A Vernacular for Coherent Logic

Sana Stojanovic, Julien Narboux, Marc Bezem, Predrag Janicic

A Vernacular for Coherent Logic*

Sana Stojanović[1], Julien Narboux[2], Marc Bezem[3], and Predrag Janičić[1]

[1] Faculty of Mathematics, University of Belgrade, Serbia
[2] ICube, UMR 7357 CNRS, University of Strasbourg, France
[3] Institute for Informatics, University of Bergen, Norway
sana@matf.bg.ac.rs,narboux@unistra.fr,
marc.bezem@ii.uib.no,janicic@matf.bg.ac.rs

Abstract. We propose a simple, yet expressive proof representation from which proofs for different proof assistants can easily be generated. The representation uses only a few inference rules and is based on a fragment of first-order logic called coherent logic. Coherent logic has been recognized by a number of researchers as a suitable logic for many everyday mathematical developments. The proposed proof representation is accompanied by a corresponding XML format and by a suite of XSL transformations for generating formal proofs for Isabelle/Isar and Coq, as well as proofs expressed in a natural language form (formatted in LaTeX or in HTML). Also, our automated theorem prover for coherent logic exports proofs in the proposed XML format. All tools are publicly available, along with a set of sample theorems.

1 Introduction

Mathematics can be done on two different levels. One level is rather informal, based on informal explanations, intuition, diagrams, etc., and typical for everyday mathematical practice. Another level is formal mathematics with proofs rigorously constructed by rules of inference from axioms. A large portion of mathematical logic and interactive theorem proving is aimed at linking these two levels. However, there is still a big gap: mathematicians still don't feel comfortable doing mathematics formally and proof assistants still don't provide enough support for dealing with large mathematical theories, automating technical problems, translating from one formalism to another, etc. We consider the following issue: there are several very mature and popular interactive theorem provers (including Isabelle, Coq, Mizar, HOL-light, see [29] for an overview), but they still cannot easily share the same mathematical knowledge. This is a significant problem, because there are increasing efforts in building repositories

* The first, second and the fourth author were partly supported by the Serbian-French Technology Co-Operation grant EGIDE/"Pavle Savić" 680-00-132/2012-09/12. The first and the fourth author are partly supported by the grant ON174021 of the Ministry of Science of Serbia. The final publication is available at http://link.springer.com.

of formalized mathematics, but — still developed within specific proof assistants. Building a mechanism for translation between different proof assistants is non-trivial because of many deep specifics of each proof assistant (there are some recent promissing approaches for this task [13]). Instead of developing a translation mechanism, we propose a proof representation and a corresponding XML-based format. The proposed proof representation is light-weight and it does not aim at covering full power of everyday mathematical proofs or full power of first order logic. Still, it can cover a significant portion of many interesting mathematical theories. The underlying logic of our representation is coherent logic, a fragment of first-order logic. Proofs in this format can be generated in an easy way by dedicated, coherent logic provers, but in principle, also by standard theorem provers. The proofs can be translated to a range of proof assistant formats, enabling sharing the same developments.

We call our proof representation "coherent logic vernacular". *Vernacular* is the everyday, ordinary language (in contrast to the official, literary language) of the people of some country or region. A similar term, *mathematical vernacular* was used in 1980's by de Bruijn within his formalism proposed for trying to *put a substantial part of the mathematical vernacular into the formal system* [10]. Several authors later modified or extended de Bruijn's framework. Wiedijk follows de Bruijn's motivation [28], but he also notices:

> *It turns out that in a significant number of systems ('proof assistants') one encounters languages that look almost the same. Apparently there is a canonical style of presenting mathematics that people discover independently: something like a* natural *mathematical vernacular. Because this language apparently is something that people arrive at independently, we might call it* the *mathematical vernacular.*

We find that this language is actually closely related to a proof language of coherent logic, which is a basis of our proof representation presented in this paper.

Our proof representation is developed also with *readable proofs* in mind. Readable proofs (e.g., textbook-like proofs), are very important in mathematical practice. For mathematicians, the main goal is often, not only a trusted, but also a clear and intuitive proof. We believe that coherent logic is very well suited for automated theorem proving with a simple production of readable proofs.

2 Background

In this section, we give a brief overview of interactive theorem proving and proof assistants, of coherent logic, which is the logical basis for our proof representation, and of XML, which is the technical basis for our proof format.

2.1 Interactive Theorem Proving

Interactive theorem proving systems (or *proof assistants*) support the construction of formal proofs by a human, and verify each proof step with respect to the

given underlying logic. The proofs can be written either in a *declarative* or in a *procedural* proof style. In the procedural proof style, the proof is described by a sequence of commands which modify the incomplete proof tree. In the declarative proof style the formal document includes the intermediate statements. Both styles are avaible in HOL-Light, Isabelle [27] and Coq proof assistants whereas only the declarative style is available in Mizar, see [30] for a recent discussion. The procedural proof style is more popular in the Coq community.

Formal proofs are typically much longer than "traditional proofs".[4] Progress in the field can be measured by proof scripts becoming shorter and yet contain enough information for the system to construct and verify the full (formal) proof. "Traditional proofs" can often hardly be called proofs, because of the many missing parts, informal arguments, etc. Using interactive theorem proving uncovered many flaws in many published mathematical proofs (including some seminal ones), published in books and journals.

