CSP-Prover

Markus Roggenbach, Swansea (Wales)
cooperation with Yoshinao Isobe, AIST (Japan)

IFIP WG 1.3 meeting, March 2007

- L. O'Reilly, Y. Isobe, M. Roggenbach: *Integrating Theorem Proving for Processes and Data*, PPL'07.
- Y. Isobe, M. Roggenbach: *A complete axiomatic semantics for the CSP stable failures model*, CONCUR'06.
- Y. Isobe, M. Roggenbach, S. Gruner: *Extending CSP-Prover by deadlock-analysis: Towards the verification of systolic arrays*, FOSE'05.
- Y. Isobe, M. Roggenbach: *A generic theorem prover of CSP refinement*, TACAS '05.

Outline

CSP-Prover

A complete axiomatic semantics for the stable failures model

CSP-CASL-Prover

CSP-Prover

The language CSP: One syntax . . .

Given an alphabet Σ (possibly infinite)

Basic processes

$$\begin{array}{rcl} P, Q & ::= & n(z_1, \ldots, z_k) \mid \mathit{Skip} \mid \mathit{Stop} \mid \mathit{Div} \\ & \mid & a \to P \mid y \to P \mid ?x : X \to P \\ & \mid & P \Box Q \mid P \sqcap Q \mid \mathit{if}\ \varphi\ \mathit{then}\ P\ \mathit{else}\ Q \\ & \mid & P \mathbin{[\![} X \mathbin{]\!]} Q \mid P \setminus X \mid P[[r]] \mid P \mathbin{;} Q \end{array}$$

(Systems of) equations

$$n(x_1, \ldots, x_k) = P$$

. . . various models

e.g.

- Traces model $\mathcal{T}$ – safety properties
- Failures divergence model $\mathcal{N}$ – livelock analysis
- Stable failures model $\mathcal{F}$ – deadlock analysis
- Stable revivals model $\mathcal{R}$ (2005) – responsiveness

Fairness: Models based on infinite traces

CSP models are denotational

Domain with antisymmetric refinement order e.g.
the domain $\mathcal{T}(A)$ of the traces model is the set of all non-empty and prefix closed subsets of $\Sigma^{*\checkmark}$; $T \sqsubseteq_{\mathcal{T}} S \Leftrightarrow S \subseteq T$.

Semantic clauses e.g.

$$
\begin{array}{rcl}
traces(Skip) & = & \{\langle\rangle, \langle\checkmark\rangle\} \\
traces(a \rightarrow P) & = & \{\langle\rangle\} \cup \{\langle a\rangle ^\frown s \mid s \in traces(P)\}
\end{array}
$$

