussion of formal mathematical libraries for automated proof checking will be picked up in Section 3.2, which will also present two such libraries that implicitly use the concept of theory graphs and compare them to the Bourbaker approach: IMPS and Mizar.

With the advent of the internet, a different form of mathematical library emerged, which Jeremy Gow and Paul Cairns call "digital libraries." [GC07] The most prominent protagonists of this type of mathematical collection are MathWorld[1] and PlanetMath[2]. Digital math libraries are collections of articles written in HTML which are connected through hyperlinks so that the user can suit her needs by jumping from article to article in a non-linear fashion. Such linked encyclopedias are not in any way optimized for processing by machines. Digital libraries will be scrutinized in greater detail in Section 3.1.

Digital libraries are focused on usability for humans, thus their creators often claim that they are suitable for mathematical self-study. However, there are systems that make more explicit use of connections between theories in order to convey mathematical knowledge to students. One such system is ActiveMath,

[2] EdNote: put examples of such systems

[1] http://mathworld.wolfram.com/

[2] http://planetmath.org/

which will be discussed in Section 3.4.

As discussed in Section 2.3, the field of mathematical standardization has diversified and reorganized itself considerably into what is now known as the field of mathematical knowledge management. Considerable efforts are made to integrate techniques from machine-oriented approaches to formalizing mathematics with more human-oriented tools, so that automation can be helpful to a broader variety of mathematicians and mathematical experts. OMDoc as a general mathematical markup format can be understood in this way. I consider the Bourbaker project itself a contribution to the integration of formal and informal methods. The common idea to most approaches is the capturing of semantic connections between informal mathematical statements and their formal counterparts. Section 3.5 provides a survey of efforts to build semantic libraries and it discusses Christoph Lange's semantic wiki SWiM as an example in this field.

3.1 Digital libraries

PlanetMath and MathWorld were already mentioned in the introduction to this chapter as examples of digital libraries. PlanetMath is a wiki-style online encyclopedia for mathematics whose articles are contributed on a voluntary basis by its community. Figure 3.1 shows a sample page from PlanetMath which states Mantel's theorem from graph theory. Articles on PlanetMath are written using LaTeX math environments that enable mathematicians to contribute to the platform in much the same way they would write journal articles. By standard, mathematical formulas are rendered as images in "png"-format but the LaTeX sources can also be displayed, which allows for shallow semantical processing of presentational mathematical content. During the authoring process, links to other PlanetMath articles can be inserted, and the system automatically links certain words to their defining articles. The example in Figure 3.1 contains links to the articles "graph," "graph order," "graph size," "set containment," "triangle," and "graph cycle." In this way, the user who later browses the article can directly jump to articles that she is not familiar with, and she can repeat this process to explore all concepts that are used in a given article.

This concept of linking makes PlanetMath very similar to the online encyclopedia Wikipedia[3], which is also based on contributions from the voluntary user community. Even though Wikipedia's focus is not on mathematical concepts, it does incorporate a fair amount of articles on mathematical theories whose quality is comparable to PlanetMath. In fact, Wikipedia and PlanetMath are both using the GNU Free Documentation License, allowing them to set up a project to integrate articles from PlanetMath into Wikipedia.

MathWorld is the oldest of the online mathematics libraries. A sample page on resultants is shown in Figure 3.2. The linked concepts include "polynomial," "determinant," and "Sylvester matrix." MathWorld's inventor Eric Weisstein released its first online version in 1995 under the name "Eric's Treasure Trove of Mathe-

[3]http://wikipedia.org/

Figure 3.1: Mantel's theorem on PlanetMath. Source: http://planetmath.org/encyclopedia/MantelsTheorem.html

matics." In 1999, the site moved under the auspices of Wolfram Research, Inc., and was renamed to its present name. Unlike with PlanetMath, articles contributed to MathWorld are reviewed without exception by the company before adding them to the website. While MathWorld is freely accessible to users on the internet, all its articles are subject to comprehensive copyright regulations. Aside its provision of encyclopedic mathematical articles, MathWorld serves as a showcase for Wolfram Research's Mathematica software with an emphasis on Mathematica functionality inside articles and frequent links to demonstration kits for the software that relate to the displayed content. Such a reference can be seen at the bottom of the excerpt displayed in Figure 3.2. The web pages themselves are generated using Mathematica, and mathematical formulas are rendered as images in "gif"-format, which does not allow for semantic analysis, and other sources are unavailable due to the mentioned copyright restrictions.

Wikipedia, PlanetMath, and MathWorld all have in common the principle of publishing articles on certain topics and linking to related articles inside the document. On the surface, this is reminiscent of theory graphs when we consider articles as theories and links between articles as theory inclusions. There are two problems with this approach that underscore how such digital libraries are less expressive than theory graphs. One problem is the blind identification of the libraries' articles with theories, the other problem is the lack of qualification of the

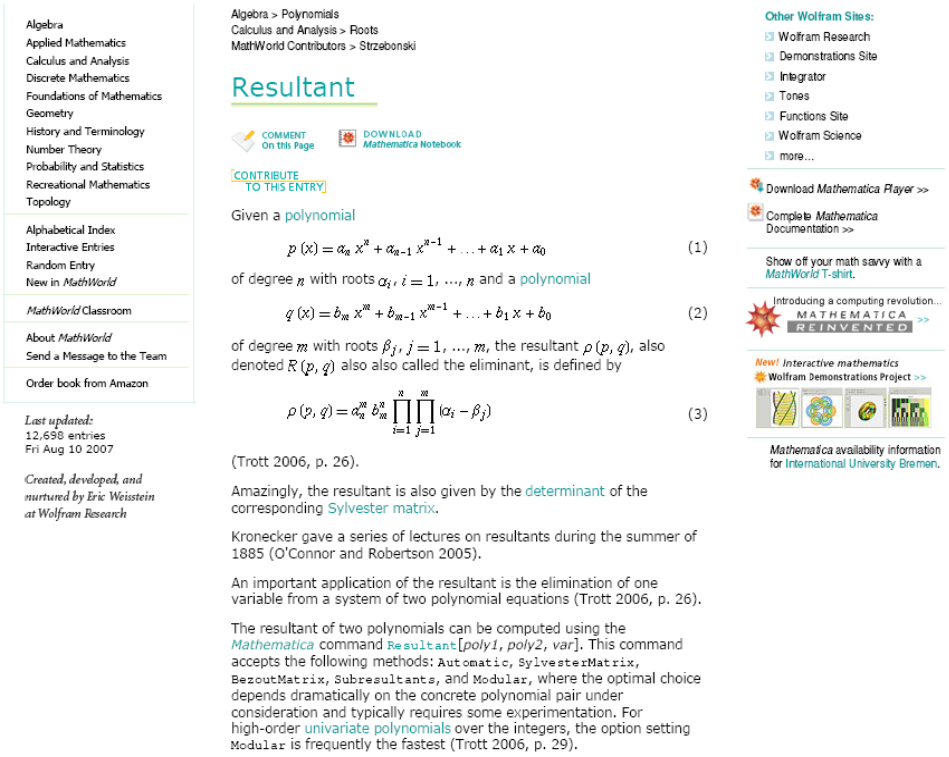

Figure 3.2: Excerpt from Resultant on MathWorld. Source: http://mathworld.wolfram.com/Resultant.html

links.

Firstly, the articles in the libraries are not confined to what one could consider theories, but the libraries give equal rank to articles explaining theorems, proofs, methods, intuitions, formulas, and often side notes. For example, all three digital libraries contain articles on the Argand diagram, which is a name for the method of rendering the complex numbers $\mathbb{C}$ as points in the real plane. Quotation 8 shows the short text from the PlanetMath article.

> "An Argand diagram is the graphical representation of complex numbers written in polar coordinates. Argand is the name of Jean-Robert Argand, the Frenchman who is credited with the geometric interpretation of the complex numbers [Biography]"

Quotation 8: Definition of the Argand diagram on PlanetMath. Source: http://planetmath.org/encyclopedia/ArgandDiagram.html

An Argand diagram does not constitute a theory in our understanding as it merely describes how to plot the members of the set of complex number in a certain way.[3] The content of this article in a theory graph would be an import[4] EdNote(3)

[3] EdNote: but thinking about it, one could construct a theory out of it if one really tried hard...
[4] Depending on how one chooses to define $\mathbb{C}$, this import is definitional or postulated.

from the theory of $\mathbb{R} \times \mathbb{R}$ into the theory of $\mathbb{C}$. Like with any other graphical representation of sets, it is hard to conceive how to axiomatize the concept of an Argand diagram. Thus the articles are more of an expository nature that aims at making the reader familiar with a certain intuition about complex numbers. The articles in all three digital libraries are also somewhat historical when they point to the role of the Argand diagram as an important step towards the acceptance of complex numbers in 19^{th} century mathematics. The PlanetMath article also links to Jean-Robert Argand's biography, but the link is formally separated from the links to "represenation," "complex numbers," and "polar coordinates."

Secondly, the links inside the libraries are not qualified. Wikipedia is the most critical library in this respect. Its articles do not only link to other mathematical articles, but to biographies, trivia, and historical side notes, without any means of distinguishing mathematical links from others. Certainly, the linked structure exists in principle, but any attempt to offer services on this structure is bound to be cluttered with a large amount of irrelevant side information which cannot be differentiated from relevant articles.

PlanetMath and MathWorld make life easier by more properly separating mathematical links and links to side information. However, even in these libraries we find articles on, for example, the use of mathematical language. PlanetMath has an article on the meaning of the word "obvious" in mathematical arguments, and similarly MathWorld describes the meaning of the word "trivial." Both libraries frequently link to such side information from mathematical articles.

Even if we assume for a moment that digital libraries were only linking mathematical articles, the links would still be unqualified. Thus links from one article to another cannot be considered real theory inclusions. Such article links can refer to any concept that might be related in any way to the given article. For example, the MathWorld article on *rings* links to *commutative rings* and *fields* as forward links to related concepts, even though these theories are based on *ring*, and not the other way around. Similarly, examples often introduce links to concepts that are not fundamental to the theory at hand.

Unqualified links make it impossible to translate statements between theories. Notice that even if we attempted to do so, a computer would not be able to infer the semantics of symbols that all these platforms render in presentational formats, or even worse, as pure images. In this way, these libraries make it impossible to extract implicit information that is buried in the depth of their linked structures. Theory graphs, however, qualify the meaning of links between theories exactly, so that every theorem is dispersed over the graph structure automatically. At the same time, the OMDoc format allows annotating theories with side information and expositional context information in much the same way as the digital libraries do. This line of thought will be picked up in Section 7 to describe how an extremely powerful digital library could be built upon a theory graph structure.

3.2 Formal libraries

Formal libraries are collections of mathematical theories, definitions, theorems, and proofs in a formal language that make the libraries suitable for machine processing. In such libraries proofs are checked for their validity and often times automated theorem provers would try to find new theorems independently, based on the supplied corpus.

In this section, we will investigate two systems in particular, William M. Farmer's IMPS[5] system and Mizar[6], which was created in the 1970s by Andrzej Trybulec. While Mizar is the oldest such system around, IMPS is fairly new and boasts explicit theory morphisms as a fundamental building principle of its library. In between these two systems, we omit the treatment of a plethora of other libraries and systems that would all be interesting in their own right. Freek Wiedijk gives an overview and a practical survey in [Wie03] of Mizar, IMPS, and 13 other systems that all handle mathematical libraries in an automated way. His survey includes names such as Stanford Research Institute's PVS[7], the Coq[8] system by the French "Institute national de recherche en informatique et en automatique," Isabelle[9] by the University of Cambridge and TU München, and Jörg Siekmann's Ωmega project which is affiliated with "Deutsches Forschungszentrum für künstliche Intelligenz," just to name a few.

Some of these systems are declarative, which means that a proof consists of listing the atomic steps of what is to be proven, and some of them are procedural, where the emphasis is on methods where one writes down how to prove a statement. John Harrison gives an overview over the different proof styles in computer aided proof systems in [Har96]. What is important to note is that declarative systems usually offer to declare variables, much like a mathematician would start a theorem or a proof with "let E be a set." These kinds of declarations establish links to other theories, which is reminiscent of theory morphisms. However, in order to make use of the exact relationship of source and target theories in such an approach, an automated system would have to extract the exact semantic meaning of such declarations. Such meanings can reach from local variables that are declared for a minor part of a proof to declaring constitutive elements inside definitions. As most of these languages lack the explicit notion of a theory, it is not conceivable to achieve such a structuring in an automated way while at the same time determining symbols that are constitutive for the signature of a theory and the appropriate imports.

[5] http://imps.mcmaster.ca/
[6] http://mizar.org/
[7] http://pvs.csl.sri.com
[8] http://coq.inria.fr/
[9] http://isabelle.in.tum.de/

3.2.1 Mizar

Let us now focus on Mizar. Mizar is the oldest formal library and proof system that is still in use, which is a quite impressive fact given the number of alternative and innovative new systems that have emerged in the past three decades. The Mizar project has accumulated the largest library out of all the formal libraries; according to their website[10] it contains over 7000 definitions and 40000 theorems. There is an active community that regularly contributes new Mizar articles. Mizar is one of the five systems listed by Freek Wiedijk in [Wie07] which he considers to come closest to the ideal of the QED Manifesto. One important reason why Mizar might be so popular might be its formal language, which is modeled after the use of natural language in mathematics. An example of a Mizar theorem is shown in Listing 3.1. The theorem comes from the Mizar article "monoid" by Grzegorz Bancerek.

```
theorem Th18:
 M is well-unital iff for a being Element of M holds
    (1.M)*a = a & a*(1.M) = a
   proof
A1:    M is well-unital iff un(M) is_a_unity_wrt op(M) by Def21;
    thus M is well-unital implies
     for a being Element of M holds un(M)*a = a & a*un(M) = a
       by A1,BINOP_1:11;
    assume
A2:  for a being Element of M holds un(M)*a = a & a*un(M) = a;
        now let a be Element of M;
          un(M)*a = a & a*un(M) = a by A2;
      hence op(M).(un(M),a) = a & op(M).(a,un(M)) = a;
     end;
    hence thesis by A1,BINOP_1:11;
  end;
```

Listing 3.1: A Mizar theorem

For a mathematician, the theorem and its proof are fairly legible, even though they are written in Mizar's formal language, which can be parsed by computers. What is interesting in light of theory graphs, however, is not the syntax of the language but the so-called environ-tag at the beginning of Mizar articles. The environ-environment declares what other theories, theorems and definitions are used in the present Mizar articles. Listing 3.2 shows an example from Andrzej Trybulec's article on enumerated sets[11].

```
environ

 vocabularies TARSKI, BOOLE;
 notations TARSKI, XBOOLE_0;
 constructors TARSKI, XBOOLE_0;
 registrations XBOOLE_0;
 definitions TARSKI, XBOOLE_0;
 theorems TARSKI, XBOOLE_0, XBOOLE_1;
 schemes XBOOLE_0;
```

Listing 3.2: Mizar environ-declaration

[10] http://mizar.org/project/

[11] http://merak.pb.bialystok.pl/mizar/mml/enumset1.miz

The environ declares to use the vocabularies, notations, etc., from the articles "TARSKI" and "BOOLE." In particular, it states the use of definitions and theorems from these articles. Now this is in direct correspondence to the theory graphs we use. In fact, a simple `imports`-declaration in OMDoc is the same as prompting the use of all symbols, definitions, axioms, and theorems from the source theory also in the target theory. Therefore, there is a rudimentary theory graph between Mizar articles. Figure 3.3 shows a full overview of the Mizar library's graph which is courtesy of Freek Wiedijk[12].

Figure 3.3: The dependency graph of Mizar theories

The most obvious shortcoming of Mizar's version of theory imports is that they are static. Morphisms that map symbols in the source signature to other symbols or even formulas in the target signature are not possible. Mizar's theory imports are meant purely as a means for including the content from other articles, so as if these articles were inserted in place of the environ-tag. This is very similar to the way programming languages like Java handle the imports of source code libraries in order to allow new programs to use already existing content. Given this function inside the Mizar language, theory imports or, more correctly, article imports are not given any meaning inside the proof system. Not only would no Mizar author

[12]the Mizar dependency graph can be obtained from: http://www.cs.ru.nl/ freek/mizar

try to express the relation between *group* and *monoid* using Mizar imports, it is downright impossible, and the Mizar system is not designed to exploit these kinds of relations. Mizar articles are also not fine-grained at all, rather, they group an often large number of definitions and theorems that relate to a certain area of mathematics, much like an introductory article in a textbook would do. Yet, for the theory-graphical method to be meaningful, fine granularity of theories is an important prerequisite. This thread will be picked up in the Bourbaki case study in Section 4.2.1.

Furthermore, the Mizar system intrinsically depends on the use of Tarski-Grothendieck set theory. While virtually all formal libraries depend on one specific logic for formalization, this is still a shortcoming that makes it impossible for these systems to capture all of mathematics. The inability of the community to either decide for one specific logic or to develop a way of dealing with multiple logics may be considered one of the reasons why the current efforts fall so very short of the QED Manifest. The theory graph model that is proposed in this work is independent of any specific kind of logic and leaves the use of specific reasoning frameworks up to the content in `FMP`-nodes of the underlying OMDoc documents.

Nonetheless is the Mizar dependency graph an interesting special case of a theory graph. Grzegorz Bancerek and Michael Kohlhase have outlined in [BK07b] a way to translate Mizar articles into OMDoc format automatically. If this translation was carried out on a large fragment of the Mizar library, this would be an excellent test case for the search engine and the visualization routines of the Bourbaker suite which are described in Chapter 5.

3.2.2 IMPS

The main people behind the IMPS system are William M. Farmer, Joshua D. Guttman, and Javier Thayer and the system is sponsored by the American Mitre Research Corporation. The IMPS system was conceived in 1995; it is now succeeded by the MathScheme system by William M. Farmer that aims at joining principles from symbolic computer algebra systems with the IMPS approach. Even though IMPS does not have a large digital library to work on, it is a very interesting system because it makes heavy use of theory morphisms. In fact, the term "little theories" was coined by William M. Farmer. The implementation of IMPS certainly comes closest to what the Bourbaker project is envisioning. Consider the definition of the theory *groups* in Listing 3.3[13]. This is one of two definitions of *groups* in the IMPS library, the other one of which does not use the theory-import from monoid and instead defines all of its axioms independently.

```
(def-language groups
    (embedded-languages monoid-language)
    (constants (inv (uu uu ))))

(def-theory groups
    (component-theories monoid-theory)
    (language groups)
```

[13]the example can be obtained from: http://imps.mcmaster.ca/theories/normed-spaces/normed-groups.html

```
  (axioms
    (right-inverse-property "forall(x:uu,x**inv(x)=e)" rewrite)))
```

Listing 3.3: IMPS definition of group using theory morphism

The first definition construct "def-language" in Listing 3.3 clearly states that *groups* is supposed to use the same symbols as *monoid*, plus a binary predicate "inv" (where "uu" denotes the base set of *monoid*). The second construct "def-theory" then states that the theory imports *monoid* as part of the definition of *groups*, and it adds one axiom to account for the existence of inverses using the "inv" predicate. But IMPS does not only support simple theory imports, but it also handles theory inclusions in an advanced way. Consider Listing 3.4[14].

```
(def-translation MONOID-THEORY-TO-ADDITIVE-RR
    (source monoid-theory)
    (target h-o-real-arithmetic)
    (fixed-theories h-o-real-arithmetic)
    (sort-pairs
     (uu rr))
    (constant-pairs
     (e 0)
     (** +)
    (theory-interpretation-check using-simplification))
```

Listing 3.4: IMPS theory translation

Here we see a postulated theory morphism from the theory of *monoid*s into the the additive structure of the reals $\mathbb{R}$. This morphism maps the base-set "uu" of the monoid to the base-set "rr" of the reals, and $e \mapsto 0$ and $** \mapsto +$ as expected, where e is the neutral element in the monoid and $**$ is its operation. The IMPS system parses such constructs as shown in Listings 3.3 and 3.4 and offers verification procedures for theorems and their proofs which are provided in the same format. The system takes into consideration axioms and assertions from ancestor theories and makes them available for proofs in descendant theories depending on the characteristics of the theory morphisms.

IMPS is based on higher-order logic with sorts, so its theory morphisms can accordingly be specified to be quite powerful. It allows to specify theory inclusions that relativize quantifiers, which allows to restrict the domain of application of the source theory axioms in the target theory. These kinds of morphisms are described by William M. Farmer in [Far93]. Such a morphism allows, for example, to define a morphism from *group* to the multiplicative group of *field* by restricting the applicability of the group axioms to non-zero elements of the field.

Obviously, the theory-graphical method is very well integrated into the IMPS system. Even though it has been around for more than decade, it has not managed to attract a large user community that would contribute to the system's library. Freek Wiedijk does not list IMPS as one of the QED-like systems in [Wie07]. We give two main reasons why this is the case and show how the Bourbaker project is solving the underlying problems.

[14] the example can be obtained from: http://imps.mcmaster.ca/theories/algebra/monoids.html

Problem 1: the IMPS syntax

IMPS has a very programming-like syntax just like virtually all the other formal libraries. While the examples in Listings 3.3 and 3.4 are still easy to understand without prior training in the syntax, this is not the case any more for general proofs. Listing 3.5 shows a short example of an IMPS theorem and its proof[15].

```
(def-theorem locality-for-monoid-prod
    "forall(f,g:[zz,uu],p,m:zz, forall(x:zz,
      p<=x and x<=m implies f(x)==g(x))
        implies monoid%prod(p,m,f)==monoid%prod(p,m,g))"
    (theory monoid-theory)
    (usages transportable-macete)

    (proof
      (
        direct-inference
        (case-split ("p<=m"))
        (induction integer-inductor ())
        (prove-by-logic-and-simplification 3)
        (apply-macete-with-minor-premises monoid-null-prod)
        )))
```

Listing 3.5: An IMPS theorem

Certainly, there are worse examples of syntaxes around, but even for these kinds of constructions it is inconceivable that a mathematician would seriously consider communicating his research in this format. The IMPS system can only handle theories, definitions and theorems that are semantically correct, so that it is impossible to extend the library with sketches or blueprints of theories, let alone theorems without proof. Still today, the IMPS library is limited to the simple example files from basic algebra and real arithmetic.

The Bourbaker approach, however, allows to build theories without the requirement of creating a system of sound definitions and theorems. Moreover, definitions, axioms, imports and theorems can largely be stated in natural language without worrying that the underlying system will reject the entry. These natural language statements have the additional advantage that the resulting theories and their content are very readable to humans, which gives the constructed theories a meaning outside the system in the first place. The Bourbaker suite itself simply injects trust in the creator of theories and theory content that the statements make sense and that the stated imports are sound.

In a second step, Bourbaker's theory graphs allow for the implementation of automatized methods such as the IMPS system's verifiers. For this, the theory contents would be annotated in parallel with their content in a formal language, which would then build the basis for proof checkers and theorem provers.

Problem 2: IMPS' logic

IMPS' second problem is that it is deeply rooted in the use of higher-order logic. This is also a problem which fundamentally all formal libraries have. Higher-order logic is desirable from the point of view of a mathematician, since it allows to

[15]source: http://imps.mcmaster.ca/theories/algebra/monoids.html

formalize virtually all of known mathematics in such a logic. On the downside, mathematicians are usually never keen on rigidly formalizing their results in any specific logical system. In published mathematical communication, it is enough for the experts to know that their definitions and results could be stated in a rigid formalism. For the rest of the time, it is much more important for them that their exposition is clear and understandable to their colleagues, which often includes logical reasoning schemes where desirable, but is not confined to them.

