Overview

• Satisfiability Modulo Theories (SMT)
  – SMT solver CVC4

• Induction techniques in CVC4 [Reynolds/Kuncak, 2015]
  – Inductive strengthening
  – Subgoal generation
  – Leveraging theory reasoning

• Experiments
Automated Reasoning

• Historically, automated reasoning meant uniform proof procedures for FOL

• More recent trend is decidable fragments
  – Equality
  – Arithmetic
  – Data structures (arrays, lists, records)
  – …
Automated Reasoning

• Examples
  – **SAT** – propositional, Boolean reasoning
    • Efficient
    • Expressive (NP) but involved encodings
      
      \[(A \lor \neg B) \land (\neg A \lor C \lor D)\]

  – **SMT** – first order, Boolean + DS reasoning
    • Loss of efficiency
    • Improves expressivity and scalability

     \[(x+1 > 0 \lor \neg a \ [x] = b) \land (\neg P \ (y) \lor y = z)\]
Articles mentioning SMT over time
Applications of SMT

• Extended static checking
• Predicate abstraction
• Model checking
• Scheduling
• Test generation
• Synthesis
• Verification
CVC4

• State-of-the-art SMT Solver
  – Developed over last 5 years as successor of CVC3
• Supports many theories:
  – Arithmetic, Arrays, Bitvectors, UF
  – Inductive/Co-inductive Datatypes
  – New: Strings, Floating Points, Finite Sets
• Has strong performance:
  – Placed 1st in 14 of 32 divisions of SMT-COMP
  – Won TFA division of CASC J7
  – Competitive for many common SMT uses
The CVC4 Team

Clark Barrett (NYU)
Cesare Tinelli (U Iowa)

Kshitij Bansal (NYU)
François Bobot (CEA)
Chris Conway (Google)

Morgan Deters (NYU)
Liana Hadarean (NYU)
Dejan Jovanović (SRI)

Tim King (Verimag)
Tianyi Liang (U Iowa)
Andrew Reynolds (EPFL)
CVC4 : Quantifiers

• Handles (universally) quantified formulas

\[ \forall x : T. \ P( x ) \]

for all x of type T

⇒ Satisfiability problem with \( \forall \) is generally undecidable

• CVC4 handles quantifiers by:
  – Heuristic instantiation (E-matching)
  – Conflict-based instantiation [FMCAD 2014]
  – Finite model finding/model-based instantiation [CADE/CAV 2013]
  – Rewrite Rules
SMT Solver

Communicate via DPLL(T) Framework

SAT Solver

Arithmetic

UF

Datatypes

\( \forall \) Module

Decision Procedures
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

Instances of axioms

(Partial) Models

When current set is unsatisfiable

UNSAT
Running Example

- **Datatype** List

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

- **Length function** \( \text{len} : \text{List} \rightarrow \text{Int} \)

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

\[
\begin{align*}
\text{len}(\text{nil}) &= 0, \\
\text{len}(\text{cons}(0, \text{nil})) &\neq 1
\end{align*}
\]

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

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

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

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

len(nil) = 0,
len(cons(0,nil)) ≠ 1,
len(cons(0,nil)) = 1 + len(nil)

∀xy. len(cons(x,y)) = 1 + len(y)

Instantiate:
Q{x → 0, y → nil}

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

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

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

Ground Solver

\text{UNSAT}

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

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

Axioms

(Negated) Conjecture

Ground Solver

\( \forall \) Module
Example #2

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

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

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

\[ \text{len(nil)} = 0, \quad \text{len}(k) < 0 \]

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

Ground Solver

\[ \forall \text{ Module} \]
Example #2

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

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

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

Partial model:
\[
\{ \ldots, \text{len}(k) = -1, k = \text{cons}(\text{hd}(k), \text{tl}(k)) \}
\]

Ground Solver

∀ Module

Instantiate:
\[
\{ x \mapsto \text{hd}(k), y \mapsto \text{tl}(k) \}
\]
Example #2

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

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

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

Partial model:

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

Ground Solver

\[ \text{len} (\text{nil}) = 0, \]
\[ \text{len} (k) < 0, \]
\[ \text{len} (k) = 1 + \text{len} (\text{tl} (k)) \]
Example #2

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

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

\text{Instantiate:} \\
\{x \mapsto \text{hd}(\text{tl}(k)), y \mapsto \text{tl}(\text{tl}(k))\}

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

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

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

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

Ground Solver

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

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

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

\[\text{Partial model: } \{..., \text{len}(k) = -3, \text{len}(\text{tl}(k)) = -2, \text{len}(\text{tl}(\text{tl}(k))) = -1, \text{tl}(\text{tl}(k)) = \text{cons(}\text{hd}(\text{tl}(\text{tl}(k))), \text{tl}(\text{tl}(\text{tl}(k))))\} \]

\[\text{...repeat indefinitely}\]
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]
  – Induction as preprocessing step to SMT solver:
    • Dafny [Leino 2012]

