Prime Implicate Generation in Equational Logic

Mnacho Echenim      Nicolas Peltier      Sophie Tourret

Grenoble Informatics Laboratory
Max Planck Institute for Informatics

July 18th, 2018
Motivations

Abduction: search for explanations

\[ \text{Theory}, \ Hyp \models \ Obs \iff \text{Theory}, \ \neg \Obs \models \neg \Hyp \]

Implicate = consequence
Prime Implicate (PI) = most general consequence

Goal

Generate all PI of formulæ in equational logic.

Why equational logic?

- Many results available in propositional logic.
- Few practical results available in more expressive logics.
- Equality required for many applications (e.g. verification).
General approach

Given an input formula in CNF:

\[
\begin{align*}
\text{Generation of implicates} \\
cSP calculus \\
\quad s \simeq t \quad u[s'] \simeq v \\
\quad [u[t] \simeq v \mid s \simeq s']
\end{align*}
\]

\[
\begin{align*}
\text{Relevancy detection?} \\
\quad \text{projection test}
\end{align*}
\]

\[
\begin{align*}
\text{Storage of relevant implicates} \\
\quad \text{normal clausal tree}
\end{align*}
\]
Dealing with the equality predicate

Propositional logic: entailment $\models$ inclusion

$$\neg A \lor D \models \neg A \lor \neg B \lor \neg C \lor F \lor D$$

ground equational clauses built on constants and functions

Example: $e \neq b \lor b \neq c \lor f(a) \simeq f(b)$

Main challenge: the transitivity and substitutivity axioms

Equational logic: entailment $\not\models$ inclusion

$$e \neq c \lor a \simeq c \models e \neq b \lor b \neq c \lor f(a) \simeq f(b)$$

Solution: projection test!
Experimental results

Benchmarks:

<table>
<thead>
<tr>
<th>B1</th>
<th>B2</th>
</tr>
</thead>
<tbody>
<tr>
<td>random</td>
<td>random</td>
</tr>
<tr>
<td>without function symbols</td>
<td>with function symbols</td>
</tr>
</tbody>
</table>

- B1: $\text{cSP}_{\text{flat}} < \text{Zres}$ [Simon & Del Val, 2001]
- B2: $\text{cSP} < \text{cSP}_{\text{flat}} < \text{Zres} < \text{SOLAR}$ [Nabeshima et al., 2010]
Examples of applications

- Bug finding
- Ontology explanation
- Knowledge base consequences
- Query on an incomplete knowledge graph
Bug finding example

Program

In:

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>♣</td>
<td>♥</td>
<td>♠</td>
</tr>
</tbody>
</table>

↓

Out:

| ♣ | ♠ | ♥ | ♦ |

Property

<table>
<thead>
<tr>
<th></th>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>♣</td>
<td>♥</td>
<td>♠</td>
</tr>
</tbody>
</table>

Counter-examples:

<table>
<thead>
<tr>
<th>i = j = 1</th>
<th>i = 1</th>
<th>j = 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>♣</td>
<td>♠</td>
<td>♠</td>
</tr>
</tbody>
</table>

Abduction:

\[ i \simeq j \lor \text{cell}(i) \simeq \text{cell}(j) \]
Results

**Theory**
correctness proofs for cSP and redundancy deletion algorithms

**Implementation**
prototypes better than the state of the art

**Publications**
- 3 workshops [IWS12, ADDCT14, PAAR14]
- 3 conferences [IJCAI13, IJCAR14, CADE15]
- 1 journal [JAIR17]
Future work

- Extension of redundancy detection to handle variables.
- Implementation in an efficient inference engine.
- Extension to theories in an SMT fashion [IJCAR18].

Thank you for your attention.