HipSpec

Automating Inductive Proofs using Theory Exploration

Dan Rosén

Koen Claessen, Moa Johansson, Nicholas Smallbone

Chalmers University of Technology

May 31, 2013

Rotate example

```
rotate Z     xs     = xs
rotate (S n) []     = []
rotate (S n) (x:xs) = rotate n (xs ++ [x])
```

```
rotate 1 [1,2,3,4] = [2,3,4,1]
rotate 2 [1,2,3,4] = [3,4,1,2]
rotate 3 [1,2,3,4] = [4,1,2,3]
rotate 4 [1,2,3,4] = [1,2,3,4]
```

Rotate example

```
rotate Z     xs     = xs
rotate (S n) []     = []
rotate (S n) (x:xs) = rotate n (xs ++ [x])
```

```
rotate 1 [1,2,3,4] = [2,3,4,1]
rotate 2 [1,2,3,4] = [3,4,1,2]
rotate 3 [1,2,3,4] = [4,1,2,3]
rotate 4 [1,2,3,4] = [1,2,3,4]
```

$$\forall \texttt{xs}.\ \texttt{rotate (length xs) xs} = \texttt{xs}$$

Rotate example

```
rotate Z     xs     = xs
rotate (S n) []     = []
rotate (S n) (x:xs) = rotate n (xs ++ [x])
```

$$\forall \texttt{xs}.\, \texttt{rotate (length xs) xs} = \texttt{xs}$$

Rotate example

```
rotate Z     xs     = xs
rotate (S n) []     = []
rotate (S n) (x:xs) = rotate n (xs ++ [x])
```

$$\forall \texttt{xs}.\ \texttt{rotate (length xs) xs} = \texttt{xs}$$

$$\begin{aligned}
&\texttt{rotate (length (x:xs)) (x:xs)} &&= \\
&\texttt{rotate (S (length xs)) (x:xs)} &&= \\
&\texttt{rotate (length xs) (xs ++ [x])} &&=
\end{aligned}$$

Rotate example

```
rotate Z     xs     = xs
rotate (S n) []     = []
rotate (S n) (x:xs) = rotate n (xs ++ [x])
```

$$\forall \texttt{xs}.\ \texttt{rotate (length xs) xs} = \texttt{xs}$$

$$\begin{aligned}
&\texttt{rotate (length (x:xs)) (x:xs)} &&= \\
&\texttt{rotate (S (length xs)) (x:xs)} &&= \\
&\texttt{rotate (length xs) (xs ++ [x])} &&=
\end{aligned}$$

Stuck!

HipSpec vs Rotate

$$\forall \texttt{xs}, \texttt{ys}.\ \texttt{rotate (length (xs ++ ys)) (xs ++ ys)} = \texttt{ys ++ xs}$$

HipSpec vs Rotate

$$\forall \texttt{xs}, \texttt{ys}.\ \texttt{rotate (length (xs ++ ys)) (xs ++ ys)} = \texttt{ys ++ xs}$$

(also requires associativity and right identity of ++)

QuickSpec: the Theory Exploration Phase

Generates a bunch of terms:

```
[]                  []++[]              qrev [] []    qrev (rev xs) []
qrev [] (rev xs)    qrev (rev xs) ys    qrev [] xs    qrev xs []
[]++qrev xs ys      qrev [] (xs++ys)    (x:xs)++[]    qrev xs ys++[]
qrev (x:[]) xs      qrev [] (x:xs)      rev []        rev (qrev ys xs)
rev (rev xs)        []++rev xs          rev xs        rev xs++ys
xs                  []++xs              xs++[]        (xs++ys)++[]
[]++(xs++ys)        xs++ys              (x:[])++xs    xs++(x:[])
```

Partitioning into Equivalence Classes

```
xs
xs++[]
[]++xs
qrev [] xs
rev (rev xs)
qrev (rev xs) []
```

```
[]
rev []
qrev [] []
[]++[]
```

```
qrev xs ys
rev (qrev ys xs)
rev xs++ys
[]++qrev xs ys
qrev [] (qrev xs ys)
qrev xs ys++[]
qrev (qrev ys xs) []
```

```
xs++ys
qrev (rev xs) ys
[]++(xs++ys)
qrev [] (xs++ys)
(xs++ys)++[]
```

```
x:xs
[]++(x:xs)
qrev [] (x:xs)
(x:xs)++[]
(x:[])++xs
qrev (x:[]) xs
```

