Using CVC4 for Proofs by Induction

Andrew Reynolds

May 28th, 2015
Overview

• Satisfiability Modulo Theories (SMT) solvers
  – Lack support for inductive reasoning
• “Induction for SMT solvers”
  With Viktor Kuncak, VMCAI 2015
  – Techniques for induction in SMT solvers
    • Subgoal generation
    • Encodings that leverage theory reasoning
    • Benchmarks/Evaluation
SMT Solvers

• SMT solvers:
  – Used in formal methods applications:
    • Software verification, automated theorem proving
  – Determine the satisfiability of:
    • Boolean combinations of ground theory constraints
      – Linear arithmetic, BitVectors, Arrays, Datatypes, etc.
  – Have limited support for quantified formulas $\forall$
    • Approaches tend to be heuristic (e.g. E-matching)
    • Often fail on simple examples
      – Notably for problems requiring inductive reasoning
SMT Solver

Communicate via DPLL(T) Framework

SAT Solver

Arithmetic

UF

Datatypes

Decision procedures

∧ Module
SMT Solver

Ground Constraints

SAT Solver

Arithmetic

UF

Datatypes

Axioms

∀

Module
SMT Solver

\[ G, \quad Q(t_1), Q(t_2), \ldots \]

\[ \forall x. Q(x) \]

SAT Solver

Ground Solver

Arithmetic

UF

Datatypes

Instances of axioms

When current set is unsatisfiable

(Partial) Models

UNSAT

Module
Running Example

• Datatype **List**

```
List := cons(hd: Int, tl: List) | nil
```

• **Length function** `len : List -> Int`

```
len(nil) = 0,
\( \forall xy. len(cons(x, y)) = 1 + len(y) \)
```
Example #1: Ground Conjecture

\[
\begin{align*}
\text{len}(\text{nil}) &= 0 \\
\forall x y. \text{len}(\text{cons}(x, y)) &= 1 + \text{len}(y) \\
\neg \text{len}(\text{cons}(0, \text{nil})) &= 1
\end{align*}
\]
Example #1

\[
\text{len}(\text{nil}) = 0, \\
\text{len}(\text{cons}(0, \text{nil})) \neq 1
\]

\[
\forall xy. \text{len}(\text{cons}(x, y)) = 1 + \text{len}(y)
\]
Example #1

\[
\text{len}(\text{nil}) = 0,
\text{len}(\text{cons}(0, \text{nil})) \neq 1
\]

\[
\forall x y. \text{len}(\text{cons}(x, y)) = 1 + \text{len}(y)
\]

Partial model:

\[
\{ \ldots, \text{len}(\text{cons}(0, \text{nil})) = 0 \} 
\]
Example #1

\[
\begin{align*}
\text{len(nil)} &= 0, \\
\text{len(cons(0,nil))} &\neq 1, \\
\text{len(cons(0,nil))} &= 1 + \text{len(nil)} \\
\forall xy. \text{len(cons(x,y))} &= 1 + \text{len(y)}
\end{align*}
\]

Instantiate:
\[Q\{x\mapsto 0, y\mapsto \text{nil}\}\]

Partial model:
\[\{..., \text{len(cons(0,nil))} = 0\}\]
Example #1

\[\text{len}(\text{nil})=0,\]
\[\text{len}(\text{cons}(0,\text{nil}))\neq 1,\]
\[\text{len}(\text{cons}(0,\text{nil}))=1+\text{len}(\text{nil})\]

\[\forall xy. \text{len}(\text{cons}(x,y))=1+\text{len}(y)\]

Since \[\text{len}(\text{cons}(0,\text{nil}))=1+\text{len}(\text{nil})=1+0=1\neq 1\]
Example #2: Quantified Conjecture

\[
\begin{align*}
\text{len(nil)} &= 0 \\
\forall xy. \text{len(cons}(x, y)) &= 1 + \text{len}(y) \\
\neg \forall x. \text{len}(x) &\geq 0
\end{align*}
\]

Axioms (Negated) Conjecture

\[
\begin{align*}
\text{Ground Solver} \\
\forall \text{ Module}
\end{align*}
\]
Example #2

\[
\begin{align*}
\text{len}(\text{nil}) &= 0 \\
\forall xy. \text{len}(\text{cons}(x, y)) &= 1 + \text{len}(y) \\
\neg \forall x. \text{len}(x) &\geq 0
\end{align*}
\]