A typical logical convention that is influencing the IMPS system is the use of relations in place of function symbols in higher order logic. When reviewing the IMPS library, this convention was apparently implemented during the development of the system. Listing 3.3 exhibits this behavior with the term "rewrite," which instructs the system to transform the function-like use of "inv" back to its intended role as a binary relation.[4] EdNote(4)

The Bourbaker system, on the other hand, does not require to fix a certain logic for defining theory graphs. Certainly, theory morphisms depend on the strength of the underlying logic so that some morphisms exist when using higher-order logic, while the same morphism cannot be expressed in first-order logic. In the worst case, client applications have to be aware of these differences when browsing the theory graph, so that the system does not claim the existence of a description logical theory morphism when the morphism really relies on higher-order logic.

Furthermore, Bourbaker's theory graphs offer simple semantic theory morphisms that are sufficient to link mathematical concepts without having to rely on any specific logic at all, which certainly comes closest to the frequent implicit use of theory translations in modern mathematics. These kinds of logic-less morphisms are enough to define added-value functionality such as searching and shallow proof assistance as described in Chapter 5. This flexibility makes Bourbaker theory graphs much more useful for a wide range applications while at the same time subsuming functionality of logic-based systems such as IMPS.

3.3 Algebraic specification languages

Formal languages are not only used to feed theorem provers. As we will see in this section, the proper design of formal languages is interesting in its own right as such languages can be used for formalizing a variety of relational sets of data. This field was vastly popularized by the need to develop languages that can be used to state and verify the architecture of software programs. The specific case that we are interested in here are the so-called algebraic specification languages which are also suitable for the formalization of mathematical content.

Essentially, we have to consider only a single algebraic specification language called *CASL*[16], which has grown out of a common effort of the community in the mid-1990s to generate one specification language that integrates the great variety

[4] EdNote: consider throwing this out, it seems to be misplaced. might cause confusion with the syntax part

[16] http://www.informatik.uni-bremen.de/cofi/wiki/index.php/CASL

of algebraic specification languages that existed at the time.

CASL is very close in spirit to our theories and their graphs. As the "CASL User Manual" states, "a basic specification declares symbols, and gives axioms and constraints." [BM98] The symbols and axioms in a specification are direct analogues to symbols and axioms in our theories. CASL extends the notion of "basic specifications" by "structural specifications" based on a signature of symbols.

Between structural specifications, CASL allows to specify signature morphisms that induce theory morphisms. Such morphisms can rename symbols and hide symbols, whereas the latter operation is not supported in our theory graphs because it does not lead to theorem propagation. As in our theory graphs, CASL identifies common symbols from multiple imports, which the authors call the "same name, same thing" principle. Listing 3.6 shows the sample specification of a *monoid* in CASL[17].

```
spec
      Monoid =
      SemiGroup
then
      ops
            e:      Elem;
            __ * __: Elem * Elem -> Elem, unit e
end
```

Listing 3.6: The specification of a *Monoid* in CASL

The listing shows how *monoid* imports from *semigroup* and defines the additional operator e. The second declaration redeclares the *semigroup*'s operation as an operation with unit e, which effectively replaces an axiom stating that e is a unit.

The standard CASL library supports stating axioms in first-order logic. In addition to basic CASL, there are a great number of extensions to the language, some of which allow statements in higher-order logic. In particular, the language extension HetCASL allows specifications that are heterogeneous in different logics. Till Mossakowski's "heterogenous tool set" (HETS) offers analysis and proof management tools for such heterogeneous specifications (see [MML06]).

This all goes a long way towards our theory graph model. But even though most of the sample applications of CASL are coming from mathematics, the language was developed with an eye on enabling the specification of software projects. This should not divert our view for the applications of CASL for our purposes, but it explains why the language does not place any emphasis on properly representing mathematics in a human-oriented way.

In spite of the many logical formalisms that CASL supports, it is not able to annotate anything with natural language statements. Items such as different kinds of assertions or definitions are unknown to the specification language. While the language is parsable by computers, its style is not flexibly extensible by annotations, multiple statements of items, and the like. Instead, every statement has to be made in full logical formalization. An augmentation of CASL by this kind

[17] source: http://www.informatik.uni-bremen.de/cofi/CASL/lib/basic/v07/Algebra_I.casl

of functionality would basically result in the kind of format that we are using. However, at that point CASL would lose its focus of being a formal specification language in the first place.

Additionally, notice that CASL's theory morphisms are not as expressive as ours. In CASL, we cannot express a theory morphism from *group* to the multiplicative group of a *field*, because we have to map *group*'s base set to the *field*'s with the 0-element removed. Mapping to expressions is not permissible in CASL.

It would be interesting to embed CASL in an OMDoc theory graph. Thus the tools that already exist for CASL would become available in OMDoc. Unfortunately, this would also require the full logical formalization of mathematical statements, which seems rather elusive for large fields of mathematics. This thread will be picked up in the discussion of future work in Chapter 7.

3.4 Interactive learning systems

The concept of using conceptual dependency graphs in mathematics is a modern teaching tool in the sciences. Such graphs serve to visualize various kinds of relations such as subsumption, membership, or equivalence of mathematical concepts and they are useful for students to gain an overview over a new field of mathematics and ease orientation therein.

ActiveMath[18][19] is an e-learning platform for mathematics developed at Deutsches Forschungszentrum für künstliche Intelligenz and the University of Saarbrücken which employs this concept. The system comprises a collection of books that contain sections on individual topics such as definitions or theorems. [MGH+06] Many items are explained with interactive graphical displays and the like, and mathematical formulas are displayed from proper semantic content (in OpenMath or Content-MathML), which can be retrieved by the student and used for semantic search within the system. Figure 3.4 shows a browser screenshot of ActiveMath's book "complete content of LeAM_Calculus."

Along with the books for self-study, the ActiveMath system offers a small selection of didactic mathematical tools, among them a function plotter, a basic computer algebra system, and a so-called concept map tool. The concept map tool is interesting here because it allows to display and manipulate mathematical dependency graphs based on the ActiveMath corpus. Figure 3.5 shows a screenshot of this tool in action with a couple of items that were dragged over from ActiveMath's "LeAm_Calculus" book.

The arrows are inserted by the user of the system, however, the tool makes suggestions where links should be placed and what links are misplaced. As can be seen in the screenshot, arrows can carry one of several tags that qualify the intended meaning of the relationship between the concepts.

Of course, linking examples to definitions does not constitute a theory graph.

[18] http://www.activemath.org/

[19] ActiveMath must not be confused with AutoMath, which is a formal mathematical language developed in the 1970s by N.G. de Bruijn

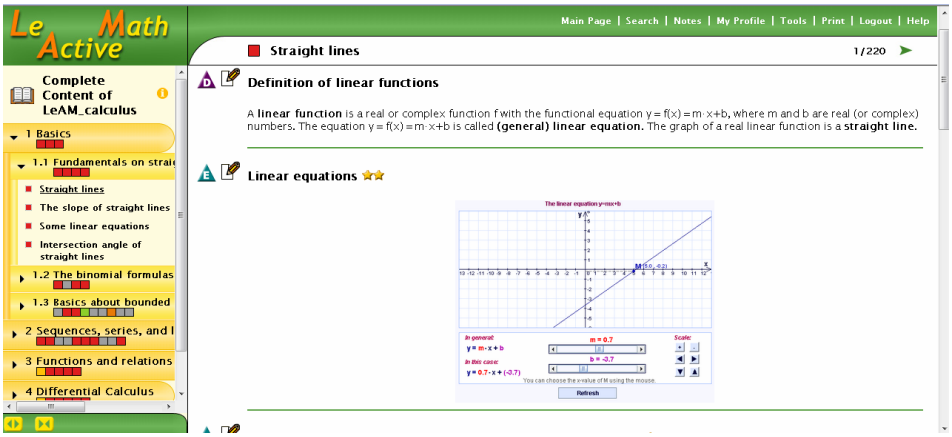

Figure 3.4: Screenshot of ActiveMath. Source: http://demo.activemath.org/ActiveMath2/main/viewBook.cmd

The ActiveMath system is not so interesting for its mathematical rigor, but more for the approach to use dependencies between theories to showcase them in a way that is particularly intuitive and natural for understanding the mathematics behind it. Drawing edges between examples and definitions would arguably not improve the education of an undergraduate student. But exploring and manipulating theory graphs just might do so by providing a point of reference that complements the linear exposition in the textbook.

Bourbaker's theory graphs have a built-in edge over other styles of exposition of mathematical content because they can be visualized, so that dependencies can be grasped in a short amount of time. And what is more, these dependencies are not just simple links that hint at any kind of relations, but their specification contains the exact nature of the morphism. The theory graphs encode a part of mathematics that is vital for every student of mathematics to understand and they do so in a handy format.[5] EdNote(5)

3.5 Semantic libraries

The term "semantic library" does not come from the literature in mathematical knowledge management. We use it here to describe systems that aim at augmenting content that would typically be found in digital libraries (see Section 3.1) with semantic markup that can subsequently be used for offering added value services, such as semantic search. As this approach is relatively new, there are not a lot of systems around that demonstrate the functionality of such libraries.

The ActiveMath library which was introduced in Section 3.4 qualifies as a semantic library because the mathematical formulas are semantically annotated and crosslinked. The formal libraries from Section 3.2 contain semantically unambigu-

[5] EdNote: now I could talk about ActiveMath's old OMDoc format and how it doesn't allow to specify theory morphisms... could insert OMDoc 1.0 snippet from http://www.activemath.org/~paul/copy_left/DagStuhlRuleMLworkshop/05.html...

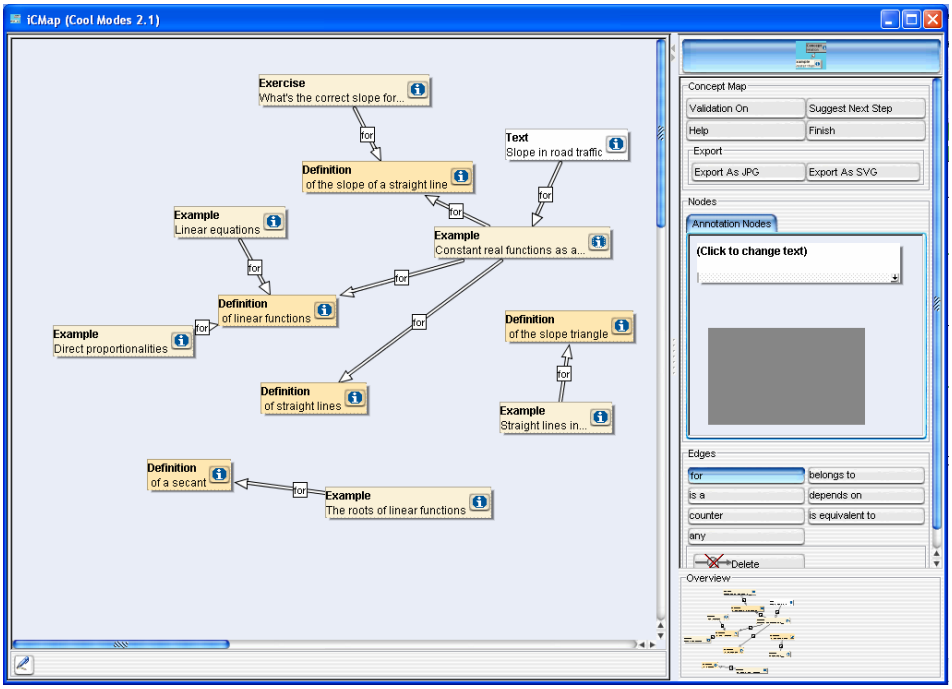

Figure 3.5: Screenshot of ActiveMath's Concept Map Tool

ous content, yet, their primary purpose is not the presentation of this content to humans, so we do not consider them semantic libraries. In this section we consider two very different approaches. The first one is Matthias Bröcheler's proposition of a decentralized "mathematical semantic web," and the second one is Christoph Lange's SWiM semantic wiki.

The idea of a mathematical semantic web is based on the annotation of mathematical concepts in description logic [Brö07]. Description logic is one of the cornerstones in research in the field of semantic webs. It is not expressive enough to fully formalize many mathematical concepts (see, e.g., [BCM+03]), but it is fairly easy to express conceptual relationships in it.

The hope is that if mathematicians would annotate their publications with description logic annotations and publish those on the world wide web, added value services become possible. The links between concepts in description logic are not fully qualified like theory morphisms in theory graphs, but they are somewhat qualified so that one can do meaningful transitive search.

It is of course questionable why description logic should be able to motivate a larger number of mathematicians to participate in the annotation than other logical formats. What is certainly interesting, though, is that the mathematical semantic web follows the concept of decentralized publishing. The annotations slumber in corners of the world wide web and wait for web crawlers to wake them and connect them with snippets of information found elsewhere.

Such a completely decentralized model is not possible with theory graphs because proving the equivalence of theories is undecidable in general. Our approach using toolbox theories is as close as we get to constructing small combinable theory

graphs. Having a more centralized theory graph is still advantageous since this graph constitutes a tangible map of current and important mathematical concepts rather than arbitrary search results.

The example of a semantic library that we are going to discuss here is the SWiM semantic wiki by Christoph Lange. Figure 3.6 shows a screenshot of SWiM's demo version.

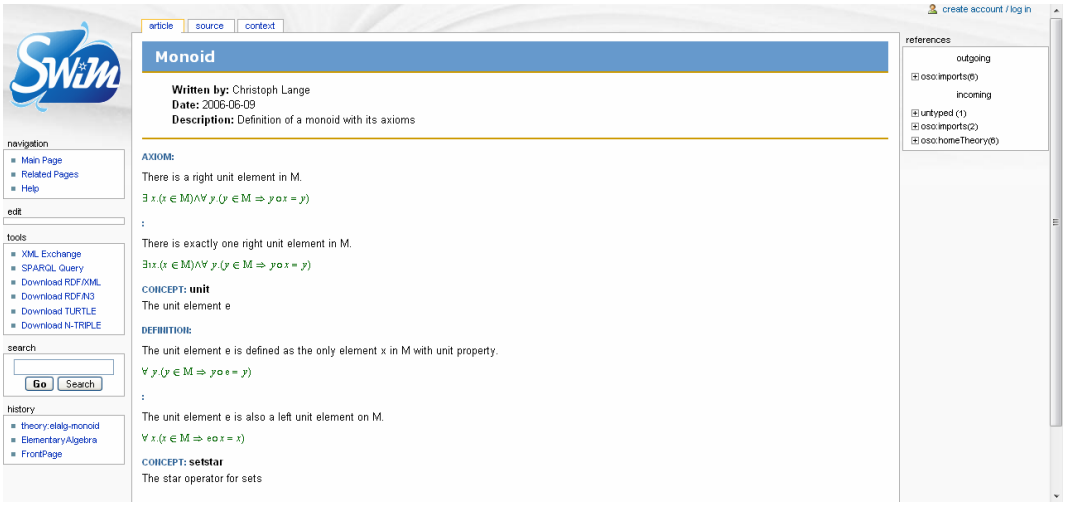

Figure 3.6: Screenshot of SWiM semantic wiki. Source: http://raspberry.eecs.jacobs-university.de:8081/SWiM

Essentially, SWiM is a wiki with much the same appearance as Wikipedia, but augmented with the capability of handling semantically annotated content (see for example [Lan07a], [Lan07b]). This is implemented using OMDoc as the underlying format, so that mathematical formulas can be authored and retrieved in their proper semantic form. Additionally, SWiM offers the option to tag wiki pages with user-defined types. For example, a page containing the Pythagorean theorem would receive the tag "theorem." This augmented type of wiki aims at enabling experts to communicate semantically critical information and at the same time to offer semantic search and the visualization of dependencies.

As such, SWiM tackles a great lot of the shortcomings of digital libraries when it comes to handling the mathematical content. The OMDoc format backing SWiM allows to specify proper theory morphisms in the same way as the Bourbaker system. As SWiM is not optimized for this task, though, it does not exploit the exact relations between theories. It merely uses the fact that there is a morphism between theories to create hyper-references and perform a morphism-blind type of graph search.

At this point it becomes clear that the SWiM system and Bourbaker could benefit from each other. A SWiM-like architecture would make for a good frame to present theory graphs if it was augmented by Bourbaker's theory graph functionality. And mathematical theories such as the ones implicit in Bourbaki (see Chapter 4) would be a nice test case for the semantic capabilities of SWiM.

Chapter 4

The Bourbaki case study

The works of Nicolas Bourbaki will serve as a case study for the value of theory graphs in this work. Specifically, we recast the first chapter of Bourbaki's Algebra I [Bou74] as a theory graph structure. Our aim is to study the behavior of mathematics in the way it is currently practiced under an organization as little theories. As this section will show, theory graphs allow for a natural expression of many connections in the book, but the versatility of mathematical exposition also highlights the limits of the theory-graphical method.

Some of the results in this chapter are inspired by the results from the "Bourbaki II"-group annotating the first ten pages of Bourbaki Algebra I in *sTeX*, a LaTeX variant that allows for semantic markup that is in principle transformable to OMDoc. The members of this group were Matthias Bröcheler, Kristina Sojakova, Bogdan Minzu, and myself. Matthias Bröcheler has used the data from the "Bourbaki II"-group for the extraction of mathematical ontologies in his Bachelor's thesis [Brö07]. The set of best practices that he derives are complementary to our heuristics presented in Section 4.2.1.

4.1 Why Bourbaki?

For the case study we needed a textual account of mathematics whose exposition is close in spirit to theory graphs already. Additionally, the text should be a maximally hard test of suitability, so it was supposed to cover the treated subject in great depth. This disqualified the vast majority of undergraduate textbooks as sources as they lacked depth and their didactic accounts for the reader were irrelevant to the theory graph structure, if not confusing in the translation process. Graduate texts, on the other hand, seemed too specific.

4.1.1 The adequacy of digital libraries

This left a hand full of encyclopedic works to chose from. At first sight, general digital libraries such as MathWorld or PlanetMath (see Section 3.1) would have been a possible candidate for transformation. In fact, the prospect of building a theory-graphical semantic library from existing digital libraries will be discussed

in Section 7. Also, a lightweight theory graph constructed from a selection of Wikipedia articles will be used in Section 4.3 to evaluate how well the created theory graphs integrate with others. Nevertheless, digital libraries have a number of disadvantages for a case study like ours.

Firstly, as they are based on voluntary contributions, we have no way to determine if they contain an in any way complete projection of mathematics. While it can be expected that currently researched topics are well presented in those libraries, it is very questionable in how far foundational articles are consistently included and connect well with the advanced material. Such good connectedness is of course vital for the shape of the resulting theory graph. And possible bias in the articles would have reduced the credibility of outcomes of the case study.

Secondly, there is no real boundary in digital libraries. As this study can impossibly cover the whole of mathematics, we would have had to make many arbitrary choices of what to cover and what not. Again, this would have called the credibility of the outcome into question.

Lastly, the usual lack of editorial review in digital libraries leaves us uncertain about whether their articles can actually be assembled in a semantically consistent way. Small imprecisions are common in digital libraries as they are in mathematics. A theorem might actually require a slightly different definition than is given in the article on the theory, there might be several contested ways of defining a certain theory (e.g., rings and commutative rings are often identified), and so on. The case study would have to make a choice on how to resolve such conflicts in each individual case, which can hardly be done consistently, if it can be done at all.

4.1.2 The adequacy of foundational encyclopedias

Excluding digital libraries from the picture, we are left with a small number volumes that try to develop the body of mathematics from first principles. The Bourbaki books belong to this group, and so do Hilbert's "Grundlagen der Geometrie" and Whitehead and Russell's "Principia Mathematica."

Hilbert's account of geometry we exclude as being too specific to be relevant throughout all of mathematics, even though its incremental axiomatic approach would map nicely into theory graphs (see Section 2.1).

The three volume "Principia Mathematica" we exclude for a completely different reason: it is too axiomatic. The authors of the books start their account with first principles and develop everything from there. However, the abstract face of set theory is largely very technical to the point where the specific details of the theory lose their ubiquity in much of contemporary mathematics. In other words, a branch of mathematics that uses its own language does not become immediately useful for other areas by just casting it as a theory graph. The three volumes of the "Principia" only arrive at covering basic real arithmetic, so even if we managed to transform all of it we would still be left with a very incomplete picture of mathematics. For the same reasons we do not consider the first Bourbaki volume on set theory [Bou68], which is more concise than the "Principia" but not less technical.

4.1.3 The adequacy of Bourbaki Algebra I

Instead of a development from first principles, the little theories approach that the we take is modular, which means that theories can be inserted at any point in the theory graph as long as consistency is vouched for by the author. Our case study is also supposed to advertise this aspect. As it will turn out, the definition of theories in *algebra* only relies on a small fraction of the results presented in *set theory*, which will be organized in a toolbox theory that underlies the created theory graph as discussed in Section 2.4.5.

Bourbaki's Algebra I treats, among others, basic algebraic structures such as *magmas*, *monoids*, *groups*, *rings*, and *fields*, linear algebra including *modules* and *vector spaces*, and tensor algebras. Thus, the scope of the book surveys a well-founded area of mathematics that makes appearances in all branches of mathematics and that is fundamental to the work of many mathematicians.

The Bourbaki texts place an emphasis on a rigorous build-up of mathematical theories, which makes them a perfect candidate for a transformation into theory graphs. The books have the property that they only refer to previous chapters in Bourbaki, which eases the proper linking of the theories. Also, the ideal of casting definitions and propositions in the most general way possible should make for a fine granularity of the resulting theory graph. Finally, the encyclopedic character of the volumes promises a thorough treatment of the covered theories that captures results that most mathematicians are not aware of in their daily work.

4.1.4 A historical account of Bourbaki

Nicolas Bourbaki was the pen name of a group of mostly French mathematicians who decided to overhaul mathematics in the 1930s. Among the founding members were Henri Cartan, André Weil, and Jean Dieudonné. Until the last book on spectral theory in 1983, the group published 9 volumes with varying membership and without acknowledging individual authorship.

From the start, the group had been influenced by the success of the axiomatic method that was practiced in the "flourishing German school (Göttingen, Hamburg, Berlin)" [Bor98, p. 373]. Their aim was a strictly self-contained presentation of mathematics, always attempting to achieve greatest generality for the presented definitions and theorems before proceeding to more specific statements. All books had complete proofs for every statement and they do not employ forward references or references to other mathematical texts except for some historic background information. While the books provide examples and exercises to supplement the presented material, no concessions to the reader are made such as presenting intuitions or heuristics on how to solve problems.

In fact, the rigorous presentation of Bourbaki's elements of mathematics also gives rise to the most articulate criticisms of the books. Even Pierre Cartier, himself a member of the third generation of Bourbachistes, concedes that "you can think of the first books of Bourbaki as an encyclopedia of mathematics [...] If you consider it as a textbook, it's a disaster." [Sen98] Cartier is referring to

the abstract character of the exposition which often introduces concepts that are meaningless to an inexperienced reader and whose relevance only becomes clear later, often in subsequent books, when other theories build on them.

It is worthwile noting that in spite of these inadequacies for teaching, the Bourbaki treatises had a great impact both on the teaching and on the practice of university mathematics in the second half of the 20^{th} century. While they might be useless as standard textbooks, they have inspired a great amount of actual textbooks for the undergraduate and graduate level that make use of the axiomatic method's logical clarity. The use of the signs $\emptyset$ for the empty set or $\Rightarrow$ for implication trace back to Bourbaki's formalization [6]. Even the "New Math" movement of the 1960s, which was aiming at concentrating the mathematics education of elementary, middle, and high school students in Europe and the US on the teaching of abstract concepts such as set theory, is partly attributed to Bourbaki's influence at the time. [Kle03] EdNote(6)

Given the thrust of its axiomatic approach to mathematics, one has to name Bourbaki's omissions of whole areas of mathematics. Cartier admits that Bourbaki never treated analysis "beyond the foundations" or geometry in its concrete form, and that they did not consider probability theory, combinatorics, algebraic topology, and logic altogether. [Sen98] Fields Medalist Timothy Gowers attributes these omissions to a divide in mathematical spirit between those mathematicians who want to build pure theories with mathematical problems as illustration, and those who primarily want to solve problems and who use theories only as tools to this end. [Gow99]

Drawing this distinction, Bourbaki is clearly on the theory-building side with its almost dogmatic axiomatic approach. Then it might become clear why rather problem-oriented fields of mathematics such as number theory or combinatorics are not treated by Bourbaki.