2.2 Coherent Logic

Coherent logic (CL) was initially defined by Skolem and in recent years it gained new attention [3,11,4]. It consists of formulae of the following form:

$$A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow \exists \boldsymbol{y}(B_1(\boldsymbol{x},\boldsymbol{y}) \vee \ldots \vee \ B_m(\boldsymbol{x},\boldsymbol{y})) \tag{1}$$

which are implicitly universally quantified, and where $0 \leq n$, $0 \leq m$, $\boldsymbol{x}$ denotes a sequence of variables $x_1, x_2, \ldots, x_k$ $(0 \leq k)$, A_i (for $1 \leq i \leq n$) denotes an atomic formula (involving zero or more of the variables from $\boldsymbol{x}$), $\boldsymbol{y}$ denotes a sequence of variables $y_1, y_2, \ldots, y_l$ $(0 \leq l)$, and B_j (for $1 \leq j \leq m$) denotes a conjunction of atomic formulae (involving zero or more of the variables from $\boldsymbol{x}$ and $\boldsymbol{y}$). For simplicity, we assume that there are no function symbols with arity greater than zero (so, we only consider symbols of constants as ground terms).

The definition of CL does not involve negation. For a single atom A, $\neg A$ can be represented in the form $A \Rightarrow \bot$, where $\bot$ stands for the empty disjunction, but more general negation must be expressed carefully in coherent logic. In order to reason with negation in general, new predicate symbols are used to abbreviate subformulas. Furthermore, for every predicate symbol R (that appears in negated form), a new symbol $\overline{R}$ is introduced that stands for $\neg R$, and the following axioms are postulated (cf. [19]): $\forall \boldsymbol{x}(R(\boldsymbol{x}) \wedge \overline{R}(\boldsymbol{x}) \Rightarrow \bot)$, $\forall \boldsymbol{x}(R(\boldsymbol{x}) \vee \overline{R}(\boldsymbol{x}))$.

CL allows existential quantifications of the conclusion of a formula, so CL can be considered to be an extension of resolution logic. In contrast to the resolution-based proving, the conjecture being proved is kept unchanged and directly proved (refutation, Skolemization and transformation to clausal form are not used). Hence, proofs in CL are natural and intuitive and reasoning is

[4] The ratio between the length of formal proof *script* and the length of the informal proof is often called the *de Bruijn factor* [2]. It varies for different parts of mathematics and for different systems, and is currently often around 4. The de Bruijn factor can be below 1 if a lot of automation can be used. It can also be well over 10 when the informal proof is rather sketchy.

constructive. Readable proofs (in the style of forward reasoning and a variant of natural deduction) can easily be obtained [3].

A number of theories and theorems can be formulated directly and simply in CL. In CL, constructive provability is the same as classical provability. It can be proved that any first-order formula can be translated into a set of CL formulas (in a different signature) preserving satisfiability [19] (however, this translation does not always preserve constructive provability).

Coherent logic is semi-decidable and there are several implemented semi-decision procedures for it [3]. ArgoCLP [24] is a generic theorem prover for coherent logic, based on a simple proof procedure with forward chaining and with iterative deepening. ArgoCLP can read problems given in TPTP form[5] [25] and can export proofs in the XML format that we describe in this paper. These proofs are then translated into target languages, for instance, the Isar language or natural language thanks to appropriate XSLT style-sheets.

2.3 XML

Extensible Markup Language (XML)[6] is a simple, flexible text format, inspired by SGML (ISO 8879), for data structuring using tags and for interchanging information between different computing systems. XML is primarily a "metalanguage"—a language for describing other customized markup languages. So, it is not a fixed format like the markup language HTML—in XML the tags indicate the semantic structure of the document, rather than only its layout. XML is a project of the World Wide Web Consortium (W3C) and is a public format. Almost all browsers that are currently in use support XML natively.

There are several schema languages for formaly specifying the structure and content of XML documents of one class. Some of the main schema languages are DTD (*Data Type Definition*), XML Schema, Relax, etc. [17]. Specifications in the form of schema languages enable automatic verification ("validation") of whether a specific document meets the given syntactical restrictions.

Extensible style-sheet language transformation (XSLT)[7] is a document processing language that is used to transform the input XML documents to output files. An XSLT style-sheet declares a set of rules (templates) for an XSLT processor to use when interpreting the contents of an input XML document. These rules tell to the XSLT processor how that data should be presented: as an XML document, as an HTML document, as plain text, or in some other form.

