Towards Testing from CSP-CASL

Markus Roggenbach, Swansea (Wales)

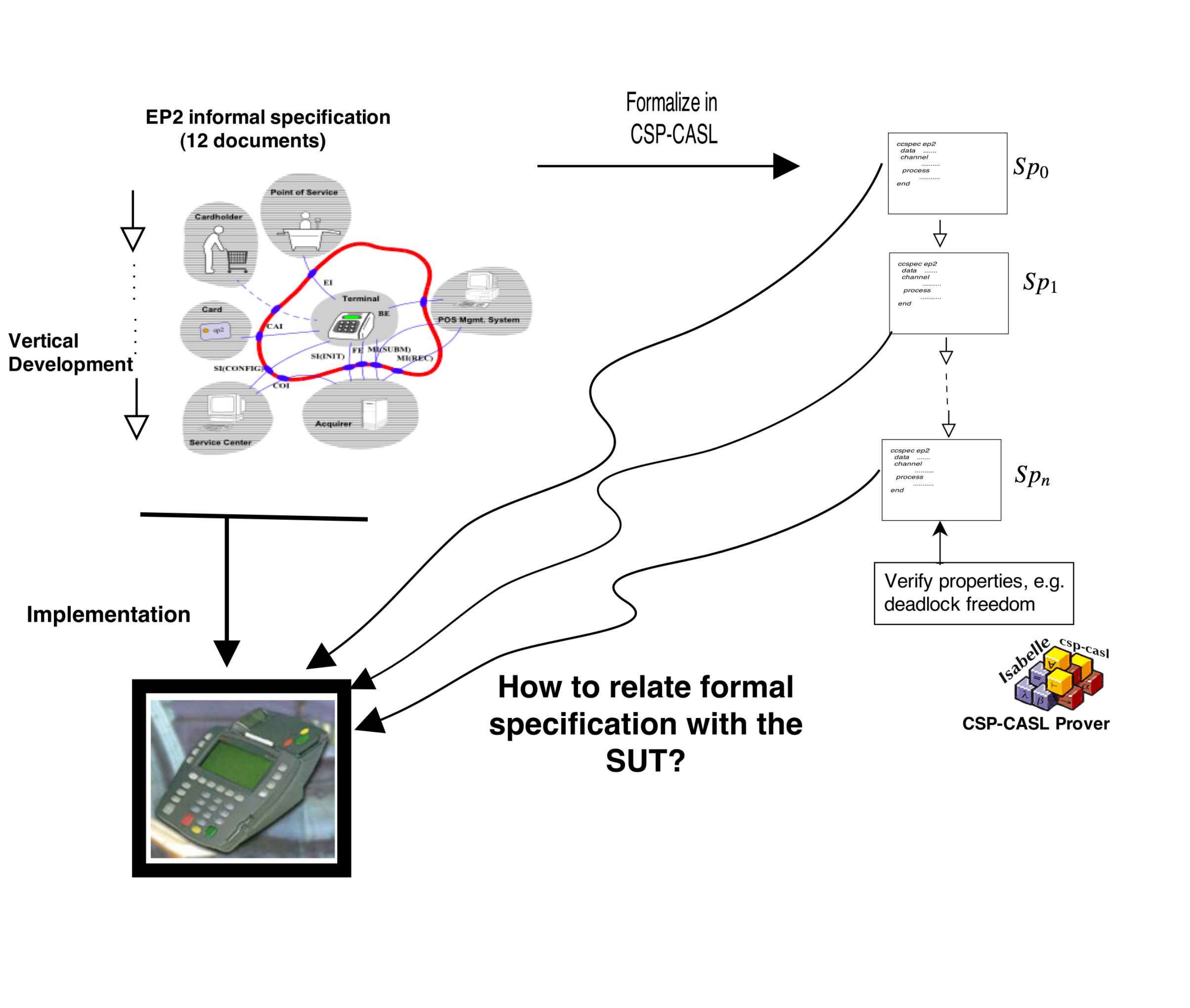
– work in progress –

ongoing cooperation with

B-H Schlingloff, Berlin & T Kahsai, Swansea

IFIP WG 1.3 meeting, January 2008

- T. Kahsai, M. Roggenbach, H. Schlingloff: *Specification-based testing for refinement*, SEFM 2007
- L. O'Reilly, Y. Isobe, M. Roggenbach: Csp-Casl-*Prover – Tool integration and algorithms for automated proof generation*, CALCO-Jnr 2007, to appear
- M. Roggenbach: *CSP-CASL: A new integration of process algebra and algebraic specification*, TCS 2006