One last serious point of criticism comes from Adrian Mathias and others in that Bourbaki ignored limitations to the axiomatic method that were known at the time. [Mat92] They find that Bourbaki was just continuing the part of Hilbert's program that was concerned with devising a unified formal system for mathematics, but at the same time omitting credit to the incompleteness theorem that Kurt Gödel had presented in 1930.

For us, starting the theory-graphical formalization of Bourbaki with the second volume on algebra instead of the first one on set theory steers clear of the last fundamental criticism. While it seems that the intensity of the debate on this issue has significantly cooled off, this work is not an attempt to take side for the choices in Bourbaki. In fact, the choice for Bourbaki is rather for pragmatic reasons. At the same time, however, the approach presented here has the potential to make the axiomatic method a more flexible tool and, with a view towards Bourbaki, to turn its mathematical rigor into a virtue even for studying mathematics.[7] EdNote(7)

[6] EdNote: the reference for this would be German Wikipedia...✓

[7] EdNote: also mention somewhere (maybe here) that Bourbaker is limited to capturing theories, and that methods and intuitions cannot really be mapped (but that Bourbaker is rather a whole new method in itself)

4.2 The transformation of Bourbaki into linked theories

For this case study, I formalized 1 through 4.2 of chapter 1 in Bourbaki Algebra I[Bou74] as little theories. I skipped the translation of proofs as they would not have added to the theory structure and I skipped examples as they are notorious for making references to theories that are not contained in the exposition (cf. the criteria in Figure 4.2). Figure 4.1 shows the resulting theory graph as displayed by the OrganicGraph applet (see Sections 5.1 and 6.6).

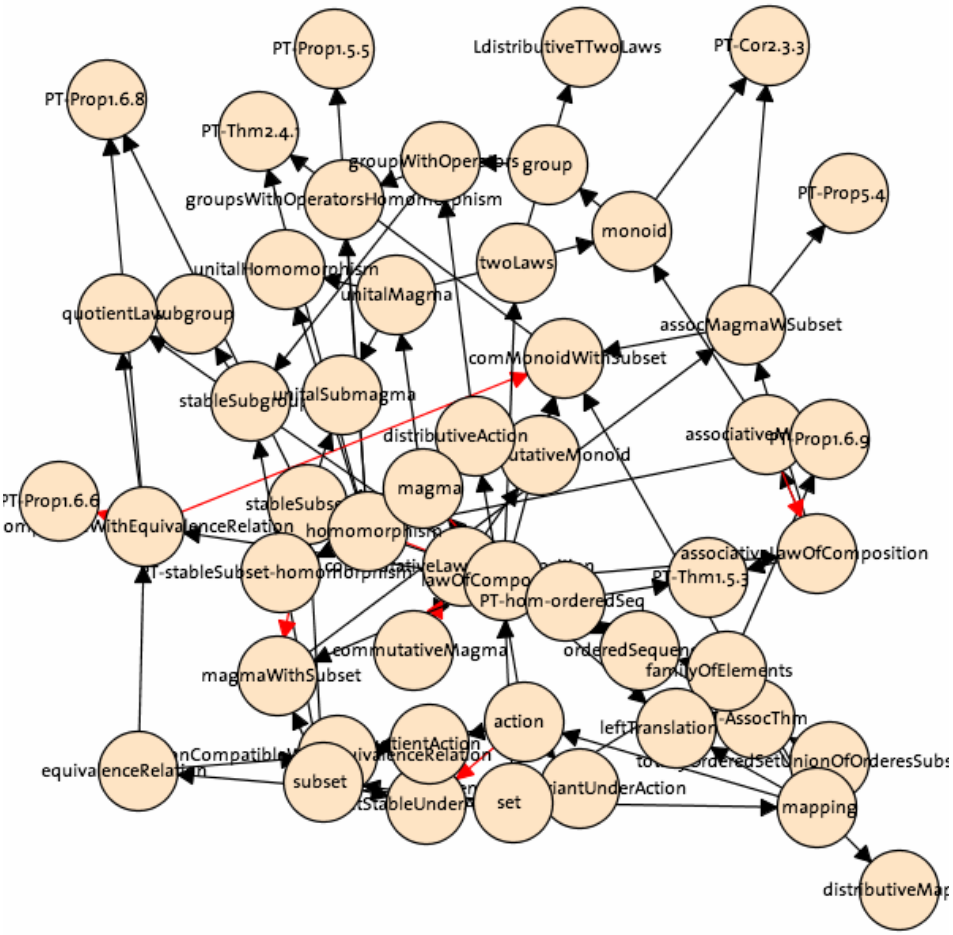

Figure 4.1: Bourbaki's theory graph

The graph contains 51 theories, 94 definitional and 10 postulated theory morphisms. 82 Symbols, 38 axioms, 30 assertions and 17 definitions were defined inside the theories. In total, the formatted OMDoc file is a document of about 80 printed pages, which is due to the markup. Even though the translation comprises just 32 pages in Algebra I, the graph is already quite complicated. It enables us to assess the text's aptitude for transformation into a theory graph structure and it allows to make predictions about the outcome of a translation of the whole text.

Of course, a translation of this size was already non-trivial. Section 4.2.1 presents the *heuristics* that I used for the translation task in order to standardize the choices that are necessary during the translation. In order to be able to assess the quality of the resulting theory graph and to compare it to the theory graph from Hungerford's Algebra book, I developed a set of *criteria* that guide the discussion in Section 4.3. These criteria will be introduced in Section 4.2.2.

4.2.1 Heuristics for the transformation

Transforming mathematical texts into theory graphs is an art rather than a science. Even Bourbaki's books, which are said to strictly adhere to the axiomatic method, do not explicitly list theories and their axioms. New theories have to be spotted from the text by the translator and their axioms have to be extracted from the given definitions. In order to ensure a uniform treatment of the texts and maximum comparability of the resulting theory graphs, this section gives a number of rules that were followed during the transformations. The five heuristics are displayed in Table 4.1.

H1	cast *definitions as axiomatized theories* and theory imports where possible
H2	achieve the *finest granularity of theories* that can be extracted from the text
H3	insert *alien theories conservatively*
H4	name *symbols by the concept* they represent
H5	insert *technical theories for propositions* that connect theories

Table 4.1: Heuristics for the transformation of texts

H1

The first heuristic refers to the method of targeting the definitions of the text for the extraction of theories. In most mathematical texts definitions are the standard way to introduce objects and concepts that are subject to certain rules. These defining rules become the candidates for axioms to the theory. This set of axioms is then reduced as much as possible by identifying theory imports that incorporate these axioms. For example, the definition stub "a group is a monoid where every element is invertible" indicates an import from the theory of monoids, plus an additional axiom on invertibility.

H2

The second heuristic describes the goal of creating small theories, as opposed to large theories that incorporate a lot of concepts. Quotation 9 shows the first definition from Bourbaki as an example.

The definition defines two concepts, that of a law of composition and that of a magma. As theories these concepts are identical as they both describe a set with a binary operation on it. In the translation, this definition is split up

> "Let E be a set. A mapping f of $E \times E$ into E is called a law of composition on E. [...] A set with a law of composition is called a magma."

Quotation 9: Bourbaki Algebra I [Bou74, p.1]

into the two theories *law of composition* and *magma*, in order to account for the distinction on the conceptual level. The identity of the two theories is expressed by assigning circular theory morphisms[1] (e.g., *magma* imports *law of composition* and *law of composition* includes *magma* by means of a postulated theory-inclusion). In general, every new concept gets its own theory if it is not completely subordinate to another defined concept. An example for such a subordinate concept would be the induced law on a stable subset. The definition is exemplified in Quotation 10. An induced law only makes sense in the context of a stable subset of a magma, therefore, the symbol, axioms and imports for the induced law are included in the theory *stable subset*.

> "A subset A of E is called stable under the law of composition $\top$ on E if the composition of two elements of A belongs to A. The mapping $(x, y) \mapsto x \top y$ of $A \times A$ into A is then called the law induced on A by the law $\top$."

Quotation 10: Bourbaki Algebra I [Bou74, p.6]

H3

The third heuristic deals with the unpleasant event of when an "alien theory," a theory which is not given in the book, has to be inserted in order to make up for references to theories outside the scope of the text. The first such instance for the texts under consideration here is the creation of a toolbox theory which contains general concepts that are commonly needed all the time. Table 4.2 shows the symbols that are defined in the toolbox theory for the Bourbaki study. As described in Section 2.4.5, such a toolbox theory can later be founded on an appropriate version of set theory, e.g., Bourbaki's first volume.

Definitions might additionally require theory imports that are nowhere defined in the text. These theories are then defined in place in accordance with what symbols and axioms are needed to create a sensible import. Alien theories are not supplied with any additional symbols, definitions, or assertions even if these might be easily inferrable from common mathematical knowledge. In the Bourbaki study, the theories *subset* and *mapping* were created as alien theories.

[1] As we do not allow definitional morphisms to be circular, one of these morphisms has to be postulated.

set	the concept of a set	{ }	set constructor
setminus	set difference	in	set inclusion for elements
proj1	canonical projection on first coordinate	x	Cartesian product of sets
proj2	canonical projection on second coordinate	=	generic equality sign
	whitespace constructor for ordered tuples	circ	function composition

Table 4.2: The symbols of Bourbaki's toolbox theory.

H4

The fourth heuristic addresses the issue of symbol names. Mathematical texts often use variables with certain properties instead of using abstract objects directly. For instance, a definition is likely to start with "let E be a set," declaring E to be an instance of the type "set," or equivalently stating that E has the property of being a set. The choices of variable names are not mimicked in the naming of the theory's symbols, instead, concepts are used directly in place for the variables. For this example this means that we would create a theory with symbol called *set* instead of *E*.

This is not always possible when multiple imports have to be distinguished. In that case, new symbols are declared that are defined by making appropriate imports. Thus, the theory of a mapping owns the two symbols *domain* and *codomain* that are identified as sets by imports.

Symbols are also renamed when the definition redefines a concept. For example, an *action* is just a certain type of *mapping* and could be axiomatized as such. Still, we chose to rename *mapping* to *action* in order to stress the underlying concept.

We allow renaming in one more situation when it is not easily inferable any more where a certain concept of a theory is coming from. In practice this hardly ever happens, and as a rule, the texts rename concepts much more often than actually necessary.

H5

The fifth heuristic is very important for the translation of propositions and theorems. Generally, theorems are added to the theory that they belong to, as they do not alter the content of the theory. However, many theorems have the fortunate property of interrelating two or more theories, so that they do not fit into either one. Quotation 11 shows a theorem from Bourbaki that relates the concepts of *monoids*, *commutative monoids*, *homomorphisms*, and *monoids of fractions* with their associated *canonical homomorphisms*.

For these kinds of assertions, special theories are created for the sole purpose of importing the necessary concepts so that the assertion can be stated. Such theories, which do not have any axioms of their own and which do not serve the purpose of defining a new concept, are called *technical theories*. Technical theories can be used to define a special kind of crosslink within theory graphs. In the Bourbaki study, we specifically mark them by a name starting with "PT."

"Let E be a commutative monoid, S a subset of E, E_S the monoid of fractions associated with S and $\varepsilon : E \to E_S$ the canonical homomorphism. Further let f be a homomorphism of E into a monoid F (not necessarily commutative) such that every element of $f(S)$ is invertible in F. There exists one and only one homomorphism $\bar{f}$ of E_S into F such that $f = \bar{f} \circ \varepsilon$."

Quotation 11: Theorem from Bourbaki Algebra I [Bou74, p.19]

4.2.2 Criteria for assessment

For the assessment of the translation results of the texts I have developed three main criteria: *clarity*, *quality*, and *compatibility*. Figure 4.2 shows the questions these criteria address. Generally, the clarity criterion addresses how easy the translation task is for the given text; the quality criterion measures data about the shape of the resulting theory graph that is important for applications such as searching in the graph; the compatibility criterion determines how well the theory graph can be combined with existing theory graphs.

- *clarity*:
  - How immediate is the *correspondence to theories* from the given text?
  - How easy is it to *determine imports* between theories?
  - How much *choice* is involved in constructing theories? Are these choices *equivalent*?
- *quality*:
  - How often does the text require imports from *alien theories*?
  - What is the *shape* of the resulting theory graph? Does the text build theories in a linear fashion or does it discuss a broad variety of independent theories?
  - How suitable is the graph for *searching* and other *added value functionalities*? (cf. Chapter 5)
- *compatibility*:
  - How *robust* is the resulting graph against ill-defined links?
  - How well is the resulting graph *extensible* by new concepts?
  - How well does the graph structure *combine with other theory graphs*?

Figure 4.2: The criteria for assessing transformations into theory graphs.

Let us dwell on the motivation for these criteria for a little while.

Clarity

Clarity is the criterion that is directly concerned with the way that the text is written. It determines how easy it is to transform the given text into a faithful theory graph version of it. For that, we examine how clearly individual theories are already worked out in the text, and whether possible theory imports and inclusions are easily identifiable at a first reading. It also addresses whether the translation task is rather mechanical or whether the text requires to make a lot of choices when building the theory graph.

A text that does not possess clarity in our sense is not useful as the basis for a general theory graph because the translation process is extremely tedious. Additionally, the outcome of translating an unclear text is questionable as it might be biased by the translator's decisions.

A document that is just a linear arrangement of theories with import references is extremely easy to transform. Introductory texts, however, that appeal to general intuition instead of stating definitions and theorems can hardly be formalized at all without injecting additional knowledge. Naturally, no text is written as just a chain of theories, but Bourbaki Algebra I and Hungerford's Algebra come surprisingly close, as will be discussed in Sections 4.3 and 4.4. Along the way, these sections will also exemplify typical problems in the transformations of the texts.

Quality

The quality criterion tries to assess the resulting theory graph when viewed in isolation. The quality of the graph naturally depends on the clarity criterion. Yet, quality rather examines the structural aptitude of the text for casting it as a theory graph.

Firstly, it asks how many theories have to be filled in by the translator because the text refers to outside concepts. As mentioned before, the introduction of alien theories is a source of bias in the translation, and possibly a source of errors. Therefore, a text is suitable for translation if references to alien theories are rare.

Secondly, we have the graph's applications in mind when we speak of its shape. A totally disconnected graph of isolated theories is certainly not very interesting. It turns out that the theory graphs both in Bourbaki's and in Hungerford's algebra books are very well connected, so we turn our focus to the fashion in which theories link to each other. A broad coverage might lead to little depth of a graph, while a linear development of theories would produce a simple path as its theory graph. Both cases are not very interesting as their structure is easily comprehensible even without our tools. Thus, a graph becomes interesting to us when it is both wide and deep, when it is both locally well-connected, and when it contains edges that span great distances.[8] EdNote(8)

This criterion offers the chance to make empirical comparisons of theory graphs. We measure the depth of the graph by the longest path on definitional morphism edges that it contains. As a measure of width, we consider the size of the largest

[8] EdNote: consider inserting sample graphs for illustration

antichain that the definitional theory graph contains. An antichain is a set of vertices of the directed graph such that there is no path between any two of them. By Dilworth's theorem, the size of the largest antichain is equal to the number of paths that are necessary to cover all vertices of the graph (cf. [Die06, p.51]). Thus, the maximum size of an antichain is an indicator of how many different morphism paths we can expect.

In addition, we consider a number of simple data about the graph: the *average out-degree* of vertices[2], the number of *leaves*[3], the number of *separation points*[4], the number of *definitional and postulated morphism edges*, and the number of morphisms that *translate a symbol to a different expression*. Leaves indicate a broad foundation within the covered area, because they allow for instant theory-imports in later development without having to worry too much about fine granularity. Separation points indicate bottleneck passages in the graph that any path has to path through. Such bottlenecks make the graph structure rather linear and thus less interesting for information extraction. We call theory morphisms that map a symbol to a different expression *intrusive morphisms*.

Finally, the quality criterion evaluates the graph's concrete suitability for searching. This assessment builds on data produced by the Bourbaker prototype's search functionality (Section 6.4). It indexes the mathematical formulas in their theories and in the transitive closure by translations along theory morphisms. Hence, an index is effectively a flattened version of the theory graph. The size of this index serves as an indicator of the complexity of search results.

Compatibility

While the quality criterion only considered the static properties of the resulting theory graph, the compatibility criterion determines how useful the graph is for application in a dynamic mathematical environment. In order to be useful, a theory graph must be extensible by new concepts and integrate with theory graphs from other sources.

Therefore, the criterion first focuses on the stability of the graph under the insertion of false links. Such a theory morphism, when constructed with little care, might make the theory and all its descendants inconsistent. As we do not have any way of assessing the consistency of natural language statements inside theories, we use the graph's topological stability to determine the robustness of the graph. By topological stability we mean the probability that a randomly inserted edge creates a cycle in the definitional theory graph. As we require the definitional theory graph to be acyclic, this would be a bad event. We approximate the graph's topological stability by repeating the experiment of inserting a random edge for one thousand times.

Furthermore, the graph should make it easy to tie in additional theories such as new results or foundations for toolbox-theories. Generally, this depends on how

[2]the average out-degree of the graph is the vertices' average number of outgoing edges

[3]a leaf is a vertex without outgoing edges

[4]a separation point is a vertex that disconnects the graph when removed

well the logical structuring of the theories is done. Inserting new theories is easiest when the theory graph is finely granulated, so that there is a high chance that it already contains the necessary theories to import. Likewise, fine granularity increases the probability that we find a suitable target for a postulated theory morphism from our theory. As a measure of granularity, we use the average number of new symbols defined in each theory. New symbols are those that are not in the image of any of the theory's imports. Symbols that are not new are just created for naming convenience. We assume, somewhat arbitrarily, an average of one new symbol per node to be an average granularity.

Finally, it should be possible to combine the constructed theory graph with other such graphs that cover the same area of mathematics. This a hard test which assesses how well chapters from this text would fit into the framework of a general purpose theory graph. As we do not have large theory graphs available, we combine the graphs with a small test graph of theories that were constructed from Wikipedia (cf. Section 3.1). The test graph is shown in Figure 4.3. To make for a realistic example, the graph contains both parts that were also covered in the translation of the algebra books (theories *semigroup*, *monoid*, *ring*) and parts that the books did not cover.

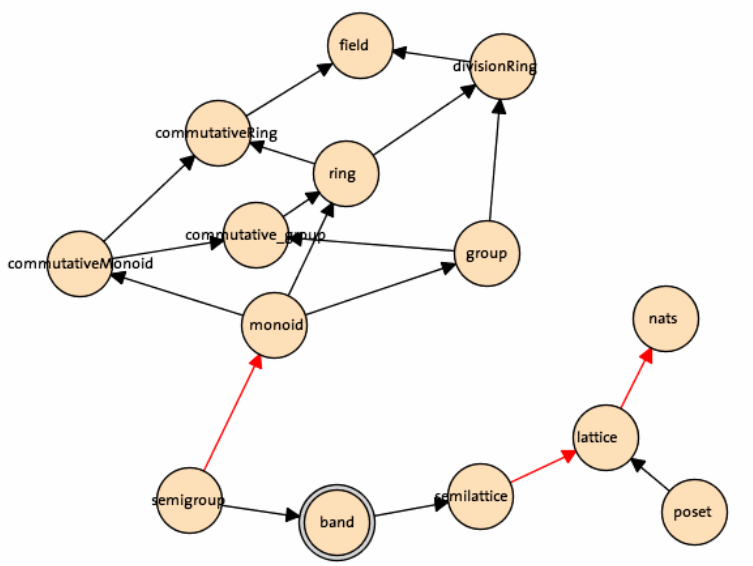

Figure 4.3: The test theory graph

4.3 Evaluation of Bourbaki's transformation

A number of things were remarkable during the translation process, even though not surprising. In hindsight, Bourbaki

- rigidly builds up theories;
- creates many stubs (i.e., theories that are not imported by any other theory);

- states propositions and theorems as soon as they can be proven.

The last finding deserves special attention. Let us call it the principle of immediate harvesting of results. This principle is less common in mathematics than one may think. Often times, theorems are stated in theories that are most commonly used. Stating these theorems in a more general theory is often possible, but it might require more work to prove it and its formalization might be less intuitive. While this behavior benefits the intuition of mathematicians, it does not necessarily suit the purpose of theory graphs. In theory graphs, we call the set of theories that a statement propagates to its *cone of impact*. Figure 4.4 displays the situation of stronger and weaker versions of theorems.

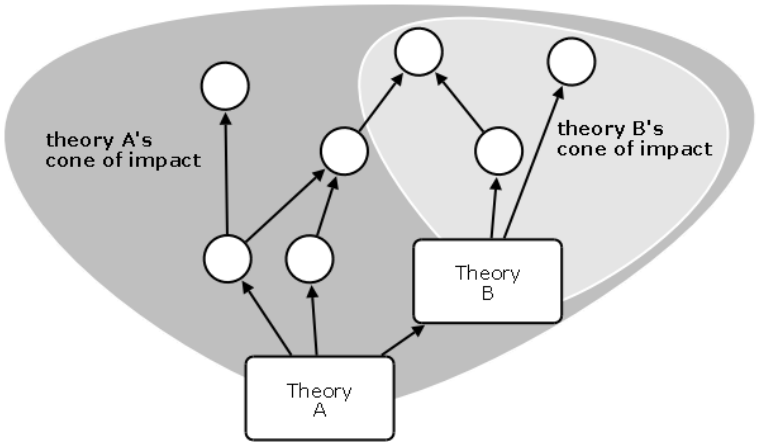

Figure 4.4: Cones of impact

As theory B imports A, its cone of impact is completely contained in A's. If our theorem is stated in A, it propagates to many more theories inside A's cone of impact. Thus, it is always most desirable to incorporate the strongest result possible and then obtain its more intuitive versions as translations into the respective theories.

Let us now have a look at the criteria:

4.3.1 Clarity assessment

Especially in the beginning, constructing the theories that faithfully correspond to the text was easy. Quotation 12 shows a progression of three definitions that are particularly nice to formalize. In all cases, exactly one theory is defined which imports from one other theory and has one additional axiom. Also, the respective axioms are stated rather unmistakably.

But these simple examples are not to say that theory extraction is a simple task in Bourbaki. The text is very dense and only statements that seem to receive special value by the authors are marked as definitions, propositions, or theorems. The absence of explanatory statements has the effect that also the text in between such statements is completely filled with on-the-fly definitions and propositions.

"DEFINITION 1. Let E be a set. A mapping f of $E \times E$ into E is called a law of composition on E. [...]"

"DEFINITION 5.A law of composition $(x, y) \mapsto x \top y$ on a set E is called associative if, for all elements x, y, z in E,

$$(x \top y) \top z = x \top (y \top z). \text{ [...]}$$

DEFINITION 8. A law of composition on a set E is called commutative if any two elements of E commute under this law. [...]"

Quotation 12: Bourbaki Algebra I [Bou74, p.1,4,7]

Quotation 13 shows an example of a short paragraph that contains both definitions and a proposition which is not even proven.

