Event-B & Cloud Provers

A. Iliasov D. Adjepon-Yamoah **P. Stankaitis**

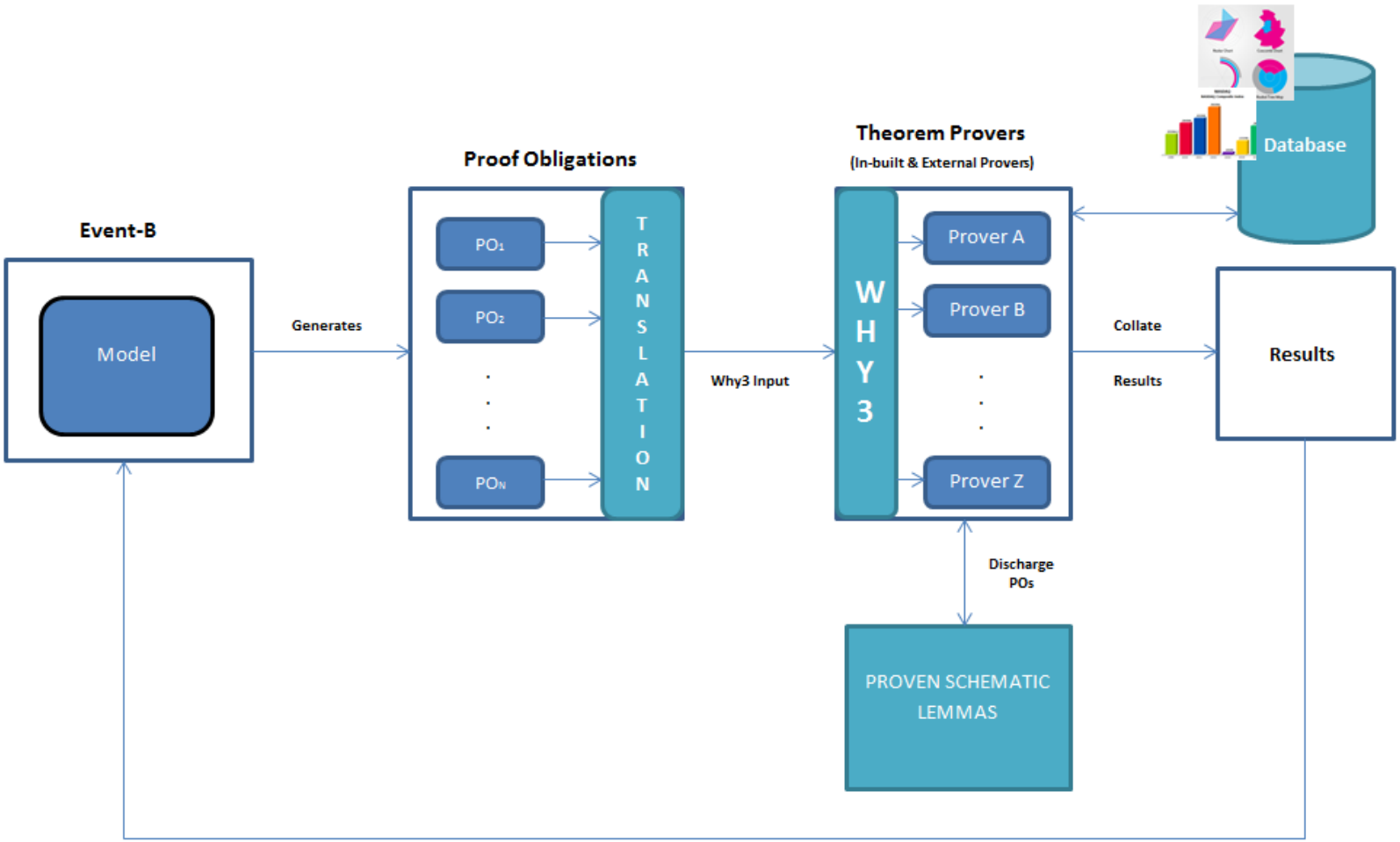
Department of Computer Science
Newcastle University

