



**m p i** max planck institut  
informatik

# Prime Implicate Generation in Equational Logic

**Mnacho Echenim   Nicolas Peltier   Sophie Touret**

**Grenoble Informatics Laboratory  
Max Planck Institute for Informatics**



July 18th, 2018

# Motivations

Abduction: search for explanations

$$\textit{Theory}, \textit{Hyp} \models \textit{Obs} \Leftrightarrow \textit{Theory}, \neg \textit{Obs} \models \neg \textit{Hyp}$$

Implicate = consequence

Prime Implicate (PI) = most general consequence

## Goal

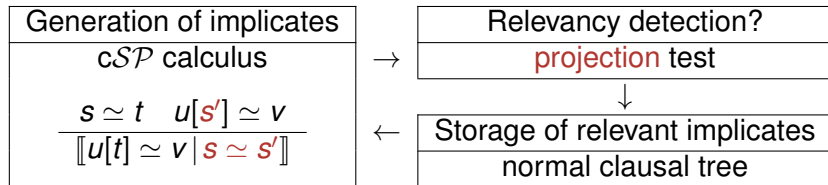
Generate **all** PI of formulæ in equational logic.

## 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:



# 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:  $e \neq b \vee b \neq c \vee f(a) \simeq f(b)$

Main challenge: the transitivity and substitutivity axioms

Equational logic: entailment  $\neq$  inclusion

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

Solution: **projection** test!

# Experimental results

Benchmarks:

B1	B2
random	random
<b>without</b> function symbols	<b>with</b> 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  $cSP$  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.