"The intersection of a family of stable subsets of E is stable; in particular there exists a smallest stable subset A of E containing a given subset X; it is said to be *generated* by X and X is called a *generating system* of A or a *generating set* of A."

Quotation 13: Bourbaki Algebra I [Bou74, p.6]

The stated definitions are also used frequently in subsequent passages. In this particular case, the proposition would become part of the theory *stable subset*, while the definition of a generating set prompts the creation of another theory. In general, Bourbaki does not limit the description of theories to definitions, and when new concepts are introduced, it usually mentions alternative names for them.

This does not create a problem for formalization but it raises the question whether all our theories are equally important. Bourbaki's authors apparently intended to place an emphasis on certain concepts and to de-emphasize others. The importance or centrality of an OMDoc theory cannot easily be determined from the document, and would make sense to give attention to these distinctions inside the theory graph.[9] EdNote(9)

Definition by ellipsis

While most theories can be formalized straight from the text with a potion of mathematical intuition, some statements resist their simple formalization. Quotation 14 shows a definition which makes casual use of a scheme to define distributivity for functions of arbitrary arities.

This is one example of the use of ellipses in Bourbaki, which are statements that use three dots "..." to signify continuation. If we stuck with the wording

[9] EDNOTE: implement this!

"Definition 5. Let $E_1, \ldots, E_n$ and F be sets and u a mapping of $E_1 \times \cdots \times E_n$ into F. Let $i \in [1, n]$. Suppose E_i and F are given the structures of magmas. u is said to be distributive relative to the index variable i if the partial mapping

$$x_i \mapsto u(a_1, \ldots, a_{i-1}, x_i, a_{i+1}, \ldots, a_n)$$

is a homomorphism of E_i into F for all fixed a_j in E_j and $j \neq i$."

Quotation 14: Bourbaki Algebra I [Bou74, p.27]

in the text, the sets E_i would have to be formalized as individual symbols to the theory. However, the number of symbols in a theory needs to be definite, therefore, this definition seems to exceed the expressiveness of the theory graph model.

Luckily, there is a work-around for this specific problem, which makes the number of sets $n \in \mathbb{N}$ a symbol to the theory and defines a totally ordered set that contains precisely the sets E_i. Note that this recovery strategy has none of the elegance of the original definition, even though it is logically correct. Generally, ellipses of all sorts may create problems for the expressiveness of theory graphs. The general treatment of ellipses goes beyond the scope of this thesis. The interested reader is referred to general articles about ellipses and logic as in [Cha87] or [vEF95].

The predicate-theory dilemma

In cases, it is hard to tell during the translation if a new theory should be created. Most of these cases are due to the definitions of properties where we cannot be sure if they will only be used in the context of the theory, or if their investigation makes sense in a separate theory. Consider the definition of *left invertible* in Quotation 15.

"Definition 6. Let E be a unital magma, $\top$ its law of composition, e its identity element and x and x' two elements of E. x' is called a left inverse of x if $x' \top x = e$. An element x of E is called left invertible if it has an inverse. A monoid all of whose elements are invertible is called a group."

Quotation 15: Adaptation from Bourbaki Algebra I [Bou74, p.15]

On the one hand, *left invertible* is a predicate that belongs to the theory *unital magma*. On the other hand, on the next page, an explicit statement is proven about left invertible elements (see Quotation 16).

Here, it looks as if statement is proven in the theory *punctured monoid*[5]. In the

[5] a *punctured monoid* is a monoid in which one element is specifically marked, i.e., one element of the base-set is made a symbol of the theory

"Proposition 3. Let E be a monoid and x an element of E. For x to be left invertible it is necessary and sufficient that the right translation by x be surjective."

Quotation 16: Adaptation from Bourbaki Algebra I [Bou74, p.16]

spirit of finely granulating the graph, such a theory should be created, however, it turns out that all later references to invertible elements only use the predicate in the parent theory. In particular, the theory *group* cannot be defined through imports from *punctured monoid*, as we want to define *group* without symbols for all the elements that the group's base set contains.[6]

This is bad because whenever the predicate *left invertible* is used from *unital magma*, there is no reference to the results proven about left invertible elements in the theory *punctured monoid*. But if there are a lot of results proven *punctured monoid* as well, then we do not want to bloat *unital magma* with these results about some defined predicate. So while proper referencing prompts us only to define the predicate, fine granularity requires the creation of another theory.

Let us call this situation the predicate-theory dilemma. This dilemma occurs quite frequently as mathematics often treats properties of objects also as objects themselves. A solution could be to define an internal concept of linkage that interrelates theories with their corresponding predicates. This is subject to future work.[10] EdNote(10)

Summary

The transformation of Bourbaki's Algebra I into linked theories is tedious as the text is very dense. It often requires the creation of several theories per page, which vouches for a fine granularity of the graph, but which makes the translation process slow.

As far as the clarity of the text is concerned, it is usually unambiguous and makes it easy to decide what the corresponding theory structure should look like. Even though Bourbaki does not explicitly structure the text in the same way as a theory graph, all the information for the transformation is included in the statements.

Many relations mathematical between concepts can be expressed in particularly nice ways in Bourbaki's theory graph. Still, we have encountered difficulties. The predicate-theory dilemma discussed in this section needs to be resolved on the side of the formal capabilities of theory graphs. The case of definition by ellipsis shows the flexibility of natural language and the limits of the theory-graphical method.

Theory imports are usually easy to identify in Bourbaki. Theories are not only built up in strict small steps, these steps are also sign-posted and tied back to the

[6]Even if we wanted to define *group* in this way, it would be impossible as we cannot have an indefinite amount of symbols in *group* or an infinite amount of imports. Compare this to "definition by ellipsis."

[10]EdNote: now I could add a couple of nice examples of the translation..

relevant concepts that were introduced earlier in the text. Thus, the translator always has an overview of the necessary imports for a new theory, and often times the text even gives explicit references to where these imported theories were first defined.

In most parts of Bourbaki, the text makes perfectly clear in what way theories are supposed to be defined and what other theories they import from. Under the application of the heuristics from Section 4.2.1, there are hardly any ambiguities what symbols and statements individual theories should contain. That said, definition by ellipsis and the predicate-theory dilemma are situations which do leave a choice how to deal with the situation. They allow different recovery strategies that might model the same expression but that lead to theory graphs which are not equivalent any more.

4.3.2 Quality assessment

Bourbaki Algebra I makes a lot of references to theories outside the text, but all these references are to theories that are defined in the first volume "Theory of Sets." They are always annotated with the exact place where in the first volume the corresponding statement can be found. At this point, Bourbaki's principle of immediate harvesting sometimes becomes a nuisance to the translator, because the text does not bother to redefine concepts such as sequences or mappings in this context, where they are now used. However, the symbols and definitions used in the first volume are guaranteed to match up perfectly with the way they are used in Algebra I. This would lead to a high degree of consistency within the theory graph if the whole of Bourbaki was formalized in this way.

Table 4.3 shows all the properties of the Bourbaki theory graph that was composed.

`theories`	51	graph height (longest path)	9
definitional `imports`	95	graph width (largest antichain)	22
postulated `inclusions`	12	average out-degree	2.1 (1.84)
`symbols`	82	leaves	19
`axioms`	38	number of separation points	9
`assertions`	27	topological stability	89%
`definitions`	17	granularity	0.76
intrusive `morphisms`	63	size of index	2833

Table 4.3: Properties of the Bourbaki theory graph

The terms in the right column were explained in Section 4.2.2. The number in brackets for average out-degree is the degree when duplicate edges are not counted. An example for duplicate edges is given by the theory *subset* that imports the *toolbox* theory twice, once for the superset and once for the subset. The resulting graph is connected. Table 4.3 also includes the 4 alien theories, *equivalence relation*, *mapping*, *subset*, and *set* (this is the name of the toolbox theory).

Let us first look at the sheer numbers on the left hand side of the table. There are more than twice as many theory morphisms as theories. Out of the total of 107 morphisms, 63 are intrusive, which means that they map a symbol to a different expression. Apparently, the graph is quite well-connected, and it is interesting as the many intrusive theory morphisms create many non-trivial translation paths. There are many more axioms than definitions as definitions have been cast as axioms inside theories where possible. For the same reason, not every theory has an axiom or a definition, since they have been replaced by appropriate theory morphisms where applicable.

What is striking when looking at the numbers on the right hand side of Table 4.3 is the low height in comparison to the width. The longest path in the definitional theory graph stretches over 9 vertices while the average path length is much lower. In order to cover the graph with paths, however, we would need 22 different paths. This shows how the text is covering the topics in depth[7]. It hints back at the principle of immediate harvesting that was observed in the beginning of Section 4.3. Generally, we would like to have more of a balance of height and width in order to obtain an interesting search graph. However, the graph's height is more a sign of the limited coverage of this case study. Given that the book abides by the principle of immediate harvesting, the height of the graph would subsequently increase throughout the chapter while the width would not grow accordingly.

The average out-degree of 2.1 shows how the graph is fanning out considerably. Even if we do not count duplicate edges then the degree is still quite high at 1.84. As a comparison, any directed tree has out-degree less than 1[8]. The high out-degree shows the great amount of theory-usage in the graph.

While the out-degree of the graph is very pleasant, the number of separation points seems to be quite high at 9. A look at the graph reveals though that most of these separation points occur right below technical theories that only import from one single theory. Such technical theories are necessary if, e.g., they import the same theory twice. These kinds of bottlenecks do not need to concern us as they are unlikely to be built upon in further development of the graph.

In total, there are 11 technical theories in the graph. These theories largely account for the 19 leaves in the graph. The rest of the leaves exemplifies the many theory stubs that would be connected to later on in the book.[11] EdNote(11)

Finally, let us focus on the graph's aptitude for search. The graph's great number of intrusive theory morphisms already suggested that the graph might be an interesting corpus for this application. Indeed, the index traces down over 2800 mathematical items in a theory graph with only 65 axioms and assertions. Of course, in a well-connected graph the size of its flattening grows exponentially. Still, 2800 is quite a high number given that no proofs and no examples were formalized on the translated 32 pages. In spite of its small size, the graph therefore promises to yield interesting search results.

In summary, Bourbaki Algebra I's theory graph makes a very good impression

[7] beware of confusion: in-depth coverage of topics leads to a wide theory graph

[8] This is because the number of edges in a tree is one less than the number of vertices.

[11] EdNote: possible other tests: maximum requation depth (probably around 3)

for our purpose. The well-connectedness of the graph, the many intrusive morphisms, and the size of the search index speak a clear language. A lack of balance in the graph should be rectified with a greater number of pages in the translation.

4.3.3 Compatibility assessment

As can be seen from Table 4.3, inserting a random edge into the definitional theory graph does not introduce a cycle in 89% of the cases. We think this makes it a fairly stable graph. For comparison, a full binary tree of depth 5 has a topological stability of 92%[9]. If a monkey was tampering with Bourbaki's theory graph, it would only destroy the graph structure in 1 out of 10 cases.

Let us have a look at the number for granularity in Table 4.3 from the preceding section. The value 0.76 means that on average, every theory inserts 0.76 new symbols that are not just renamed symbols from one of its ancestors. We believe that this is a decent granularity, but this statement only becomes meaningful in comparison to Hungerford's theory graph, which is discussed in Section 4.4.

The impression of fine granularity is affirmed when we combine Bourbaker's theory graph with the test graph. Admittedly, the test graph consists of fairly standard theories. They combine perfectly with Bourbaker's theories, though, so that we could connect four theories on each side as listed in Table 4.4.

Bourbaki theory graph		test graph
associative law of composition	$\longleftrightarrow$	*semigroup*
group	$\longleftrightarrow$	*group*
monoid	$\longleftrightarrow$	*monoid*
commutative monoid	$\longleftrightarrow$	*commutative monoid*

Table 4.4: Vertices connected between the Bourbaki graph and the test graph

The combined graph has 65 vertices and 110 definitional and 21 postulated theory morphisms, out of which 75 are intrusive. The topological stability of the combined graph remains high at 89%. This is interesting because it means that if we were constructing the test graph and we had made a couple of preliminary connections to background knowledge in the Bourbaki graph, the overall stability would not decrease. Judging from this small test, this seems to make the Bourbaki a good candidate for a database that could serve working mathematicians as an encyclopedic working theory graph.

On the downside, the number of separation points increased from 11 in the Bourbaki graph to 13 in the combined graph. Intuitively, when we extend a graph by additional knowledge, we would want the number of separation points rather to decrease than to increase. This hints at the requirement to integrate the two

[9]The topological stability of a binary graph of depth d is easily seen to be given by $1 - \frac{1}{(2^{d+1}-1)^2}\sum_{i=0}^{d}(i+1)2^i$.

theory graphs as closely as possible, which cannot be done properly with the rather small coverage of this case study.

4.4 A comparison to Hungerford's "Algebra"

As a direct comparison to Bourbaki's aptitude for theory-graphical formalization, we applied the same translation procedure to a more modern standard algebra textbook by Thomas W. Hungerford [Hun74]. The translation comprises the first four sections in the book's chapter on groups. Also, here we omitted the formalization of proofs, examples, and exercises. Covering 17 pages, this study is not as comprehensive as the Bourbaki study, but the results allow for a comparison of the two texts. Apart from groups, the chapter covers semigroups, monoids, cyclic groups and cosets under equivalence relations. Figure 4.5 shows the resulting theory graph.

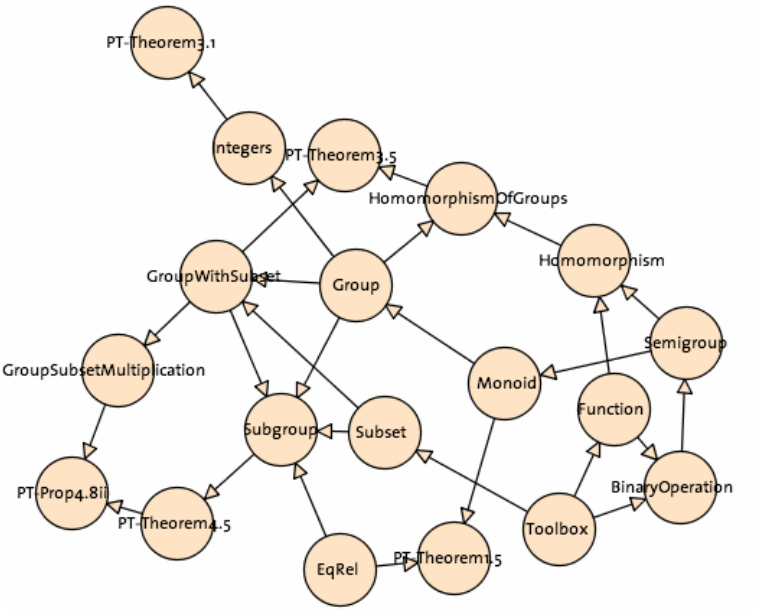

Figure 4.5: Hungerford's theory graph

The graph has 19 theories, 33 definitional and 5 postulated theory morphisms. 47 Symbols, 11 axioms, 26 assertions and 16 definitions were defined inside the theories. The formatted OMDoc file is a document of almost 40 printed pages, which is about half as much as the Bourbaki study's OMDoc file.

Let us first have a look at the clarity assessment for Hungerford's text.

4.4.1 Clarity assessment

The first impression of the chapter on groups is promising. Quotation 17 shows the first definition in that chapter.

"DEFINITION 1.1. A **semigroup** is a nonempty set G together with a binary operation on G which is

- associative: $a(bc) = (ab)c$ for all a, b, $c \in G$;

a **monoid** is a semigroup G which contains a

- (two-sided) identity element $e \in G$ such that $ae = ea = a$ for all $a \in G$.

A **group** is a monoid G such that

- for every $a \in G$ there exists a (two-sided) inverse element $a^{-1} \in G$ such that $a^{-1}a = aa^{-1} = e$.

"

Quotation 17: Excerpt from definition 1.1 in Hungerford "Algebra" [Hun74, p.24]

The way this definition is structured, it is basically a ready-made theory graph on paper. The three theories that are spelt out in bold font are given axioms as clearly marked separate items. Even though the axiomatic method is frequently employed in the field of algebra, this seems to support its great influence in the second half of the twentieth century as discussed in Section 4.1.4.

Imprecisions

However, the impression fades the deeper we get into the translation process. The first, and certainly still minor problem, are many small imprecisions that do not confuse a mathematics student, but that create some surprises for the translation process. As an example, consider the definition in Quotation 18.

"DEFINITION 2.1. Let G and H be semigroups. A function $f : G \to H$ is a **homomorphism** provided $f(ab) = f(a)f(b)$ for all $a, b \in G$."

Quotation 18: Excerpt from definition 2.1 in Hungerford "Algebra" [Hun74, p.30]

There are two different binary operations involved in this definition, namely that from G and that from H. The text does not distinguish the two, which we have to do in the theory that we create because otherwise the laws from our two imports are identified. This would in turn identify the two underlying sets of the laws, so if we are not careful and insert separate symbols for the two laws, then we get $G = H$ as the "faithful" translation of the text. It is to Bourbaki's credit that in there, such differences were always stressed and the translator could translate without vesting much understanding in the nature of the definition.

Another example along the same lines is given in Quotation 19.

"Definition 2.4. Let G be a group and H a nonempty subset that is closed under the product in G. If H is itself a group under the product in G, then H is said to be a **subgroup** of G."

Quotation 19: Excerpt from definition 2.4 in Hungerford "Algebra" [Hun74, p.31]

Again, no distinction is made between the law on G and the induced law on H. As in the example above, we have to draw this distinction if we do not want G and H to be identified by our imports. This distinction is made in the corresponding definition in Bourbaki.

These examples are not supposed to be biased towards Bourbaki's style of exposition. I myself believe that Hungerford's style is much more friendly to the human reader who can easily figure out the above technicalities as soon as she starts working with these objects. However, for the sake of a faithful translation of the text, these examples introduce a choice component that should be avoided in the translation process.

The predicate-theory dilemma revisited

The bigger problem with Hungerford's text is of a different nature. It is directly related to the predicate-theory dilemma that was discussed in Section 4.3.1. As it turns out, a good part of the text's definitions and theorems are stated in terms of predicates which can hardly be transformed into a plastic theory graph structure. Quotation 20 shows an early example of this.

"Proposition 1.3 Let G be a semigroup. Then G is a group if and only if for all $a, b \in G$ the equations $ax = b$ and $ya = b$ have solutions in G."

Quotation 20: Proposition 1.4 in Hungerford "Algebra" [Hun74, p.25]

If we stick to the text, this proposition calls for the definition of a predicate "group" inside the theory *semigroup*. As discussed before, this is not very desirable as it cuts the connection with the theory *group*. As the proposition essentially introduces a stronger axiomatization of *group*, we could either exchange the *group*'s axioms or we create a new theory with the proposition's axioms that is completely equivalent to *group*. Both meanings are not intended by the author so that the translator has to make yet another decision in the realm of the predicate-theory dilemma.

Similar problems arise with the concepts *subgroup*, *left* and *right congruence*, *monomorphism* and *isomorphism*. In most of these cases, characterizations of these theories are given after the initial definition that are smaller and therefore better suited for the axiomatization of the concept. The definitions that are initially given, however, have the tendency to be more intuitive and therefore better suited for the human reader.

As before, Hungerford's style is very understandable for student's of the subject matter who do not have a problem to relate predicates and their associated theories. But unless the predicate-theory dilemma can be solved in a satisfactory manner, the transformation of Hungerford's text into theory graphs remains problematic.

In summary, we see that Hungerford's text is exerted to deliver well-defined concepts that enable a fluent translation. However, small imprecisions afford more attention to detail on the translator's side and the predicate-theory dilemma is lurking behind many of his statements.

4.4.2 Quality assessment

Table 4.5 shows the symbols defined in the toolbox theory of Hungerford's theory graph. It is very similar to Bourbaki's toolbox with the difference that Hungerford frequently uses the natural numbers.

set	the concept of a set	.	multiplication on $\mathbb{N}$
in	set inclusion for elements	{ }	set constructor
x	Cartesian product of sets	0	zero element in $\mathbb{N}$
=	generic equality sign	intersection	set intersection
N	the set of natural numbers $\mathbb{N}$	lessthan	the natural order $\leq$ on $\mathbb{N}$
+	addition on $\mathbb{N}$		

Table 4.5: The symbols of Hungerford's toolbox theory.

Hungerford's Algebra makes frequent reference to alien theories. Many of these theories are shortly treated in the introductory chapter "Prerequisites and Preliminaries." However, as already mentioned the text makes frequent reference to natural number and integer arithmetic which is nowhere defined. Instead, Hungerford appeals to the reader's intuition, which is certainly justified in the scope of his book. If we were to insert an underlying theory graph in place of the toolbox theory we would have to make a tedious verification that our integer arithmetic is consistent with the way Hungerford employs it.

Table 4.6 shows the properties of the Hungerford theory graph with the values from Bourbaki as a comparison.

The table includes the five alien theories *subset*, *function*, *integers*, *equivalence relation*, and *toolbox*. As before, the number in brackets for the average out-degree is the average degree when we do not count duplicate edges.

When comparing the values in the table we have to keep in mind that the Hungerford study covers only about half the amount of pages as the Bourbaki study. Still the data reveals some interesting trends. On some figures the Hungerford graph clearly falls behind the Bourbaki graph, while on others the two are conspicuously close together.

Hungerford's number of theories, axioms, and intrusive theory morphisms are considerably less than half of Bourbaki's values. At the same time, the number

	Hungerford	Bourbaki
pages translated	17	32
`theories`	19	51
definitional `imports`	31	95
postulated `inclusions`	6	12
`symbols`	47	83
`axioms`	11	38
`assertions`	24	27
`definitions`	16	17
intrusive `morphisms`	23	63
graph height (longest path)	9	9
graph width (largest antichain)	5	22
average out-degree	2 (1.53)	2.1 (1.84)
leaves	4	19
number of separation points	0	9
topological stability	70%	89%
granularity	1.63	0.76
size of index	1494	2833

Table 4.6: Properties of the Hungerford theory graph

of symbols in Hungerford's theories exceeds half of Bourbaki's symbols while the numbers are fairly equal for assertions and definitions. These figures appear to be a direct consequence of the above discussion that Hungerford makes more frequent use of predicates instead of defining theories. The use of predicates leads to the introduction of symbols, definitions, and assertions, while corresponding definitions of theories would employ theories, morphisms, and axioms. The fact that Hungerford's graph employs more definitions and assertions than axioms and that it contains relatively few intrusive theory morphisms lead us to conclude that the graph is not as well-connected as Bourbaki's graph.

The most prominent difference between the two graphs is the ratio of width to height. Both of the graphs have the same diameter, but the width of Hungerford's graph is strikingly low in comparison to Bourbaki. In a way, Hungerford's graph is more balanced than Bourbaki. However, its lack of width is an indicator of a not as deep coverage, although it is important to notice that the graph would be wider if more of its predicates could be formalized as theories. It seems, though, as if Hungerford rather builds a theory tower, while Bourbaki is building a theory platform.

In line with this observation is that the translation of Hungerford does not create a great amount of theory stubs as Bourbaki does. The three leaves that the graph creates are surprisingly few, which consolidates the impression of a theory tower. Given the low number of leaves, the number of separation points is also lower in Hungerford's graph. Notice, however, that Hungerford's graph will still

become wider in subsequent chapters when different kinds of groups are treated.

Lastly, let us have a look at the sizes of the search indexes. Here, Hungerford seems to be absolutely competitive. Under the assumption that the size of the index should grow exponentially[12] for well-connected graphs, Hungerford's index even seems to be sensationally large. This holds true especially when we realize that Hungerford's translation contains 35 assertions and axioms which is only slightly more than half of the number that Bourbaki has. EdNote(12)

At a closer look, Hungerford's theories contain 67 mathematical objects indexed inside axioms and assertions while Bourbaki has 153 such objects. The apparent edge that Hungerford has over Bourbaki is due to equal height of the graphs and the overall better balance of Hungerford's graph.