Skolemize: statement (does not) hold for fresh constant \(k\)

\[
\neg \text{len}(k) \geq 0
\]
Example #2

len(nil) = 0,
len(k) < 0

∀xy. len(cons(x, y)) = 1 + len(y)

Ground Solver

∀ Module
Example #2

len(nil) = 0, 
len(k) < 0

∀xy. len(cons(x, y)) = 1 + len(y)

Ground Solver

Partial model:
{..., len(k) = -1, k = cons(hd(k), tl(k))}
Example #2

\

\begin{align*}
\text{len(nil)} &= 0, \\
\text{len(k)} &< 0, \\
\text{len(cons(hd(k), tl(k)))} &= 1 + \text{len(tl(k))} \\
\forall xy. \text{len(cons(x, y))} &= 1 + \text{len(y)}
\end{align*}

\begin{align*}
\text{Instantiate:} \\
\{x \mapsto \text{hd(k)}, y \mapsto \text{tl(k)}\}
\end{align*}

\begin{align*}
\text{Partial model:} \\
\{\ldots, \text{len(k)} = -1, k = \text{cons(hd(k), tl(k))}\}
\end{align*}
Example #2

\[\text{len(nil)} = 0,\]
\[\text{len(k)} < 0,\]
\[\text{len(k)} = 1 + \text{len(tl(k))}\]

\[\forall xy. \text{len(cons(x, y))} = 1 + \text{len(y)}\]
Example #2

\[
\forall xy. \text{len}(\text{cons}(x, y)) = 1 + \text{len}(y)
\]

Partial model:
{...,
\text{len}(k) = -1,
\text{len}(\text{tl}(k)) = -2,
\text{tl}(k) = \text{cons}(\text{hd}(\text{tl}(k)), \text{tl}(\text{tl}(k)))}
Example #2

\[
\begin{align*}
\text{len}(\text{nil}) &= 0, \\
\text{len}(\text{k}) &< 0, \\
\text{len}(\text{k}) &= 1 + \text{len}(\text{tl}(\text{k})) \\
\text{len}(\text{cons}(\text{hd}(\text{tl}(\text{k})), \text{tl}(\text{tl}(\text{k})))) &= 1 + \text{len}(\text{tl}(\text{tl}(\text{k})))
\end{align*}
\]

\[\forall xy. \text{len}(\text{cons}(x, y)) = 1 + \text{len}(y)\]

Ground Solver

\[
\begin{align*}
\{ x \mapsto \text{hd}(\text{tl}(\text{k})), y \mapsto \text{tl}(\text{tl}(\text{k})) \}
\end{align*}
\]

Partial model:

\[
\{ ..., \text{len}(\text{k}) = -1, \text{len}(\text{tl}(\text{k})) = -2, \\
\text{tl}(\text{k}) = \text{cons}(\text{hd}(\text{tl}(\text{k})), \text{tl}(\text{tl}(\text{k}))) \}
\]
Example #2

\[
\begin{align*}
\text{len(nil)} &= 0, \\
\text{len}(k) &< 0, \\
\text{len}(k) &= 1 + \text{len}(\text{tl}(k)) \\
\text{len}(\text{tl}(k)) &= 1 + \text{len}(\text{tl}(\text{tl}(k)))
\end{align*}
\]

\[\forall xy. \text{len}(\text{cons}(x, y)) = 1 + \text{len}(y)\]
Example #2

\[ \forall xy. \text{len}(\text{cons}(x, y)) = 1 + \text{len}(y) \]

Partial model:

\{ ..., \text{len}(k) = -1, \text{len}(\text{tl}(k)) = -2, \text{len}(\text{tl}(\text{tl}(k))) = -3, \text{tl}(\text{tl}(k)) = \text{cons}(\text{hd}(\text{tl}(\text{tl}(k))), \text{tl}(\text{tl}(\text{tl}(k)))) \} \]
Example #2

len(nil) = 0,  
len(k) < 0,  
len(k) = 1 + len(tl(k))  
len(tl(k)) = 1 + len(tl(tl(k)))  

∀xy. len(cons(x, y)) = 1 + len(y)

Partial model:
{..., len(k) = -1, len(tl(k)) = -2, len(tl(tl(k))) = -3,  
  tl(tl(k)) = cons(hd(tl(tl(k))), tl(tl(tl(k))))}
Challenge: Inductive Reasoning