• No SMT solvers support induction natively
  ⇒ 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 smallest CE to property:
  $$k = \text{cons}(\text{hd}(k), \text{tl}(k)) \Rightarrow \text{len}(\text{tl}(k)) \geq 0$$
Example #2: revised

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

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

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

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

Ground Solver

\text{UNSAT}

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

\forall Module
Skolemization with Inductive Strengthening

• General form:

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

  – For well-founded relation $R$

• Extends for multiple variables

• Common examples 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
    • QuickSpec [Claessen et al 2010]

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

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

(Negated) Conjecture

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

Axioms

Ground Solver

\forall \text{ Module}
**Subgoal Generation in SMT**

<table>
<thead>
<tr>
<th>Ground Solver</th>
<th>∀ Module</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \neg \forall x. \text{len} (\text{rev}(x)) = \text{len}(x) )</td>
<td>( \forall xy. \text{len}(\text{cons}(x, y)) = 1 + \text{len}(y) )</td>
</tr>
<tr>
<td>( \neg \forall x. \text{len}(x) = \mathbb{Z} )</td>
<td>( \forall x. \text{len}(x) = \mathbb{Z} )</td>
</tr>
<tr>
<td>( \neg \forall x. \text{app}(x, \text{nil}) = x )</td>
<td>( \forall x. \text{app}(x, \text{nil}) = x )</td>
</tr>
<tr>
<td>( \neg \forall xy. \text{len}(\text{app}(x, y)) = \text{plus}(\text{len}(x), \text{len}(y)) )</td>
<td>( \forall xy. \text{len}(\text{app}(x, y)) = \text{plus}(\text{len}(x), \text{len}(y)) )</td>
</tr>
</tbody>
</table>

- Enumerate subgoals
  - For each, either:
    - (a) it has a c.e.,
    - (b) holds universally

⇒ Use “splitting-on-demand”
Subgoal Generation in SMT

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

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

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

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

\[ \neg \forall x. \text{app}(x, \text{nil}) = x \]

\[ \forall x. \text{app}(x, \text{nil}) = x \]

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

\[ \forall y. \text{len}(\text{app}(x, y)) = \text{plus}(\text{len}(x), \text{len}(y)) \]
Subgoal Generation in SMT

- Each subgoal can be inductively strengthened
  - I.e. holds for a **smallest** counterexample

<table>
<thead>
<tr>
<th>Subgoal</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{len} (\text{rev}(k_0)) \neq \text{len}(k_0) ), ( \text{len} (\text{rev}(\text{tl}(k_0))) = \text{len}(\text{tl}(k_0)) )</td>
</tr>
<tr>
<td>( \text{len}(k_1) \neq Z ), ( \text{len}(\text{tl}(k_1)) = Z )</td>
</tr>
<tr>
<td>( \text{app}(k_2, \text{nil}) \neq k_2 ), ( \text{app}(\text{tl}(k_2), \text{nil}) = \text{tl}(k_2) )</td>
</tr>
<tr>
<td>( \text{len}(\text{app}(k_3, k_4)) \neq \text{plus}(\text{len}(k_3), \text{len}(k_4)) ), ...</td>
</tr>
</tbody>
</table>

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

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

\[ \forall x. \text{app}(x, \text{nil}) = x \]

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

- When negated s.g. is SAT, deduce ground information
- When negated s.g. is UNSAT, it can be used as axiom

\[
\begin{align*}
\text{len}(\text{rev}(k_0)) & \neq \text{len}(k_0), \\
\text{len}(\text{rev}(\text{tl}(k_0))) & = \text{len}(\text{tl}(k_0))
\end{align*}
\]

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

\[
\begin{align*}
\text{len}(k_1) & \neq Z, \text{len}(\text{tl}(k_1)) = Z, k_1 = \text{cons}(0, \text{nil}) \\
\forall x. \text{len}(x) & = Z \\
\text{app}(k_2, \text{nil}) & \neq k_2, \text{app}(\text{tl}(k_2), \text{nil}) = \text{tl}(k_2) \\
\forall x. \text{app}(x, \text{nil}) & = x \\
\end{align*}
\]

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

• Generate subgoals by enumeration
• Main challenge: scalability
• Keys to success:
  – Generate s.g. in a fair manner (smaller s.g. first)
  – Filter out s.g. that are not useful
Subgoal Filtering

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

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

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

• Filtering based on counterexamples:
  \( \forall x. \text{len}(x) = \text{len}(\text{app}(x, x)) \)
  – False, since we know \( \text{len}(x) \neq \text{len}(\text{app}(x, x)) \) for \( x \neq \text{nil} \)

⇒ Using techniques, typically can remove >95% subgoals
Experiments : Benchmarks

• Four benchmark sets:

  1. Isaplanner  [Johansson et al 2010]
     • List, Nats, Trees, (some) higher-order functions
  2. Clam      [Ireland 1996]
     • Lists, Nats, Sets
       – Designed specifically to require subgoals
  3. Hipspec   [Claessen et al 2013]
     • Lists, Nats
       – e.g. : sum of n cubes is square of nth triangle number
  4. Leon
     • Amortized Queues, Binary search trees, Leftist Heaps
Experiments : Encodings