EP2 informal specification
(12 documents)
Formalize in
CSP-CASL
Vertical
Development
Cardholder
Point of Service
Terminal
Card
POS Mgmt. System
Service Center
Acquirer
Sp_0
Sp_1
Sp_n
Implementation
Verify properties, e.g.
deadlock freedom
How to relate formal
specification with the
SUT?
CSP-CASL Prover

Outline

A specification exercise in CSP-CASL

A conformance relation

Relating Refinement and Conformance

A specification exercise in CSP-CASL

Refinement 'slogans'

CASL: 'less models'

CSP: 'less non-determinism'

CSP-CASL: 'less models or less non-determinism'

Setting up the interface

ccspec $\text{BCALC0} =$

data sort $Number$

ops $0, 1 : Number;$

$\_\_ + \_\_ : Number \times Number \rightarrow ?\ Number$

channel

$Button, Display : Number$

process

$$P_0 = (?x : Button \rightarrow P_0) \sqcap (!y : Display \rightarrow P_0)$$

$Button!0 \rightarrow Button!0$ is 'left open' behaviour.

Alternating buttons and display

ccspec $\textrm{BC}\textsc{alc}1 =$
data sort $Number$
ops $0, 1 : Number;$
$\_\_ + \_\_ : Number \times Number \to ? \ Number$
channel
$Button, Display : Number$
process
$P_1 = ?x : Button \to !y : Display \to P_1$

$Button!0 \to Button!0$ is 'unwanted' behaviour.

Fixing the displayed value

ccspec $\text{BC}\textsc{alc}2 =$

data sort $Number$

ops $0, 1 : Number;$

$\_\_ + \_\_ : Number \times Number \rightarrow ? \ Number$

channel

$Button, Display : Number$

process

$$P_2 = ?x : Button \rightarrow Display!x \rightarrow ?y : Button \rightarrow Display!(x + y) \rightarrow P_2$$

$Button!0 \rightarrow Display!0 \rightarrow Button!1 \rightarrow Display!1$ is 'left open' behaviour.

Basic arithmetic

ccspec $\mathrm{BC{\scriptsize ALC}3} =$

data sort $Number$

ops $0, 1 : Number;$

$\_\_ + \_\_ : Number \times Number \to ?\ Number$

axioms $0 + 0 = 0;\ 0 + 1 = 1;\ 1 + 0 = 1;\ \neg(0 = 1)$

channel

$Button, Display : Number$

process

$$P_3 = ?x : Button \to Display!x$$
$$\to ?y : Button \to Display!(x + y) \to P_3$$

$Button!0 \to Display!0 \to Button!1 \to Display!1$ is 'intended' behaviour.

1-bit arithmetic

ccspec $\mathrm{BC{\small ALC}4} =$

data CARDINAL [**op** $WordLength : Nat = 1$]

with sort $CARDINAL \mapsto Number$ **reveal** ...

channel

$Button, Display : \; Number$

process

$$
\begin{aligned}
P_4 \; &= ?x : Button \rightarrow \; Display!x \\
&\rightarrow ?y : Button \rightarrow \; Display!(x + y) \rightarrow P_4
\end{aligned}
$$

monomorphic data, no internal non-determinism:
behaviour either 'unwanted' or 'intended'