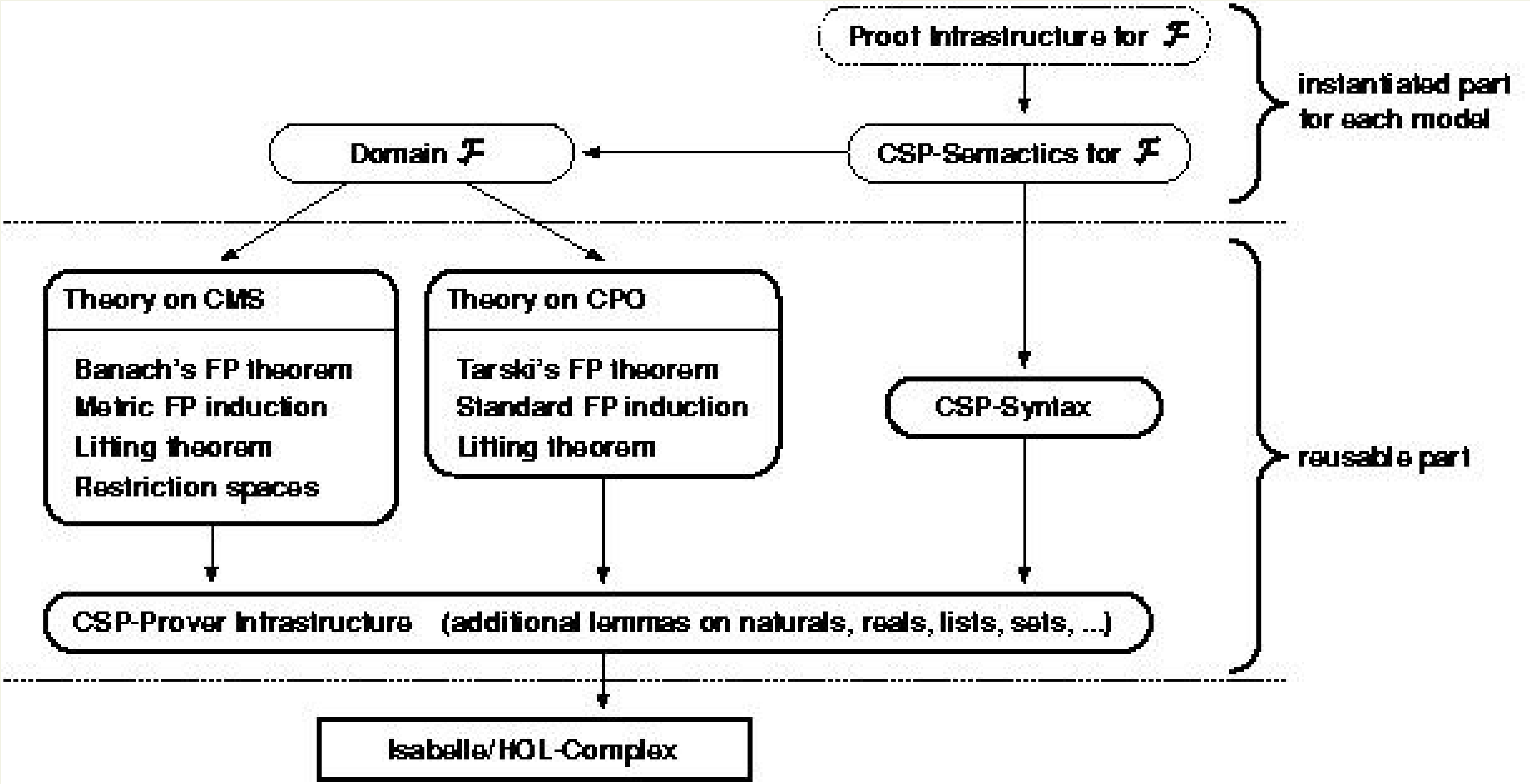
Fixed Point Theory Tarski & cpo or Banach & cms

CSP-Prover (TACAS 05)

Refinement proofs ‘$P \sqsubseteq Q$’ over different models

- Based on Isabelle-HOL (Complex)
- Generic architecture
- Currently implemented models: $\mathcal{F}$ (and $\mathcal{T}$)
- Deep encoding
- Different fixed-point theories (cms, cpo)

Implemented Isabelle Theories



An example proof

$$
\begin{array}{lcl}
NoLoss & = & coin \rightarrow item \rightarrow NoLoss \\
 & & \sqcap\ coin \rightarrow NoLoss \\
UnfairVM & = & button \rightarrow coin \rightarrow coin \rightarrow item \rightarrow UnfairVM
\end{array}
$$

claim:

$$NoLoss \sqsubseteq_{\mathcal{F}} UnfairVM \setminus \{button\}$$

Case studies

- Dining mathematicians (infinite state)
- EP2 dialogues (industrial application)
- Deadlock analysis of systolic algorithms (parametrised problems)
- Verification of process algebraic laws (Analysing CSP)
- Completeness proof of our axiomatic semantics (Analysing CSP)

Users

- Qinetic – industrial (& military?) applications
- TU Berlin (Timed CSP)

Alternative approaches

- Tej/Wolff: HOL-CSP (1997, 2003)
  - Based on Isabelle/HOL
  - Flat encoding
  - New partial order on processes
- Schneider/Dutertre (1997, 2002)
  - Based on PVS
  - Traces-model only
  - Tailored for security-protocols
- Badban/Fokkink/Groote/Pang/van de Pol: 'μCRL'-Prover (2005)
  - Based on PVS
  - Axiomatic


Computing
W
Our choice for book of the month
KATHY REICHS
CROSS BONES
SAVE
COMPUTER
Categories for Software Engineering
Algorithms for VLSI Design Automation
REALLY EASY
STEP-BY-STEP
COMPUTER

A complete axiomatic semantics for the CSP stable failures model

Complete Axiomatic Semantics for CSP

Best known result:

Finitely non-deterministic CSP over a finite alphabet
(Roscoe 1998, improving Brookes 1983)

Here:

Relative completeness for CSP over an arbitrary alphabet;
oracles for set theory & natural numbers
(side conditions require theorems on sets and naturals)

Completeness w.r.t. which language?