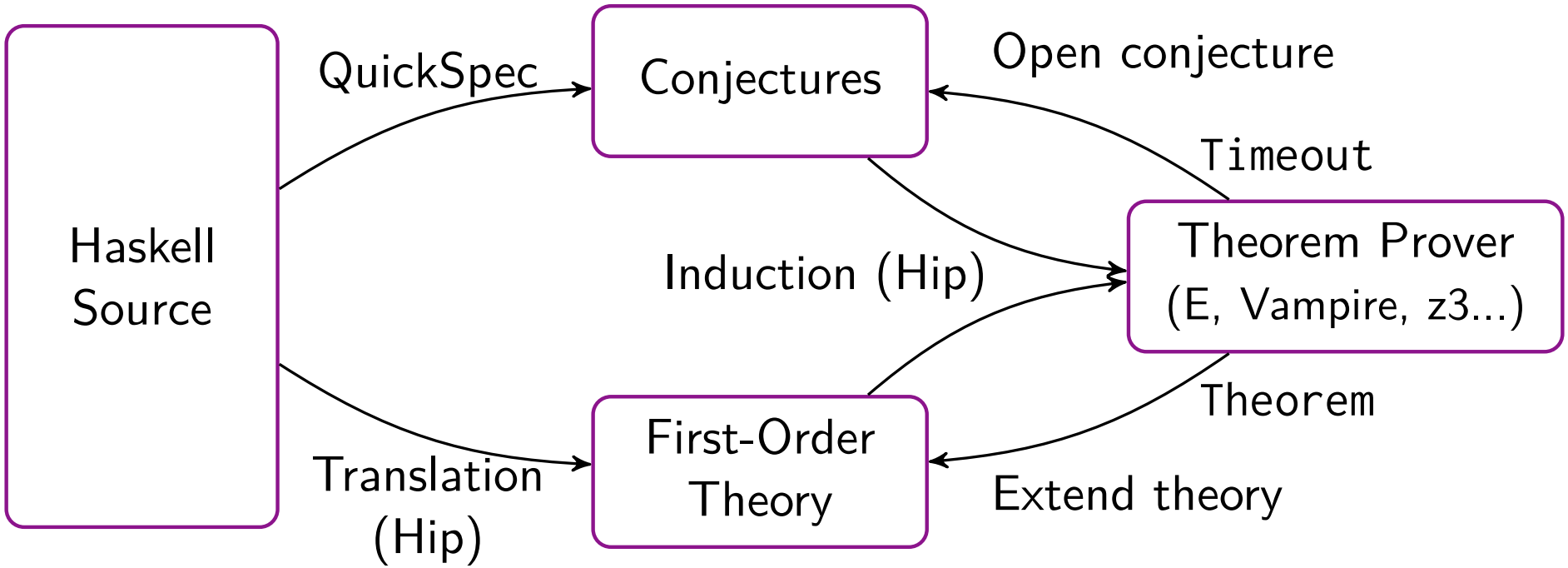
```
rev xs
qrev xs []
[]++rev xs
qrev [] (rev xs)
```