AI4FM workshop, 2015

Outline

Outline

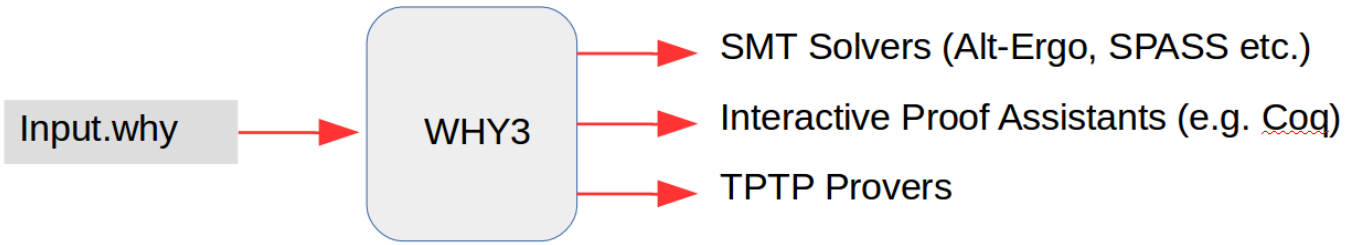
Visual Concept



Outline

Why3 Tool

Provides interface to theorem provers like SMT solvers (e.g Z3, Spass), TPTP provers, interactive proof assistants (e.g Coq)



Why3 Tool

Input is a collection of small units - theories, where as theory may contain functions, type declarations, predicates, axioms, lemmas and goals.

WHY3 Tool: Atelier B vs WHY3

Machine	# of PO	Unproved by Atelier B	Unproved by Why3
Automaton	4	0	0
Automaton_context_i	10	**8**	9
Automaton_i	229	71	**0**
Automaton_transitions	189	7	**0**
Automaton_transitions_i	1678	25	**0**
Boom_detectors_i	16	0	0
Configuration_i	7	4	**0**
Indicators_i	12	0	0
Lamps_bells_i	4	0	0
Timer	3	1	**0**
Timer_i	10	0	0
Track_circuit	2	0	0
Track_circuit_i	1	0	0
Train_detector_i	4	0	0
Warning_section	2	0	0
Warning_section_i	59	11	**10**
Warning_section_r	17	2	**0**
Total	2247	129	**19**

Number of proof obligation not discharged by the tools. [1]

[1] David Mentr, Claude March, Jean-Christophe Fillitre, Masashi Asuka (2012) Discharging Proof Obligations from Atelier B Using Multiple Automated Provers