• This example requires induction
• Existing techniques
  – Within inductive theorem provers:
    • ACL2 [Chamarthi et al 2012]
    • HipSpec [Claessen et al 2013]
    • IsaPlanner [Johansson et al 2010]
    • Zeno [Sonnex et al 2012]
    • SPASS/Pirate
    • …
  – Induction as preprocessing step to SMT solver:
    • Dafny [Leino 2012]
• No SMT solvers support induction \textit{natively}
  \[\Rightarrow\text{Until now, in CVC4}\]
Solution: Inductive Strengthening

• Given negated conjecture:

\[ \neg \forall x. \text{len}(x) \geq 0 \]

• Assume property does not hold for fresh k:

\[ \neg \text{len}(k) \geq 0 \]

AND

• Assume k is the \textit{smallest} CE to property:

\[ k = \text{cons}(\text{hd}(k), \text{tl}(k)) \Rightarrow \text{len}(\text{tl}(k)) \geq 0 \]
Example #2: revised

len(nil)=0,
len(k)<0,
k=cons(hd(k),tl(k)) \Rightarrow
len(tl(k)) \geq 0,
len(k)=1+len(tl(k))

∀xy.\,\text{len(cons(x,y))}=1+\text{len(y)}
Example #2: revised

\[
\begin{align*}
\text{len}(\text{nil}) &= 0, \\
\text{len}(k) &< 0, \\
k &= \text{cons}(\text{hd}(k), \text{tl}(k)) \implies \\
\text{len}(\text{tl}(k)) &\geq 0, \\
\text{len}(k) &= 1 + \text{len}(\text{tl}(k))
\end{align*}
\]

\[
\forall xy. \text{len}(\text{cons}(x, y)) = 1 + \text{len}(y)
\]

Ground Solver

UNSAT

Since \(0 > \text{len}(k) = 1 + \text{len}(\text{tl}(k)) \geq 1\)
Skolemization with Inductive Strengthening

• General form:

$$\forall x. P(x) \lor (\neg P(k) \land \forall y. (y < k \Rightarrow P(y)))$$

– For well-founded relation “<“

• Extends for multiple variables

• Common examples of “<“ in SMT:
  – (Weak) structural induction on inductive datatypes
    • Assume property holds for direct children of k of same type
  – (Weak) well-founded induction on integers
    • Assume property holds for (k-1), with base case 0
Challenge: Subgoal Generation

• Unfortunately, inductive strengthening is not enough
• Consider conjecture:

\[ \forall x. \text{len}(\text{rev}(x)) = \text{len}(x) \]

– where \text{rev} is axiomatized by:

\[
\begin{align*}
\text{rev}(\text{nil}) &= \text{nil}, \\
\forall xy. \text{rev}(\text{cons}(x, y)) &= \text{app}(\text{rev}(y), \text{cons}(x, \text{nil}))
\end{align*}
\]

• To prove, requires induction, and “subgoals”:

\[ \forall xy. \text{len}(\text{app}(x, y)) = \text{plus}(\text{len}(x), \text{len}(y)) \]

\[ \forall xy. \text{plus}(x, y) = \text{plus}(y, x) \]
Generating candidate subgoals

• How to generate necessary subgoals?
  – Idea: Enumerate/prove them in a principled way
  • HipSpec [Claessen et al 2013]

\[
\forall x. \text{len}(x) = Z \\
\forall x. \text{len}(x) = S(Z) \\
\forall x. \text{app}(x, \text{nil}) = \text{nil} \\
\forall x. \text{app}(x, \text{nil}) = x \\
\forall x. \text{app}(x, \text{nil}) = \text{cons}(0, x) \\
\ldots \\
\forall xy. \text{plus}(x, y) = \text{plus}(x, 0) \\
\forall xy. \text{plus}(x, y) = \text{plus}(y, x) \\
\ldots \\
\forall xy. \text{len}(\text{app}(x, y)) = \text{plus}(\text{len}(x), \text{len}(y)) \\
\ldots 
\]
Subgoal Generation in SMT

Ground Constraints

Axioms

\( \forall \) Module

Ground Solver

UNSAT

When current set is unsatisfiable

(Partial) Models
Subgoal Generation in SMT

Ground Constraints

Instances of Axioms

Axioms

Quantifier Instantiation

Ground Solver

∀ Module

UNSAT
Subgoal Generation in SMT

Ground Constraints

Instances of Axioms

Axioms

Subgoal

\[ \neg \forall x. P(x) \lor \forall x. P(x) \]