3 Proof Representation

The proposed proof representation is very usable and expressive, yet very simple. It uses only a few inference rules, a variant of the rules given in [4]. Given a set of coherent axioms AX and a coherent conjecture $A_1(\boldsymbol{x}) \wedge \ldots \wedge A_n(\boldsymbol{x}) \Rightarrow$

[5] http://www.cs.miami.edu/~tptp/
[6] http://www.w3.org/XML/
[7] http://www.w3.org/Style/XSL/

$\exists \boldsymbol{y}(B_1(\boldsymbol{x},\boldsymbol{y}) \vee \ldots \vee\ B_m(\boldsymbol{x},\boldsymbol{y}))$, the goal is to prove, using the rules given below, the following (where $\boldsymbol{a}$ denote a vector of new symbols of constants):

$$AX, A_1(\boldsymbol{a}) \wedge \ldots \wedge A_n(\boldsymbol{a}) \vdash \exists \boldsymbol{y}(B_1(\boldsymbol{a},\boldsymbol{y}) \vee \ldots \vee\ B_m(\boldsymbol{a},\boldsymbol{y}))$$

The rules are applied in a forward manner, so they can be read from bottom to top. In the rules below we assume:

- $ax \in AX$ is a formula of the form (1) (page 3);
- $\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ denote vectors of constants (possibly of length zero);
- in the rule *mp*, $\boldsymbol{b}$ are fresh constants;
- $\boldsymbol{x}$ and $\boldsymbol{y}$ denote vectors of variables (possibly of length zero);
- $A_i(\boldsymbol{x})$ ($B_i(\boldsymbol{x},\boldsymbol{y})$) have no free variables other than from $\boldsymbol{x}$ (and $\boldsymbol{y}$);
- $A_i(\boldsymbol{a})$ are ground atomic formulae;
- $B_i(\boldsymbol{a},\boldsymbol{b})$ and $B_i(\boldsymbol{c})$ are ground conjunctions of atomic formulae;
- $\underline{\Phi}$ denotes the list of conjuncts in Φ.

$$\frac{\Gamma, ax, \underline{A_1(\boldsymbol{a}) \wedge \ldots \wedge A_n(\boldsymbol{a})}, B_1(\boldsymbol{a},\boldsymbol{b}) \vee \ldots \vee B_m(\boldsymbol{a},\boldsymbol{b}) \vdash P}{\Gamma, ax, \underline{A_1(\boldsymbol{a}) \wedge \ldots \wedge A_n(\boldsymbol{a})} \vdash P}\ \textit{mp}\ (\textit{modus ponens})$$

$$\frac{\Gamma, \underline{B_1(\boldsymbol{c})} \vdash P \quad \ldots \quad \Gamma, \underline{B_n(\boldsymbol{c})} \vdash P}{\Gamma, B_1(\boldsymbol{c}) \vee \ldots \vee B_n(\boldsymbol{c}) \vdash P}\ \textit{cs}\ (\textit{case split})$$

$$\frac{}{\Gamma, \underline{B_i(\boldsymbol{a},\boldsymbol{b})} \vdash \exists \boldsymbol{y}(B_1(\boldsymbol{a},\boldsymbol{y}) \vee \ldots \vee\ B_m(\boldsymbol{a},\boldsymbol{y}))}\ \textit{as}\ (\textit{assumption})$$

$$\frac{}{\Gamma, \bot \vdash P}\ \textit{efq}\ (\textit{ex falso quodlibet})$$

None of these rules change the goal P, which helps generating readable proofs as the goal can be kept implicit. Note that the rule *mp* actually combines universal instantiation, conjunction introduction, modus ponens, and elimination of (zero or more) existential quantifiers. This seems a reasonable granularity for an inference step, albeit probably the maximum for keeping proofs readable. Compared to [20] which defines the notion of obvious inference rule by putting constraints on an automated prover, our position is: the obvious inferences are the ones defined by the inference rules above. Compared to the rules given in [4], we choose to separate the *case split* rule (disjunction elimination) and the *ex falso quodlibet* rule from the single combined rule in [4], in order to improve readability. Case distinction (split) is an important way of structuring proofs that deserves to be made explicit. Also, *ex falso quodlibet* could be seen as a *case split* with zero cases, but this would be less readable.

Any coherent logic proof can be represented in the following simple way (*mp* is used zero or more time, *cs* involves at least two other *proof* objects):

$$\textit{proof}\ ::=\ \textit{mp}^*\ (\textit{cs}(\textit{proof}^{\geq 2}) \mid \textit{as} \mid \textit{efq})$$

4 XML Suite for CL Vernacular

The proof representation described in Section 3 is used as a basis for our XML-based proof format. It is developed as an interchange format for automated and interactive theorem provers. Proofs (for Coq and Isabelle/Isar) that are produced from our XML documents are fairly readable. The XML documents themselves can be read by a human, but much better alternative is using translation to human readable proofs in natural language (formatted in LaTeX, for instance). The proof representation is described by a DTD `Vernacular.dtd`. As an illustration, we show some fragments:

```
...
<!--******** Theory **************-->
<!ELEMENT theory (theory_name, signature, axiom*) >
<!ELEMENT theory_name (#PCDATA)>
<!ELEMENT signature (type*, relation_symbol*, constant*) >
<!ELEMENT relation_symbol (type*)>
<!ATTLIST relation_symbol name CDATA #REQUIRED>
<!ELEMENT type (#PCDATA)>
<!ELEMENT axiom (cl_formula)>
<!ATTLIST axiom name CDATA #REQUIRED>
...
```