Outline

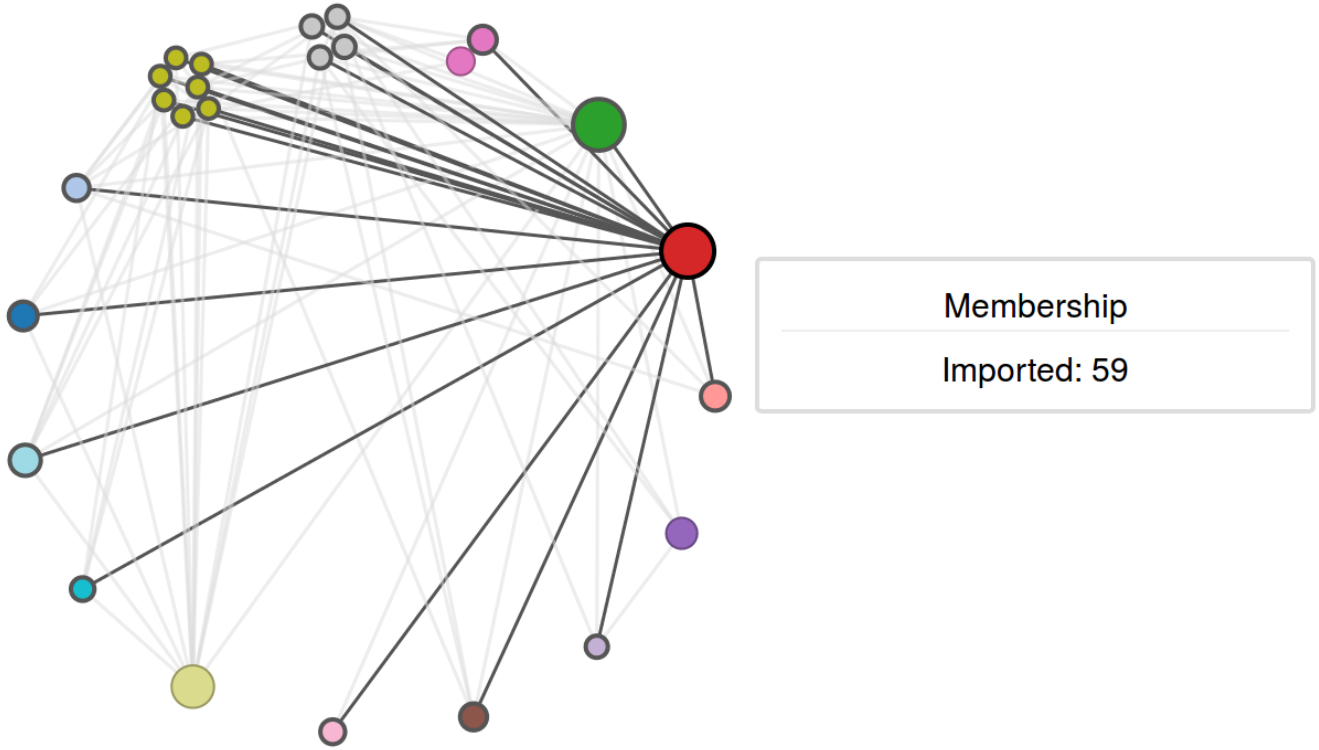
Translation and Axiomatization

- Event-B to Why3 *theory* input notation

In total 67 functions, 8 predicates, 3 constant relations were translated and defined by 78 axioms and 115 lemmas.

Translation and Axiomatization

In order to implement filtering mechanism operators were separated.



Example of Set Size:

```
function card (set 'a) : int

axiom card_def0:
forall s: set 'a. finite s -> card s >= 0

axiom card_def1:
forall x : 'a.
card (singleton x) = 1

axiom card_def2:
forall s: set 'a.
  ((finite s) /\ is_empty s) -> (card s) = 0

lemma card_def3:
forall s, t: set 'a.
(finite t /\ (subsetprop s t)) -> (card s) < (card t)
```

```
lemma card_def4:
forall s : set 'a, t : set 'b, f : rel 'a 'b.
(finite t /\ (mem f (s >->> t))) -> (card s = card t)

lemma lemma_def5: (*eProver 12s*)
forall s : set 'a, t : set 'b, f : rel 'a 'b.
((card s = card t) /\ mem f (s >-> t)) ->
mem f (s -->> t)

lemma lemma_card6: (*eProver 19s*)
forall s, t: set 'a.
finite t /\ (forall x : 'a. mem x s -> mem x t) ->
(card s) <= (card t)
```

Translation & Axiomatization

Simple mistake in translation file could result in inconsistent theory and thus allow to prove anything. Possible mistakes: missing finite statement, bi-implication used instead of implication.

Quick check: use dummy lemma e.g. $1 = 2$

Outline

Results

Model	Total POs	Open, built-in	Open, built-in + FA	Open, Open, built-in + z3($c \cdot \star$)
prime15r3	625	281/62[2]	18	201
paxos3a3	348	121	4	27
fishers	82	14	0	14

[2] The second figure includes POs discharged by SMT plug-in.

Outline

Server-side scenario language

- A client generates *n* verification conditions;
- These are sent, individually, to a cloud-based service;
- Each verification condition is treated by a *verification scenario* that may involve several provers;
- Scenario results are collated and, if necessary, some prover instances are terminated before they complete;
- An adjudicated response is communicated back to the client.

Server request API

The service accepts as inputs *sequents* $s \in \mathcal{S}$ of the form: $(\tau, l, \mathbf{T}, \mathbf{I}, \mathbf{H}, G)$ where

- τ defines the mathematical notation used for defining types, hypothesis and the goal. It can be, for instance, Classical B, Event-B, why3, SMT-LIB, and so on;
- l defines the client time-out in milliseconds;
- $\mathbf{T}$ is a set of types used in the sequent;
- $\mathbf{I}$ is a set of typed free identifiers occurring in the sequent;
- $\mathbf{H}$ is a set of sequent hypothesis;
- G is the sequent goal.

Scenario examples

$$\mathrm{pr}_1(c \cdot \star)$$

Apply some prover pr_1 to an input sequent

Scenario examples

$$FA \equiv \mathrm{pr}_1(c \cdot \star) \vee \mathrm{pr}_2(c \cdot \star) \vee \mathrm{pr}_1(c \cdot \star) \vee \ldots$$

Make use of multiple verification tools, trying them concurrently and report any (first) result.

Scenario examples

$$SA \equiv \mathrm{pr}_1(c \cdot \star) \wedge \mathrm{pr}_2(c \cdot \star) \wedge \mathrm{pr}_3(c \cdot \star) \wedge \ldots$$

Increase the confidence by adjudicating prover results.

Plans

- Machine learning to match a theorem with a most suitable prover and scenario;
- Server API with more input notations: Event-B, Classical B
- Non-real time operation mode: upload conditions on the cloud and they are processed in some spare and results are communicated back automatically.
- User-definable axiomatization extensions: must be proven and will be automatically integrated.
- Support for other input notations (of an interest to anybody?)

Outline

Demonstration