Hip: The Haskell Inductive Prover

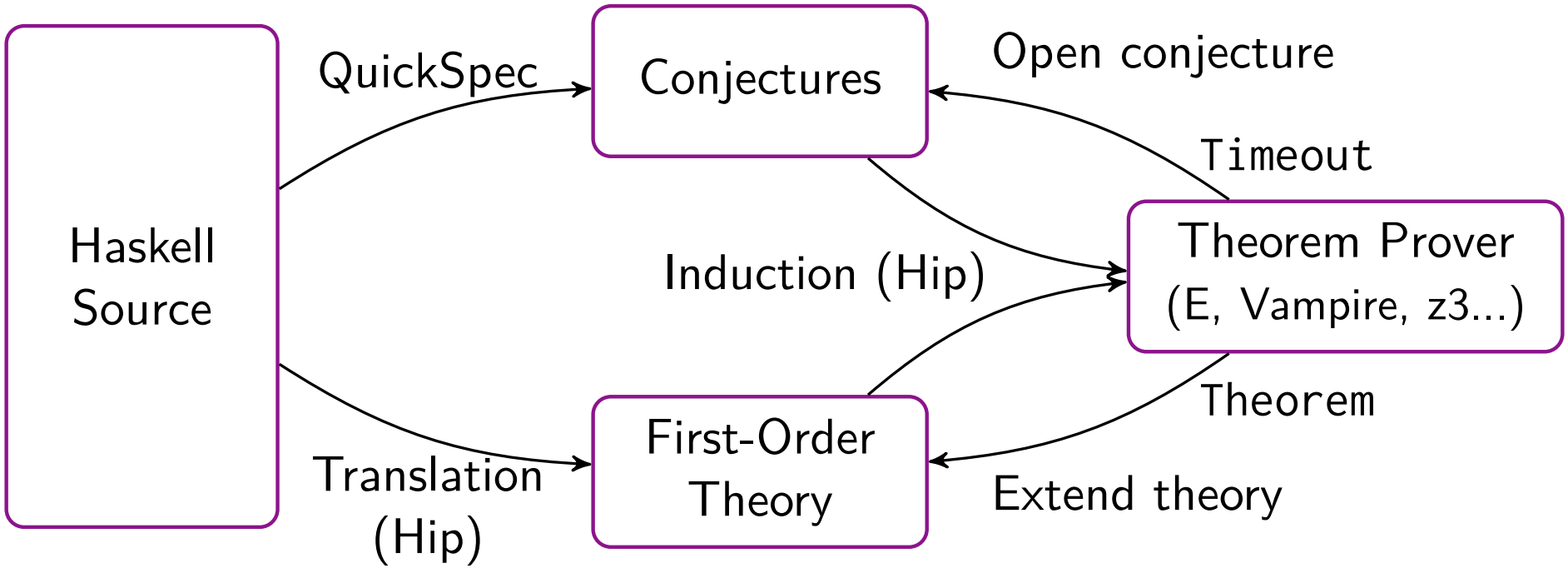
- Translate to typed first order logic
- Apply structural induction

Also supports higher-order functions and partial application

Overview of HipSpec



Overview of HipSpec



- Try to prove "smallest" unproved equation
- Terminate everything is proved (or when the current theory cannot prove any more open conjectures)

Rotate, revisited

$$\forall \texttt{xs}.\ \texttt{rotate (length xs) xs} = \texttt{xs}$$

$$\begin{aligned}
\texttt{rotate (length (x:xs)) (x:xs)} &= \\
\texttt{rotate (S (length xs)) (x:xs)} &= \\
\texttt{rotate (length xs) (xs ++ [x])} &=
\end{aligned}$$

Stuck!

HipSpec the Theory Exploration System

Saturate a theory and have it nicely presented

HipSpec the Theory Exploration System

Saturate a theory and have it nicely presented

- `data Integer = Positive Nat | Negative Nat`
- `data BinNat = Zero | ZeroAnd BinNat | OneAnd BinNat`

Conjecturing Conditionals

```
isort :: [Nat] -> [Nat]

insert :: Nat -> [Nat] -> [Nat]

sorted :: [Nat] -> Bool
```

$$\forall\, \texttt{xs}.\, \texttt{sorted (isort xs)} = \texttt{True}$$

Requires:

$$\forall\, \texttt{xs}.\, \texttt{sorted xs} = \texttt{True} \Rightarrow \texttt{sorted (insert x xs)} = \texttt{True}$$

Conclusions

- Evaluate your programs!
- Completeness up to a certain depth:
  If the lemma is there, HipSpec will eventually try to prove it!

github.com/danr/hipspec

What is HipSpec?

Haskell source

```
rev [] = []
rev (x:xs)
  = rev xs ++ [x]

prop_rev xs
  = rev (rev xs) =:= xs
```

Hip

Haskell Inductive Prover

- FOL translation
- Apply induction
- Success , or stuck!

QuickSpec

Eq-theory from testing:

```
rev (xs ++ ys)
  = rev ys ++ rev xs
xs ++ [] = []
xs ++ (ys ++ zs) =
  (xs ++ ys) ++ zs
```

HipSpec

Use these as lemmas!!