1. Base encoding (dt), e.g. defined plus as:
   \[\forall x.\text{plus}(Z,x)=x\]
   \[\forall xy.\text{plus}(S(x),y)=S(\text{plus}(x,y))\]

2. Theory encoding (dtt)
   – Rephrase axioms/conjectures in terms of “+”

3. Theory-isomorphism encoding (dti)
   – Keep encoding, provide mappings to theory symbols:
     • Injection “toInt” from dt to int, with axiom:
     \[\forall xy.\text{toInt}(\text{plus}(x,y))=\text{toInt}(x)+\text{toInt}(y)\]

\[\Rightarrow 2,3 \text{ allow SMT solver to leverage theory reasoning} \]
   – Thus, we get subgoals for “free”, e.g.:
     \[A \models\text{\lowercase{\boldmath $T$}} \forall xy.\text{plus}(x,y)=\text{plus}(y,x)\]
## Results: SMT solvers

<table>
<thead>
<tr>
<th>dt</th>
<th>Isaplan</th>
<th>Clam</th>
<th>HSpec</th>
<th>Leon</th>
</tr>
</thead>
<tbody>
<tr>
<td>z3</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>cvc4</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>cvc4+i</td>
<td>68</td>
<td>16</td>
<td>12</td>
<td>29</td>
</tr>
<tr>
<td>cvc4+ig</td>
<td>75</td>
<td>38</td>
<td>17</td>
<td>34</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dti</th>
<th>Isaplan</th>
<th>Clam</th>
<th>HSpec</th>
<th>Leon</th>
</tr>
</thead>
<tbody>
<tr>
<td>z3</td>
<td>35</td>
<td>9</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>cvc4</td>
<td>34</td>
<td>7</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>cvc4+i</td>
<td>64</td>
<td>16</td>
<td>11</td>
<td>37</td>
</tr>
<tr>
<td>cvc4+ig</td>
<td>67</td>
<td>25</td>
<td>12</td>
<td>39</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>dtt</th>
<th>Isaplan</th>
<th>Clam</th>
<th>HSpec</th>
<th>Leon</th>
</tr>
</thead>
<tbody>
<tr>
<td>z3</td>
<td>35</td>
<td>9</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>cvc4</td>
<td>34</td>
<td>7</td>
<td>2</td>
<td>8</td>
</tr>
<tr>
<td>cvc4+i</td>
<td>64</td>
<td>16</td>
<td>11</td>
<td>37</td>
</tr>
<tr>
<td>cvc4+ig</td>
<td>67</td>
<td>25</td>
<td>12</td>
<td>39</td>
</tr>
</tbody>
</table>

| Total     | 85      | 50   | 26    | 45   |

- **300 second timeout**

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

• With subgoals, solved +37 for dti encoding
  – Only solved +1 when filtering turned off
• Most subgoals were small: term size $\leq 3$
• Can find simpler goals than by manual inspection:
  – For conjecture:
    $$\forall xy.\text{count}(x,y) = \text{count}(\text{insort}(x),y)$$
  – We thought it would require:
    $$\forall xy.\text{count}(\text{ins}(y,x),y) = \text{S}(\text{count}(x,y)), \forall xyz. y \neq z \Rightarrow \text{count}(\text{ins}(y,x),z) = \text{count}(x,z)$$
  – CVC4 found:
    $$\forall xyz.\text{count}(\text{ins}(y,x),z) = \text{count}(\text{cons}(y,x),z)$$

$\Rightarrow$ Suffices to prove conjecture, which CVC4 did fully automatically
## Comparison with Other Provers

<table>
<thead>
<tr>
<th></th>
<th>Isaplan</th>
<th>Clam</th>
<th>HSpec</th>
<th>Leon</th>
</tr>
</thead>
<tbody>
<tr>
<td>cvc4+ig (dti)</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>Hspec</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><strong>Total</strong></td>
<td><strong>85</strong></td>
<td><strong>50</strong></td>
<td><strong>26</strong></td>
<td><strong>45</strong></td>
</tr>
</tbody>
</table>

- Translated/evaluated in previous studies
- Tools tend to perform well on benchmarks they are tuned for
  - CVC4 competitive with state-of-the-art inductive theorem provers
Summary

• Techniques for Induction in CVC4
• Best performance by making use of:
  – Theory reasoning (dti encoding)
  – Subgoal generation
• Competitive with inductive theorem provers
Future Work

• Improvements to subgoal generation
  – Filtering heuristics
  – User-guided/interactive approaches
• Incorporate more induction schemes
• Completeness criteria
  – Identify cases approach is guaranteed to succeed
• Standard format for inductive theorem provers
• Applications:
  – Use within Leon verification tool (EPFL)
  – Synthesis of recursive functions
Thanks!

• CVC4 publicly available:
  – Induction techniques:
    • Enabled by "--quant-ind"
    • More details, see VMCAI 2015 paper

• Questions?