Verification of Functional Programs
Tools

Andrés Sicard-Ramírez

EAFIT University

Semester 2014-1

Description

- Automatic inductive theorem prover for proving Haskell properties

Zeno (An Automated Prover for Properties of Recursive Data Structures)

Description

- Automatic inductive theorem prover for proving Haskell properties
- The tool can discover necessary auxiliary theorems

Zeno (An Automated Prover for Properties of Recursive Data Structures)

Description

- Automatic inductive theorem prover for proving Haskell properties
- The tool can discover necessary auxiliary theorems
- The proofs can be verified in Isabelle

Zeno (An Automated Prover for Properties of Recursive Data Structures)

Description

- Automatic inductive theorem prover for proving Haskell properties
- The tool can discover necessary auxiliary theorems
- The proofs can be verified in Isabelle
- From a test suit for IsaPlanner, Zeno can prove more properties than IsaPlanner and ACL2s (ACL2 sedan)

Demo

See source code in the course web page.

Zeno

Demo

See source code in the course web page.

Presentation (slides)

Sophia Drossopoulou. Zeno. A theorem prover for inductively defined properties (IFIP WG2.1, 2011) (`https://wp.doc.ic.ac.uk/sd/`)

Zeno

Demo

See source code in the course web page.

Presentation (slides)

Sophia Drossopoulou. Zeno. A theorem prover for inductively defined properties (IFIP WG2.1, 2011) (https://wp.doc.ic.ac.uk/sd/)

Limitations

Zeno works only with terminating functions and total and finite values.

Zeno

Material

- Sonnex, William, Drossopoulou, Sophia and Eisenbach, Susan [2012]. Zeno: An Automated Prover for Properties of Recursive Data Structures. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2012). Ed. by Flanagan, Cormac and König, Barbara. Vol. 7214. Lecture Notes in Computer Science. Springer, pp. 407–421
- Sonnex, William, Drossopoulou, Sophia and Eisenbach, Susan [Feb. 2011]. Zeno: A Tool for the Automatic Verification of Algebraic Properties of Functional Programs. Tech. rep. Imperial College London.
- Web
  http://www.haskell.org/haskellwiki/Zeno

Zeno

Installation (Zeno 0.2.0.1 tested with GHC 7.0.4)

```
$ cabal unpack zeno
```

```
$ cd zeno-0.2.0.1
```

```
# Remove from zeno.cabal:
if impl(ghc >= 7)
   ghc-options:  -with-rtsopts="-N"
```

```
$ cabal install
```

Zeno

Installation (Zeno 0.2.0.1 tested with GHC 7.0.4)

```
$ cabal unpack zeno
$ cd zeno-0.2.0.1
# Remove from zeno.cabal:
if impl(ghc >= 7)
   ghc-options:  -with-rtsopts="-N"
$ cabal install
```

Remark

For installing/using different versions of GHC the stow command is your friend (see `http://www1.eafit.edu.co/asr/tips-and-tricks.html`).

HipSpec (Automating Inductive Proofs of Program Properties)

HipSpec [Claessen, Johansson, Rosén and Smallbone 2012] is based on:

- Hip [Rosén 2012]
- QuickSpec [Claessen, Smallbone and Hughes 2010]
- Theorem provers (e.g. E, Vampire and Z3)

Hip (Haskell Inductive Prover)

- Automatically prove properties about Haskell programs including partial and potentially infinite values.

Hip (Haskell Inductive Prover)

- Automatically prove properties about Haskell programs including partial and potentially infinite values.
- Subset of Haskell $\rightarrow$ intermediate language $\rightarrow$ first-order logic

Hip (Haskell Inductive Prover)

- Automatically prove properties about Haskell programs including partial and potentially infinite values.
- Subset of Haskell $\rightarrow$ intermediate language $\rightarrow$ first-order logic
- Induction techniques
  - Definitional equality
  - Structural induction
  - Scott's fixed-point induction
  - Approximation lemma

Hip (Haskell Inductive Prover)

- Automatically prove properties about Haskell programs including partial and potentially infinite values.
- Subset of Haskell → intermediate language → first-order logic
- Induction techniques
  - Definitional equality
  - Structural induction
  - Scott's fixed-point induction
  - Approximation lemma
- The higher-order (co)-induction principles are handled at the meta-level.

Hip (Haskell Inductive Prover)

- Automatically prove properties about Haskell programs including partial and potentially infinite values.
- Subset of Haskell → intermediate language → first-order logic
- Induction techniques
  - Definitional equality
  - Structural induction
  - Scott's fixed-point induction
  - Approximation lemma
- The higher-order (co)-induction principles are handled at the meta-level.
- The first-order reasoning is handled by off-the-shelf theorem provers (E, Prover9, SPASS, Vampire and Z3).

