

Towards a Tool for Featherweight OCL: A Case Study On Semantic Reflection

Delphine Longuet, Frederic Tuong
and
Burkhardt Wolff

Université Paris-Sud

Abstract

We show how modern proof environments comprising code generators and reflection facilities can be used for the effective construction of a tool for OCL. For this end, we define a UML/OCL meta-model in HOL, a meta-model for Isabelle/HOL in HOL, and a compiling function between them over the vocabulary of the libraries provided by Featherweight OCL. We use the code generator of Isabelle to generate executable code for the compiler, which is bound to a USE tool-like syntax integrated in Isabelle/Featherweight OCL. It generates for an arbitrary class model an object-oriented datatype theory and proves the relevant properties for casts, type-tests, constructors and selectors automatically.

**Isabelle:
The “Eclipse” of
Formal Methods
Tools
(not just a theorem prover)**

The Isabelle System Framework

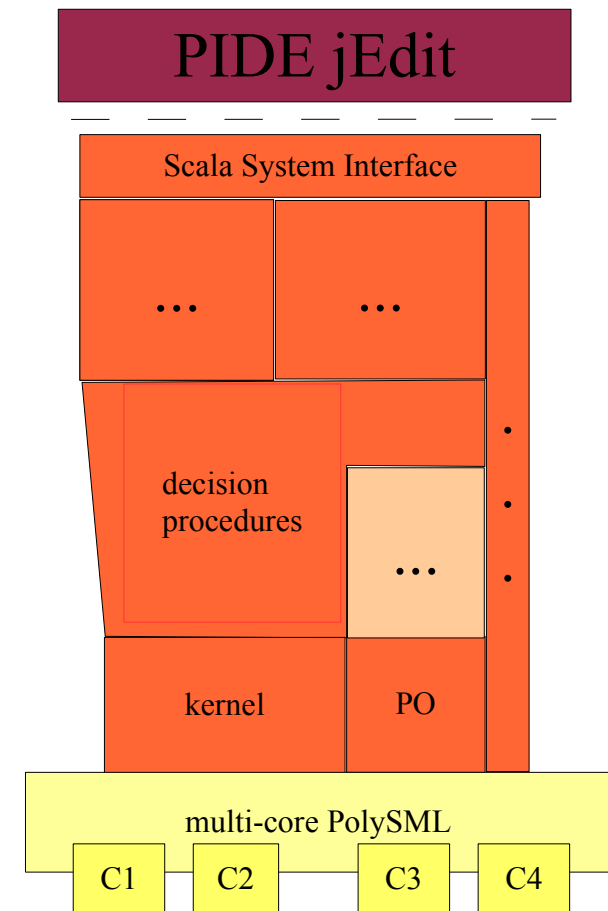
Isabelle Architecture

- Modern Isabelle Architecture consists of 5 identifiable layers
 - SML layer
 - Kernel & Proof Object Layer
 - Tactic Layer and decision procedures
 - Isar Engine
 - PIDE Framework and Interface Layer

Isabelle Architecture

Observation:

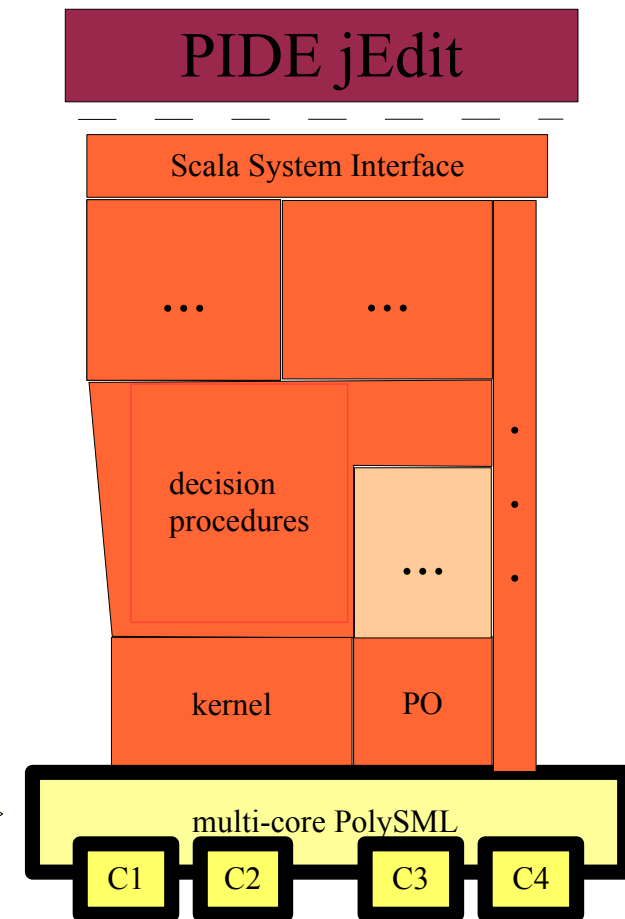
Effective parallelization is a **PERVASIVE PROBLEM**,
that must be addressed



Isabelle Architecture

- In detail:

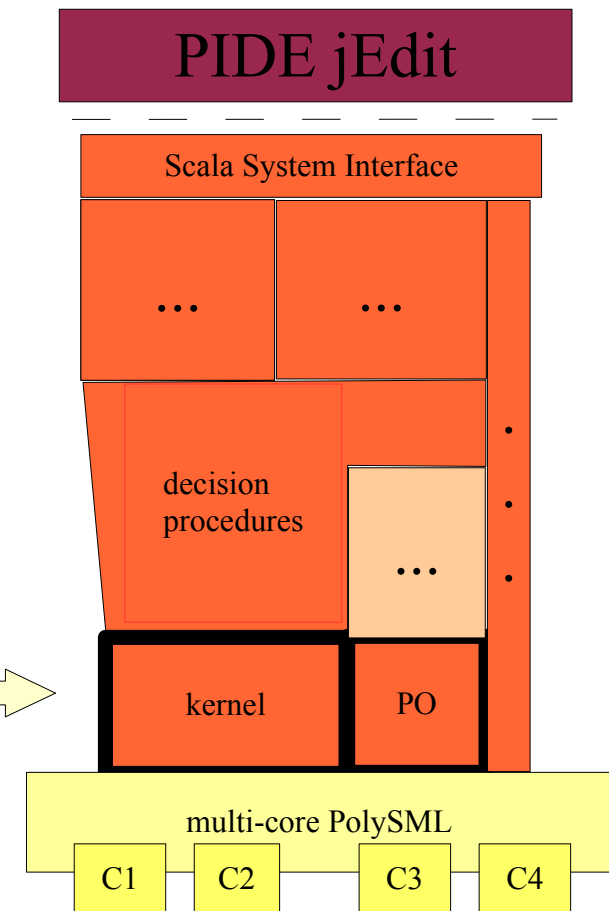
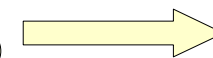
on the execution platform layer



Isabelle Architecture

- In detail:

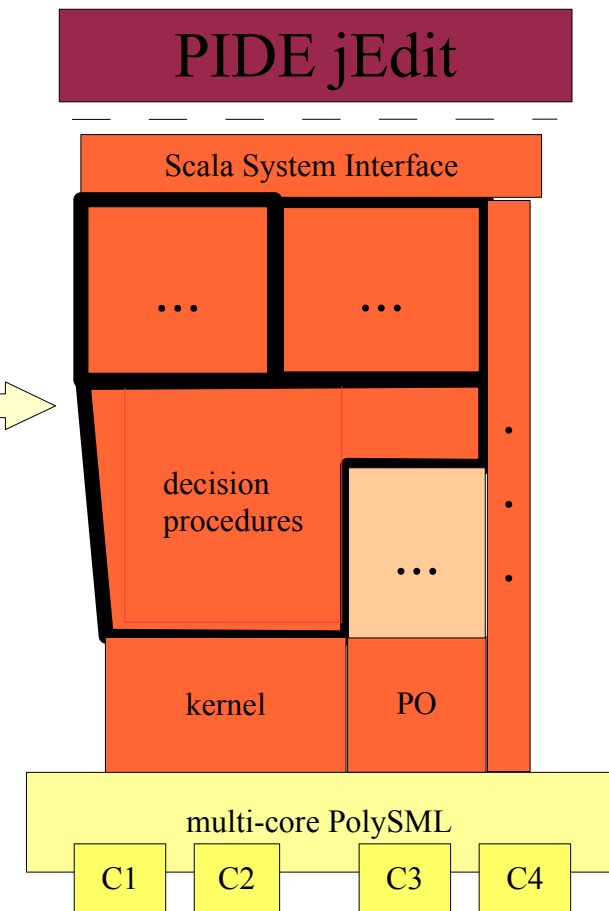
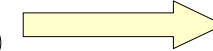
on the kernel layer



Isabelle Architecture

- In detail:

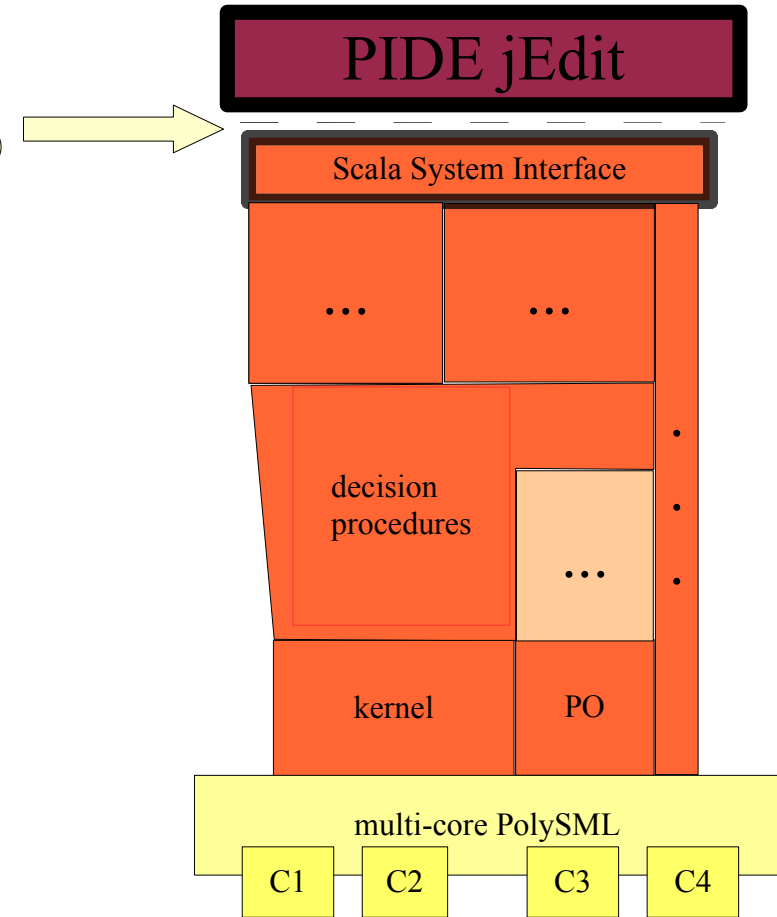
on layer of procedures and packages



Isabelle Architecture

- In detail:

on the interface layer
PIDE framework + Editor



Idea:

Let's reuse this rich
system Framework

(not only the logical
meta-language HOL)

to construct an OCL Tool !

Why is this necessary:

- A concrete class-models semantics consist of
 - an denotational object universe
 - definitions / proofs for accessors
 - definitions / proofs for tests
 - definitions / proofs for casts
 - definitions / proofs for type-tests

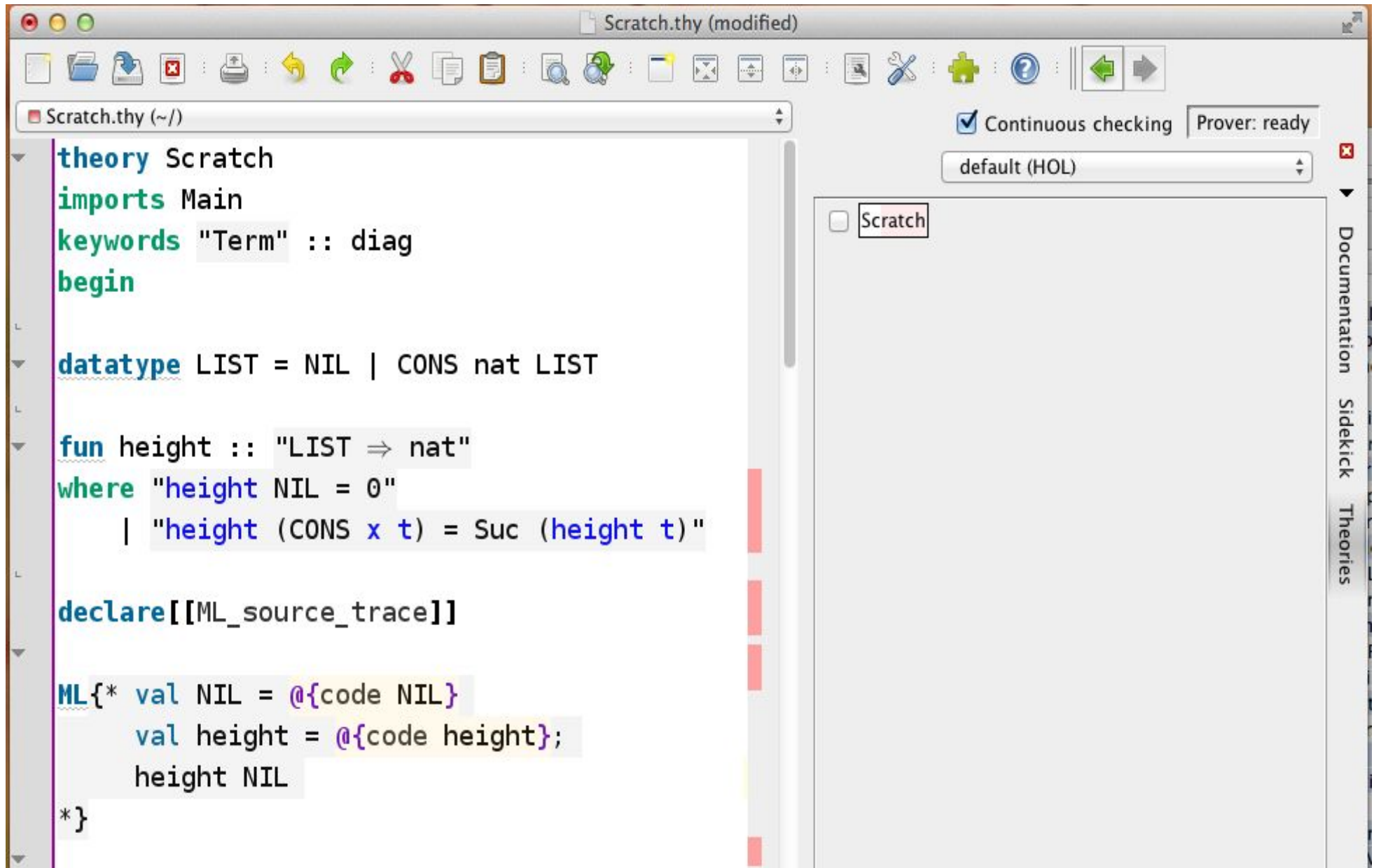
... in Featherweight OCL, this induces hundreds of definitions and proofs.