Ground Solver

\[ \forall \text{ Module} \]

UNSAT

• Subgoal \( P(x) \) either:
  a. has a counterexample,
  b. holds universally
Subgoal Generation in SMT

- Ground Constraints
- Instances of Axioms
- Axioms

\[ \neg \exists x. P(x) \lor \exists x. P(x) \]

Subgoal \( P(x) \) either:
- a. has a smallest ce,
- b. holds universally

\( \neg P(k) \land \forall y. (y < k \Rightarrow P(y)) \lor \forall x. P(x) \)

- Ground Solver
- UNSAT
- ∀ Module

• Subgoal \( P(x) \) either:
  a. has a smallest ce,
  b. holds universally
Subgoal Generation in SMT

Ground Constraints

Instances of Axioms

\neg P(k)

(\neg P(k) \land \forall y. (y<k \Rightarrow P(y)))

\forall x. P(x)

• First, assume:
  \neg P(k) \land \forall y. (y<k \Rightarrow P(y))

Ground Solver

UNSAT

\forall Module
Subgoal Generation in SMT

Ground Constraints

Instances of Axioms

Axioms

\( \forall x. P(x) \)

\( \neg P(k) \land \forall y. (y < k \Rightarrow P(y)) \)

\( \bigvee \forall x. P(x) \)

Ground Solver

\text{UNSAT}

• If unsuccessful:
  - Assert \( \forall x. P(x) \)
• Managed by conflict analysis mechanism

\forall \text{ Module}
Subgoal Generation in SMT

Ground Constraints
Instances of Axioms
Active/Failed Subgoals

Axioms
Proven Subgoals

Ground Solver

∀ Module

When current set is unsatisfiable
UNSAT

(Partial) Models
Subgoal Generation : Challenges

• Main challenge: scalability

• Keys to success:
  – Enumerate subgoals in a fair manner (smaller first)
  – Do not consider subgoals that are not useful
Subgoal Filtering

• Given: \( \forall x. \text{len}(\text{rev}(x)) = \text{len}(x) \)

• Filtering based on “active” symbols:
  \( \forall xy. \text{count}(x, y) = \text{count}(\text{rev}(x), y) \)
  – Irrelevant, if conjecture is not related to “\text{count}”

• Filtering based on canonicity:
  \( \forall x. \text{len}(x) = \text{len}(\text{app}(x, \text{nil})) \)
  – Redundant, if we know \( \forall x. x = \text{app}(x, \text{nil}) \)

• Filtering based on counterexamples:
  \( \forall x. \text{len}(x) = \text{len}(\text{app}(x, x)) \)
  – Falsified, e.g. if partial model contains \( \text{len}(t) \neq \text{len}(\text{app}(t, t)) \)

\( \Rightarrow \) Typically can remove >95-99% subgoals
Evaluation: Benchmarks

- Four benchmark sets (in SMT2):
  1. IsaPlanner [Johansson et al 2010]
  2. Clam [Ireland 1996]
  3. HipSpec [Claessen et al 2013]
  4. Leon
    - Amortized Queues, Binary search trees, Leftist Heaps

- Three encodings:
  - Base encoding
  - (2 variants of) Theory encoding
    - Take advantage of builtin theory reasoning of SMT solver
Base Encoding

• All functions over datatypes:

Nat := S(P:Nat) | Z
List:= cons(hd:Int,tl:List) | nil

∀x. plus(Z,x)=x
∀xy. plus(S(x),y)=S(plus(x,y))
len(nil)=Z
∀xy. len(cons(x,y))=S(len(y))
...

¬∀x. len(rev(x))=len(x)
Base Encoding

• All functions over datatypes:

\[
\begin{align*}
\text{Nat} & : = S(P: \text{Nat}) \mid Z \\
\text{List} & : = \text{cons}(hd: \text{Int}, tl: \text{List}) \mid \text{nil}
\end{align*}
\]

\[
\begin{align*}
\forall x. \text{plus}(Z, x) = x \\
\forall xy. \text{plus}(S(x), y) = S(\text{plus}(x, y)) \\
\text{len}(\text{nil}) = Z \\
\forall xy. \text{len}(\text{cons}(x, y)) = S(\text{len}(y)) \\
\ldots
\end{align*}
\]

\[
\begin{align*}
\forall xy. \text{len}(\text{app}(x, y)) &= \text{plus}(\text{len}(x), \text{len}(y)) \\
\forall xy. \text{plus}(x, y) &= \text{plus}(y, x)
\end{align*}
\]

\[
\neg \forall x. \text{len}(\text{rev}(x)) = \text{len}(x)
\]

Necessary Subgoals for UNSAT
Theory Encoding

• All functions over datatypes:

Nat := S(P:Nat) | Z
List := cons(hd:Int, tl:List) | nil

∀x.\text{plus}(Z, x) = x
∀xy.\text{plus}(S(x), y) = S(\text{plus}(x, y))
len(nil) = Z
∀xy.\text{len}(\text{cons}(x, y)) = S(\text{len}(y))
...