Data type and equality

```
data Prop a = a :=: a
```

```
(=:=) :: a -> a -> Prop a
(=:=) = (:=:)
```

Hip - Definitional Equality

Example

From combinatory logic (see, e.g., Hindley and Seldin [2008]).

Hip - Definitional Equality

Example

From combinatory logic (see, e.g., Hindley and Seldin [2008]).

```
k :: a -> b -> a
k x _ = x
```

Hip - Definitional Equality

Example

From combinatory logic (see, e.g., Hindley and Seldin [2008]).

```
k :: a -> b -> a
k x _ = x
```

```
s :: (a -> b -> c) -> (a -> b) -> a -> c
s f g x = f x (g x)
```

Hip - Definitional Equality

Example

From combinatory logic (see, e.g., Hindley and Seldin [2008]).

```
k :: a -> b -> a
k x _ = x

s :: (a -> b -> c) -> (a -> b) -> a -> c
s f g x = f x (g x)

id :: a -> a
id x = x
```

Example

From combinatory logic (see, e.g., Hindley and Seldin [2008]).

```
k :: a -> b -> a
k x _ = x
s :: (a -> b -> c) -> (a -> b) -> a -> c
s f g x = f x (g x)
id :: a -> a
id x = x
prop_skk_id :: Prop (a -> a)
prop_skk_id = s k k =:= id
```

Hip - Structural Induction

Example

```
data N = Z | S N
```

Hip - Structural Induction

Example

```
data N = Z | S N
```

- Structural recursion on total and finite values

$$\frac{P\ \mathsf{Z} \qquad \forall x.\, P\ x \Rightarrow P(\mathsf{S}\ x)}{\forall x.\, x \text{ total and finite} \Rightarrow P\ x}$$

Hip - Structural Induction

Example

```
data N = Z | S N
```

- Structural recursion on total and finite values

$$\frac{P\ \mathsf{Z} \qquad \forall x.\, P\ x \Rightarrow P(\mathsf{S}\ x)}{\forall x.\, x \text{ total and finite} \Rightarrow P\ x}$$

- Structural recursion on partial and potentially infinite values

$$\frac{P\ \bot \qquad P\ \mathsf{Z} \qquad \forall x.\, P\ x \Rightarrow P(\mathsf{S}\ x) \qquad P \text{ admissible}}{\forall x. P\ x}$$

Hip

Limitations

Hip cannot use auxiliary theorems and theories.

Hip

Limitations

Hip cannot use auxiliary theorems and theories.

Installation

Hip is now developed as a part of the HipSpec, so it is not stand-alone maintained.

You can install Hip from `https://github.com/asr/hip` using GHC 7.6.3.

References

Claessen, Koen, Johansson, Moa, Rosén, Dan and Smallbone, Nicholas (2012). HipSpec: Automating Inductive Proofs of Program Properties. Workshop on Automated Theory Exploration (ATX), at IJCAR 2012. URL: http://www.cse.chalmers.se/~jomoa/ (visited on 25/05/2013) (cit. on p. 12).

Claessen, Koen, Smallbone, Nicholas and Hughes, John (2010). QuickSpec: Guessing Formal Specifications Using Testing. In: Tests and Proofs (TAP 2010). Ed. by Fraser, Gordon and Garfantini, Gordon. Vol. 6143. Lecture Notes in Computer Science. Springer, pp. 6–21 (cit. on p. 12).

Hindley, J. Roger and Seldin, Jonathan P. (2008). Lambda-Calculus and Combinators. An Introduction. Cambridge University Press (cit. on pp. 19–23).

Rosén, Dan (2012). Proving Equational Haskell Properties Using Automated Theorem Provers. MA thesis. University of Gothenburg (cit. on p. 12).

Sonnex, William, Drossopoulou, Sophia and Eisenbach, Susan (Feb. 2011). Zeno: A Tool for the Automatic Verification of Algebraic Properties of Functional Programs. Tech. rep. Imperial College London (cit. on p. 9).

References

Sonnex, William, Drossopoulou, Sophia and Eisenbach, Susan (2012). Zeno: An Automated Prover for Properties of Recursive Data Structures. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS 2012). Ed. by Flanagan, Cormac and König, Barbara. Vol. 7214. Lecture Notes in Computer Science. Springer, pp. 407–421 (cit. on p. 9).