Lets automate that.

What is the “Output”?

- Well, as in Eclipse, this is not so easy to point out ...
 - a derived theory (thousands of lemmas and their proofs)
 - a set-up for provers
 - a set-up for code-generators
 - a set-up for document generation
 - well, and user-interaction, a GUI, a code-generator that can be reused.

An Isabelle Isar Document



The screenshot shows the Isabelle IDE interface. The main editor displays the following Isar code for a theory named 'Scratch':

```
theory Scratch
imports Main
keywords "Term" :: diag
begin

datatype LIST = NIL | CONS nat LIST

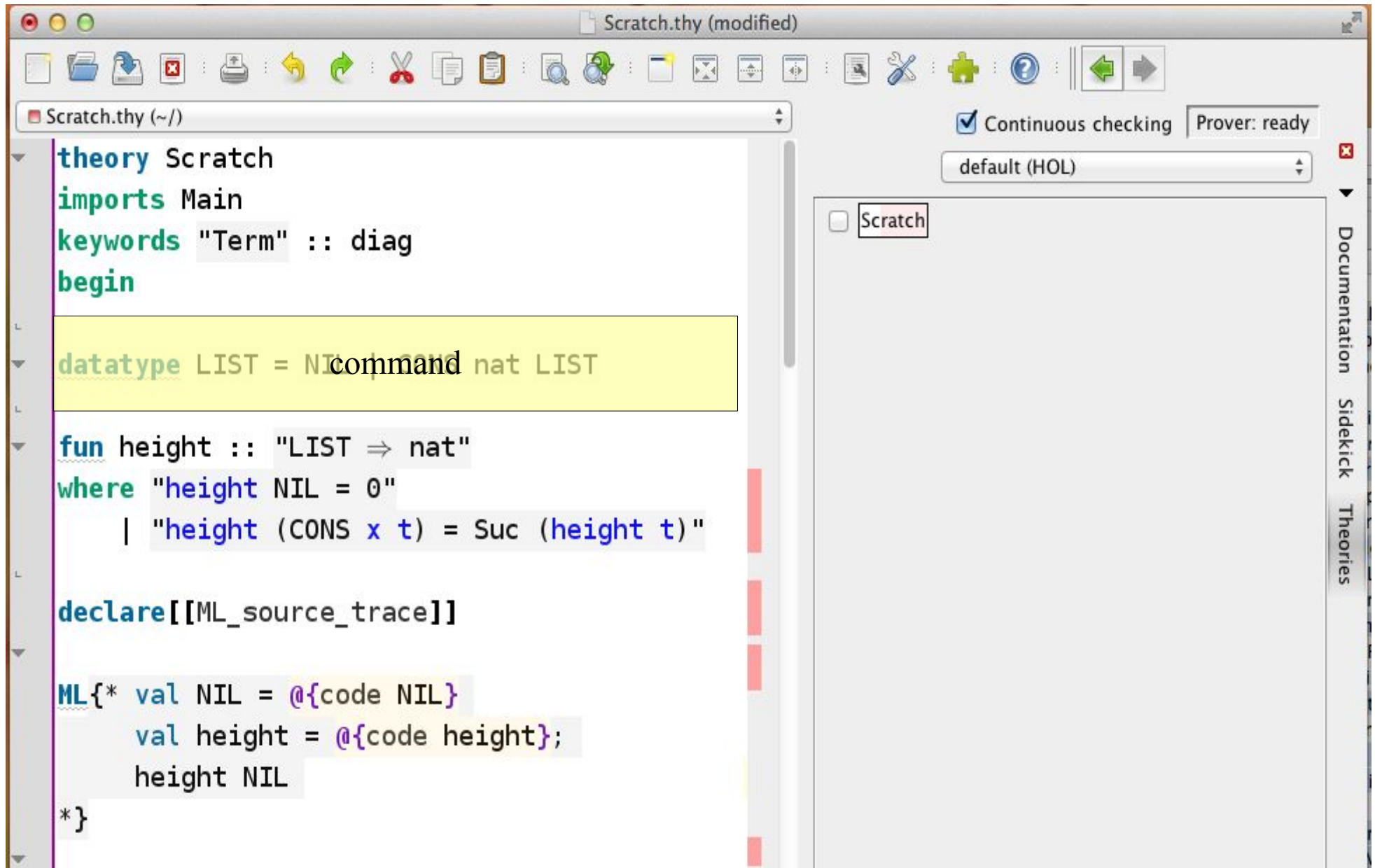
fun height :: "LIST  $\Rightarrow$  nat"
where "height NIL = 0"
      | "height (CONS x t) = Suc (height t)"

declare [[ML_source_trace]]

ML{* val NIL = @{code NIL}
     val height = @{code height};
     height NIL
*}
```

The right-hand side of the interface features a 'Prover: ready' status and a dropdown menu set to 'default (HOL)'. Below this, a list of theories is shown, with 'Scratch' currently selected and highlighted.