¬∀x.\text{len}(\text{rev}(x)) = \text{len}(x)
Theory Encoding #1

• All functions over datatypes:

\[
\begin{align*}
\text{Nat} & := S(P: \text{Nat}) \mid Z \\
\text{List} & := \text{cons}(\text{hd}: \text{Int}, \text{tl}: \text{List}) \mid \text{nil}
\end{align*}
\]

\[
\begin{align*}
\forall x. 0 + x &= x \\
\forall xy. (x + 1) + y &= (x + y) + 1 \\
\text{len}(\text{nil}) &= 0 \\
\forall xy. \text{len}(\text{cons}(x, y)) &= \text{len}(y) + 1 \\
\end{align*}
\]

\[\Rightarrow \text{Replace uninterp. functions with theory functions, e.g. plus} \rightarrow +\]

\[\neg \forall x. \text{len} (\text{rev}(x)) = \text{len}(x)\]
Theory Encoding #1

- All functions over datatypes:

\[ \forall x. 0 + x = x \]
\[ \forall xy. (x + 1) + y = (x + y) + 1 \]
\[ \text{len(nil)} = 0 \]
\[ \forall xy. \text{len(cons(x, y))} = \text{len(y)} + 1 \]

\[ \text{Nat} := S(P: \text{Nat}) \mid Z \]
\[ \text{List} := \text{cons(hd: Int, tl: List)} \mid \text{nil} \]

⇒ Replace uninterpreted functions with theory functions, e.g. \text{plus} \rightarrow +

Downside: quantifiers + theory symbols can be hard

\[ \neg \forall x. \text{len(rev(x))} = \text{len(x)} \]
Theory Encoding

• All functions over datatypes:

Nat := S(P:Nat) | Z
List := cons(hd:Int,tl:List) | nil

∀x. plus(Z,x)=x
∀xy. plus(S(x),y)=S(plus(x,y))
len(nil)=Z
∀xy. len(cons(x,y))=S(len(y))
...

¬∀x. len(rev(x))=len(x)
Theory Encoding #2

• All functions over datatypes:

Nat := S(P:Nat) | Z
List := cons(hd:Int, tl:List) | nil

∀x. plus(Z,x) = x
∀xy. plus(S(x), y) = S(plus(x, y))
len(nil) = Z
∀xy. len(cons(x, y)) = S(len(y))
...

toInt(zero) = 0, ∀x. toInt(S(x)) = 1 + toInt(x)
∀xy. toInt(plus(x, y)) = toInt(x) + toInt(y)
...

¬∀x. len(rev(x)) = len(x)
Theory Encoding #2

• All functions over datatypes:

Datatype Definitions

Nat := S(P:Nat) | Z
List := cons(hd:Int,tl:List) | nil

Function Definitions

∀x.plus(Z,x)=x
∀xy.plus(S(x),y)=S(plus(x,y))
len(nil)=Z
∀xy.len(cons(x,y))=S(len(y))
...

Mapping
toInt : Nat → Int

toInt(zero)=0,
∀x.toInt(S(x))=1+toInt(x)
∀xy.toInt(plus(x,y))=toInt(x)+toInt(y)
...