The above fragment describes the notion of theory. (Definitions, formalized as pairs of coherent formulae, are used as axioms.) A file describing a theory could be shared among several files with theorems and proofs.

```
...
<!--******** Theorem **************-->
<!ELEMENT theorem (theorem_name, cl_formula, proof+)>
<!ELEMENT theorem_name (#PCDATA)>
<!ELEMENT conjecture (name, cl_formula)>

<!--******** Proof **************-->
<!ELEMENT proof (proof_step*, proof_closing, proof_name?)>
<!ELEMENT proof_name EMPTY>
<!ATTLIST proof_name name CDATA #REQUIRED>

<!--******** Proof steps **************-->
<!ELEMENT proof_step (indentation,modus_ponens)>
<!ELEMENT proof_closing (indentation, (case_split|efq|from),
  (goal_reached_contradiction|goal_reached_thesis))>
...
```

The above fragment describes the notion of a theorem and a proof. As said in Section 3, a proof consists of a sequence of applications of the rule modus ponens and closes with one of the remaining proof rules (*case split*, *as*, or *efq*). Within the last three, there is the additional information on whether the proof closes by $\perp$ (by detecting a contradiction) or by detecting one of the disjuncts

from the goal. This information is generated by the prover and can be used for better readability of the proof but also for some potential proof transformations. Within each proof step there is also the information on indentation. This information, useful for better layout, tells the level of subproofs and as such can be, in principle, computed from the XML representation. Still, for convenience and simplicity of the XSLT style-sheets, it is stored within the XML representation.

We implemented XSL transformations from XML format to Isabelle/Isar (`VernacularISAR.xls`), Coq (`VernacularCoqTactics.xls`), and to a natural language (English) in LaTeX form and in HTML form (`VernacularTex.xls` and `VernacularHTML.xls`).

The translation from XML to the Isar language is straightforward and each of our proof steps is trivially translated into Isar constructs.[8] Naturally, we use native negation of Isar (and Coq) instead of defined negation in coherent logic.

The translation to Coq has been written in the same spirit as the Isar output despite the fact proofs using tactics are more popular in Coq than declarative proofs. We refer to the assumptions by their statement instead of their name (for example: `by cases on (A = B \/ A <> B)`). Moreover, when we can, we avoid to refer to the assumptions at all. We did not use the declarative proof mode of Coq because of efficiency issues. We use our own tactics to implement the inference rules of CL to improve readability. Internally, we use an Ltac tactic to get the name of an assumption. The forward reasoning proof steps consist of applications of the `assert` tactic of Coq. Equality is translated into Leibniz equality.

The translation to LaTeX and HTML includes an additional XSLT style-sheet that optionally defines specific layout for specific relation symbols (so, for instance, $(A, B) \cong (C, D)$ can be the layout for `cong(A,B,C,D)`).

The developed XSLT style-sheets are rather simple and short — each is only around 500 lines long. This shows that transformations for other target languages (other theorem provers, like Mizar and HOL light, LaTeX with other natural languages, MathML, OMDoc or TPTP) can easily be constructed, thus enabling wide access to a single source of mathematical contents.

Our automated theorem prover for coherent logic ArgoCLP exports proofs in the form of the XML files that conforms to this DTD. ArgoCLP reads an input theory and the conjecture given in the TPTP form (assuming the coherent form of all formulae and that there are no function symbols or arity greater than 0). ArgoCLP has built-in support for equality (during the search process, it uses an efficient union-find structure) and the use of equality axioms is implicit in generated proofs. The generated XML documents are simple and consist of three parts: `frontpage` (providing, for instance, the author of the theorem, the prover used for generating the proof, the date), `theory` (providing the signature and the axioms) and, organized in chapters, a list of conjectures or theorems with their proofs. This way, some contents (`frontpage` and `theory`) can be shared by a number of XML documents. On the other hand, this also enables simple

[8] The system Isabelle has available a proof method `coherent` based on a internal theorem prover for coherent logic. Our Isar proofs do not use this proof method.

construction of bigger collections of theorems. The following is one example of an XML document generated by ArgoCLP:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE main SYSTEM "Vernacular.dtd">
<?xml-stylesheet href="VernacularISAR.xsl" type="text/xsl"?>

<main>
<xi:include href="frontpage.xml" parse="xml"
    xmlns:xi="http://www.w3.org/2003/XInclude"/>
<xi:include href="theory_thm_4_19.xml" parse="xml"
    xmlns:xi="http://www.w3.org/2003/XInclude"/>

<chapter name="th_4_19">
<xi:include href="proof_thm_4_19.xml" parse="xml"
    xmlns:xi="http://www.w3.org/2003/XInclude"/>
</chapter>
</main>
```

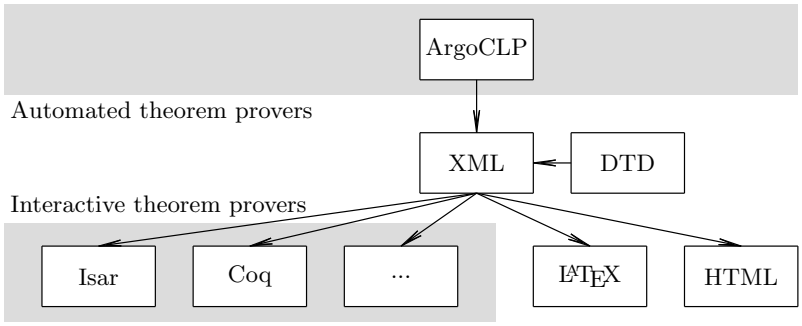

Fig. 1. Architecture of the presented framework

The overall architecture of the framework is shown in Figure 1.[9]

5 Examples

Our XML suite for coherent logic vernacular is used for a number of proofs generated by our prover ArgoCLP. In this section we discuss proofs of theorems from the book *Metamathematische Methoden in der Geometrie*, by Wolfram Schwabhäuser, Wanda Szmielew, and Alfred Tarski [21], one of the twenty-century mathematical classics. The theory is described in terms of first-order logic, it uses only one sort of primitive objects — points, has only two primitive

[9] The whole of our XML suite, along with a collection of theorems is available online from http://argo.matf.bg.ac.rs/downloads/software/clvernacular.zip.

predicates (*cong* or arity 4 and *bet* of arity 3, intuitively for congruence and betweenness) and only eleven axioms. The majority of theorems from this book are in coherent logic or can be trivially transformed to belong to coherent logic. After needed transformations, the number of theorems in our development (238) is somewhat larger than in the book [23].

Here we list a proof of one theorem (4.19) from Tarski's book. The theorem was proved by ArgoCLP (using the list of relevant axioms and theorems produced by a resolution theorem prover), the proof was exported in the XML format, and then transformed to a proof in natural language by appropriate XSL transformation ($(A, B) \cong (C, D)$ is an infix notation for $cong(A, B, C, D)$ and it denotes that the pairs of points (A, B) and (C, D) are congruent, $bet(A, B, C)$ denotes that the point B is between the points A and C, $col(A, B, C)$ denotes that the points A, B and C are collinear).

Theorem 1 (th_4_19). *Assuming that* $bet(A, B, C)$ *and* $AB \cong AD$ *and* $CB \cong CD$ *it holds that* $B = D$.

Proof:

1. It holds that $bet(B, A, A)$ (using th_3_1).
2. From the fact(s) $bet(A, B, C)$ it holds that $col(C, A, B)$ (using $ax\_4\_10\_3$).
3. From the fact(s) $AB \cong AD$ it holds that $AD \cong AB$ (using th_2_2).
4. It holds that $A = B$ or $A \neq B$.
   5. Assume that: $A = B$.
      6. From the fact(s) $AD \cong AB$ and $A = B$ it holds that $AD \cong AA$.
      7. From the fact(s) $AD \cong AA$ it holds that $A = D$ (using ax_3).
      8. From the fact(s) $A = B$ and $A = D$ it holds that $B = D$.
      9. The conclusion follows from the fact(s) $B = D$.
   10. Assume that: $A \neq B$.
      11. It holds that $A = C$ or $A \neq C$.
         12. Assume that: $A = C$.
            13. From the fact(s) $bet(A, B, C)$ and $A = C$ it holds that $bet(A, B, A)$.
            14. From the fact(s) $bet(A, B, A)$ and $bet(B, A, A)$ it holds that $A = B$ (using th_3_4).
            15. From the fact(s) $A \neq B$ and $A = B$ we get contradiction.
         16. Assume that: $A \neq C$.
            17. From the fact(s) $A \neq C$ it holds that $C \neq A$.
            18. From the fact(s) $C \neq A$ and $col(C, A, B)$ and $CB \cong CD$ and $AB \cong AD$ it holds that $B = D$ (using th_4_18).
            19. The conclusion follows from the fact(s) $B = D$.
      20. The conclusion follows in all cases.
21. The conclusion follows in all cases.

QED

Below is the same proof in Isabelle/Isar form:

```
lemma th_4_19 :  assumes  "bet A B C" and "cong A B A D" and
"cong C B C D"  shows "(B = D)"

proof -

have "bet B A A"  by (rule th_3_1)
from ‘bet A B C‘ have "col C A B" by (rule ax_4_10_3)
from ‘cong A B A D‘ have "cong A D A B" by (rule th_2_2)
have "A = B ∨ A ~= B" by (subst disj_commute, rule excluded_middle)
   show ?thesis
   proof(cases "A = B")
     case True
      from ‘cong A D A B‘ and ‘A = B‘ have "cong A D A A" by simp
      from ‘cong A D A A‘ have "A = D" by (rule ax_3)
      from ‘A = B‘ and ‘A = D‘ have "B = D" by simp
      from ‘B = D‘ show ?thesis by assumption
   next
     case False
      have "A = C ∨ A ~= C" by (subst disj_commute, rule
excluded_middle)
         show ?thesis
         proof(cases "A = C")
           case True
            from ‘bet A B C‘ and ‘A = C‘ have "bet A B A" by simp
            from ‘bet A B A‘ and ‘bet B A A‘ have "A = B" by (rule
th_3_4)
            from ‘A ~= B‘ and ‘A = B‘ have "False" by (rule notE)
            from this show ?thesis by (rule FalseE)
         next
           case False
            from ‘A ~= C‘ have "C ~= A" by (rule not_sym)
            from ‘C ~= A‘ and ‘col C A B‘ and ‘cong C B C D‘ and
‘cong A B A D‘ have "B = D" by (rule th_4_18)
            from ‘B = D‘ show ?thesis by assumption
         qed
   qed
qed
end
```

Below is the same proof in Coq form:

```
Theorem th_4_19 : ∀ (A:point) (B:point) (C:point) (D:point), (bet A B C
∧ cong A B A D ∧ cong C B C D) → B = D.
Proof.
 intros.
 assert (bet B A A) by applying (th_3_1 B A ) .
 assert (col C A B) by applying (ax_4_10_3 A B C ) .
 assert (cong A D A B) by applying (th_2_2 A B A D ) .
 assert (A = B ∨ A ≠ B) by applying (ax_g1 A B ) .
 by cases on (A = B ∨ A ≠ B).
 - {
    assert (cong A D A A) by (substitution).
    assert (A = D) by applying (ax_3 A D A ) .
    assert (B = D) by (substitution).
    conclude.
    }
 - {
    assert (A = C ∨ A ≠ C) by applying (ax_g1 A C ) .
    by cases on (A = C ∨ A ≠ C).
    - {
       assert (bet A B A) by (substitution).
       assert (A = B) by applying (th_3_4 A B A ) .
       assert (False) by (substitution).
       contradict.
       }
    - {
       assert (C ≠ A) by (substitution).
       assert (B = D) by applying (th_4_18 C A B D ) .
       conclude.
       }
    }
Qed.
```

From the set of individual theorems (238), the prover ArgoCLP completely automatically proved 85 (36%) of these theorems and generated proofs in the XML format. We created a single XML document that contains all proved theorems and other theorems tagged as conjectures. The whole document matches the original book by Schwabhäuser, Szmielew, and Tarski and can be explored in the LaTeX (or PDF) form, HTML or as Isabelle or Coq development.[10]

[10] Translating the XML document with 85 proofs by to Isabelle, Coq, HTML, LaTeX (and then to PDF) takes altogether around 20s on a PC with AMD Opteron 6168. The resulting Isabelle document is verified in 30s, and the Coq document in 6s.

6 Related Work

In [28], Wiedijk proposes a mathematical vernacular that is in a sense the common denominator of the proof languages of Hyperproof, Mizar and Isabelle/Isar. We agree with his conclusion in the last sentence of the quotation in the introduction, but we think that the three proof languages were *not* discovered independently. Natural deduction has been introduced by the Polish logicians Łukasiewicz and Jaśkowski in the late 1920's, in reaction on the formalisms of Frege, Russell and Hilbert. The term *natural deduction* seems to have been used first by Gentzen, in German:

> *Ich wollte zunächst einmal einen Formalismus aufstellen, der dem wirklichen Schließen möglichst nahe kommt. So ergab sich ein "Kalkül des natürlichen Schließens".* (First of all I wanted to set up a formalism that comes as close as possible to actual reasoning. Thus arose a "calculus of natural deduction".)—Gentzen, Untersuchungen über das logische Schließen (Mathematische Zeitschrift 39, pp.176–210, 1935)

The qualifier *natural* was of course particularly well-chosen to express that the earlier formalisms were unnatural! As this was indeed the case, natural deduction quickly became the predominant logical system, helped by the seminal work by Gentzen on cut-elimination. (Ironically, this technical work in proof theory is best carried out with proofs represented in *sequent calculus*, using natural deduction on the meta-level.)

It should thus not come as a surprise that the vernacular we propose also is based on natural deduction. One difference with Wiedijk's vernacular is that ours is based on coherent logic instead of full first-order logic. This choice is motivated in Section 2.2 (easier semi-decision procedure and more readable proofs). Another difference is that Wiedijk allows proofs to be incomplete, whereas we stress complete proof objects. This difference is strongly related to the fact that Wiedijk's vernacular is in the first place an input formalism for proof construction, whereas our vernacular is an output formalism for proof presentation and export of proofs to different proof assistants. As far as we know, the mathematical vernacular proposed by Wiedijk's has not been implemented on its own, although Hyperproof, Mizar and Isabelle/Isar are developed using the same ideas.

A number of authors independently point to this or similar fragments of first-order logic as suitable for expressing significant portions of standard mathematics (or specifically geometry), for instance, Avigad et.al. [1] and Givant and Tarski et.al. [26,21] in the context of a new axiomatic foundations of geometry. A recent paper by Ganesalingam and Gowers [12] is also related to our work. Their goal is comparable to ours: full automation combined with human-style output. They propose inference rules which are very similar to our coherent logic based proof system. For example, their rule *splitDisjunctiveHypothesis* corresponds to the rule *case split*, *deleteDoneDisjunct* corresponds to *as*, *removeTarget* corresponds to *as* (with length of $\mathbf{y}$ greater than 0), *forwardsReasoning* corresponds to the rule *mp*. Yet, some rules they proposed are not part of our set of rules. The logic they use is full first-order, with a plan to include second-order features

(this would also be perfectly possible for coherent logic, which is the first-order fragment of *geometric* logic, which is in turn a fragment of higher-order logic, see [8]). Upon closer inspection, the paper by Ganesalingam and Gowers seems to stay within the coherent fragment, and proofs by contraposition and contradiction are delegated to future work. We find some support for our approach in the observation by Ganesalingam and Gowers that it will be hard to avoid that such reasoning patterns are applied in "inappropriate contexts". On the other hand, the primary domain of application of their approach is metric space theory so far, with the ambition to attack problems in other domains as well. It would be very interesting to test the two approaches on the same problem sets. One difference is that [12] insists on proofs being faithful to the thought processes, whereas we would be happy if the prover finds a short and elegant proof even after a not-so-elegant proof search. Another difference is that we are interested in portability of proofs to other systems. To our knowledge, the prover described in [12] is not publicly available.

Compared to OMDoc [16], our proof format is much more specific (as we specify the inference rules we use) and has less features. It can be seen as a specific set of `methods` elements of the `derive` element of OMDoc.

An alternative to using coherent logic provers would be using one of the more powerful automated theorem provers and exploiting existing and ongoing work on proof reconstruction and refactoring (see, for example, [22,5,14]). This is certainly a viable option. However, reconstructing a proof from the log of a highly optimized prover is difficult. One problematic step is deskolemization, that is, proof reconstruction from a proof of the skolemized version of the problem. (The most efficient provers are based on resolution logic, and clausification including skolemizing is the first step in the solution procedure.) What can be said about this approach in its current stage is that more theorems can be proved, but their proofs can still be prohibitively complicated (or use additional axioms). It has been, however, proved beneficial to use powerful automated theorem provers as preprocessors, to provide hints for ArgoCLP.

The literature contains many results about exchanging proofs between proof assistant using deep or shallow embeddings [18,15]. Boessplug, Cerbonneaux and Hermant propose to use the $\lambda\Pi$-calculus as a universal proof language which can express proof without losing their computational properties [9]. To our knowledge, these works do not focus on the readability of proofs.

7 Conclusions and Further Work

Over the last years a lot of effort has been invested in combining the power of automated and interactive theorem proving: interactive theorem provers are now equipped with trusted support for SAT solving, SMT solving, resolution method, etc [7,6]. These combinations open new frontiers for applications of theorem proving in software and hardware verification, but also in formalization of mathematics and for helping mathematicians in everyday practice. Exporting proofs in formats such as the presented one opens new possibilities for ex-

porting readable mathematical knowledge from automated theorem provers to interactive theorem provers. In the presented approach, the task of generating object-level proofs for proof assistants or proofs expressed in natural language is removed from theorem provers (where it would be hard-coded) and, thanks to the interchange XML format, delegated to simple XSLT style-sheets, which are very flexible and additional XSLT style-sheets (for additional target formats) can be developed without changing the prover. Also, different automated theorem provers can benefit from this suite, as they don't have to deal with specifics of proof assistants.

The presented proof representation is not intended to serve as "the mathematical vernacular". However, it can cover a significant portion of many interesting mathematical theories while it is very simple.

Often, communication between an interactive theorem prover and an external automated theorem prover is supported by a verified, trusted interface which enables direct calling to the prover. On the other hand, our work yields a common format which can be generated by different automated theorem provers and from which proofs for different interactive theorem provers can be generated. The advantage of our approach relies on the fact that the proof which is exported is not just a certificate, it is meant to be human readable.

The current version of the presented XML suite does not support function symbols of arity greated than 0. For the future work, we are planning to add that support to the proof format and to our ArgoCLP prover.

In the current version, for simplicity, the generated Isar and Coq proofs use tactics stronger than necessary. We will try to completely move to basic proofs steps while keeping simplicity of proofs. Beside planning to further improve existing XSLT style-sheets, we are also planning to implement support for additional target languages such as OMDoc.

Acknowledgements. We are grateful to Filip Marić for his feedback and advices on earlier phases of this work.

References

1. Jeremy Avigad, Edward Dean, and John Mumma. A Formal System for Euclid's Elements. *The Review of Symbolic Logic*, 2009.
2. Henk Barendregt and Freek Wiedijk. The Challenge of Computer Mathematics. *Philosophical Transactions of the Royal Society*, 363(1835):2351–2375, 2005.
3. Marc Bezem and Thierry Coquand. Automating Coherent Logic. In Geoff Sutcliffe and Andrei Voronkov, editors, *12th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning — LPAR 2005*, volume 3835 of *Lecture Notes in Computer Science*. Springer-Verlag, 2005.
4. Marc Bezem and Dimitri Hendriks. On the Mechanization of the Proof of Hessenberg's Theorem in Coherent Logic. *Journal of Automated Reasoning*, 40(1), 2008.
5. Jasmin Christian Blanchette. Redirecting Proofs by Contradiction. In Jasmin Christian Blanchette and Josef Urban, editors, *Third International Workshop*

on Proof Exchange for Theorem Proving, PxTP 2013, Lake Placid, NY, USA, June 9-10, 2013, volume 14 of *EPiC Series*, pages 11–26. EasyChair, 2013.
6. Jasmin Christian Blanchette, Sascha Böhme, and Lawrence C. Paulson. Extending Sledgehammer with SMT Solvers. *Journal of Automated Reasoning*, 51(1):109–128, 2013.
7. Jasmin Christian Blanchette, Lukas Bulwahn, and Tobias Nipkow. Automatic Proof and Disproof in Isabelle/HOL. In Cesare Tinelli and Viorica Sofronie-Stokkermans, editors, *Frontiers of Combining Systems, 8th International Symposium, Proceedings*, volume 6989 of *Lecture Notes in Computer Science*, pages 12–27. Springer, 2011.
8. Andreas Blass. Topoi and Computation. *Bulletin of the EATCS*, 36:57–65, 1998.
9. Mathieu Boespflug, Quentin Carbonneaux, and Olivier Hermant. The $\lambda\Pi$-calculus Modulo as a Universal Proof Language. In *Second Workshop on Proof Exchange for Theorem Proving (PxTP)*, volume 878 of *CEUR Workshop Proceedings*, pages 28–43. CEUR-WS.org, 2012.
10. Nicolaas Govert de Bruijn. The Mathematical Vernacular, a Language for Mathematics with Typed Sets. In Dybjer et al., editor, *Proceedings of the Workshop on Programming Languages*, 1987.
11. John Fisher and Marc Bezem. Skolem Machines and Geometric Logic. In Cliff B. Jones, Zhiming Liu, and Jim Woodcock, editors, *4th International Colloquium on Theoretical Aspects of Computing — ICTAC 2007*, volume 4711 of *Lecture Notes in Computer Science*. Springer-Verlag, 2007.
12. Mohan Ganesalingam and William Timothy Gowers. A fully automatic problem solver with human-style output. *CoRR*, abs/1309.4501, 2013.
13. Cezary Kaliszyk and Alexander Krauss. Scalable LCF-Style Proof Translation. In Sandrine Blazy, Christine Paulin-Mohring, and David Pichardie, editors, *Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings*, volume 7998 of *Lecture Notes in Computer Science*, pages 51–66. Springer, 2013.
14. Cezary Kaliszyk and Josef Urban. PRocH: Proof Reconstruction for HOL Light. In Maria Paola Bonacina, editor, *Automated Deduction - CADE-24 - 24th International Conference on Automated Deduction*, volume 7898 of *Lecture Notes in Computer Science*, pages 267–274. Springer, 2013.
15. Chantal Keller and Benjamin Werner. Importing HOL Light into Coq. In *ITP*, pages 307–322, 2010.
16. Michael Kohlhase. An OMDoc primer. In *OMDoc – An Open Markup Format for Mathematical Documents [version 1.2]*, volume 4180 of *Lecture Notes in Computer Science*, pages 33–34. Springer Berlin Heidelberg, 2006.
17. Dongwon Lee and Wesley W. Chu. Comparative analysis of six xml schema languages. *SIGMOD Record*, 29(3):76–87, 2000.
18. Steven Obua and Sebastian Skalberg. Importing HOL into Isabelle/HOL. In Ulrich Furbach and Natarajan Shankar, editors, *Automated Reasoning*, volume 4130 of *Lecture Notes in Computer Science*, page 298–302. Springer Berlin Heidelberg, 2006.
19. Andrew Polonsky. *Proofs, Types and Lambda Calculus*. PhD thesis, University of Bergen, 2011.
20. Piotr Rudnicki. Obvious inferences. *Journal of Automated Reasoning*, 3(4):383–393, 1987.
21. Wolfram Schwabhäuser, Wanda Szmielew, and Alfred Tarski. *Metamathematische Methoden in der Geometrie*. Springer-Verlag, Berlin, 1983.

22. Steffen Juilf Smolka and Jasmin Christian Blanchette. Robust, Semi-Intelligible Isabelle Proofs from ATP Proofs. In Jasmin Christian Blanchette and Josef Urban, editors, *Third International Workshop on Proof Exchange for Theorem Proving, PxTP 2013*, volume 14 of *EPiC Series*, pages 117–132. EasyChair, 2013.
23. Sana Stojanović, Julien Narboux, and Predrag Janičić. Synergy Between Interactive and Automated Theorem Proving in Formalization of Mathematical Knowledge: A Case Study of Tarski's Geometry. *Submitted for publication*, 2014.
24. Sana Stojanović, Vesna Pavlović, and Predrag Janičić. A Coherent Logic Based Geometry Theorem Prover Capable of Producing Formal and Readable Proofs. In Pascal Schreck, Julien Narboux, and Jürgen Richter-Gebert, editors, *Automated Deduction in Geometry*, volume 6877 of *Lecture Notes in Computer Science*. Springer, 2011.
25. G. Sutcliffe. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning*, 43(4):337–362, 2009.
26. Alfred Tarski and Steven Givant. Tarski's system of geometry. *The Bulletin of Symbolic Logic*, 5(2), June 1999.
27. Markus Wenzel. Isar - A Generic Interpretative Approach to Readable Formal Proof Documents. In Yves Bertot, Gilles Dowek, André Hirschowitz, C. Paulin, and Laurent Théry, editors, *Theorem Proving in Higher Order Logics (TPHOLs'99)*, volume 1690 of *Lecture Notes in Computer Science*, pages 167–184. Springer, 1999.
28. Freek Wiedijk. Mathematical Vernacular. Unpublished note. http://www.cs.ru.nl/~freek/notes/mv.pdf, 2000.
29. Freek Wiedijk, editor. *The Seventeen Provers of the World*, volume 3600 of *Lecture Notes in Computer Science*. Springer, 2006.
30. Freek Wiedijk. A Synthesis of the Procedural and Declarative Styles of Interactive Theorem Proving. *Logical Methods in Computer Science*, 8(1), 2012.