

Isabelle/DOF

Extended Abstract and Tool Demonstration

Achim D. Brucker   



University of Exeter, Department of Computer Science, UK

Idir Ait-Sadoune  

Université Paris-Saclay, CentraleSupélec, LMF, France

Nicolas Méric  

Université Paris-Saclay, CNRS, LMF, France

Burkhart Wolff  

Université Paris-Saclay, CNRS, LMF, France

Abstract

Isabelle/DOF is an ontology framework built on top of Isabelle/HOL. It enables the formal development of ontologies and continuous conformity-checking of integrated documents that contain formal developments. These integrated documents can include text, code, definitions, proofs, and user-programmed constructs, supporting a wide range of formal methods within Isabelle/HOL. In contrast to conventional ontology definition languages like OWL, Isabelle/DOF generates *theories* of meta-data which were used, when executable, for runtime-checks enforcing the conformity of documents to the given ontology. Isabelle/DOF is designed to trace typed metadata in integrated documents, providing support for navigation within Isabelle's IDE and enhancing the document generation process.

In this extended abstract and tool demonstration, we will briefly describe Isabelle/DOF, as recently published in the Archive of Formal Proofs [2].

2012 ACM Subject Classification Theory of computation → Interactive proof systems; Computing methodologies → Ontology engineering; Information systems → Ontologies; Theory of computation → Interactive proof systems; Theory of computation → Higher order logic

Keywords and phrases Ontologies, Formal Development, Formal Documents, Isabelle/HOL, Software Engineering

Digital Object Identifier 10.4230/LIPICs...

1 Introduction

Isabelle/DOF [1] is a framework extending Isabelle/HOL [6] with ontological modelling and document authoring features. Its development has been motivated by the vision to bridge the gap between formal verification and document authoring, enabling users to create structured documents with formal links between informal and formal content.

The illustration in Figure 1 gives an overview over the architecture: Isabelle/DOF extends the Isabelle/HOL core by constructs allowing for the specification of formal ontologies (right-hand side) using an Ontology Definition Language (ODL) and it also provides documentation constructs (left-hand side) for text-, definition-, term-, proof-, code-, and user-defined elements. In particular, Isabelle/DOF can enforce document conformance to a given ontology.

Isabelle/DOF is a new kind of ontological modelling and document validation tool. In contrast to conventional languages like OWL [7] and development environments such as Protégé [5], it brings forward our concept of *deep ontologies*, i. e., ontologies represented inside a logical language such as HOL rather than a conventional programming language like Java. Deep ontologies generate strongly typed meta-information specified in form of

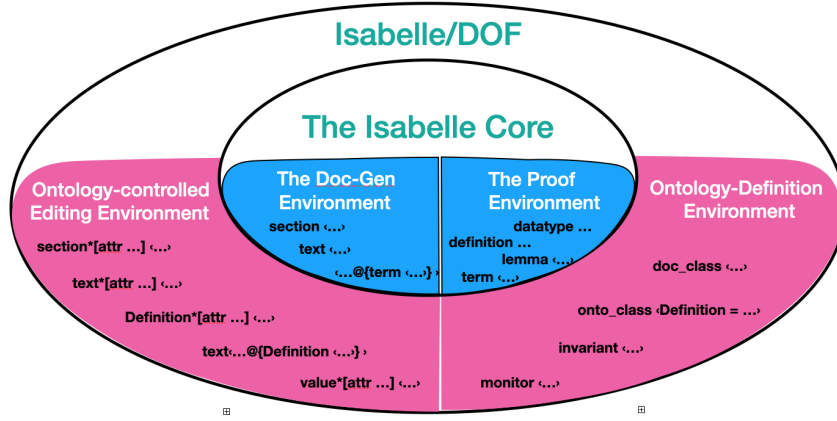


© Author: Please provide a copyright holder;

licensed under Creative Commons License CC-BY 4.0

Leibniz International Proceedings in Informatics

LIPICs Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany



■ **Figure 1** The Ontology Environment Isabelle/DOF.

HOL-theories allowing both efficient execution **and** logical reasoning about meta-data. They generate a particular form of checked annotations called *anti-quotation* to be used inside code and texts. Deeply integrated into the Isabelle ecosystem [4], thus permitting continuous checking and validation also allow ontology-aware navigation inside large documents with both formal and informal content.

2 Ontological Annotated Text in Isabelle/DOF: An Example

We will detail this by example of annotated text in a document. We will assume a given ontology. The Isabelle’s `text` ... -element or `ML` ... -code-elements are extended to the corresponding Isabelle/DOF elements:

```
text*[label::cid, attrib_def1,...,attrib_defn]<... annotated text ... >
ML*[label::cid, attrib_def1,...,attrib_defn]<... annotated code ... >
```

where *cid* is an identifier of an ontological class introduced in the underlying ontology together with attributes belonging to this class defined in ODL. For example, if an ontology provides a concept *Definition*, we can do the following:

```
text*[safe::Definition, name=safety]<Our system is safe if the following holds ...>
```

The Isabelle/DOF command `text*` creates an instance *safe* of the ontological class *Definition* with the attribute *name* and associates it to the text inside the <...>-brackets.

We call the content of these brackets the *text-context* (or *ML-context*, respectively). Of particular interest for this paper is the ability to generate a kind of semantic macro called *anti-quotation*, which is continuously checked and whose evaluation uses information from the background theory of this text element.

For example, we might refer to the above definition in another text element:

```
text*[...]<As stated in @{Definition <safe>}, . . . >
```

where Isabelle/DOF checks on-the-fly that the reference “safe” is indeed defined in the document and has the right type (it is not an *Example*, for example), generates navigation information (i.e. hyperlinks to *safe* as well as the ontological description of *Definition* in the Isabelle IDE). Finally, it generates specific documentation markup in the generated PDF document, e.g.:

```
As stated in Def. 3.11 (safety), ...
```

where the underline may be blue because the layout description configured for this ontology says so. Moreover, this can be used to generate an index containing, for example, all definitions. Similarly, this also works for an ontology providing concepts such as “objectives”, “claims” and “evidences”. Invariants may be stated in an ontological class that finally enforce properties such as that “all claims correspond to evidences in the global document”, and “all evidences must contain at least one proven theorem”, etc. In contrast to a conventional type-setting system, Isabelle can additionally type-check formulas, so for example:

```
text*[...]⟨The safety distance is defined by @{term  $dist_{safe} = \text{sqrt}(d - a \cdot (\Delta t)^2$ )}\dots⟩
```

where functions like $dist_{safe}$, sqrt , etc., have to be defined in the signature and logical context or background theory of this formula. Anti-quotations as such are not a new concept in Isabelle; the system comes with a couple of hand-programmed anti-quotations like $\text{@}\{term \dots\}$. In contrast, Isabelle/DOF *generates* anti-quotations from ontological classes in ODL, together with checks generated from data-constraints (or: class invariants) specified in HOL.

3 Ontological Classes: An Example

In the previous section, we said that “The Isabelle/DOF command **text*** creates an instance *safe* of the ontological class *Definition*”.

Technically, this means that a term for an extensible record is generated from an ontological definition like:

```
onto-class Definition = Math_Content +
  name      :: string
  is_formal :: bool <- False
invariant L :: name ≠ ""
```

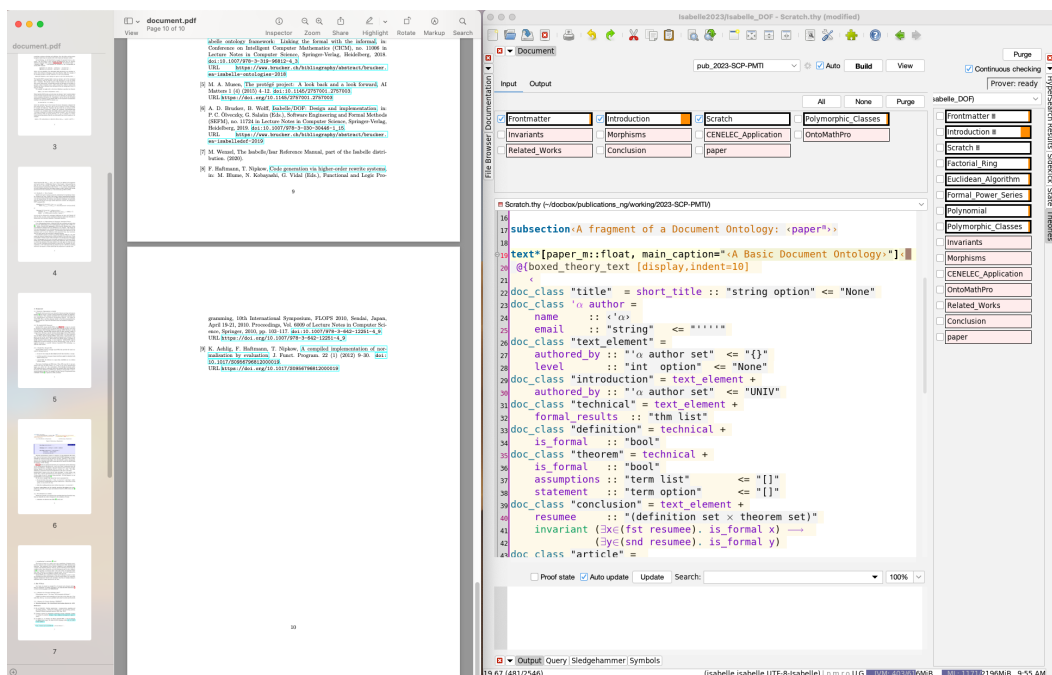
having the record fields *name* of type *string* (set to “safety”) and the boolean field of default-value *False*. Ontological classes may have invariants expressing data-constraints, which were checked at object-creation time.

An *Ontology* is just a conventional theory file containing a collection of ontological definitions. Note that since an ontology is represented internally by a theory extension in the logical context of Isabelle/HOL, it is possible to formally verify properties over ontological classes: for example, one may define a conversion function between the instances of two different ontological classes and formally prove, that this so-called *ontological mapping* is injective and preserves the invariant.

Ontologies may define all sorts of meta-data useful for structuring, queries and exchange-formats. The most common use of ontologies, so far, consists in the *generation* of LaTeX and PDF documents. For this end, Isabelle/DOF provides a number of advanced technologies to configure the documentation generator and thus to provide specific layout for instances of particular ontological classes. The interested reader may consult the Isabelle/DOF reference manual provided by the Archive of Formal Proofs [2].

4 Conclusion

We briefly presented how Isabelle/DOF integrates ontological document authoring into Isabelle/HOL. This integration is also supported by Isabelle’s document generation producing PDF output (see Figure 2). The consistency checking of Isabelle/DOF ensures that all references are valid and pointing to elements of the correct type. As such, Isabelle/DOF



■ **Figure 2** Editing a paper in Isabelle/DOF together with its PDF preview.

significantly contributes to the overall quality and consistency of authored documents, in particularly when developed collaboratively.

Availability. Isabelle/DOF is available in the Archive of Formal Proofs (AFP) as entry “Isabelle/DOF” [2] under a 2-clause BSD-license (SPDX-License-Identifier: BSD-2-Clause). Additional extensions (e.g., ontologies, document templates, and examples) are available on Zenodo [3]. The development version is available at https://git.logicalhacking.com/Isabelle_DOF/Isabelle_DOF/.

References

- 1 Achim D. Brucker, Idir Ait-Sadoun, Paolo Crisafulli, and Burkhart Wolff. Using the Isabelle ontology framework: Linking the formal with the informal. In *Intelligent Computer Mathematics (CICM)*, number 11006 in LNCS. Springer, 2018. doi:10.1007/978-3-319-96812-4_3.
- 2 Achim D. Brucker, Nicolas Méric, and Burkhart Wolff. Isabelle/DOF. *Archive of Formal Proofs*, April 2024. https://isa-afp.org/entries/Isabelle_DOF.html, Formal proof development.
- 3 Achim D. Brucker and Burkhart Wolff. Isabelle/DOF. Zenodo, August 2019. doi:10.5281/zenodo.3370482.
- 4 Achim D. Brucker and Burkhart Wolff. Isabelle/DOF: Design and implementation. In Peter C. Ölveczky and Gwen Salaün, editors, *Software Engineering and Formal Methods (SEFM)*, number 11724 in LNCS. Springer, 2019. doi:10.1007/978-3-030-30446-1_15.
- 5 Mark A. Musen. The protégé project: A look back and a look forward. *AI Matters*, 1(4):4–12, jun 2015. doi:10.1145/2757001.2757003.
- 6 Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL—A Proof Assistant for Higher-Order Logic*, volume 2283 of LNCS. Springer, 2002. doi:10.1007/3-540-45949-9.
- 7 Kunal Sengupta and Pascal Hitzler. Web ontology language (OWL). In *Encyclopedia of Social Network Analysis and Mining*, pages 2374–2378. 2014. doi:10.1007/978-1-4614-6170-8_113.