⇒ Allows SMT solver to make use of theory reasoning on demand
Above axioms imply, e.g.
∀xy.plus(x,y)=plus(y,x)

¬∀x.len(rev(x))=len(x)

Negated Conjecture
Theory Encoding #2

• All functions over datatypes:

\[
\begin{align*}
\text{Nat} & := S(P:\text{Nat}) \mid \text{Z} \\
\text{List} & := \text{cons}(\text{hd}:	ext{Int},\text{tl}:\text{List}) \mid \text{nil}
\end{align*}
\]

\[
\begin{align*}
\forall x. \text{plus}(Z,x) & = x \\
\forall xy. \text{plus}(S(x),y) & = S(\text{plus}(x,y)) \\
\text{len}(\text{nil}) & = \text{Z} \\
\forall xy. \text{len}(\text{cons}(x,y)) & = S(\text{len}(y)) \\
& \ldots
\end{align*}
\]

\[
\begin{align*}
\text{toInt}(\text{zero}) & = 0, \forall x. \text{toInt}(S(x)) = 1+\text{toInt}(x) \\
\forall xy. \text{toInt}(\text{plus}(x,y)) & = \text{toInt}(x) + \text{toInt}(y) \\
& \ldots
\end{align*}
\]

\[
\begin{align*}
\forall xy. \text{len}(\text{app}(x,y)) & = \text{plus}(\text{len}(x),\text{len}(y))
\end{align*}
\]

\[
\neg \forall x. \text{len}(\text{rev}(x)) = \text{len}(x)
\]

Datatype Definitions

Function Definitions

Mapping toInt : Nat → Int

Necessary Subgoals for UNSAT
## Results: SMT solvers

<table>
<thead>
<tr>
<th></th>
<th>base</th>
<th>th1</th>
<th>th2</th>
</tr>
</thead>
<tbody>
<tr>
<td>z3</td>
<td>35</td>
<td>72</td>
<td>75</td>
</tr>
<tr>
<td>cvc4</td>
<td>29</td>
<td>63</td>
<td>68</td>
</tr>
<tr>
<td>cvc4+i</td>
<td>204</td>
<td>180</td>
<td>240</td>
</tr>
<tr>
<td>cvc4+ig</td>
<td>260</td>
<td>201</td>
<td>277</td>
</tr>
</tbody>
</table>

- Results for 311 benchmarks from 4 classes
- 300 second timeout

- cvc4+i: with induction
- cvc4+ig: with induction + subgoal gen.
Results: Subgoal Generation

• With subgoals, solved +37 for th2 encoding
  – Only solved +1 when filtering turned off
• Overhead of subgoal generation was small:
  – 30 cases (out of 933) was 2x slower
  – 9 cases (out of 933) went solved -> unsolved
• Most subgoals were small: term size ≤ 3
  – Some were non-trivial (not discovered manually)
Results: Subgoal Generation

• Conjecture:

\[ \forall x. \text{count}(n, \text{sort}(x)) = \text{count}(n, x) \]

\[ \Rightarrow \text{Number of times } n \text{ occurs in a list is unchanged after sorting} \]

• We thought it would require subgoals:

\[ \forall x. \text{count}(n, \text{insert}(n, x)) = \text{count}(n, x) + 1 \]
\[ \forall x. n \neq m \Rightarrow \text{count}(n, \text{insert}(m, x)) = \text{count}(n, x) \]

• CVC4 instead found the sufficient subgoal:

\[ \forall x. \text{count}(n, \text{insert}(m, x)) = \text{count}(n, \text{cons}(m, x)) \]

\[ \Rightarrow \text{Proved original conjecture fully automatically with a simpler proof} \]
Comparison with Other Provers

- Translated/evaluated in previous studies
- CVC4 fairly competitive

<table>
<thead>
<tr>
<th>Benchmark class</th>
<th>Isaplanner</th>
<th>Clam</th>
<th>HipSpec</th>
<th>Leon</th>
</tr>
</thead>
<tbody>
<tr>
<td>cvc4+ig:th2</td>
<td>80</td>
<td>39</td>
<td>18</td>
<td>42</td>
</tr>
<tr>
<td>ACL2</td>
<td>73</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Clam</td>
<td></td>
<td>41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dafny</td>
<td>45</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hipspec</td>
<td>80</td>
<td>47</td>
<td>26</td>
<td></td>
</tr>
<tr>
<td>Isaplanner</td>
<td>43</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Zeno</td>
<td>82</td>
<td>21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>85</td>
<td>50</td>
<td>26</td>
<td>45</td>
</tr>
</tbody>
</table>
Future Work

Improvements to subgoal generation
  – Filtering heuristics
  – Configurable approaches for signature of subgoals

Incorporate more induction schemes

Completeness criteria
  – Identify cases approach is guaranteed to succeed

Better comparison with other tools

Applications:
  – Tighter integration with Leon (http://leon.epfl.ch)
Thanks!

• CVC4 publicly available:
  – Induction techniques:
    • Enabled by “--quant-ind”

• Benchmarks (SMT2) available: