IDE: Introduction, Destruction, Elimination

Mihails Milehins

March 17, 2025

Abstract

The document presents a reference manual for the command **mk_ide** developed for the object logic *Isabelle/HOL* (e.g., see [2]) of the formal proof assistant *Isabelle* [3]. The command provides means for the automated synthesis of the introduction, destruction and elimination rules from the definitions of predicates stated in Isabelle/HOL.

Acknowledgements

The author would like to acknowledge the assistance that he received from the users of the mailing list of Isabelle in the form of answers given to his general queries.

Furthermore, the author would like to acknowledge the positive impact of [4] and [7] on his ability to code in Isabelle/ML. Moreover, the author would like to acknowledge the positive role that numerous Q&A posted on the Stack Exchange network (especially Stack Overflow and TeX Stack Exchange) played in the development of this work.

The author would also like to express gratitude to all members of his family and friends for their continuous support.

Contents

1 Introduction

1.1 Background

This document presents a reference manual for the framework IDE. The framework IDE can be used for the automated synthesis of the introduction, destruction and elimination rules from the definitions of predicates stated in the object logic Isabelle/HOL of the proof assistant Isabelle.

The primary functionality of the framework is available via the *Isabelle/Isar* [5, 6] command **mk-ide**. Given a definition of a predicate in Isabelle/HOL, the command can synthesize introduction, destruction and elimination rules for this definition. The rules are stated in a certain predetermined format that is meant to be both natural and convenient for the end user and the tools for classical reasoning available in Isabelle/HOL.

1.2 Related and previous work

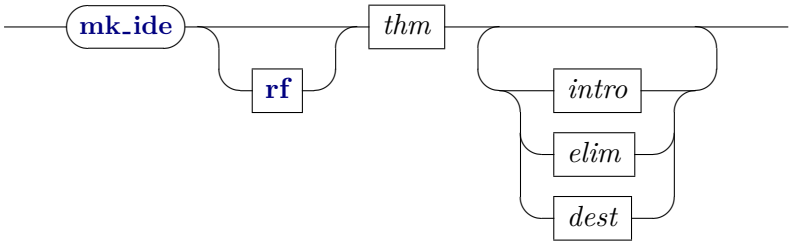
The standard distribution of Isabelle provides the *attribute elim-format* [8] that can be used for the conversion of the destruction rules to the elimination rules. The primary functionality of this attribute is reused in the implementation of the command **mk-ide**.

Furthermore, Isabelle offers several definitional packages that provide similar rules automatically for the constants created by these definitional packages [8]. However, to the best knowledge of the author, none of these packages can generate such rules for arbitrary predicates. Perhaps, in the future, the approaches can be unified or integrated in some manner.

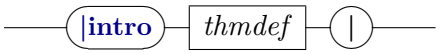
2 Syntax

This subsection presents the syntactic categories that are associated with the command **mk-ide**. It is important to note that the presentation is only approximate.

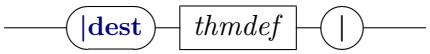
mk-ide : *local-theory* → *local-theory*



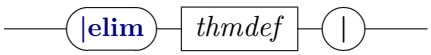
intro



dest



elim



mk-ide (**rf**) *def-thm* |*intro name*[*attrbs*]| converts the definition *def-thm* into an introduction rule, followed by the application of the functionality associated with the optional keyword **rf** and the attributes *attrbs* to this rule. The result of the application of the attributes *attrbs* is stored in the local context under the name *name*. *def-thm* is meant to be a fact that consists of exactly one theorem of the form

$$A\ a_1\ \ldots\ a_n \simeq f_1\ a_1\ \ldots\ a_n \wedge \ldots \wedge f_m\ a_1\ \ldots\ a_n,$$

where n and m are natural numbers, A is a constant predicate in n arguments, $\simeq$ is either the meta-logic equality or the object logic equality, $a_1\ \ldots\ a_n$ are schematic variables and $f_1\ \ldots\ f_m$ are suitable predicates in n arguments (however, there are further implicit restrictions). The resulting introduction rule is expected to be stated in the format

$$f_1\ a_1\ \ldots\ a_n \Longrightarrow \ldots \Longrightarrow f_m\ a_1\ \ldots\ a_n \Longrightarrow A\ a_1\ \ldots\ a_n$$

prior to the application of the functionality associated with the keyword **rf** and the attributes *attrbs*. If the optional keyword **rf** is passed as an argument to the command, then the output of the command (prior to the application of the attributes) is formatted using an algorithm associated with the attribute *rule-format* [8].

mk-ide (**rf**) *def-thm* |*dest name*[*attrbs*]| converts the definition *def-thm* into one or more destruction rules, followed by the application of the functionality associated with the optional keyword **rf** and the attributes *attrbs* to each destruction rule. Given the theorem *def-thm* in the format described above, the command provides m destruction rules of the form

$$A\ a_1\ \ldots\ a_n \Longrightarrow f_i\ a_1\ \ldots\ a_n$$

for each *1≤i≤m* prior to the application of the functionality associated with the keyword **rf** and the attributes *attrbs*.

mk-ide (**rf**) *def-thm* |*elim name*[*attrbs*]| converts the definition *def-thm* into an elimination rule, followed by the application of the functionality associated with the optional keyword **rf** and the attributes *attrbs* to each destruction rule. Given the theorem *def-thm* in the format described above, the elimination rule has the format

$$A\ a_1\ \ldots\ a_n \Longrightarrow (f_1\ a_1\ \ldots\ a_n \Longrightarrow \ldots \Longrightarrow f_m\ a_1\ \ldots\ a_n \Longrightarrow P) \Longrightarrow P$$

prior to the application of the functionality associated with the keyword **rf** and the attributes *attrbs*.

It is possible to combine the keywords |*intro*, |*dest* and |*elim* in a single invocation of the command.

3 Examples

In this section, some of the capabilities of the framework IDE are demonstrated by example. The example is based on the definition of the constant *monoid* from the standard library of Isabelle/HOL [1] and given by

$$monoid\ f\ z \equiv semigroup\ f \wedge (\forall a.\ f\ z\ a = a) \wedge (\forall a.\ f\ a\ z = a)$$

mk-ide rf *monoid-def*[*unfolded monoid-axioms-def*]
|*intro monoidI*|
|*dest monoidD*|
|*elim monoidE*|

The invocation of the command **mk-ide** provides the theorem *monoidI* given by

$$[\![semigroup\ f;\ \bigwedge a.\ f\ z\ a = a;\ \bigwedge a.\ f\ a\ z = a]\!] \Longrightarrow monoid\ f\ z,$$

the fact *monoidD* given by

$$monoid\ f\ z \Longrightarrow semigroup\ f$$
$$monoid\ f\ z \Longrightarrow f\ z\ a = a$$
$$monoid\ f\ z \Longrightarrow f\ a\ z = a$$

and the theorem *monoidE* given by

$$[\![monoid\ f\ z;\ [\![semigroup\ f;\ \bigwedge a.\ f\ z\ a = a;\ \bigwedge a.\ f\ a\ z = a]\!] \Longrightarrow W]\!] \Longrightarrow W.$$

References

[1] Isabelle/HOL Standard Library, 2020. URL https://isabelle.in.tum.de/website-Isabelle2020/dist/library/HOL/HOL/index.html.

[2] O. Kunčar and A. Popescu. Comprehending Isabelle/HOL's Consistency. In H. Yang, editor, *Programming Languages and Systems*, volume 10201, pages 724–749. Springer, Heidelberg, 2017. ISBN 978-3-662-54433-4.

[3] L. C. Paulson. Natural Deduction as Higher-Order Resolution. *The Journal of Logic Programming*, 3(3):237–258, 1986.

[4] C. Urban. *The Isabelle Cookbook: A Gentle Tutorial for Programming Isabelle/ML*. 2019.

[5] M. Wenzel. Isar — A Generic Interpretative Approach to Readable Formal Proof Documents. In Y. Bertot, G. Dowek, L. Théry, A. Hirschowitz, and C. Paulin-Mohring, editors, *Theorem Proving in Higher Order Logics*, volume 1690, pages 167–183. Springer, Heidelberg, 1999. ISBN 978-3-540-66463-5.

[6] M. Wenzel. Isabelle/Isar — a Generic Framework for Human-Readable Proof Documents. *Studies in Logic, Grammar and Rhetoric*, 10(23):277–297, 2007.

[7] M. Wenzel. The Isabelle/Isar Implementation. 2019.

[8] M. Wenzel. The Isabelle/Isar Reference Manual. 2019.