CSP_{TP} : (CSP for Theorem Proving) – fully abstract w.r.t. $\mathcal{F}$

$$
\begin{array}{lll}
P & ::= & \ldots \\
 & | & !_{set}\, X : \Delta \bullet P(X) \quad \%\% \text{ internal choice over } \Delta \subseteq \mathbb{P}(\Sigma) \\
 & | & !_{nat}\, n : N \bullet P(n) \quad \%\% \text{ internal choice over } N \subseteq \mathbf{N} \\
 & | & P \downarrow n \quad \%\% \text{ depth restriction to } n \in \mathbf{N}
\end{array}
$$

CSP_{Roscoe} (as used for theoretical studies)

$$
\begin{array}{lll}
P & ::= & \ldots \\
 & | & \sqcap S \quad \%\% \text{ internal choice over } S \subseteq P
\end{array}
$$

CSP_{M} (CSP for the model checker FDR)

$$
\begin{array}{lll}
P & ::= & \ldots \\
 & | & P \sqcap Q \quad \%\% \text{ finite internal choice}
\end{array}
$$

Key insight: No need for process equations

Let $X = P(X)$ have a solution S thanks to Tarski.

Then

$$S \;=\; \sqcup_{n \in \mathbf{N}} [\![P^n(Div)]\!]_{\mathcal{F}}$$

Key insight: No need for process equations

Let $X = P(X)$ have a solution S thanks to Tarski.

Then

$$\begin{array}{rcl} S & = & \sqcup_{n\in\mathbf{N}}[\![P^n(Div)]\!]_{\mathcal{F}} \\ & = & \cup_{n\in\mathbf{N}}[\![P^n(Div)]\!]_{\mathcal{F}} \end{array}$$

Key insight: No need for process equations

Let $X = P(X)$ have a solution S thanks to Tarski.

Then

$$\begin{aligned} S &= \sqcup_{n \in \mathbf{N}} [\![P^n(Div)]\!]_{\mathcal{F}} \\ &= \cup_{n \in \mathbf{N}} [\![P^n(Div)]\!]_{\mathcal{F}} \\ &= [\![\sqcap \{ P^n(Div) \mid n \in \mathbf{N} \}]\!]_{\mathcal{F}} \end{aligned}$$

Key insight: No need for process equations

Let $X = P(X)$ have a solution S thanks to Tarski.

Then

$$
\begin{aligned}
S &= \sqcup_{n \in \mathbf{N}} [\![P^n(Div)]\!]_{\mathcal{F}} \\
&= \cup_{n \in \mathbf{N}} [\![P^n(Div)]\!]_{\mathcal{F}} \\
&= [\![\sqcap \{ P^n(Div) \mid n \in \mathbf{N} \}]\!]_{\mathcal{F}} \\
&= [\![!_{nat}\, n : \mathbf{N} \bullet P^n(Div) \}]\!]_{\mathcal{F}}
\end{aligned}
$$

Key insight: No need for process equations

Let $X = P(X)$ have a solution S thanks to Tarski.

Then

$$
\begin{aligned}
S &= \sqcup_{n \in \mathbf{N}} [\![P^n(Div)]\!]_{\mathcal{F}} \\
&= \cup_{n \in \mathbf{N}} [\![P^n(Div)]\!]_{\mathcal{F}} \\
&= [\![\sqcap \{ P^n(Div) \mid n \in \mathbf{N} \}]\!]_{\mathcal{F}} \\
&= [\![!_{nat}\, n : \mathbf{N} \bullet P^n(Div) \}]\!]_{\mathcal{F}}
\end{aligned}
$$

Theorem: All classical CSP operators are continuous over $\mathcal{F}$.

Axiom system $A_{\mathcal{F}}$ ($\sim$ 80 cond. eq.)

1. Congruence axioms, e.g.

$$P = Q \Rightarrow P \downarrow n = Q \downarrow n$$

2. Basic axioms, e.g.

$$P \downarrow n = Div \quad (P \downarrow n) \downarrow m = P \downarrow \min(n, m)$$

3. Distributivity axioms, e.g.

$$(P_1 \sqcap P_2) \downarrow n = (P_1 \downarrow n) \sqcap (P_2 \downarrow n)$$

4. Step laws, e.g.

$$(?\, x : A \to P(x)) \downarrow (n + 1) = ?\, x : A \to (P(x) \downarrow n)$$

5. *Skip* & *Div* axioms, e.g.

$$Skip \downarrow (n + 1) = Skip \quad Div \downarrow n = Div$$

Changes compared to Roscoe'98

Language

1. Inclusion of the depth restriction operator $P \downarrow n$
2. Exclusion of recursion

Axioms

1. Two axioms needed to be corrected – see below
2. Added axioms for $\downarrow$ and infinite internal choice
3. Three additional axioms on the process Div

Correction of two step laws by Roscoe

$P \rhd Q := (P \sqcap Stop) \mathbin{\Box} Q$

Roscoe's version

$$
\begin{array}{rl}
 & (P \rhd P') \mathbin{[\![\, X \,]\!]} (Q \rhd Q') \\
= & (P \mathbin{[\![\, X \,]\!]} Q) \rhd ((P' \mathbin{[\![\, X \,]\!]} (Q \rhd Q')) \sqcap ((P \rhd P') \mathbin{[\![\, X \,]\!]} Q'))
\end{array}
$$

Correction of two step laws by Roscoe

$P \rhd Q := (P \sqcap Stop) \;\Box\; Q$

Roscoe's version

$$\begin{array}{rl} & (P \rhd P') \,[\![X]\!]\, (Q \rhd Q') \\ = & (P \,[\![X]\!]\, Q) \rhd ((P' \,[\![X]\!]\, (Q \rhd Q')) \sqcap ((P \rhd P') \,[\![X]\!]\, Q')) \end{array}$$

Correction

Let $P = (?x : A \rightarrow P'(x)) \rhd P''$, $Q = (?x : B \rightarrow Q'(x)) \rhd Q''$

$$\begin{array}{rcl} P \,[\![X]\!]\, Q & = & (?x : ((X \cap A \cap B) \cup (A - X) \cup (B - X)) \rightarrow \\ & & \mathit{if}\ (x \in X)\ \mathit{then}\ (P'(x) \,[\![X]\!]\, Q'(x)) \\ & & \mathit{else\ if}\ (x \in A \cap B)\ \mathit{then}\ ((P'(x) \,[\![X]\!]\, Q) \sqcap (P \,[\![X]\!]\, Q'(x))) \\ & & \mathit{else\ if}\ (x \in A)\ \mathit{then}\ (P'(x) \,[\![X]\!]\, Q)\ \mathit{else}\ (P \,[\![X]\!]\, Q'(x))) \\ & & \rhd ((P'' \,[\![X]\!]\, Q) \sqcap (P \,[\![X]\!]\, Q'')) \end{array}$$

Bill Roscoe's reaction

$>$ my colleague Yoshinao Isobe (AIST, Japan) and I found
$>$ counter examples to the step laws for . . .

You are right about them...

I think that, implicitly, it demonstrates that, soon, presentations of similar models and axiom schemes will only be "complete" once they have been accompanied by similar mechanised theorem proving.

Main Result

$A_{\mathcal{F}}$ is sound and complete, i.e.

$$A \vdash P = Q \Leftrightarrow [\![P]\!]_{\mathcal{F}} = [\![Q]\!]_{\mathcal{F}}$$

Soundness: lots of work, however boring . . .

Completeness: based on normalisation

Main Result

$A_{\mathcal{F}}$ is sound and complete, i.e.

$$A \vdash P = Q \Leftrightarrow [\![P]\!]_{\mathcal{F}} = [\![Q]\!]_{\mathcal{F}}$$

Soundness: lots of work, however boring . . .

Completeness: based on normalisation

Remark: $P \sqsubseteq Q$ is equivalent to $P = P \sqcap Q$

Step 1: Sequentialising

'remove hiding, renaming, parallel composition'

1. Definition of 'full sequential form' $SeqProc_\Sigma$
2. Definition of a sequentializing function by induction on the process structure, e.g.

$$P \parallel [\, X \,] \parallel_{seq} Skip = !\, n : N \bullet (P'(n) \parallel [\, X \,] \parallel_{seq} Skip)$$
$$\text{for } P = !\, n : N \bullet P'(n) \in SeqProc_\Sigma$$
$$R \parallel [\, X \,] \parallel_{seq} Skip = (?\, x : (A - X) \to (P'(x) \parallel [\, X \,] \parallel_{seq} Skip)) \Box Q$$
$$\text{for } R = \ (?\, x : A \to P'(x)) \Box Q \in SeqProc_\Sigma$$

3. Theorem: $Seq(P) \in SeqProc_\Sigma$ and $A_{\mathcal{F}} \vdash P = Seq(P)$

Step 2: Normalisation

1. Adaption of Rosoe's full normal form to infinite alphabets:

$$((?x : A \rightarrow P(x)) \,\Box\, Q) \sqcap (!_{set}\, X : \Delta \bullet (?x : X \rightarrow \mathit{Div}))$$

$\bigcup \Delta \subseteq A$,
$(\exists\, X_0 \in \Delta.\, X_0 \subseteq X \subseteq A) \Rightarrow X \in \Delta$,
$P(x)$ is in full normal form, $Q \in \{\mathit{Skip}, \mathit{Div}\}$

Results:

- Does not capture all processes
- Theorem:
  For all P, Q in normal from holds: $[\![P]\!]_{\mathcal{F}} = [\![Q]\!]_{\mathcal{F}} \Leftrightarrow P = Q$.

Normalization (continued)

2. Definition of processes in 'extended full normal form' $XNormProc_{\Sigma}$:

$$!_{nat}\, n : \mathbf{N} \bullet P(n)$$

   - all $P(n)$ in full normal form,
   - $[\![P(n)]\!]_{\mathcal{F}} = [\![(!nat\ \ n : \mathbf{N} \bullet P(n)) \downarrow n]\!]_{\mathcal{F}}$

3. Theorem: $\forall\, P, Q \in XNormProc_{\Sigma} : P = Q \Leftrightarrow [\![P]\!]_{\mathcal{F}} = [\![Q]\!]_{\mathcal{F}}$

4. Define $XNorm(P) =!_{nat}\, n : \mathbf{N} \bullet (Norm_{(n)}(Seq(P)))$.

5. Theorem:

   $XNorm(P) \in XNormProc_{\Sigma}$ and $A_{\mathcal{F}} \vdash\ P =\ XNorm(P)$

Gluing things together

Let $[\![P]\!]_{\mathcal{F}} = [\![Q]\!]_{\mathcal{F}}$.

We know

- $A_{\mathcal{F}} \vdash \; P = \; XNorm(P)$
- $A_{\mathcal{F}} \vdash \; Q = \; XNorm(Q)$

As $A_{\mathcal{F}}$ is sound, we have

- $[\![XNorm(P)]\!]_{\mathcal{F}} = [\![P]\!]_{\mathcal{F}} = [\![Q]\!]_{\mathcal{F}} = \; [\![XNorm(Q)]\!]_{\mathcal{F}}$

As $[\![XNorm(P)]\!]_{\mathcal{F}} = [\![XNorm(Q)]\!]_{\mathcal{F}} \in XNormProc_{\Sigma}$:

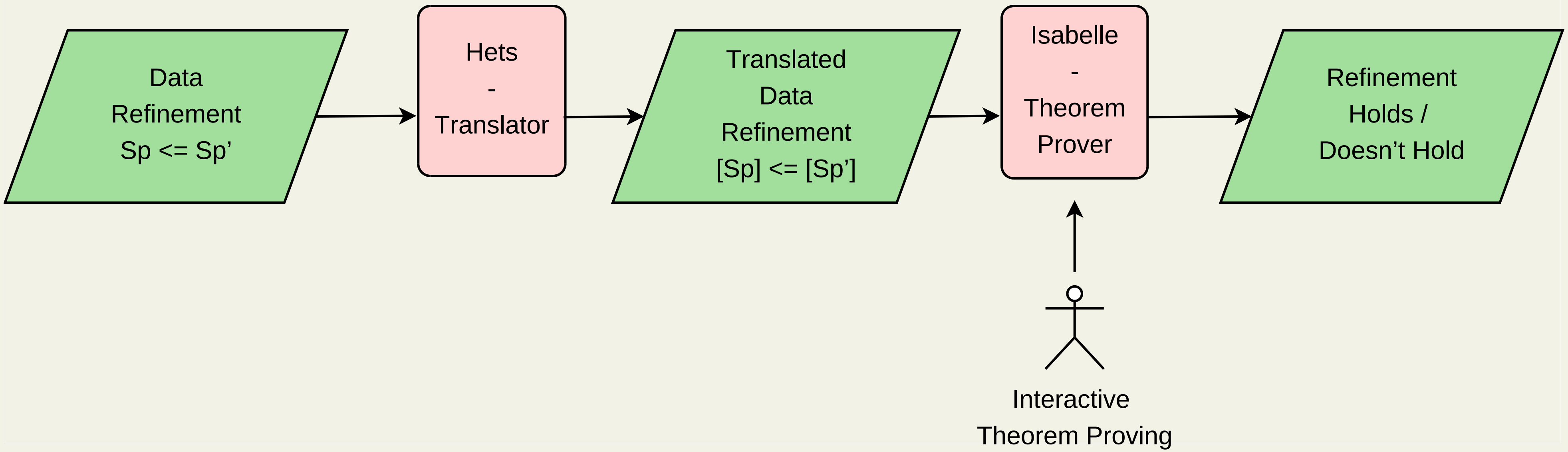
$$XNorm(P) = XNorm(Q)$$

Towards Integrated Theorem Proving for Processes and Data

with Liam O'Reilly

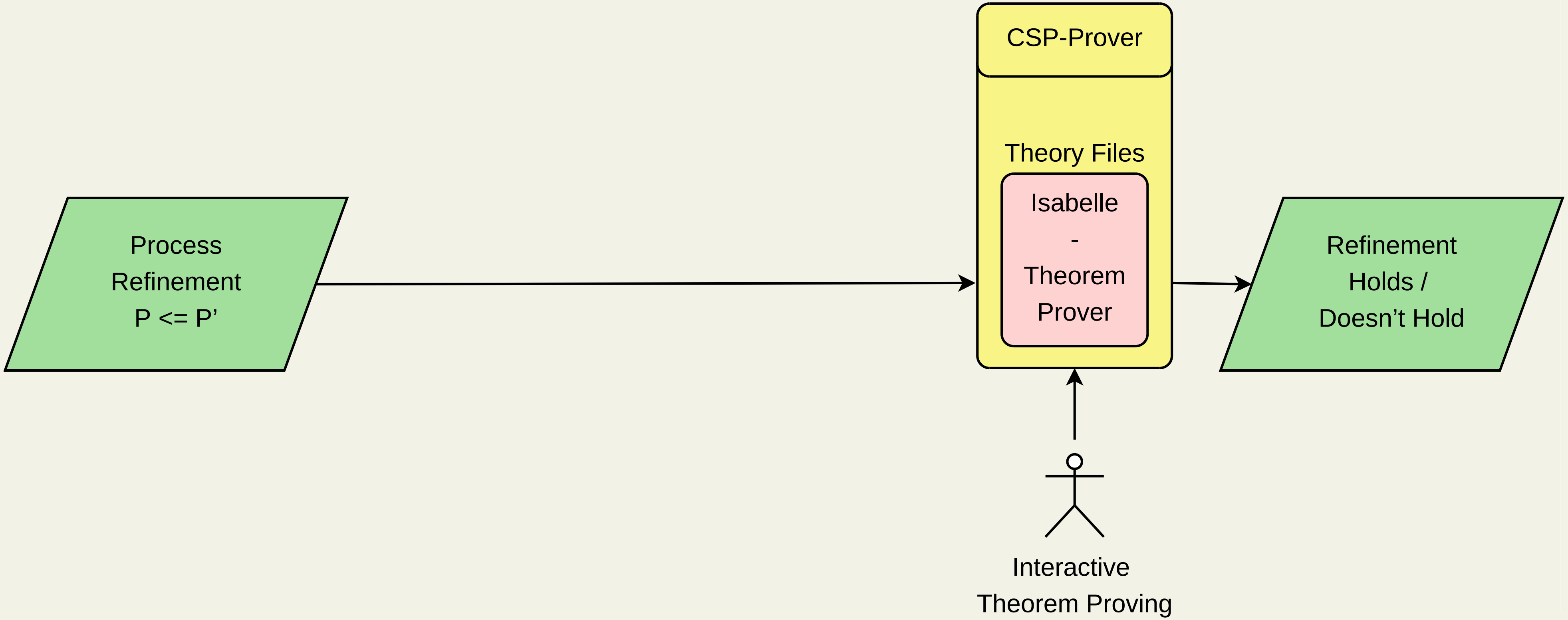
Software Architecture of CSP-CASL-Prover

Theorem Proving on CASL alone



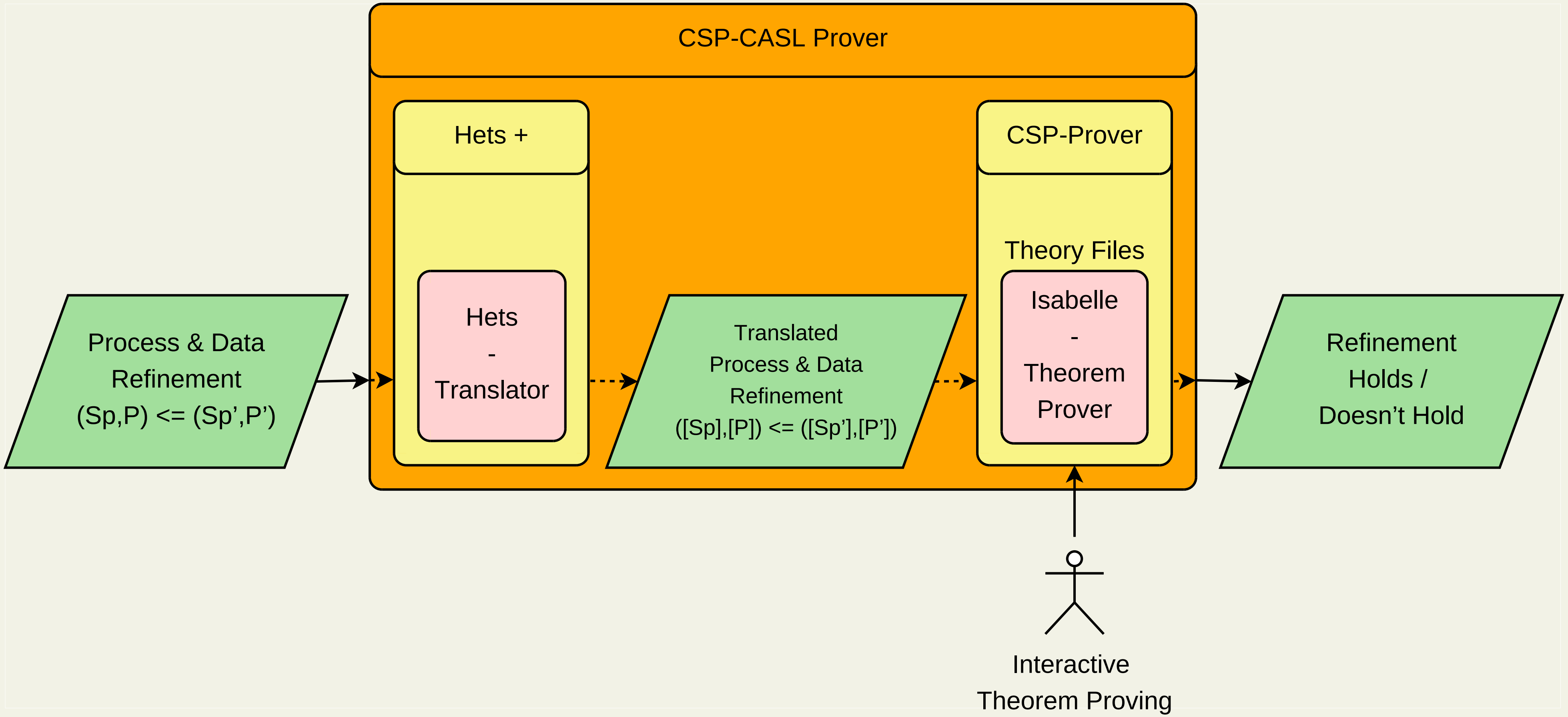
Software Architecture of CSP-CASL-Prover

Theorem Proving on CSP alone

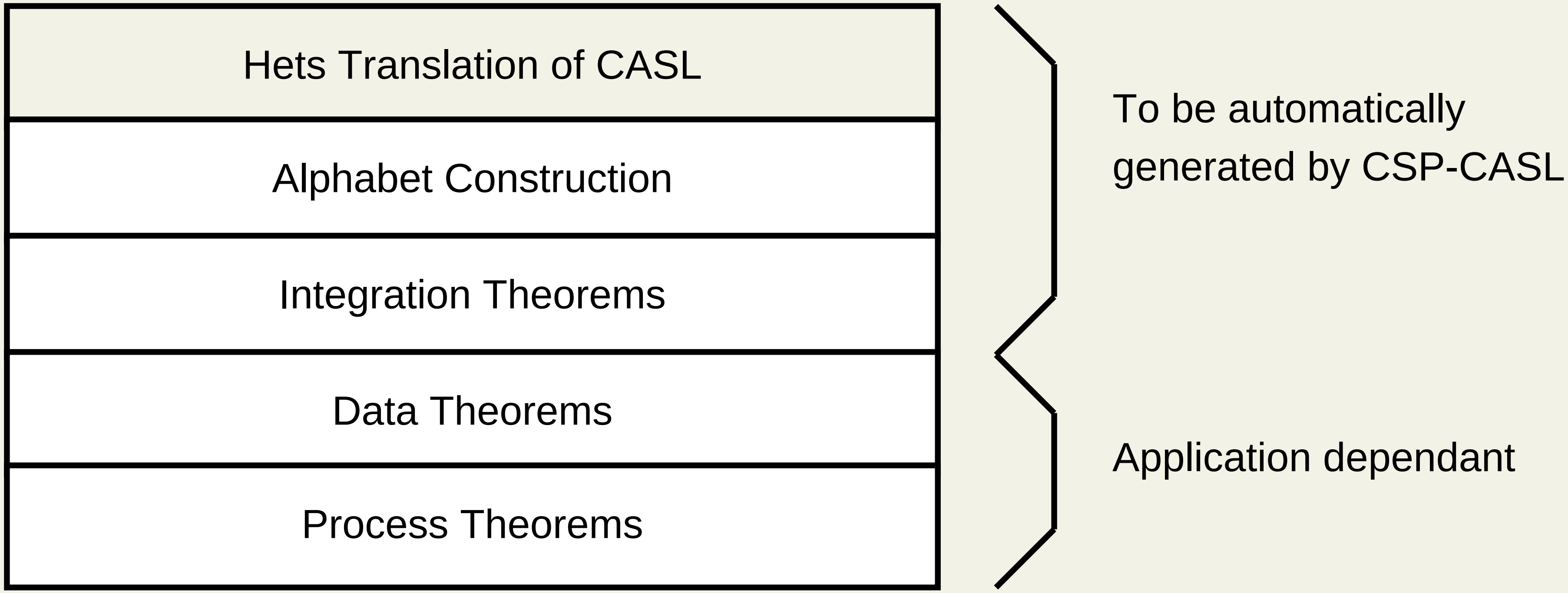


Software Architecture of CSP-CASL-Prover

Theorem Proving on CSP-CASL



Prototypical Structure of the Associated Theory File



One of the 4 challenges: Subsorting

data	**sorts**	$S < T$
	ops	$c : S;\ d : T$
	axiom	$c = d$
process	$c \rightarrow SKIP \ \|\| \ d \rightarrow SKIP$	

is equivalent to (i.e. $<=$ and $>=$ hold)

data	**sorts**	$S < T$
	ops	$c : S;\ d : T$
	axiom	$c = d$
process	$c \rightarrow SKIP$	

Summary & Future Work

Summary

- CSP-Prover works well
- Complete axiomatic semantics for $\mathcal{F}$
- First steps towards CSP-CASL-Prover

Future Work

- CSP-Prover Version 4 (polished syntax & improved tactics)
- Integration of CSP-Prover with FDR
- Models currently ‘under construction’
  - traces model $\mathcal{T}$
  - stable revivals model $\mathcal{R}$ – MPhil of Gift Samuel
- Implementing CSP-CASL-Prover –
  - MPhils of Andy Gimblett & Liam O’Reilly
- Testing
  - PhD of Teme Kahsai
- Analysing Parallel Algorithms

My group in Swansea (at a visit in Gregygnog)

CSP channels

A': set of 'basic' communications

C: set of channel names; $dom(c) \subseteq A' \times \ldots \times A'$ for $c \in C$

Alphabet with channels:

$$A := A' \cup \bigcup_{c \in C} \{c.x_1. \ldots .x_n \mid (x_1, \ldots, x_n) \in dom(c)\}$$

Sending communication a over c:

$$c!a := c.a$$

Receiving value x over c:

$$c?x \to P(x) \;\; := \; ?y : \{c.x \mid x \in dom(c)\} \to P(\pi_2(y))$$