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

$$Theory, Hyp \models Obs \quad \Leftrightarrow \quad Theory, \neg Obs \models \neg Hyp$$

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:

Generation of implicates
c$\mathcal{SP}$ calculus
$$\frac{s \simeq t \quad u[s'] \simeq v}{[\![u[t] \simeq v \mid s \simeq s']\!]}$$

→

Relevancy detection?
projection test

↓

Storage of relevant implicates
normal clausal tree

←

Dealing with the equality predicate

Propositional logic: entailment = inclusion

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

ground equational clauses built on constants and functions

Example: $e \not\simeq b \vee b \not\simeq c \vee f(a) \simeq f(b)$

Main challenge: the transitivity and substitutivity axioms

Equational logic: entailment ≠ inclusion

$$e \not\simeq c \vee a \simeq c \models e \not\simeq b \vee b \not\simeq c \vee f(a) \simeq f(b)$$

Solution: projection test!

Experimental results

Benchmarks:

B1	B2
random	random
without function symbols	**with** function symbols

B1

cSP_flat < Zres [Simon & Del Val, 2001]

B2

cSP < cSP_flat < Zres < 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:

	i	j	
♣	♡	♠	♢

↓

Out:

♣	♠	♡	♢

Property

	i	j	
♣	♡	♠	♢

$\neq$

♣	♠	♡	♢

Counter-examples:

$i = j = 1$
♣

,

$i = 1$	$j = 2$
♠	♠

Abduction: $i \simeq j \vee \text{cell}(i) \simeq \text{cell}(j)$

Results

Theory

correctness proofs for c$\mathcal{SP}$ 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.