An Isabelle Isar Document



The screenshot shows the Isabelle IDE interface. The main editor displays the following code:

```
theory Scratch
imports Main
keywords "Term" :: diag
begin

datatype LIST = NIL | CONS nat LIST

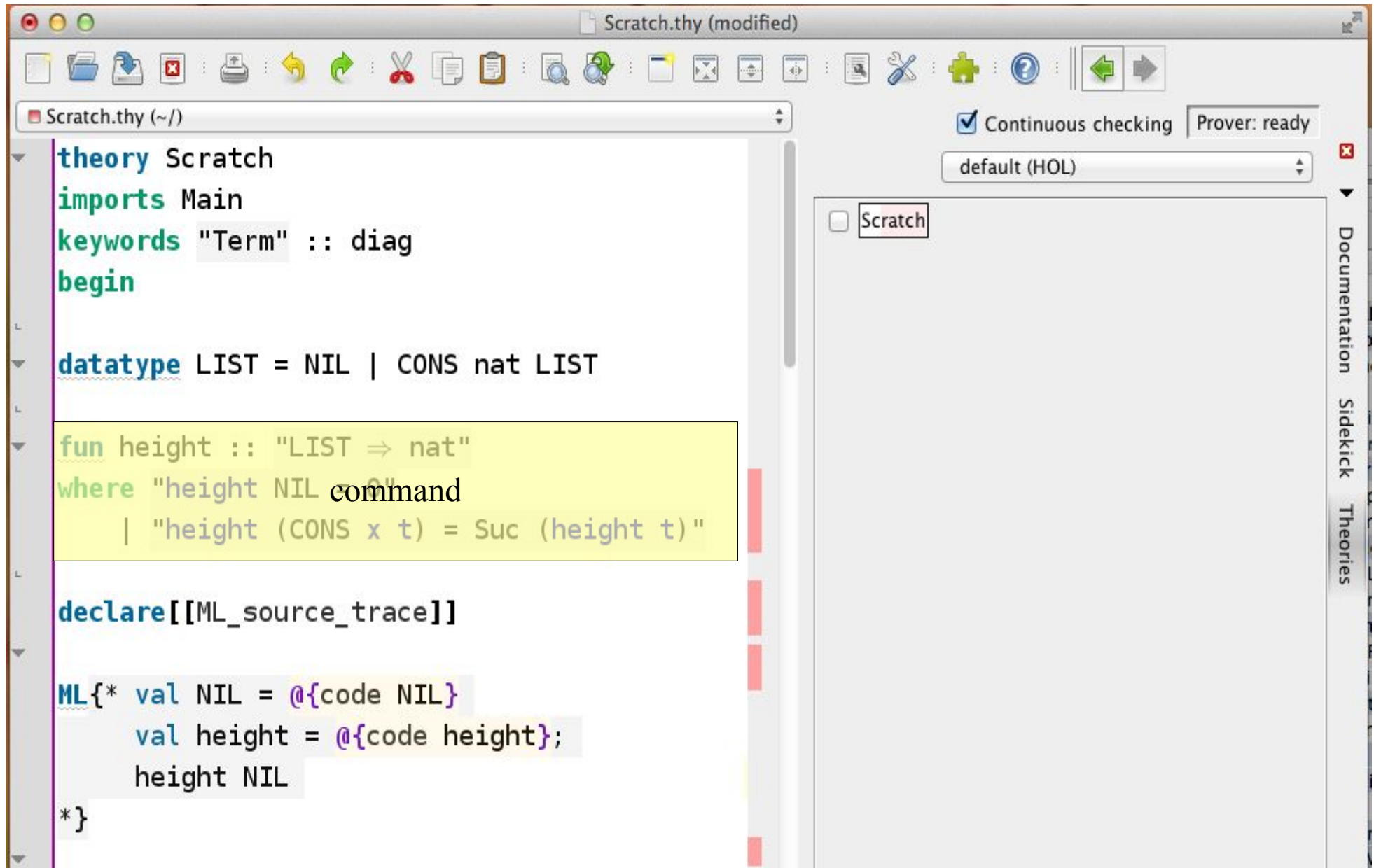
fun height :: "LIST ⇒ nat"
where "height NIL = 0"
      | "height (CONS x t) = Suc (height t)"

declare [[ML_source_trace]]

ML{* val NIL = @{code NIL}
     val height = @{code height};
     height NIL
*}
```

The right-hand side of the interface shows the prover status: "Continuous checking" is checked, and the prover is "ready". The default HOL is selected. A "Scratch" checkbox is visible in the right pane.

An Isabelle Isar Document



The screenshot shows the Isabelle IDE interface. The main editor displays the following Isabelle code:

```
theory Scratch
imports Main
keywords "Term" :: diag
begin

datatype LIST = NIL | CONS nat LIST

fun height :: "LIST  $\Rightarrow$  nat"
where "height NIL command
      | "height (CONS x t) = Suc (height t)"

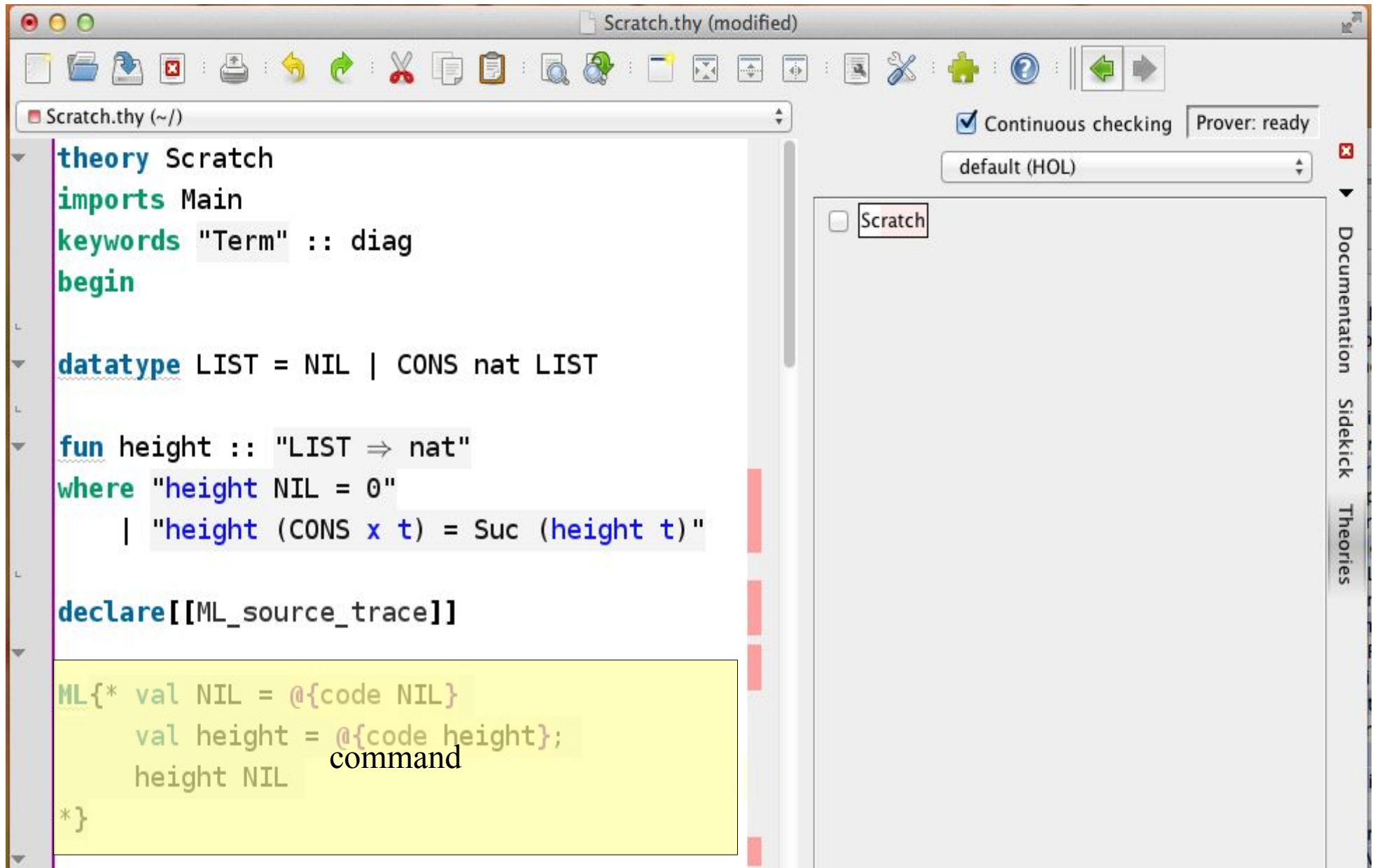
declare [[ML_source_trace]]

ML{* val NIL = @{code NIL}
     val height = @{code height};
     height NIL
*}
```

The code is color-coded: **theory** and **imports** are blue, **keywords** is green, **datatype** is blue, **fun** is blue, **where** is green, **declare** is blue, and **ML** is blue. The function signature and the **ML** block are highlighted in yellow.

The IDE interface includes a toolbar with various icons, a file browser showing 'Scratch.thy (~/)', a 'Continuous checking' checkbox (checked), a 'Prover: ready' status, a dropdown menu set to 'default (HOL)', and a sidebar with 'Documentation', 'Sidekick', and 'Theories' sections. The 'Scratch' theory is currently selected in the sidebar.

An Isabelle Isar Document



The screenshot shows the Isabelle IDE interface. The main editor displays the following Isar code for a theory named 'Scratch':

```
theory Scratch
imports Main
keywords "Term" :: diag
begin

datatype LIST = NIL | CONS nat LIST

fun height :: "LIST  $\Rightarrow$  nat"
where "height NIL = 0"
      | "height (CONS x t) = Suc (height t)"

declare [[ML_source_trace]]

ML{* val NIL = @code NIL
     val height = @code height;
     height NIL
*}
```

The code is color-coded: 'theory' and 'imports' are blue, 'keywords' is green, 'datatype' is blue, 'fun' is blue, 'where' is green, and 'declare' is blue. The ML block is highlighted in yellow. The word 'command' is written in black text below the ML code.

The right-hand side of the IDE shows the 'Prover: ready' status, a dropdown menu set to 'default (HOL)', and a list of theories with 'Scratch' selected.

Instead of Standard Commands ...

- ... we redefine our own **commands** inside the Isabelle Framework
 - for classes
 - ... with attributes and operations
 - ... and types
 - associations
 - invariant declarations
 - operation contracts

The result at a glimpse

The image displays two Isabelle2013-2 IDE windows side-by-side, illustrating the result of a theorem search.

Left Window: Isabelle2013-2 - Scratch.thy

```
theory Scratch imports Main
  keywords "Term" :: diag begin

datatype LIST = NIL | CONS nat LIST

fun height :: "LIST  $\Rightarrow$  nat"
where "height NIL = 0"
      | "height (CONS _ t) = Suc (height t)"

declare [[ML_trace]]

ML{* val NIL = @{code NIL}
     val height = @{code height}
     val _ = height NIL *}

ML{* Outer_Syntax.command @{command_spec "Term"}
  " <Term> reads and prints an arbitrary HOL term "
  (Parse.term >> (Isar_Cmd.print_term o pair [])) *}

Term "height a + height b = height b + height a"

find_theorems

end
```

Right Window: Isabelle2013-2 - Bank_AnalysisModel.thy

```
theory Bank_AnalysisModel imports "../src/UML_Main"
begin

generation_syntax [deep (generation_semantics [analysis,
  [ in SML module_name M (no_signatures) ]

Class Savings < Account
  Attributes maximum : Real End

Association clients
  Between Bank [1 .. *] Role banks
  Client [1 .. *] Role clients End

Context c: Savings
  Inv A : `0.0 <_real (c .maximum)`
  Inv B : `c .moneybalance  $\leq$ _real (c .maximum)
  and 0.0  $\leq$ _real (c .moneybalance)`

(* 2385 generated UML/OCaml theorems, among others: *)
thm up_OclAny_down_Savings_cast up_OclAny_down_Account_cast up_A...
```

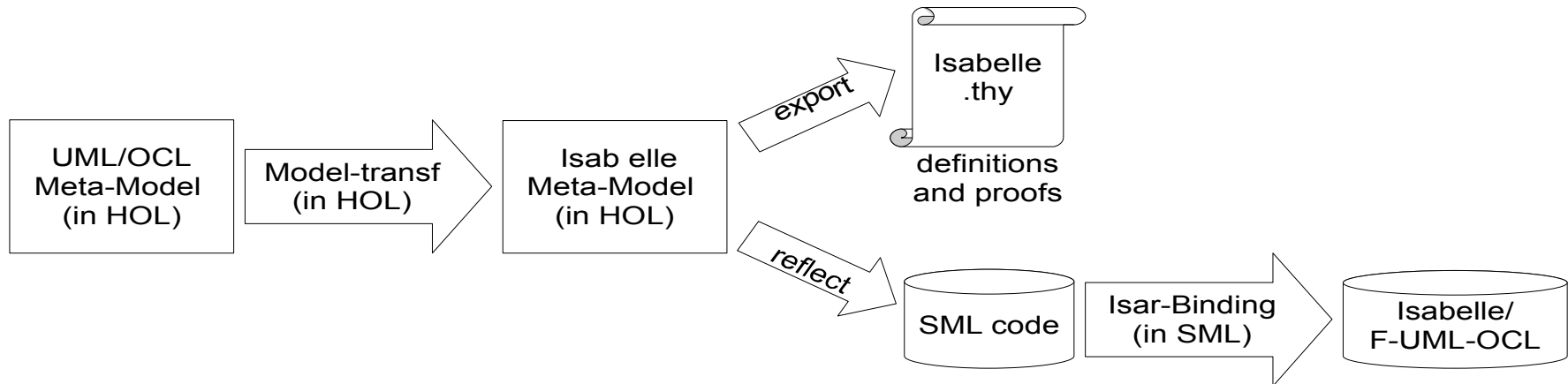
Info Window (over Scratch.thy):

found 14579 theorem(s) (40 displayed):

- Code_Generator.holds: Code_Generator.holds
- Code_Evaluation.arity_term_of_fun:

Bottom status bar: 21,1 (570/589) | 65,86 (3754/4663) | (isabelle,sidekick,UTF-8-Isabelle)Nmr o UG 327/2548MB1:45 PM

The technique at a glimpse



- Note that this “model-transformation” also generates the family of
 - declarations for constructors, accessors casts and tests
 - the proofs for the lemmas (concerning strictness, null, up-cast-down-cast, down-cast-upcast, constructors-destructors, tests, ...)

Experimental Results at a glimpse

- The compiler is about 10000 lines of code
- Some generation info:

c	depth c	depth 5	depth 4	depth 3	depth 2	depth 1
12	(1, c) 14K				(3, 2) 12K	(c , 1) 11K
14	(1, c) 20K			(2, 3) 17K		(c , 1) 16K
20	(1, c) 52K				(4, 2) 39K	(c , 1) 39K
30	(1, c) 155K		(2, 4) 121K		(5, 2) 115K	(c , 1) 115K
39	(1, c) 330K			(3, 3) 240K		(c , 1) 240K
42	(1, c) 409K				(6, 2) 288K	(c , 1) 294K
56	(1, c) 964K				(7, 2) 649K	(c , 1) 661K
62	(1, c) 1.3M	(2, 5) 907K				(c , 1) 882K
72	(1, c) 2M				(8, 2) 1.3M	(c , 1) 1.3M
84	(1, c) 3.3M			(4, 3) 2.1M		(c , 1) 2.1M
90	(1, c) 4.2M				(9, 2) 2.5M	(c , 1) 2.5M

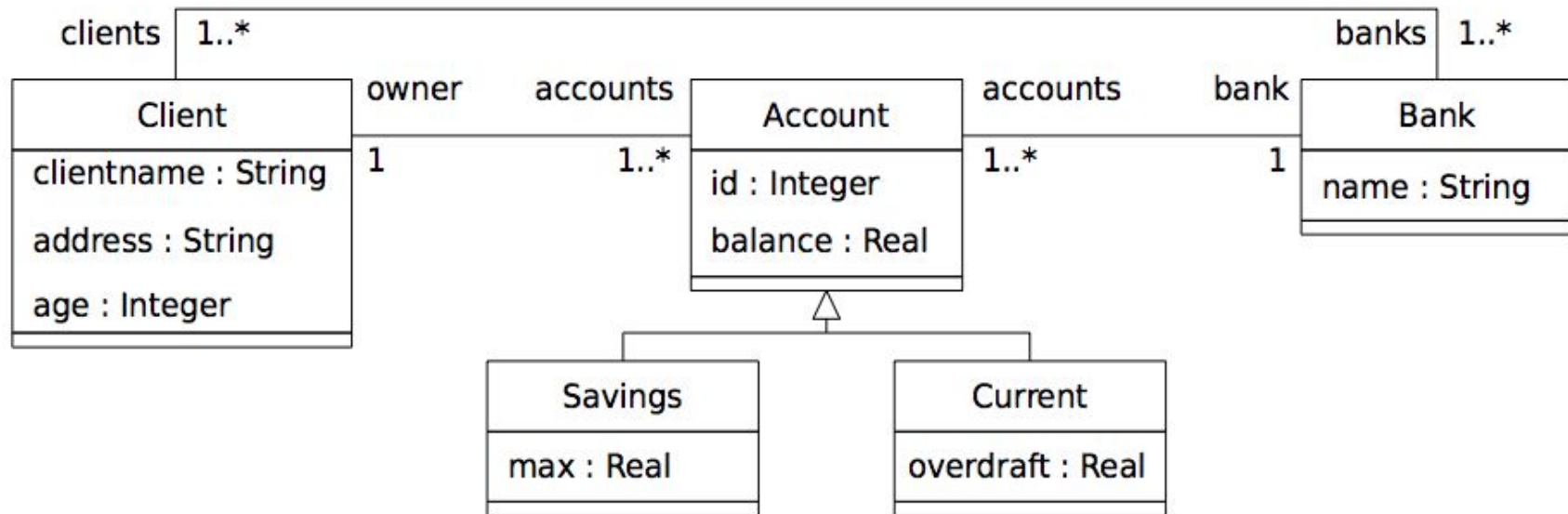
Fig. 5. Number of theorems generated

A Summary

- Formal **semantic-centric view** of tool construction (based on Higher-order Logic in Isabelle/HOL)
- A technique to Embed OCL deeply on the System Framework of OCL.
- Technique amenable to a wide range of (text-based) domain specific languages (DSL)'s and semantic-based model-transformations.

Demo V

Running Example



Running Example

```
Class Bank
  Attributes
    name      : String
End

Class Client
  Attributes
    clientname : String
    address    : String
    age        : Integer
End

Class Account
  Attributes
    id        : Integer
    balance   : Real
End

Class Savings < Account
  Attributes
    max       : Real
End
```

```
Class Current < Account
  Attributes
    overdraft : Real
End

Association clients
  Between Bank [1 .. *]
    Role banks
  Client [1 .. *]
    Role clients End

Association accounts
  Between Account [1 .. *]
    Role accounts
  Client [1]
    Role owner End

Association bankaccounts
  Between Account [1 .. *]
    Role accounts
  Bank [1]
    Role bank End
```

Running Example

```
Context c: Savings
```

```
Inv A : '0 < (c .max)'
```

```
Inv B : 'c .balance <= (c .max) and 0 <= (c .balance)'
```

```
Context c: Current
```

```
Inv A : '25 < (c .owner .age) implies (c .overdraft = 0)'
```

```
Inv B : 'c .owner .age <= 25 implies (c .overdraft = -250)'
```

```
Context c: Client
```

```
Inv A : 'c .accounts ->collect(banks) = c .banks'
```