4.4.3 Compatibility Assessment

A direct comparison of topological stabilities reveals a gap between Bourbaki's and Hungerford's graphs. The value of 70% that Hungerford achieves is substantially less than Bourbaki's 90%. Notice, however, that we cannot directly compare the two values because the topological stability of a graph depends both on its size and on its type. When we consider a graph that is formed of two identical copies of the Hungerford graph, the topological stability increases to 85%. This is still less than Bourbaki's value which indicates that Hungerford's graph is really less stable than Bourbaki's.

The result on topological stability is consistent with the observation that Hungerford's graph is more reminiscent of a tower, which accounts for the lack of stability. In Section 4.3.3, we compared Bourbaki's stability to a binary tree. For trees the topological stability tends to 1 as the graph grows. This is different for linear graphs, whose topological stability tends to $\frac{1}{2}$ as the graph size tends to ∞[10]. For a linear graph of length 9, the diameter of Hungerford's graph[13], the topological stability is 44%. Apparently, the graph's stability is right in between these two models. EdNote(13)

The largest difference we find in the granularities of the two graphs. Hungerford's value of 1.63 is more than twice the number of Bourbaki (0.76). This means that on average, every one of Hungerford's theories defines one more new symbol than every Bourbaki theory. Again, a smaller graph has more symbols because it needs to introduce many of them. But even if the Hungerford graph was twice as large without defining any additional symbol, it would still be coarser grained than the Bourbaki graph.

The lack of topological stability and its coarse granularity render Hungerford's graph rather unfit for dynamic applications. At least with our test graph, Hungerford's graph can be integrated at three points. In both graphs, the theory nodes *semigroup*, *monoid*, and *group* match up. This is one less node than we could match up with Bourbaki (see Table 4.4). Interestingly, the combined graph has a high

[12] EdNote: note: for paths, this number would grow linearly...

[10] The topological stability of a linear graph of length n is given by $1 - \frac{n+1}{2n}$.

[13] EdNote: consider introducing abbreviations for these graphs, like H and B

stability rate, namely 84%, which is higher than either Hungerford's (70%) or the test graph's (76%).

4.4.4 Combining Bourbaki and Hungerford

After discussing Bourbaki's and Hungerford's theory graphs separately, we shall now try to integrate the two graphs. Figure 4.6 shows the resulting theory graph.

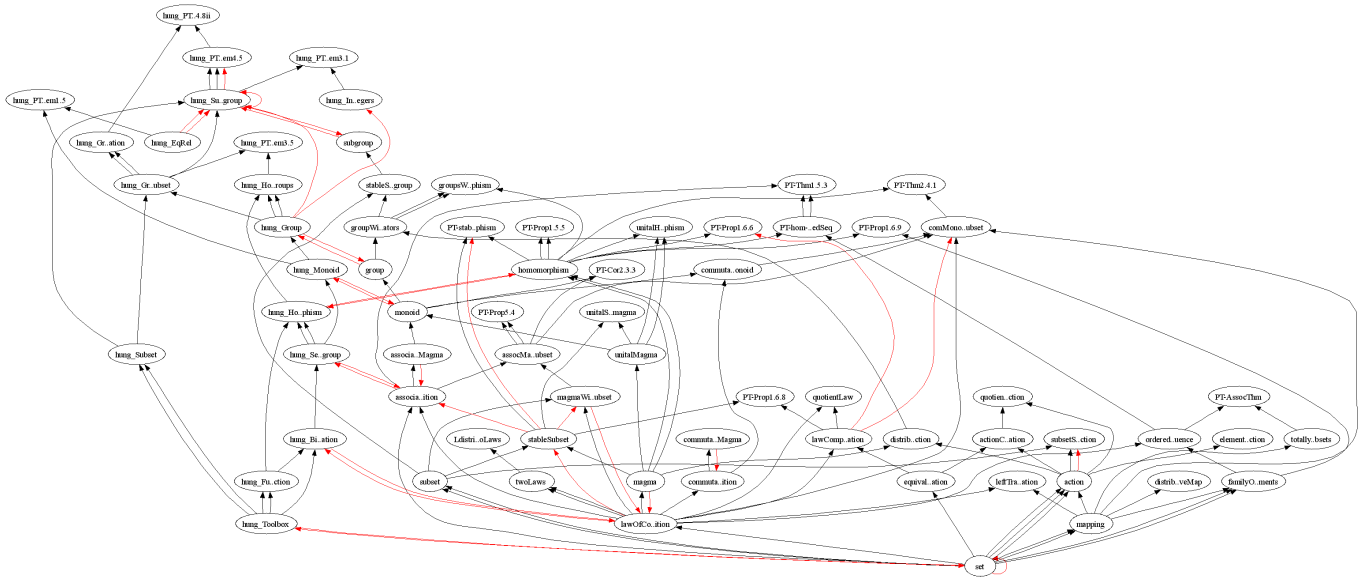

Figure 4.6: Bourbaki's and Hungerford's theory graphs combined

Observe the doubly red connections that sew the two theory graphs together. In total, we connected seven pairs of theories in this way. Table 4.7 lists the theories that have been identified in the combined graph.

Bourbaki		Hungerford
subgroup	$\longleftrightarrow$	subgroup
group	$\longleftrightarrow$	group
monoid	$\longleftrightarrow$	monoid
associative law	$\longleftrightarrow$	semigroup
homomorphism	$\longleftrightarrow$	homomorphism
law of composition	$\longleftrightarrow$	binary operation
set	$\longleftrightarrow$	toolbox

Table 4.7: Identified theories in the combined theory graph

We owe it to the great standardization that we can combine the two theory graphs without much effort. The identified theories are logically equivalent.

According to all our criteria, the resulting theory graph is of very high quality. Out of a total of 158 theory morphisms, 100 are intrusive, which promises rich

search results. The combined graph has 8 separation points which is less than the Bourbaki graph alone. The graph is fairly balanced with a width of 24 and a height of 15. The topological stability remains high at 86%, while the granularity is fine with a value of 0.8. The full table of data for the combined graph can be found in Table A.1 in the Appendix.

The combination of the two theory graphs shows how the theory-graphical method can powerfully combine theories from different sources. It allows the conclusion that we could build up a truly large library with data from various sources without compromising the quality of linkage in the graph.

4.5 Conclusion of the case study

In this chapter we have shown that the formalization of an important part of mathematics is possible. In order to test the practicability of the approach, we chose Bourbaki's Algebra I as a source of standardized mathematical knowledge in natural language. For comparison, we selected Hungerford's Algebra as a standard textbook in the area of algebra.

In Section 4.2.1 we introduced five simple heuristics that served as a guide in the translation process. The heuristics were sufficient for the task except for the situations described in Sections 4.3.1 and 4.4.1. There were different sources to these problems. The case of definition by ellipsis exposed limits to the theory-graphical method which requires a fixed amount of symbols and imports in a way that mathematical vernacular can brush over. The predicate-theory dilemma shows a lack of semantic connection between theories in the graph and predicate symbols that speak about their owning theories. Finally, small imprecisions in the texts were pitfalls in the translation process that cannot be remedied by simple heuristics.

Imprecisions mostly occurred in Hungerford's text while Bourbaki seems very exerted to avoid any source of unclarity. The predicate-theory dilemma was in issue in both texts, however, it proved to be much more problematic in Hungerford's book.

In Section 4.2.2 we developed three criteria to benchmark our transformation results, *clarity* of the text, *quality* of the theory graph, and *compatibility* with other theory graphs. Even though both texts proved to be compatible with the translation procedure, Bourbaki assumes the leadership in all of these categories. Bourbaki's translation results in a very fine-grained theory graph that makes use of theory morphisms in most all definitions. Unlike Hungerford, Bourbaki follows the principle of immediate harvesting of results which leads to the construction of a "theory platform" that allows for great flexibility in extending the graph.

Finally, we combined Hungerford's and Bourbaki's theory graphs in Section 4.4.4. The result is a very stable, fine-grained, well balanced large theory graph that mends all the shortcomings of Hungerford's graph and improves the overall aptitude for our purpose. The combination of the graphs is an excellent display of the theory-graphical method's modular approach that allows for the flexible

combination of theories from various different sources.

Chapter 5

Theory graph tools

In Chapters 2 and 4 we introduced the notational basics for writing up theory graphs and showed that the content of general mathematical resources such as textbooks can be transformed into theory graphs. All this time we have been vague about what this transformation is actually good for. This is going to be rectified in this chapter.

Along with the case study, I developed a program that demonstrates various applications of theory graphs. I called it the Bourbaker suite prototype. In this chapter we want to introduce the two main applications of theory graphs that the Bourbaker prototype supports. The focus here is going to be on theoretical considerations and design choices that make these applications useful for mathematicians. The exact details of the implementation will be presented in Chapter 6.

The Bourbaker prototype can

- visualize theory graphs in an interactive manner (Section 5.1),
- perform intelligent search for mathematical formulas on the flattened version of the theory graph and display the search results in a meaningful way (Section 5.2), and
- offer the user to edit theory graphs with a graphical frontend that does not require her to write any XML syntax (Section 5.3).

We consider these program parts to be the basic functionality for theory graphs to be useful at all. The case study in Chapter 4 comprises almost 6000 lines of OMDoc syntax, which would have been very tedious to write by hand. Bourbaker's theory editor did not only save me from a sprained wrist and the destruction of certain keys on my keyboard, but it also kept track of theories and their content which is indispensable help for documents with more than 30 theories.

The visualization of the theory graph makes our approach tangible. The pictures in Chapter 4 show samples of static theory graphs that are a direct result of the semantic formalization of theories and their morphisms. However, we believe that visualization should not end there because theory graphs are nothing

static. They are rather dynamical objects whose visualization should be interactive in order to adapt to the user's purpose. We use the theory graph's structure explicitly as a dynamic map of mathematics that the working mathematician can locate himself on to find directions. The Bourbaker prototype implements one such interactive map. The ideas behind its implementation are presented in Section 5.1.

Finally, our theory graphs could just as well be printed on paper if we were not offering some mechanized bookkeeping to make their flattening manageable. Bourbaker implements search functionality that enables the user to retrieve formulas all over the graph. While implementations of semantic search already exist and Bourbaker is using one of them, searching on theory graphs needs to take into account the structural localization of search queries and the exponential explosion of data in the flattened theory graph. Section 5.2 contains the guiding principles for how Bourbaker is dealing with these issues.

5.1 Visualization of theory graphs

The case study in Chapter 4 has shown that theory graphs can become quite messy even when their vertex count is still rather small. However, their irregular structure also makes them interesting, and we would like to make this structural information available to mathematicians. A proper visualization of a theory graph can be used like a map that a mathematician locate theories on and study their connections with other points of interest.

Section 5.1.1 presents a number of guidelines that makes large theory graphs manageable. In Section 5.1.3 we describe the implementation of a number of these principles in Bourbaker's OrganicGraph visualization applet.

5.1.1 Principles in theory graph visualization

The greatest challenge in developing an interactive theory graph display is the possible large size of the graph. In order to be useful, theory graphs have to be large, so large that a simple planar projection of the graph becomes too complex to grasp.

Our approach is based on the idea that the user is usually not interested in the whole of the graph, but rather only in parts of it. Often times, a single theory is the *center of interest*. When paths in the graph are investigated, it is usually the source and the target theory that form the center of interest. All nodes that are far away from the center of interest can largely be neglected, which reduces the number of nodes that we have to consider.

In the following we discuss various ways how to specifically deal with the visualization in theory graphs.

Hiding and collapsing

Firstly, it is customary to hide nodes that are not of immediate interest. This can be done in two ways. Either, the graph is displayed as an *exploration graph*, with

fringes indicating edges to further vertices where the display cuts off the graph. Figure 5.1 exemplifies such an exploration graph.

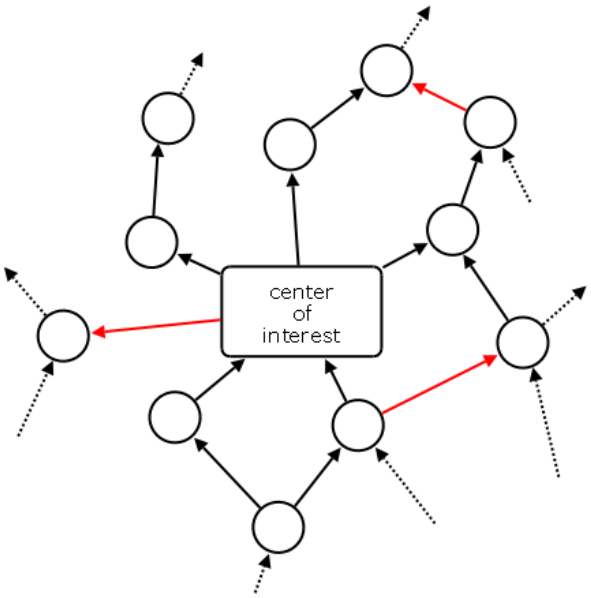

Figure 5.1: A sample exploration graph

Alternatively, collections of vertices can be collapsed into single vertices that represent the collection vertices of minor interest. The collapse of a group of vertices into one corresponds to forming a graph minor[1] of the original theory graph. Both of these approaches can be combined to clip the interesting parts of the theory chart.

Varying node size

A second way to illustrate large graphs is to vary the size in which the nodes are displayed. Nodes of interest may be inflated while uninteresting nodes are scaled down. In this way, vertices that are remote from the center of interest appear rather small.

Their size can be computed as a function of the distance to the center of interest, and depending on how fast this function decays, this method of display can be very similar to cut-off exploration graphs. Similarly, long strands of simple inheritance may suggest a great distance between theories, while in reality nothing interesting is happening. Along with technical theories, such nodes can be compressed in size so that they take up less of the presentation space. If we reduce the size of linear chains in this way, this procedure is reminiscent of forming a topological minor[2] of the theory graph.

[1] A minor X of a graph G is a graph that can be constructed from G by repeated deletion of edges and vertices and by contraction of sets of vertices into a single vertex. See e.g. [Die06] for details.

[2] A graph X is a topological minor of a graph G if we can obtain G by replacing edges in X by independent paths. For details, see e.g. [Die06]

Ordering vertices hierarchically

One last interesting way of displaying large theory graphs draws on the idea of collapsing nodes in the graph. The exploration of the graph starts with collapsing all nodes into one. Let us call such nodes supernodes in order to distinguish them from the original, indivisible theory nodes of the graph. Supernodes that contain just one node are identified with that node. Expanding a supernode is the division of the supernode into a group of supernodes, which themselves form a partition of the collection of nodes that the original supernode was standing for. In this way, the whole theory graph can be obtained by expanding all supernodes exhaustively. We call this interactive model of presenting a theory graph a *resolution graph*[3]. Figure 5.2 shows an example of the way resolution graphs work.

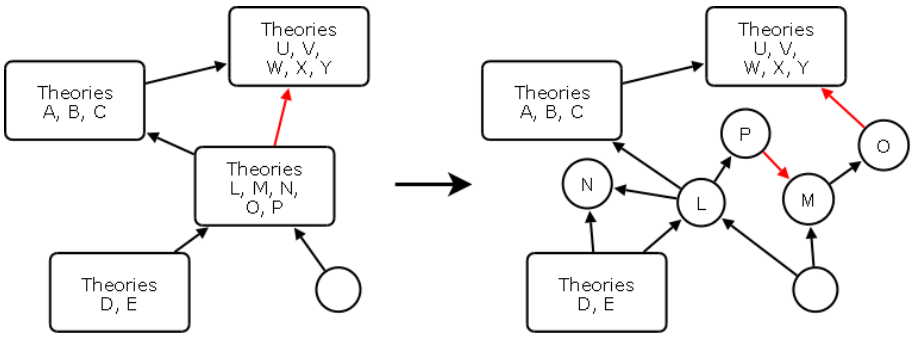

Figure 5.2: An example of a resolution graph

Resolution graphs allow for the intuitive modeling of fields, areas, and branches of mathematics and they enable the structural analysis of mathematical theories on various levels of specificity. This approach makes resolution graphs a supreme model of the way in which mathematicians view mathematics. However, its greatest advantage is also its biggest problem, because it is not clear at all in which way one should choose which theories to conglomerate.

At the macro level, this could be done using a manual division into established fields of mathematics, while some simple heuristics could associate theories at the micro level. But the division of the middle ground between coarse-grained fields and fine-grained sister theories remains elusive and would either require an enormous amount of manual classification or intelligent classification heuristics.

5.1.2 Using the graph visualization and tweaking the graph

The above visualization techniques are all helpful for navigating theory graphs. Which ones to apply greatly depends on what the graph is supposed to be used for. Several applications are conceivable.

At the micro level, one might want to investigate the immediate vicinity of a theory in order to find close relationships to other theories or to determine the

[3]The term "resolution graph" is also sometimes used for graphs that describe a certain kind of bifurcation in real dynamics.

properties of a specific theory morphism. When we are interested in the theory morphism paths that a specific statement marks between the source and the target theory, then the source and the target are the nodes of interest, possibly together with a selection of nodes along the paths. Section 5.2.3 introduces search variants that are trimmed or based at one theory. The interesting vertices are the base node and the satellite nodes which statements are imported from or exported to.

When visualizing all these situations in the graph, the emphasis is always placed on a selection of nodes that mark our center of interest. The nodes in the center of interest can be emphasized on by focusing on them, enlarging them, or cutting out or collapsing vertices that are remote from the center. Let us call a theory graph in which there is an emphasis on a small selection of special nodes a tweaked theory graph.

Non-tweaked theory graphs are useful too, for example, when we want to investigate the structure of whole areas of mathematics and the relations between these areas. In order to retain orientation in a vast theory graph, this visualization mode must be able to show only part of the graph. The graph should be movable, scalable, magnifiable, and rotatable. Ideally the graph itself can be rearranged in an interactive way, such as adjusting edge length or vertex positions.

5.1.3 Developing an interactive theory graph model

The Bourbaker suite implements a graph visualization applet that satisfies most of the considerations in the preceding section. The applet is developed separately under the name OrganicGraph and it can be used to visualize general graphs, even though it is optimized for displaying theory graphs. Figure 5.3 shows a screenshot of Bourbaki's theory graph that exemplifies some of OrganicGraph's features.

In the figure, we can see that *comMonoidWithSubset* is toggled in the bottom left corner. The picture might look messy in the static picture, but dynamic interaction with the graph enables display lets us investigate any connection we like.

OrganicGraph constructs an interactive and non-static directed two-dimensional graph projection with node focus for tweaked theory graphs and exponential size decay for nodes away from the center of interest. To achieve an optimal spread of theory nodes, vertices are modeled as repelling particles in a physical system and edges are transformed into springs that try to keep neighboring nodes a together at a certain distance against the forces between the particles.

Every node is labeled with the theory's name. As can be seen in Figure 5.3, OrganicGraph displays definitional theory morphisms as black edges and postulated inclusions in red. Nodes can be selected in order to stop them from moving. Multiple nodes can be selected by holding down the shift key. Vertices can be moved by dragging and dropping them in a different place in the graph. Thus, the graph can be manipulated dynamically.

After initialization, OrganicGraph gives a general overview over the theory graph. Double-clicking on a node causes the node to be toggled and the graph

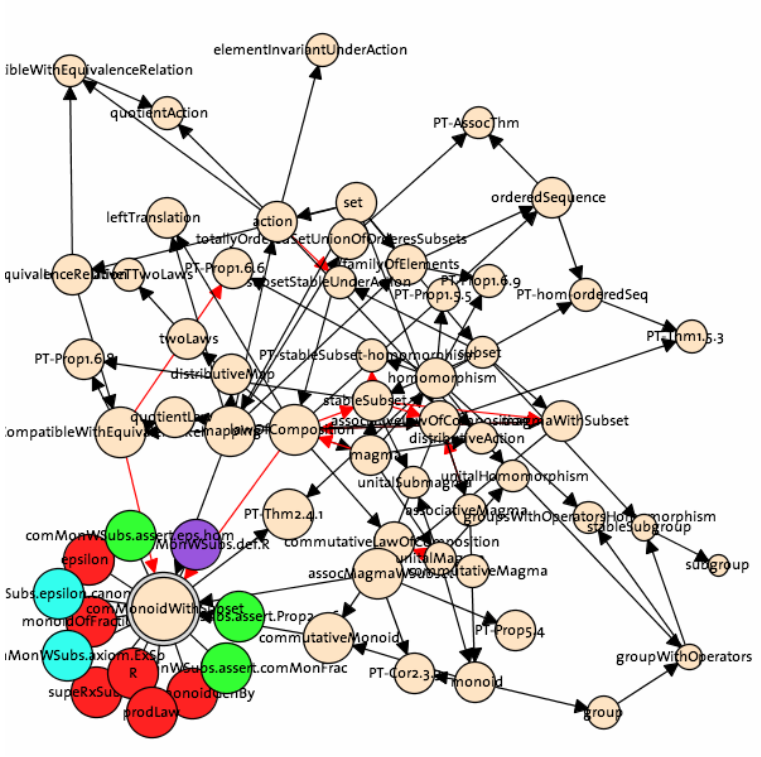

Figure 5.3: OrganicGraph displaying the Bourbaki theory graph

to become tweaked, enlarging the toggled node and scaling the size of the other nodes down exponentially with respect to the distance from the toggled node.

Toggled nodes are bear satellite nodes that correspond to the content of the theory. In Figure 5.3, these are the colorful nodes grouped around *comMonoidWithSubset*. Symbols are shown in red, axioms in blue, assertions in green, and definitions in purple.

The OrganicGraph interacts with Bourbaker's search functionality. Selecting theories or search results in the search window cause the corresponding theories to get toggled or morphism trails of statements to be marked in the graph.

5.2 Searching in the presence of theory morphisms

As mentioned in the introduction to this chapter, theory graphs are of not much use if we cannot efficiently retrieve its content. Bourbaker's search functionality is therefore one of its most important functions. The first hurdle on the way to efficient searching is making use of the semantic information that we have constructed for mathematical formulas in the course of the translation.

This task is already much more difficult to do than purely textual search, but fortunately there are already systems that can perform semantic search in mathematical formulas. The Bourbaker prototype is using Ioan Şucan's "MathWebSearch" (MWS) [KŞ06, KŞ07]. Section 5.2.1 will give a short description how

MWS works.

It is important to note that the MWS system does not support search queries that mix natural language and mathematical formulas. Stefan Anca has developed a search engine called "MaTeSearch" [Anc07] that combines traditional text search and the semantic search from MWS. MaTeSearch could be used in place of MWS if it offers a similar query interface.

MWS can easily be used to search for mathematical formulas in theory graphs. All that is needed is to write out the flattened theory graph into pairs of formulas and location expressions. MWS then does the indexing of the pairs and provides for their retrieval corresponding to search queries. Bourbaker then does the matching of search results with the saved location expressions and extracts the formulas' morphism trails from there.

In the following, this section discusses searching on theory graphs, especially the search for various statements in specified theories. It turns out that the simple retrieval of mathematical formulas from a theory graph is not very useful at all. But the plastic structure of the graph has a lot more to offer. If we focus the search on specific theories, a number of questions can be answered by using the search. Section 5.2.2 contains some examples how mathematicians can use the search in theory graphs.

In order to enable the user to focus her search only on theories that she is interested in, the Bourbaker prototype three specific kinds of search that limit the search space in the graph: *based search*, *trimmed search*, and *transitively trimmed search*. They all allow the focus on one single theory of interest at which we "tweak" the theory graph for the search. We call this way of searching the tweaked search paradigm and discuss it in Section 5.2.3.

Unlike standard search engines, the statement of search results in theory graphs have to refer back to the structure of the theory graph in order to guide the user through the extreme wealth of formulas in the flattened graph. The presentation of search results is elaborated in Section **??**

5.2.1 Semantic search for mathematical formulas

In standard text search, all the computer does is matching strings of characters. This principle is not powerful enough for mathematical formulas. Suppose we were searching for the expression

$$f(x) = x.$$

Here x is a variable, and we would like to treat it as such. For example, if our database contained the formula $f(z) = z$, text search would not be able to match the two expressions. But even if we augment text search by the concept of variables, such a search would find expressions like $f(x) = 1$, because the system does not necessarily understand that we would like the two x-variables to be the same.

Ioan Şucan's MathWebSearch addresses all these issues. It uses substitution tree indexing for fast and efficient retrieval of search results (see [KŞ06]). It supports queries in XML-syntax and a simplified string syntax for stating search requests. The XML syntax extends MathML and OpenMath for queries and will

not be treated here because it usually is too cumbersome for humans to write directly. Nonetheless, such queries are handed on by the Bourbaker search function and are thus supported.

The string query syntax is quite simple and can be used for direct queries. The "@"-modifier marks variables, and identical variables are treated as identical by the system. Function application is written in brackets. Our query from above would therefore read as $= (\mathbf{f}(@\mathbf{x}), @\mathbf{x})$ in the string syntax. Notice that for the search, we have to write formulas in prefix notation. If we wanted the function f to be treated as a variable as well, we would write $= (@\mathbf{f}(@\mathbf{x}), @\mathbf{x})$. Multiple arguments are separated by commas, so a search query might look like this: $= (\mathbf{f}(@\mathbf{x}, @\mathbf{y}), @\mathbf{x}(@\mathbf{y}))$.

Both the string syntax and the XML queries can be entered directly into Bourbaker's search window in order to start the search in the theory graph.

5.2.2 Using the search

The flattening of the theory graph quickly leads to a vast number of indexed formulas even for graphs of moderate size. Bourbaki's theory graph with just 51 vertices produced an index of 2800 entries. In this situation, a general search for a certain formula often leads to many irrelevant hits in theories that are unrelated to the information that the user wanted to get.

As an example, consider the query $= (@\mathbf{op}(@\mathbf{x}, @\mathbf{op}(@\mathbf{y}, @\mathbf{z})), @\mathbf{op}(@\mathbf{op}(@\mathbf{x}, @\mathbf{y}), @\mathbf{z}))$. This equation represents the law of associativity, and as such it will create a hit in the theory of *semigroups*, where it is originally stated, and all the theories that import *semigroup*, for example, *monoids*, *groups*, *rings*, *fields*, the *reals*, the *complex numbers*, and so on. If the user was interested in whether or not the law of associativity holds for matrix multiplication, this vast number of search results would become cumbersome to review. In order to reduce the number of rather redundant hits, it is makes sense to give the user control where in the theory graph to look for results.

Let us consider a few motivating examples that should clarify what searching in theory graphs can be used for. As in the example above, the user might want to verify that a certain statement holds in a given theory. She may then just enter the respective statement into the search mask and receive a quick affirmative answer in case the statement holds.

The user might also want to investigate a specific statement in a given theory, and how it translates under morphisms of the theory. This can be used to refute conjectures in a fast way, because the conjectured statement may lead to a contradiction with a statement in one of the theory's descendants.

More generally, the user might want to know how the axiom of commutativity reads after translation to another theory. Or what theory the division operation that can be done in fields comes from. Or she might want to find all statements about connectedness in a *topological space*. In all these cases, the user can design search queries to the theory graph that contain the answer to her question, and by focusing on her theory of interest she can narrow down the results to what she

is interested in.

5.2.3 The tweaked search paradigm

The examples in the preceding section have in common that they focus their search only on part of the theory graph. Obviously, knowing where to look for answers is a great advantage for producing relevant search hits, which is why the Bourbaker prototype makes use of such information. It allows the user to *trim*, *base*, or *transitively trim* her search at a theory in the graph. Figure 5.4 shows a screenshot of Bourbaker's search panel that offers the different search options.

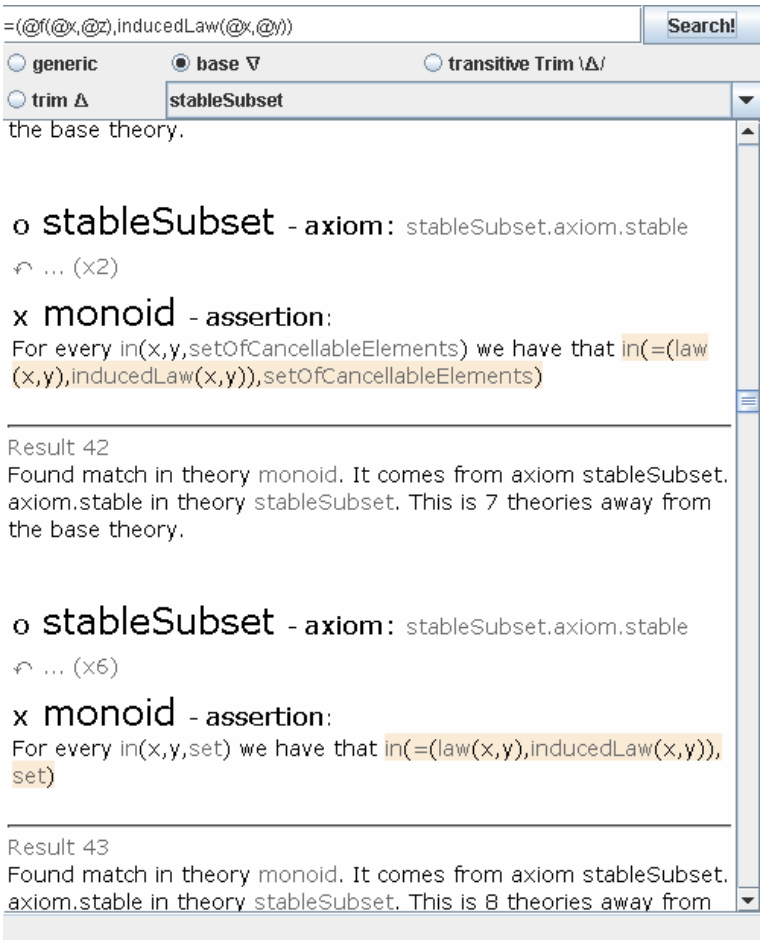

Figure 5.4: A screenshot of a Bourbaker theory graph search

Trimming the search means that only results within the specified theory are shown, which limits the search space to that theory and all its ancestors. We call this search space the trimmed cone of the search. This is useful for searching all the valid assertions in a theory of choice.

Basing the search at a theory is the converse operation that only shows search results that originate from that theory, so the search space is limited to the theory and its descendants. In analogy to the trimmed cone we call the space of based

search the *based cone*. This enables the user to show the morphism paths that a statement is taking starting from a specific theory.

Finally, transitively trimmed search is a generalization of trimmed search. It addresses the issue that often times, one is not only searching for assertions that are valid within the specific theory, but that closely derive from it. An example are technical theories that we have met plenty of in the case study in Chapter 4. These theories merely contain one statement relating two or more theories, but as such they cannot be stated in either one of the theories they import. Such statements we would also like to find when we search for among the ancestors of a theory in the theory graph. Transitively trimmed search therefore also finds statements that are descendants of theories in the trimmed search cone.

Notice, however, that we have to limit the distance of theories from the trimmed search cone in order to widen the search space not too much. Especially if there is a toolbox theory from which there is a path to every theory in the graph, transitively trimmed search would otherwise search the whole theory graph, and we would not have gained any control. The Bourbaker prototype therefore limits its transitively trimmed search to theories that are not farther than two imports away from the trimmed cone.

The Bourbaker project implements trimmed, based, and transitively trimmed search in a straight-forward way by suppressing search results that do not correspond to the specified search mode. Notice that this is certainly not the most efficient way of dealing with tweaked search queries. A scalable implementation of searching in theory graphs would try to exploit the explicit structure given by the theory graph in order to reduce search time. For this, MWS's term indexing procedure would have to be augmented or replaced by a classical graph search algorithm.

5.2.4 Presentation of search results

Unlike classical search results, the results of a query to a theory graph need greater care in presentation than merely stating the matched instance.

For example, the query $@\mathbf{op}(@\mathbf{x}, @\mathbf{op}(@\mathbf{x}, @\mathbf{y}))$ might locate the statement $\mathrm{lcm}(x, \gcd(x, y) = x$ in the theory of natural numbers. Without further explanation, it is unclear where this statement comes from and why its truth should be believed at all. This might either be a theorem in the natural numbers which has an attached proof or it might be the inclusion of the absorption law for lattices[14], which relates to lcm and gcd by a postulated theory-inclusion. Stating context information of how the search engine arrived at this match is therefore very important. EdNote(14)

The context information is generated from the morphism path and from the information contained in its theories and morphism elements. Figure 5.4 in the preceding section shows the search results of a based search. Figure 5.5 shows another example of a transitively trimmed search together with its visualization in Bourbaki's theory graph.

[14] EdNote: if this example was used before, reference it here!

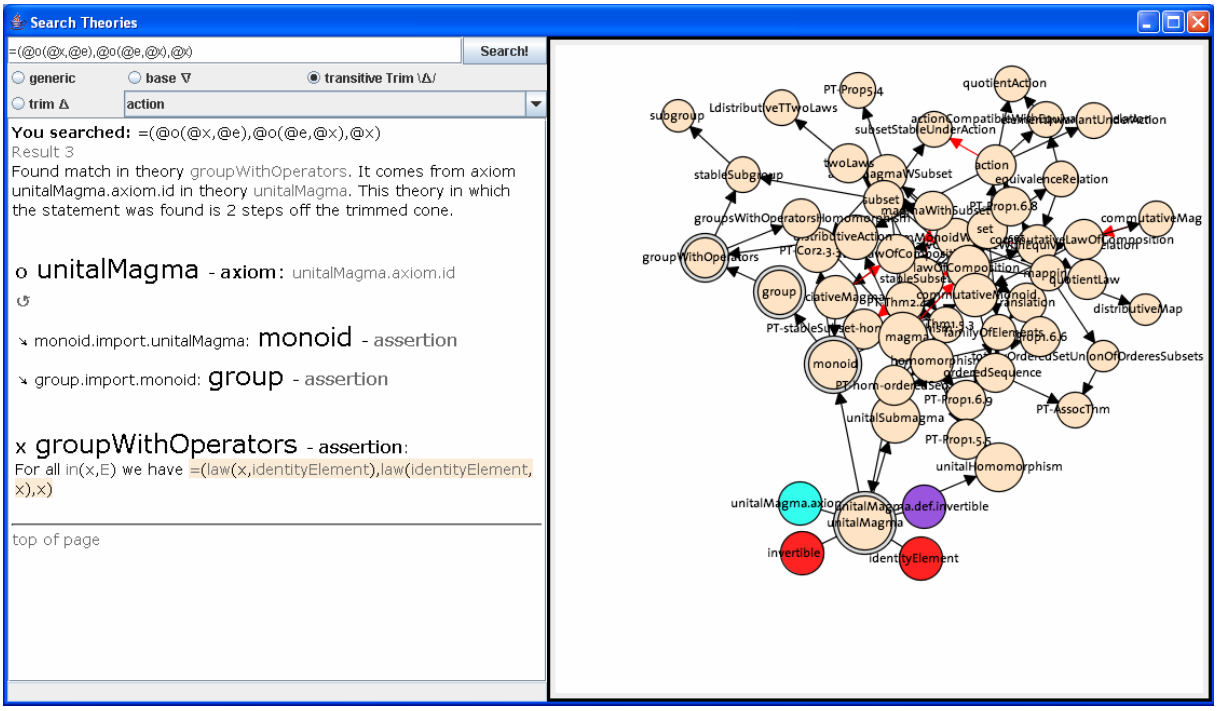

Figure 5.5: An example of a transitively trimmed search with the theory graph illustration attached

The left side of the figure shows the presentation of one search result for the query $= (@\mathbf{o}(@\mathbf{x}, @\mathbf{e}), @\mathbf{o}(@\mathbf{e}, @\mathbf{x}), @\mathbf{x})$.

Hit explanation

The first thing that is stated is in what theory the formula was found, and where it originally comes from. In our example we have found a hit in an axiom of *unital magma*, which is translated through three morphisms into the theory *group with operators*.

As we had specified to use transitively trimmed search, the hit explanation also states how far away *group with operators* is from the trimmed cone. If we had used based or trimmed search instead, the explanation would state how far away the hit is from the theory we based or trimmed our search at.

The morphism trail

The second part of the search result presentation is the statement of the morphism trail that the formula has taken. In our example, it was translated from *unital magma* to *monoid* to *group* before getting to the target theory *group with operators*.

In the beginning, the morphism trail is collapsed and only the source and the target theory are shown (see Figure 5.4). A click on the $\rightsquigarrow$-symbol expands the morphism trail and a click on the $\circlearrowleft$-symbol collapses it again.

The statement that the matched formula was found in is written out in its

translated version underneath the target theory. The formula in which the match occurred is highlighted with bisque background color. By clicking on the links at the individual steps of the morphism trail, the translated version of the statement at every individual step can be shown. In this way the exact trail of the formula can be scrutinized.

Of course, we have to be careful with the translated versions of the axiom. As mentioned in Chapter 2, the translation of formulas inside natural language statements is only possible under certain conditions. The original statement can be retrieved by clicking on the source theory, and from there on the user can assess if the translations make sense.

Graphical visualization

In addition to the textual presentation of search results, Bourbaker also highlights the involved theories in the theory graph. In our example, we toggled the source theory by clicking on it in the hit explanation on the left hand side. The OrganicGraph additionally marks the nodes in the morphism trail to emphasize it in the graph.

5.2.5 Towards shallow proof assistance

The search functionality presented above is a structural added value service. We can search no matter what the content of the theory graph is. But we could do more with theory graphs if we prepared them in the right way.

The key to more functionality in the graph is more semantic annotation of theory nodes. Given the capabilities of computers today, this requires formal logical languages. The logical markup does not have to be visible to humans, but humans can greatly benefit from it. This approach links back to the theory graph as a bulletin board.

Consider the example of where we want to introduce a new theory to the theory graph. Instead of manually selecting theories as imports and targets for theory inclusions, automated inclusion detection mechanisms could provide for a quick integration of the theory if the theories in question are logically specified. The same morphism detection mechanism could also increase the connectivity in the graph on a constant basis. Immanuel Normann is researching formula matching algorithms for this purpose in his PhD thesis [NK07].

Suggesting appropriate theory morphisms is one instance of what we call *shallow proof assistance*. Without committing to the details of the system, the user can get rudimentary functionality from automated reasoning. Notice that for the above example to work, the user would still have to express his results in a formal language. Her formalization efforts would be locally limited to her theory, though, and no familiarity with the rest of the corpus is required.

Additional shallow proof assistance tasks can include decidable queries to the graph whether certain statements hold in a given theory or basic interactive proof search tools as they already exist today for some of the formal languages.

5.3 Editing theory graphs

Writing OMDoc syntax by hand is possible in principle, but in practice it is very tedious. While its XML-tags are understandable, they are not primarily made for humans to read so that OMDoc source text is not very appealing to the eye.

This would not matter if we were using the OMDoc tags only for occasional structuring commands. However, the theory graphs that we marked up in Chapter 4 make constant use of XML-tags. What is more, the markup of theory morphisms and symbols makes constant reference to all the theories in the document so that scrolling through the document to find the proper terms easily takes up most of the time once the theory graph has grown a little. And scrolling around in a 3000 line OMDoc document to find a specific reference in several possible theories amounts to finding a needle in a stack of hay.

There are essentially two ways to remedy this problem. The first one would be an extension of the text editor by what is known as "tab-completion." While the translator still has to write proper OMDoc syntax, the editor would assist her by automatically completing references to theories and their symbols automatically after the first couple of characters have been typed. This takes care of avoiding broken links through typos and, if designed in a proper way, it can also offer help simple help on what symbol is contained in what theory.

For the translation tasks here, I have chosen a second approach to editing theory graphs. The Bourbaker suite implements a graphical theory editor that does not require the translator to write any OMDoc syntax at all. Instead, all elements in the document are organized in lists that can be edited using the mouse. Figure 5.6 shows a screenshot of Bourbaker's theory editor.

The *list* in the top left corner shows the current theories in the graph. When we choose one of the *theories*, the other lists in the top half of the window are filled with the respective elements inside that theory. Then clicking on an element in any of these lists causes the bottom half of the editor window to change into a panel for editing the chosen element. In the figure, the last *assertion* in the list is currently edited.

The *edit panel for assertions* that we see in Figure 5.6 has a list on the left side which contains text passages and OpenMath elements in their proper order. Text is inserted in the text field in the middle, while OpenMath elements are modified using the *tree structure* on the right. Within the OpenMath tree, the type of nodes can be changed by choosing the appropriate type from the right click context menu. If we choose "*symbol*" as the type of a node, we can set its *theory* from a list of the currently available theories. Once the theory is set, the specific *symbol* in that theory can be chosen from a second list.

The edit panels for *definitions* and *axioms* look similar to the one in our example. The edit panels for *theories* and *symbols* only allow the modification of their names. The edit panel for *imports* is more complicated as it allows for the addition and specification of *symbol mappings*.

The *edit menu* at the top of the window offers the addition of elements to any of the six lists. It also offers an option to write the current theory structure out to

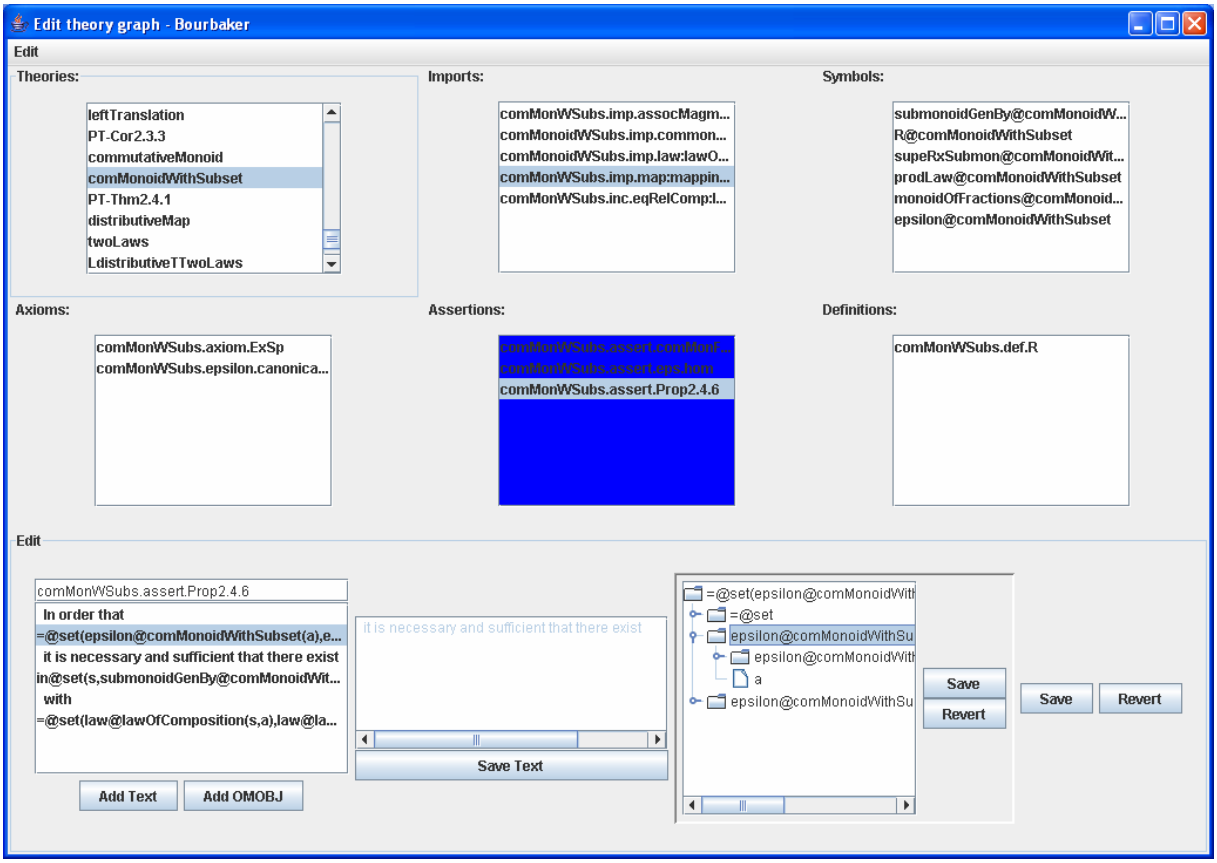

Figure 5.6: A screenshot of Bourbaker's theory editor

a proper OMDoc document. This document can then be saved from Bourbaker's main window.

In practice, the theory editor has proved to be of great help in the translation of Bourbaki's Algebra I and Hungerford's Algebra. With a little bit of practice it is easy to achieve a workflow that would be impossible to mimic in a text editor. The complete Hungerford translation that was presented in Section 4.4 took only about eight hours using the graphical editor. For comparison, the text-based markup of the first ten pages of Bourbaki took a cumulative of about 20 hours for the members of the Bourbaki II group[4].

Even after ours of editing the theory graph, we can be assured that the resulting OMDoc syntax will be valid in the end. And the bookkeeping of theories and their symbols is of great value once the theory graph is bigger than 20 theories. This shows that the theory editor is an indispensable tool for the creation of theory graphs.

[4]The Bourbaki II group used sTeX for the markup of Bourbaki whose syntax is slightly simpler to write than OMDoc.

Chapter 6

The prototype

The Bourbaker suite was implemented as a prototype to demonstrate the use of theory graphs. Its architecture can be seen in Figure 6.1. The Bourbaker suite takes care of handling in- and output to OMDoc documents, building an internal memory representation of the theory graph, communicating with an instance of the MWS server over a socket, and it offers interfaces for editing theories and their imports, searching theory graphs, and displaying graphs dynamically.

6.1 Development

The Bourbaker suite is written in Java and was implemented using Eclipse SDK[1] version 3.2.1 with the Java SE Development Kit 6[2]. The graphical user interface is using the standard Java Swing components. Parsing and writing of OMDoc files is done using the JDOM 1.0 API[3], which relies on a standard SAX parser for parsing. XPath-functionality for querying XML files is provided by the Jaxen 1.1.1 API[4]. Visualization in the OrganicGraph package uses the Processing library[5] together with the Traer.Physics package[6].

6.2 The main classes

At the time when the coding of the Bourbaker suite was started, it was not clear what parts and what functionality it would eventually comprise. It therefore received a rather open architecture based on managing classes that offer standard functionality and runtime information on their area of operations. These confluence classes are called OMDocManager, TheoryManager, and SearchManager. On top comes the graphical user interface for convenient user interac-

[1] http://www.eclipse.org/
[2] http://java.sun.com/
[3] http://www.jdom.org/
[4] http://jaxen.org/
[5] http://processing.org/
[6] http://www.cs.princeton.edu/~traer/physics/

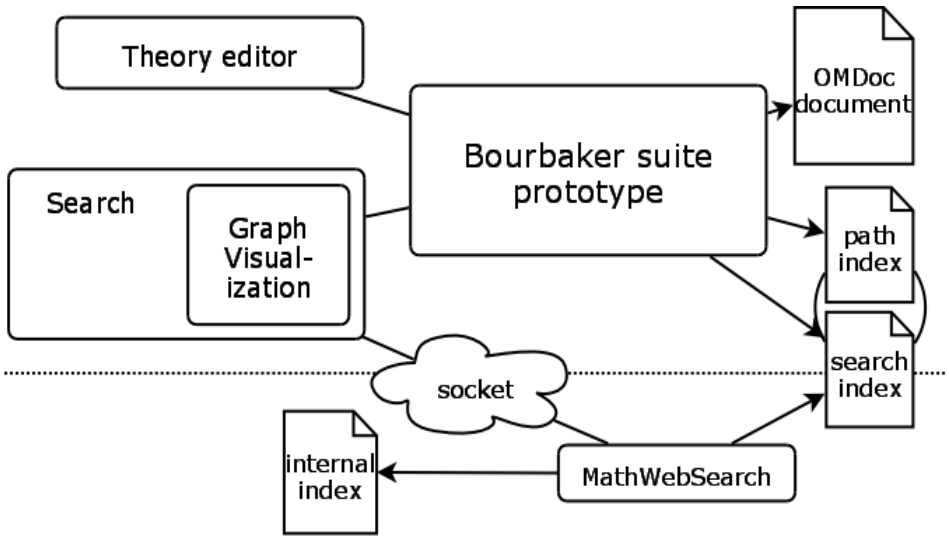

Figure 6.1: Architecture of the Bourbaker suite

tion in the class MainWindow. The description of the individual classes can be found in Chapter B in the Appendix.

Three main areas of the Bourbaker suite have emerged which roughly match up with the three managers: the internal theory graph representation classes, the theory editing tool, and the search functionality. Figure 6.2 is showing a logical class diagram of the main classes and the three areas. The class diagram of the OrganicGraph visualization tool is shown in Figure 6.3. OrganicGraph is the main feature of Bourbaker's graph visualization, but it was developed as a separate applet. It will be discussed in Section 6.6.

The Bourbaker suite is unifying two approaches to handling OMDoc theory graphs. On the one hand, the suite builds an in-memory representation of the theory graph in order to allow for dynamic editing and display of the graph structure. However, theory graphs only exhibit their real power when they are large, and large theory graphs are hardly manageable in main memory in a dynamic way. Consequently, the search functionality does not use the in-memory representation of the theory graph but works directly with XPath expressions that only require the local parsing of files when queried. Therefore, Bourbaker's search scales to theory graphs of truly interesting proportions. When operating on such large graphs, the in-memory graph representation and its visualization would have to be clipped to a fraction of the graph that is small enough to handle.

6.3 The memory representation of the theory graph

The in-memory presentation of OMDoc elements enables editing and visualizing the underlying theory graph structure. For the Bourbaker suite, the OMDoc-specific element nodes that are needed were more or less faithfully mapped to

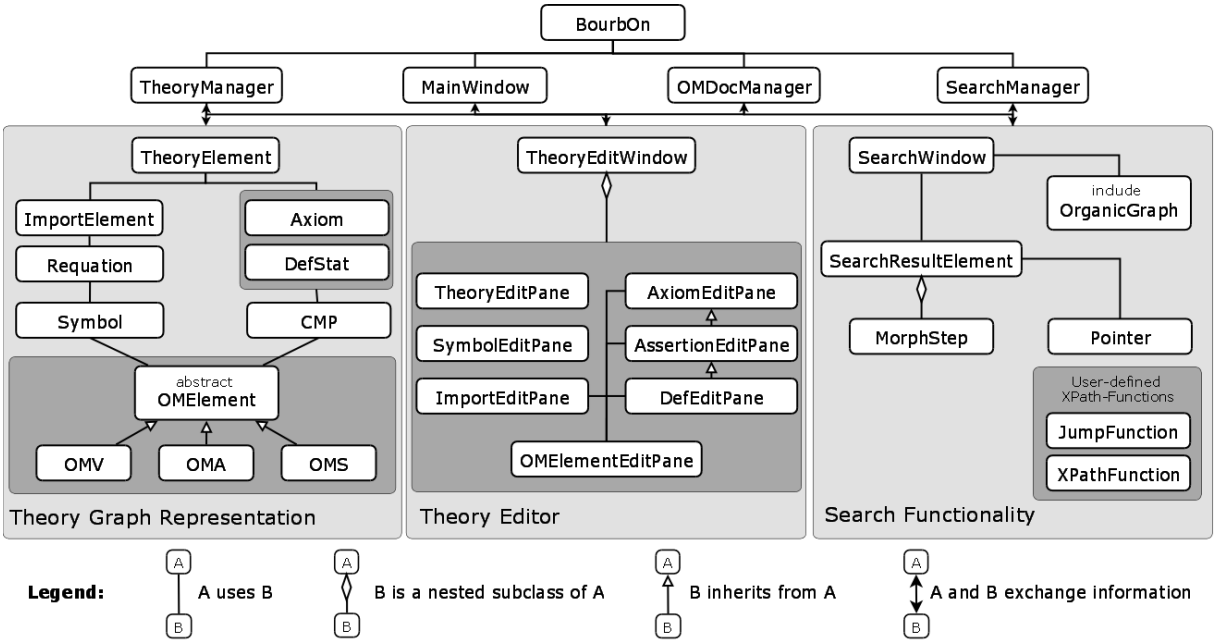

Figure 6.2: Logical class diagram of the Bourbaker suite

classes with corresponding names.

The classes were constructed as smart agent classes rather than mere container classes. They construct their own representation when given a JDOM XML node of their corresponding type, and they recursively call the respective class constructors for child nodes. For the representation of mathematical formulas in OMDoc, the Bourbaker suite supports OpenMath content, whose nested structure is represented by the classes that inherit from OMElement.

6.4 The search functionality

This area of the Bourbaker suite implements its search capabilities. Upon entering a query, the SearchManager class establishes a socket connection to the MWS server and retrieves the returned search results from the document structure. As mentioned before, Bourbaker's search does not make use of the in-memory theory graph so that it scales to large graphs. Instead, it uses the XPath XML addressing language to reference element nodes in OMDoc documents directly.

XPath is recommended by the W3C (see [CD99]). Consider the following query in the XPath language:

$$//\mathbf{def} : \mathbf{theory}[@\mathbf{xml} : \mathbf{id} =' \mathbf{set}']/\mathbf{def} : \mathbf{definition}[\mathbf{1}]/\mathbf{descendant} :: \mathbf{om} : \mathbf{OMOBJ}[\mathbf{3}]$$

The XPath expression points to elements in the XML document. In our example, it specifies all the `OMOBJ`-elements that are the third descendant of the first `definition` in `theories` with the name "set." The tags "xml," "def," and "om" are shorthands for namespaces. The text inside rectangular brackets restricts the

selection of elements. The statement $@\mathbf{xml} : \mathbf{id} =' \mathbf{set}'$ specifies that the attribute "id" has to exist and be equal to the value "set" while the numbers 1 and 3 refer to the position of the specified elements in document order.

Generally, XPath expressions select a whole node-set of elements that match the XPath expression. These node sets can contain multiple elements or be empty depending on whether the expression generates hits inside the XML document. The Bourbaker prototype always uses XPath expressions for addressing unique document nodes, which is well supported by the structure of the generated OMDoc documents.

There are alternatives to using simple XPath for querying XML documents. XPointer[7] is an extension of XPath that specifically aims at extracting uniquely specified element nodes. XQuery[8] is more powerful than XPath in that it has a more general query syntax. For our purpose here, the power of XPointer and XQuery were not needed.

Bourbaker introduces two function extensions to the XPath language in order to suite the handling of theory graphs. Table 6.1 shows the XPath-signatures of the two functions.

Function:	*node-set*	**jump**(*node-set*)
Function:	*string*	**xpath**(*node-set*)

Table 6.1: Signatures of the jump and xpath functions

Firstly, XPath does not natively support referencing into other XML documents or into other parts of the same document that are independent of the currently matched nodes. The jump-function solves the problem. It takes as its argument an XML attribute node that has a URL as its value. The function is defined to return the XML node-set which is specified by that URL. The jump-function enables XPath expressions to jump from point to point in several XML documents before retrieving the result set of elements. This is necessary for the dynamic referencing of virtual elements in flattened theories. It also frees XPath queries from the confinement to only browsing up or down the XML element tree.

Secondly, XPath does not come with a standard function to construct an XPath expression that uniquely points to a given XML node. This is solved by the xpath-function in a custom way that relies on the grouping of the OMDoc document into top-level theories. Notice that Table 6.1 states the function's return type as *string* since the XPath specification does not contain string-sets. In fact, Jaxen allows for an implementation that returns a whole list of XPath-strings that correspond one-to-one to the input node-set. In this sense, Bourbaker's implementation of the xpath-function is nonstandard.

[7] http://www.w3.org/TR/xptr/
[8] http://www.w3.org/XML/Query/

6.5 The theory editor

Bourbaker's theory editor is a tool to ease the creation of OMDoc theory graphs and theory content. It is implemented as a graphical user interface in the class TheoryEditWindow that offers to manipulate the in-memory theory graph directly. For this reason, the editor is in constant communication with the TheoryManager. Using the theory editor has many advantages over writing OMDoc by hand in an editor. It

- provides a structured overview over theories and the symbols, imports, axioms, assertions, and definitions they contain;
- renders remembering the specific names and requirements for XML elements and attributes superfluous;
- gives direct access to the theories' symbols in all context menus, thus insuring the absence of broken references in the resulting document.

TheoryEditWindow offers the option to write the theory structure into a memory representation of an OMDoc document, which can then be saved from the MainWindow. The theory editor is implemented as a single window with multiple edit panels that offer text fields, lists, and hierarchical trees for modifying the content of theories. See also Section 5.3 for the layout of the theory editor and Chapter B in the Appendix for the class description of the theory editor.

6.6 The OrganicGraph applet

The OrganicGraph applet was implemented using the Processing graphing library[9]. OrganicGraph's main class extends Processing's PApplet which makes it an applet. Figure 6.3 shows the class diagram of the OrganicGraph applet.

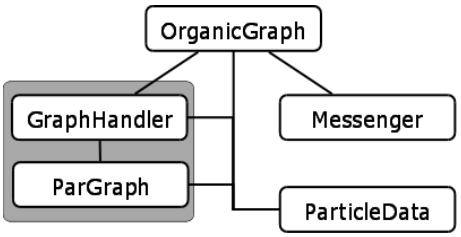

Figure 6.3: The OrganicGraph class diagram

[9]http://www.processing.org/

Chapter 7

Future Work

The Bourbaki case study and the Bourbaker prototype are merely a first step towards a completely new approach to offering advanced added-value functionality to mathematicians. They show that casting mathematics in the form of theory graphs is both *possible* and *desirable*.

This chapter first considers a few immediate steps that should be taken in order to improve the Bourbaker suite. Most of these suggestions are well-defined projects on the basis of this work that aim at integrating it with current developments in the OMDoc language. This discussion is contained in Section 7.1.

In the second part of this chapter, we paint the big picture of where we believe the Bourbaker approach should be going. This blueprint is not a ready-made roadmap but all our suggestions draw on well-established ideas. We motivate why we think Bourbaker is going this way and why we believe it is possible. The construction of the big picture is done in Section 7.2.

7.1 Immediate steps

Virtually all parts of the Bourbaker prototype can be improved on in functionality and user-friendliness. In particular, the theory editor is a rather preliminary implementation. Given the importance of such a tool, a development into a strong standardized program would make sense. Furthermore, OrganicGraph's graph visualization does not implement all of the visualization principles that were mentioned in Section 5.1.

An improved version of the Bourbaker prototype should also be able to handle presentations of mathematical formulas that are more intuitive than OpenMath's prefix notation. OMDoc already offers so-called presentation elements that form the necessary basis for implementing such an improvement.

Apart from the straight-forward improvement of the Bourbaker suite, there are several new developments around the OMDoc language whose implementation would greatly benefit the performance and the architecture of the prototype. First of all, a new *OMDoc version (1.8)* is being developed that will simplify many structural issues. Then, the implementation of an OMDoc API called *JOMDoc* has

recently been completed. Finally, the KWARC[15] group is developing an database called *OMBase* that specifically handles OMDoc content. We discuss these items in reversed order. EdNote(15)

7.1.1 OMBase

As the Bourbaker suite was planned only as a proof-of-concept prototype, little care was taken of an efficient implementation. While the in-memory representation of our theory graphs is unproblematic, the retrieval of search results occasionally takes unacceptably long. This is not MWS' fault, which is delivering the search results extremely fast. However, the search results then have to be parsed from the OMDoc documents using XPath queries which requires constant read access to the file.

Obviously, such a task would better be implemented using a database that can process the queries faster than the file system. The OMBase system that is currently in planning within the KWARC group is such a database that is optimized for handling OMDoc content. OMBase would be particularly useful if it was able to handle the new OMDoc version, which is discussed in Section 7.1.3.

7.1.2 JOMDoc

During this summer, Kristina Sojakova has implemented a Java API[1] for OMDoc called JOMDoc. The JOMDoc API maps OMDoc elements to faithful representations as class instances in Java which offer convenience functions for manipulating OMDoc elements and their contents. JOMDoc parses OMDoc files, checks if they are valid OMDoc, and it writes back the in-memory structure to the document on the hard drive.

Essentially, JOMDoc replaces all of Bourbaker's in-memory representation classes. Integrating JOMDoc as the basic interface between OMDoc and the algorithms of the TheoryManager (see Chapter 6) would greatly improve Bourbaker's architecture. Firstly, Bourbaker does not take too much care of validating the OMDoc document. If the document is not valid, Bourbaker behaves in undefined ways. Secondly, the JOMDoc API encapsulates document-related functionality that is right now scattered between the various classes of the in-memory representation and the TheoryManager.

As an immediate consequence, the JOMDoc API makes it possible to divide up the Bourbaker suite into several independent parts. On the one hand, Bourbaker's theory editor can be developed independently with just just JOMDoc as its backbone. On the other hand, the OrganicGraph theory graph visualization can be directly built on top of the JOMDoc API.

[15] EdNote: introduce these guys beforehand

[1] application programming interface

7.1.3 OMDoc 1.8

OMDoc is not a final specification of how to write semantically annotated mathematical content but it evolves in order to better suit its users needs. Version 1.8 will be an extension of the version 1.2 which offers several improvements over the current version.

The first novelty will be `assertion` elements that specify their own imports. They are placed outside all theories on the top level of the document. Imports for `assertions` are a good model of context definitions in mathematical statements such as "Let E be a group." This new kind of element is a perfect generalization of the technical theories that we have used to model theorems that relate several theories.

The greater innovation of OMDoc 1.8 will be a logical addressing language into OMDoc documents. Paths in this language are not strictly bound to the XML structure of the document any more so that they effectively replace the use of XPath in Bourbaker's search. There is no complete specification of the syntax yet, but queries to the document may then look something like this:

$$//\textbf{theory} : \textbf{group}/\textbf{import} : \textbf{monoid}/\textbf{axiom} : \textbf{associativity}$$

This query would specify the translated version of the *monoid*'s associativity axiom inside *group*. This syntax is clearly much easier to use than the generic XPath queries Bourbaker is using.

The new logical paths are particularly interesting in the context of an integration with the OMBase system. If OMBase was capable of handling such queries, Bourbaker's search functionality would become a very streamlined application.

Along the same lines, Bourbaker would profit from an integration of the new paths into the JOMDoc API. Such an implementation is only envisioned at the moment. The JUMP and XPATH-functions that we have introduced in Section 5.2 might serve as yardsticks for the backend implementation of such a logical path language.

7.2 Beyond the prototype

We are convinced that the theory-graphical method is better than both digital libraries such as PlanetMath and formal verification languages such as Mizar. Unlike digital libraries, our theory graph structures offer rich information about the underlying mathematics. This work has shown how the structural information opens the door for a wealth of useful applications. And unlike formal libraries, theory graphs are very close in spirit to the way that mathematicians are already working today.

In this section, we therefore propose the establishment of a central theory graph resource for mathematics. Such a resource would replace digital libraries as we find them today and it would act as a *bulletin board* for the integration of formal systems, such as those presented in Section 3.2.

Section 7.2.1 motivates the building of theory graphs as a community effort given the necessity of a large corpus for truly interesting applications. It also addresses why our approach is more promising than past formalization efforts like Mizar or IMPS. Section 7.2.2 discusses how theory graphs and their applications revolutionize digital libraries. We show that Bourbaker is suitable for the use as a dynamic community-based tool. We describe how the theory graph system benefits from community contributions and how the community profits at the same time. Finally, we claim in Section 7.2.3 that theory graphs have the truly unique potential to change the way in which mathematicians work. Instead of a large collection of small journals, the theory graph could be a place for mathematicians to publish, relieving them of requirements such as the re-statement of already known results.

7.2.1 Building a large theory graph

If Bourbaker's theory graph is supposed to be useful, it has to grow immensely. Not only does it have to be big, it also has to at least cover whole branches of mathematics in order to become interesting for mathematicians. It is clear that such a formalization effort is impossible to realize by one person in the style in which our case study has been conducted.

It would therefore be desirable if we could recruit a large user community that is willing to contribute to the build-up of the theory graph. At that point, we have a hen-and-egg problem. Why would anybody contribute to a theory graph that is still too small to be useful? We are aware that this hurdle is quite a high one. Even the size of Mizar's database, which after all, contains 40000 theorems, is too small a corpus to be relevant to the majority of mathematicians. Notice that even PlanetMath has only about 7000 articles (according to Wikipedia). A user community of the size of PlanetMath's would be sufficient, though, to develop a large theory graph that covers many parts of mathematics adequately.

We claim that Bourbaker can also attract a large user community, possibly even larger than those of current digital libraries. Notice first that the contribution to theory graphs is far easier than the contribution to a formal library. Our theory graphs do not require contributors to cast their results in specific formal languages. In fact, we do not even require that their results are necessarily true. The handling of such content will be discussed in Section 7.2.2.

Given the right tools, contributing to theory graphs is just as easy as writing a journal article. In the next two sections we even claim that it is easier. But when the work is done for a journal article, it can be integrated into a theory graph without great effort. Being aware of the local axiomatization and imports of one's theories is something that is necessary for mathematicians anyways. The rest is ordinary copy and paste.

This shows that with theory graphs it makes sense to not only build a reference library for college students but to involve top-of-the-line research directly. The approach to building a theory graph should therefore rest on two pillars.

On the one hand, formalization efforts such as Bourbaki's can offer a working

basis that demonstrates the power of the graph and motivates mathematicians to contribute. In the same vein, the theory graph must serve as a digital library that forms only a slightly more powerful resource than the established libraries. The theory graph model is easy to understand and to contribute to, so there is no reason why it should not be possible in the same way as for PlanetMath and MathWorld. At least in the case of PlanetMath, content might be transferred to start up the graph system.

On the other hand, the system should be open for current research in order to insure that Bourbaker can cover top research. Gaining contributions from researchers is certainly the harder nut to crack. But as discussed in Section 2.5, the theory graph model has things to offer here, too. For instance, for large projects it might serve as a status tool that keeps track of current developments. In particular, the public prominence of publishing on Bourbaker offer mathematicians the chance to be recognized for their work beyond the boundaries of their field. The theory graph serves as a map here which mathematicians can put their own theories and theorems on.

The start-up of the theory graph system is certainly the most critical phase in its operation. However, there are many immediate gains for a large group of people to contribute, which makes the theory graph model a very promising approach.

7.2.2 The next generation of digital libraries

Bourbaki's books were a collaborative effort, so building a reference corpus of mathematics based on the axiomatic method suggests itself. As we have seen in the previous section, the theory graph model is better suited for managing mathematics than digital libraries. Now we want to discuss how such a shared theory graph could evolve in a dynamical way.

First and foremost, this work has shown that our approach is *modular*. This means that theory graphs can integrate content from various different sources. The main problem with free contribution to the graph is the danger of ill-defined links. But similarly to digital libraries, the community that introduces bad content can also commit to purging such content from the system.

By keeping track of the popularity of theories, the system would contribute to standardizing the usage of terms in mathematics. The same holds for theory morphisms. Morphisms that are frequently used, for example, by the search functionality, are more popular. Such links can be trusted more than others because they are under constant surveillance by a large user group. Rankings on the theory graph can in turn be used to emphasize on theories and morphisms of higher quality. This could, for example, be used to sort search hits and in the visualization of the theory graph.

Using this mechanism, versions of theories, theorems, and morphisms evolve dynamically and incorrect content is quickly discarded. Notice that such a popularity ranking closely mirrors the way in which mathematics evolves as a whole. Branches of mathematics are established, are a hot topic for a while, yet, often decline again once it has become clear that they do not lead to further interesting

insights.

The same is done in community-based theory graph model. Thus we claim that the theory graph model does not only faithfully model mathematical content but that, in fact, it models the mathematical community itself.

This community consists of many small interest groups which form so-called "communities of practice." Modeling communities of practice is itself an area of research in mathematical knowledge management (see e.g. [KMM07]). Different communities of practice are interested in different branches of mathematics. The theory graph model accommodates different branches for all of them.

One such community of practice is the mechanized reasoning community. For them, the theory graph offers a fantastic prospect. A large central theory graph acts as a huge *bulletin board* which they can fill with formal versions of the respective theories. Automated reasoners can then reason in perfect analogy with mathematics as it is really practiced. The possibility of annotating the graph in this way is already present in the OMDoc format.

Theorem provers may then discover new theorems or new links in the graph and have them marked as such. The way from automated discovery to application in mathematics would thus become a lot shorter.

Additionally, automated reasoners may verify theorems and theory morphisms. Such graph entities could then be trusted much more than others, which gives automated reasoning a direct application in contemporary mathematics.

7.2.3 Changing the way mathematics works

Finally, the impact of establishing a central theory graph for mathematics might effectively change the way in which mathematics is working. Publishing on such a central graph might become more rewarding than submitting one's result to a journal because the results become immediately available to a large community of mathematicians.

Changing the workflow in mathematics was also the dream expressed in the QED Manifesto. Yet, unlike formal libraries, the theory graph model does not require changing mathematics altogether. The main change in our approach comes about by changing the publishing behavior.

A central theory graph is just as well suited for publication of mathematical results as are journals. The peer-review process is taken over by the user community and can be made anonymous if so desired. In place of journals, the graph provides the opportunity for news feeds to publish new results in specific branches of the theory graph.

Publishing in the graph also has advantages for the publishing mathematicians. Instead of repeating well-known results before presenting their own research, they merely have to tie in their results in the theory graph. Thus they can re-use content and concentrate on advancing research in their field.

The content of this section is certainly far from its implementation. Right now it is just a vision. But we believe that it is better a vision than the QED Manifesto because it is more suitable for contemporary mathematics and because it is more

realistically implementable. The works of mathematics have changed in the past and it is conceivable that they will continue changing in the future. At the same time, mathematics has been largely unaffected by the revolution of information technology. The content of this work may just be the first step towards a different kind of mathematics, one that makes work easier for mathematicians and releases unknown synergetic power.

Chapter 8

Conclusion

In this thesis we have presented a case study and a prototype for mapping mathematics using theory graphs. The case study has shown that a formalization of general mathematical content is *feasible*. The Bourbaker prototype demonstrates that such a formalization is *desirable* because it enables us to manage mathematical content in a novel way and offer many applications to aid working mathematicians.

Using Bourbaki's "Algebra I" and Hungerford's "Algebra" as samples allowed us to conclude that theory graphs are very close in spirit to contemporary mathematical texts. The case study also revealed differences in the suitability of different mathematical texts to be casted as theory graphs. In our study, Bourbaki carried over Hungerford in all the *criteria* that we used to assess our translation results. We managed to integrate Bourbaki's and Hungerford's theory graph without considerable effort, which demonstrates the *modularity* and *extensibility* of theory graphs.

As a proof of concept, the Bourbaker prototype implements searching on theory graphs. We developed the *tweaked search paradigm* in order to adapt semantic search to the structural conditions of the theory graph. The prototype's visualization of theory graphs makes these graphs *tangible*. It is making the inherent structure of our case study's theory graphs explicit. Thus, we arrived at *mapping* a small part of mathematics using theory graphs.

Our results pave the way for a broader application of theory graphs as a tool in mathematics. The theory graphs that we built act as a *bulletin board* by offering the possibility of annotation by many different parties. This thesis therefore reveals a promising prospect of theory graphs in the field of mathematical knowledge management.

Appendix A

Graph data

The numbers in brackets for average out-degrees are the average out-degrees without counting duplicate edges.[16] EdNote(16)

[16] EdNote: regularize tables all over with these numbers here

	Bourbaki	Hungerford	test graph
pages translated	32	17	-
`theories`	51	19	14
definitional `imports`	95	31	15
postulated `inclusions`	12	6	5
intrusive `morphisms`	63	23	8
`symbols`	82	47	18
`axioms`	38	11	17
`assertions`	27	24	3
`definitions`	17	16	2
graph height (longest path)	9	9	6
graph width (largest antichain)	22	5	4
average out-degree	2.1 (1.84)	1.95 (1.53)	1.43 (1.36)
leaves	19	4	2
number of separation points	9	0	5
topological stability	89%	70%	76%
granularity	0.76	1.63	0.64
size of index	2833	1494	-

	Bourbaki + test graph	Hungerford + test graph	Bourbaki + Hungerford
`theories`	65	33	70
definitional `imports`	110	48	126
postulated `inclusions`	21	16	32
intrusive `morphisms`	75	37	100
`symbols`	100	65	129
`axioms`	55	28	49
`assertions`	30	27	51
`definitions`	19	18	33
graph height (longest path)	9	10	16
graph width (largest antichain)	25	9	24
average out-degree	2.02 (1.8)	1.94 (1.7)	2.26 (1.94)
leaves	21	6	23
number of separation points	13	4	7
topological stability	89%	84%	86%
granularity	0.66	1.21	0.8

Table A.1: Properties of the case study's theory graph

Appendix B

The classes of the Bourbaker prototype

The main classes

BourbOn:
: This is the root class that contains the main method. It instantiates and keeps static references to TheoryManager, OMDocManager, SearchManager, and MainWindow.

OMDocManager:
: The class OMDocManager handles all input and output from and to OMDoc and text files. It retrieves the memory representation of the XML document structure and makes it available to the program. It stores the used namespaces for later reference and sets the context for the Jaxen engine to work. It also delegates XPath queries.

TheoryManager:
: The TheoryManager is responsible for the working memory representation of the theory graph. It constructs the internal theory representation structure from the provided XML elements and maintains the list of currently registered theories. It is also the starting point for the recursive creation of the search index and the back-transformation of the theory structure into OMDoc tags.

SearchManager:
: This class processes search queries and handles the connection to the MWS server over a socket connection. It composes the search result list in the form of SearchResultElements and filters it depending on the used search paradigm.

MainWindow:
: The class MainWindow provides for the central graphical user interface. It displays the current OMDoc document and offers links to the other windows SearchWindow and TheoryEditWindow. It also offers buttons for loading and saving OMDoc files.

The classes of the theory graph's in-memory representation

TheoryElement:
: For every `theory` in an OMDoc document, one instance of the class TheoryElement is created. It contains lists linking to the `symbol`s, `imports`, `axiom`s, `definition`s and `assertion`s that are contained in the theory. The TheoryElement also takes care of linking translated symbols in axioms, assertions, and definitions with the original symbols in their home theories.

ImportElement:
: The class ImportElement represents an `imports`-element from OMDoc. It maintains whether the import is global and postulated and it maintains a list of Requations that point to the translations under the given morphism.

Axiom:
: The Axiom class represents an OMDoc `axiom` of the theory. Its main content is an instance of a CMP.

DefStat:
: DefStat is similar to Axiom. It has an additional field for `definition`s to point to what symbol is being defined.

Requation:
: The Requation class keeps references to the source `symbol` and the term it is mapped to.

CMP:
: The CMP is a wrapper class for OMDoc `CMP`s and `FMP`s. Note that for this program, `CMP`s and `FMP`s do not need to be handled differently. It keeps its text and OpenMath content in its proper order and delegates translation requests to the OMElements it contains.

Symbol:
: Foremost, the Symbol class is a wrapper for the home `theory` and the `OMOBJ` representation of an OMDoc `symbol`. However, Symbols are more versatile than this and can be used to represent translated `symbol`s while simultaneously maintaining reference to the home `theory` of the `symbol`.

OMElement:
: An OMElement is an abstract class that serves to represents OpenMath tags. It provides abstract method functionality for the classes OMA, OMS, and OMV such as presentation formats for indexing, printing, and OMDoc, and it has an abstract stub for handling Requation elements.

OMA:
: OMA inherits from OMElement and implements its abstract methods. OMA represents OpenMath's function application tag. It keeps references to a function OpenMath object and a list of arguments of arbitrary size.

OMS:
: OMS inherits from OMElement and implements its abstract methods. It stands for an OpenMath symbol. It maintains the name of the `symbol` and the content dictionary attribute (`cd`) in their unlinked form as strings.

OMV:
: OMV inherits from OMElement and implements its abstract methods. OMV represents an OpenMath variable. It has a simple string field for the name of the variable.

The search functionality classes

SearchResultElement:
: Instances of the class SearchResultElement are the internal representation of search results returned by the MWS server. Its main component is a list of translation steps that a formula and its associated assertion or axiom undergo from their source to their target `theory`. Translation steps are represented by the nested subclass MorphStep. It also provides the required methods for building runtime representations of the translations on the morphism trail.

MorphStep:
: MorphStep is a nested subclass of SearchResultElement. It represents a single node on the morphism trail of a statement in the theory graph. It maintains references to the considered `theory`, the ImportElement at this step, and the statement's CMP.

Pointer:
: A Pointer is the custom representation of a URL (uniform resource locator) and an Jaxen XPath. The URL is supposed to resolve to an OMDoc document and the XPath is an OMDoc custom XPath string on the XML-structure of this document. The Bourbaker implementation uses Pointers exclusively to reference unique OMDoc elements.

JumpFunction:
: The JumpFunction class implements the "jump"-function extension to the XPath language. It takes as input a list of JDOM attribute nodes, reads out the URLs they contains, and assuming that this URLs reference theories in an OMDoc document, it retrieves the corresponding `theory` element nodes. The JumpFunction is registered with the Jaxen engine by OMDocManager.

XPathFunction:
: The XPathFunction class implements the "xpath"-function, which is an extension to the XPath language. It is fundamentally the inverse operation to retrieving a document node for a given XPath. Given a list of document nodes, the "xpath"-command returns a list of XPaths as strings that point uniquely to the given element. This implementation of XPathFunction works under the assumption that every node is contained inside some `theory` which is uniquely

identified by an XML-ID inside an OMDoc document. From the theory node downwards, the XPath expression is built step by step, using IDs when available and document node positions otherwise, so that the created XPaths identify the supplied elements uniquely.

The theory editor classes

TheoryEditWindow:
This class provides a simple graphical user interface for manipulating in-memory theory graph structures. It is listing symbols, imports, axioms, assertions and definitions depending on the chosen theory. It has a versatile panel that is filled with an instance of TheoryEditPane, SymbolEditPane, ImportEditPane, AxiomEditPane, AssertionEditPane, or DefEditPane depending on what the user chooses to edit. All these classes and the class OMElementEditPane are implemented as nested subclasses.

TheoryEditPane:
This panel provides a simple text edit field for changing the name of a theory.

SymbolEditPane:
This panel provides a simple text edit field for changing the name of a symbol.

ImportEditPane:
With the ImportEditPane one can change the name of the `imports` element and the theory it is importing from. Furthermore, it allows to add `requation` elements to the import and edit the `symbol` translation. For manipulating the target expression of a translated symbol it uses an instance of OMElementEditPane.

AxiomEditPane:
The AxiomEditPane enables the user to edit text and OpenMath content of an `axiom`'s `CMP` in their proper sequential order. It incorporates an instance of OMElementEditPane for revising OMElements.

AssertionEditPane:
AssertionEditPane inherits from AxiomEditPane and offers the same functionality.

DefEditPane:
DefEditPane inherits from AssertionEditPane and adds an extra text field for editing the `definition`'s "for"-attribute.

The classes of the OrganicGraph applet

OrganicGraph:
The root class OrganicGraph extends the PApplet class from the Processing library, which makes it a Java applet without swing functionality. Organ-

icGraph re-implements the methods setup() and draw() which are being called on by PApplet. In setup(), the class initializes a physical system of repelling particles that are held together by springs to form a graph. The draw() method is called perpetually to draw discs to represent particles and arrows between discs to represent springs. OrganicGraph maintains the list of particles and springs and keeps a reference to an instance of GraphManager.

Messenger:
The Messenger class is OrganicGraph's communication interface with applications that embed it. It allows the embedding program to register action listeners for instances of OrganicGraph applets, which will then receive events when nodes in the graph are selected, toggled, or moved. It also allows to reload the contents of an OrganicGraph by means of providing a ParGraph.

GraphManager:
The GraphManager class maintains both directed and undirected adjacency maps of the displayed particle system in order to offer convenient graph operations. It uses breadth-first search in order to calculate shortest paths from a node, which is used for adjusting the size of nodes when one vertex is toggled. It implements an iterative depth-first deepening method that calculates longest path distances, which is used to determine spring lengths. [17] EdNote(17)

ParGraph:
A ParGraph is a wrapper class for an adjacency list representation of a graph.

ParticleData:
ParticleData is a struct-like class for storing context information to particles, i.e., graph nodes.

[17] EdNote: the smartest way to do that would be computing the graph minor by modding out by strongly connected components, then compute a topological ordering and then find max paths in linear time...

Bibliography

[ABC+03] Ron Ausbrooks, Stephen Buswell, David Carlisle, Stéphane Dalmas, Stan Devitt, Angel Diaz, Max Froumentin, Roger Hunter, Patrick Ion, Michael Kohlhase, Robert Miner, Nico Poppelier, Bruce Smith, Neil Soiffer, Robert Sutor, and Stephen Watt. Mathematical Markup Language (MathML) version 2.0 (second edition). W3C recommendation, World Wide Web Consortium, 2003. Available at `http://www.w3.org/TR/MathML2`.

[AHMS02] Serge Autexier, Dieter Hutter, Till Mossakowski, and Axel Schairer. The development graph manager MAYA. In *Proceedings 9th International Conference on Algebraic Methodology And Software Technology, AMAST2002*. Springer-Verlag, LNCS 2422, 2002.

[Anc07] Ştefan Anca. MaTeSearch a combined math and text search engine. Bachelor's thesis, Jacobs University Bremen, 2007.

[Asc04] Michael Aschbacher. The status of the classification of the finite simple groups. *Notices of the American Mathematical Society*, 51(7):736–740, 2004.

[BCC+04] Stephen Buswell, Olga Caprotti, David P. Carlisle, Michael C. Dewar, Marc Gaetano, and Michael Kohlhase. The Open Math standard, version 2.0. Technical report, The Open Math Society, 2004. `http://www.openmath.org/standard/om20`.

[BCM+03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BK07a] Grzegorz Bancerek and Michael Kohlhase. The mizar mathematical library in omdoc. submitted, 2007.

[BK07b] Grzegorz Bancerek and Michael Kohlhase. Towards a MIZAR mathematical library in OMDoc format. In Matuszewski and Zalewska [MZ07], pages 265–275.

[BM98] Michel Bidoit and Peter D. Mosses. *CASL User Manual*, volume 2900 of *LNCS*. Springer Verlag, 1998.

[Bor98] Armand Borel. Twenty-five years with Nicolas Bourbaki, 1949-1973. *Notices of the American Mathematical Society*, 45(3):373–380, 1998.

[Bou68] Nicolas Bourbaki. *Theory of Sets*. Elements of Mathematics. Springer Verlag, 1968.

[Bou74] Nicolas Bourbaki. *Algebra I*. Elements of Mathematics. Springer Verlag, 1974.

[BPSM97] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. Extensible Markup Language (XML). W3C Recommendation TR-XML, World Wide Web Consortium, December 1997. Available at `http://www.w3.org/TR/PR-xml.html`.

[Brö07] Matthias Bröcheler. A mathematical semantic web. Bachelor's thesis, Jacobs University Bremen, 2007.

[CD99] James Clark and Steve DeRose. XML Path Language (XPath) Version 1.0. W3C recommendation, The World Wide Web Consortium, November 1999. Available at `http://www.w3.org/TR/1999/REC-xpath-19991116`.

[Cha87] Wyn Chao. *On Ellipsis*. PhD thesis, University of Massachusssetts, 1987.

[Die06] Reinhard Diestel. *Graph Theory*. Graduate Texts in Mathematics. Springer Verlag, 2006.

[EFT94] Heinz-Dieter Ebbinghaus, Jörg Flum, and Wolfgang Thomas. *Mathematical Logic*. Undergraduate Texts in Mathematics. Springer Verlag, second edition, 1994.

[End02] Herbert B. Enderton. *A Mathematical Introduction to Logic*. Harcourt Academic Press, 2002.

[Euc56] Euclid. *The Thirteen Books of Euclid's Elements*. Dover Publications, 1956.

[Far93] William M. Farmer. Theory interpretation in simple type theory. In *HOA'93, an International Workshop on Higher-order Algebra, Logic and Term Rewriting*, volume 816 of *LNCS*, Amsterdam, The Netherlands, 1993. Springer Verlag.

[FGT92] William Farmer, Josuah Guttman, and Xavier Thayer. Little theories. In D. Kapur, editor, *Proceedings of the 11th Conference on Automated Deduction*, volume 607 of *LNCS*, pages 467–581, Saratoga Springs, NY, USA, 1992. Springer Verlag.

[FGT93] William M. Farmer, Joshua D. Guttman, and F. Javier Thayer. IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning*, 11(2):213–248, October 1993.

[Fre07] Gottlob Frege. *Begriffsschrift und andere Aufsätze*. Olms Verlag, 2007.

[GC07] Jeremy Gow and Paul Cairns. Closing the gaph between formal and digital libraries of mathematics. In Matuszewski and Zalewska [MZ07], pages 249–263.

[Gow99] W. T. Gowers. *Mathematics: Frontiers and Perspectives*, chapter The two cultures of mathematics. American Mathematical Society, 1999.

[Har96] John Harrison. Proof style. In Eduardo Giménez and Christine Paulin-Möhring, editors, *Types for Proofs and Programs: International Workshop TYPES'96*, volume 1512 of *LNCS*, pages 154–172. Springer Verlag, 1996.

[HB99] David Hilbert and Paul Bernays. *Grundlagen der Geometrie*. Teubner Verlag, 13. edition, 1899.

[Hun74] Thomas W. Hungerford. *Algebra*. Graduate Texts in Mathematics. Springer Verlag, 1974.

[Kle03] David Klein. *Mathematical Cognition*, chapter A Brief History of American K-12 Mathematics Education. Information Age Publishing, 2003.

[KMM07] Michael Kohlhase, Achim Mahnke, and Christine Müller. Managing variants in document content and narrative structures. forthcomming, 2007.

[Koh06] Michael Kohlhase. OMDoc – *An open markup format for mathematical documents [Version 1.2]*. Number 4180 in LNAI. Springer Verlag, 2006.

[KŞ06] Michael Kohlhase and Ioan Şucan. A search engine for mathematical formulae. In Tetsuo Ida, Jacques Calmet, and Dongming Wang, editors, *Proceedings of Artificial Intelligence and Symbolic Computation, AISC'2006*, number 4120 in LNAI, pages 241–253. Springer Verlag, 2006.

[KŞ07] Michael Kohlhase and Ioan Şucan. System description: MathWebSearch 0.3, a semantic search engine. submitted to CADE 21, 2007.

[Lan07a] Christoph Lange. Swim – a semantic wiki for mathematical knowledge management. In Paul Libbrecht, editor, *Mathematical User Interfaces Workshop 2007*, 2007.

[Lan07b] Christoph Lange. Towards a Semantic Wiki for Science. `http://kwarc.info/projects/swim/pubs/swimplus-resprop.pdf`, February 2007. Research proposal for a Ph. D. thesis.

[MAH06] T. Mossakowski, S. Autexier, and D. Hutter. Development graphs – proof management for structured specifications. *Journal of Logic and Algebraic Programming*, 67(1-2):114–145, 2006.

[Mat92] Adrian R. D. Mathias. The ignorance of Bourbaki. *Mathematical Intelligencer*, 14(3):4–13, 1992.

[MGH+06] Erica Melis, Giorgi Goguadze, Martin Homik, Paul Libbrecht, Carsten Ullrich, and Stefan Winterstein. Semantic-aware components and services of activemath. *British Journal of Educational Technology*, 37(3):405–423, May 2006.

[MML06] Till Mossakowski, Christian Maeder, and Klaus Lüttich. Hets: The heterogeneous tool set. In OMDoc – *An open markup format for mathematical documents [Version 1.2]* [Koh06], chapter 26.13.

[MZ07] Roman Matuszewski and Anna Zalewska, editors. *From Insight to Proof*, volume 10 of *Studies in Logic, Grammar and Rhetoric*. University of Białystok, 2007.

[NK07] Immanuel Normann and Michael Kohlhase. Extended formula normalization for ϵ-retrieval and sharing of mathematical knowledge. In Manuel Kauers, Manfred Kerber, Robert Miner, and Wolfgang Windsteiger, editors, *MKM/Calculemus 2007*, number 4573 in LNAI, pages 266–279. Springer Verlag, 2007. in press.

[QED95] The QED manifesto. Internet Report `http://www.rbjones.com/rbjpub/logic/qedres00.htm`, 1995.

[RKM07] Florian Rabe, Michael Kohlhase, and Normen Müller. A module system for mathematical theories. forthcomming, 2007.

[Sen98] Marjorie Senechal. The continuing silence of Bourbaki - an interview with Pierre Cartier, June 18, 1997. *The Mathematical Intelligencer*, (1):22–28, 1998.

[Sol95] Ron Solomon. On finite simple groups and their classification. *Notices of the American Mathematical Society*, 42(2):331–339, 1995.

[vEF95] Jan van Eijck and Nissim Francez. Verb-phrase ellipsis in dynamic semantics. pages 29–59. Kluwer, 1995.

[Wie03] Freek Wiedijk. Comparing mathematical provers. In Andrea Asperti, Bruno Buchberber, and James Harold Davenport, editors, *Mathematical Knowledge Management, MKM'03*, number 2594 in LNCS, pages 188–202. Springer Verlag, 2003.

[Wie07] Freek Wiedijk. The QED Manifesto revisited. In Matuszewski and Zalewska [MZ07], pages 121–133.

[WR10] Alfred North Whitehead and Bertrand Russell. *Principia Mathematica*, volume I. Cambridge University Press, Cambridge, Great Britain; second edition, 1910.

Index

Şucan, Ioan, 74
AssertionEditPane, 101
AxiomEditPane, 101
Axiom, 99
BourbOn, 98
CMP, 99
DefEditPane, 101
DefStat, 99
GraphManager, 102
ImportEditPane, 101
ImportElement, 99
JumpFunction, 100
MainWindow, 98
Messenger, 102
MorphStep, 100
OMA, 99
OMDocManager, 98
OMElement, 99
OMS, 100
OMV, 100
OrganicGraph, 101
ParGraph, 102
ParticleData, 102
Pointer, 100
Requation, 99
SearchManager, 98
SearchResultElement, 100
SymbolEditPane, 101
Symbol, 99
TheoryEditPane, 101
TheoryEditWindow, 101
TheoryElement, 99
TheoryManager, 98
XPathFunction, 100

ActiveMath, 24, 37
alien theory, 47, 50
Anca, Stefan, 75
antichain, 51
average out-degree, 51

Bancerek, Grzegorz, 30, 32
based search, 77
based search cone, 78
big theory, 6
Bourbaker, 83
  development, 83
  in-memory theory graph, 84
  main classes, 83
  search, 85
Bourbaker prototype, 69
Bourbaker suite, *see* Bourbaker prototype
Bourbaki, 41
  Algebra I, 41, 43
  clarity, 53
  compatibility, 59
  quality, 57
  Theory of Sets, 42
Bourbaki, Nicolas, *see* Bourbaki, 43
Bröcheler, Matthias, 39
bulletin board, 90

Cartan, Henri, 43
Cartier, Pierre, 43
CASL, 35
center of interest, 70, 73
communities of practice, 93
cone of impact, 53
criteria for text evaluation, 49
  clarity, 50
  compatibility, 51
  quality, 50

definition by ellipsis, 54
Dieudonné, Jean, 43
digital library, 24

Eclipse, 83
Euclid, 5
Euler's polyhedron theorem, 9
exploration graph, 70

Farmer, William M., 6, 29, 32
flattening, 17
formal libraries, 29
formal library, 24
Frege, Gottlob, 5
future work, 88

Gödel, Kurt, 44
Gowers, Timothy, 44
graph
    extensibility, 49
    leaf, 51
    robustness, 49
graph data, 96
graph depth, 50
graph minor, 71
graph tools, 69
graph width, 50
Graphical User Interface, 83
Guttman, Joshua D., 32

heuristics
    alien theories, 47
    concept-driven theories, 46
    definitions through axioms, 46
    definitions through graph structure, 46
    fine granularity, 46
    symbol renaming, 48
    technical theories, 48
heuristics for graph transformation, 46
Hilbert, David, 6
Hungerford, Thomas W., 60

identification of symbols, 17
IMPS, 6, 24, 32
intrusive morphism, 51

Java, 83
Jaxen, 83
JDOM, 83
JOMDoc, 89

Kohlhase, Michael, 13, 19, 32

Lange, Christoph, 40
little theories, 6

Müller, Normen, 19
Math Web Search, 75
Mathematical knowledge management, *see* MKM
mathematical knowledge management, 25
Mathias, Adrian, 44
MathWorld, 24
minor, *see* graph minor
Mizar, 24, 30
MKM, 11, *see* mathematical knowledge management
morphism trail, 79
Mossakowski, Till, 36
MWS, *see* Math Web Search

New Math, 44
Normann, Immanuel, 80

OMDoc, 12, 13
OrganicGraph, 73

Perelman, Grigori, 23
PlanetMath, 24, 25
predicate-theory dilemma, 55, 62
principle of immediate harvesting, 53
Processing, 83
proof obligation, 16

Rabe, Florian, 19
resolution graph, 72
Russell, Bertrand, 5

search
    presentation, 78
semantic library, 38
separation point, 51
shallow proof assistance, 80
Sojakova, Kristina, 89
spring, 73
supernode, 72
SWiM, 25

technical theory, 48

Thayer, F. Javier, 32
theory graph, 9
    flattened, 17, 51
    in-memory representation, 84
    robustness, 51
    topological stability, 51
    visualitation, 70
theory import
    seetheory morphism
      definitional, 16
theory interpretation, *see* theory morphism
theory morphism, 6, 7
    definitional, 15, 16
    detection, 80
    postulated, 16
    structural, 15
theory translation, *see* theory morphism
topological exponential map, 15
topological minor, 71
Traer, 83
transformation, 46
    heuristics, 46
transitively trimmed search, 77
translation, *see* transformation
trimmed search, 77
trimmed search cone, 77
Trybulec, Andrzej, 29
tweaked search, 77
tweaked theory graph, 73

unqualified link, 28

Weil, André, 43
Weisstein, Eric W., 25
Whitehead, Alfred N., 5
Wiedijk, Freek, 29
Wikipedia, 25

XML, 12
XPath, 83