Refinements

$$\textrm{BC}\textsc{alc}0 \rightsquigarrow_{\mathcal{F}} \textrm{BC}\textsc{alc}1 \rightsquigarrow_{\mathcal{F}} \textrm{BC}\textsc{alc}2 \rightsquigarrow_{\mathcal{F}} \textrm{BC}\textsc{alc}3 \rightsquigarrow_{\mathcal{F}} \textrm{BC}\textsc{alc}4$$

process refinement: 'constant data part'

$$\textrm{BC}\textsc{alc}0 \rightsquigarrow_{\mathcal{F}}^{process} \textrm{BC}\textsc{alc}1 \rightsquigarrow_{\mathcal{F}}^{process} \textrm{BC}\textsc{alc}2$$

data refinement: 'constant process part'

$$\textrm{BC}\textsc{alc}2 \rightsquigarrow^{data} \textrm{BC}\textsc{alc}3 \rightsquigarrow^{data} \textrm{BC}\textsc{alc}4$$

process refinement and data refinement imply CSP-CASL refinement

A conformance relation

Test case

Given:

- (Sp, P) CSP-CASL specification

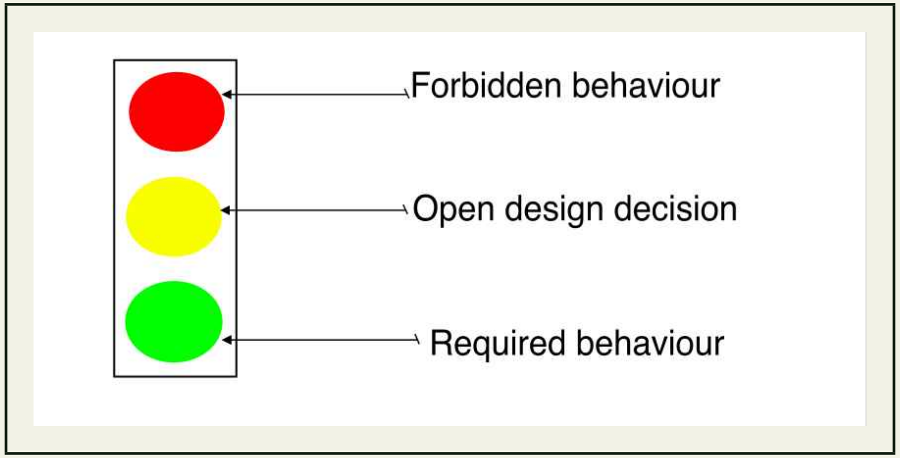
A test case T is any linear CSP-CASL process in the signature of Sp.

$$\boxed{\text{e.g.} \quad Button!0 \rightarrow Display!0 \rightarrow Button!1 \rightarrow Display!1 \rightarrow Stop}$$

Remark: In the paper also terms with variables.

Colouring test cases

The colour of test T with respect to (Sp, P) is a value in $\{red, yellow, green\}$.



Formal definition of colouring

For consistent Sp :

- colour(T) $=$ green iff
  for all $M \in \mathbf{Mod}(Sp)$ and all variable evaluations $\nu : X \to M$:
  (a) $traces([\![T]\!]_\nu) \subseteq traces([\![P]\!]_{\emptyset:\emptyset\to\beta(M)})$ and
  (b) for all $tr = \langle t_1, \ldots t_n\rangle \in traces([\![T]\!]_\nu)$, $1 \leq i \leq n$:
  $(\langle t_1, \ldots, t_{i-1}\rangle, \{t_i\}) \notin failures([\![P]\!]_{\emptyset:\emptyset\to\beta(M)})$
- colour(T) $=$ red iff
  for all models $M \in \mathbf{Mod}(Sp)$ and all variable evaluations $\nu : X \to M$:
  $traces([\![T]\!]_\nu) \not\subseteq traces([\![P]\!]_{\emptyset:\emptyset\to\beta(M)})$
- colour(T) $=$ yellow otherwise.

Remark: works also for arbitrary, non-linear processes T.

Basic properties

- $STOP$ is alway green
- prefixes of green test cases are green
- extensions of red test cases are red

- $(SP, P) \rightsquigarrow_{\mathcal{T}} (Sp, T)$ for a green test case T
- $(SP, P) \rightsquigarrow_{\mathcal{T}} (Sp, \Box \, \{T \mid color(T) = green\})$

- However: these refinment results do not carry over to $\mathcal{F}$

Syntactic characterization theorems

colour(T)= green w.r.t. (Sp, P) semantical definition

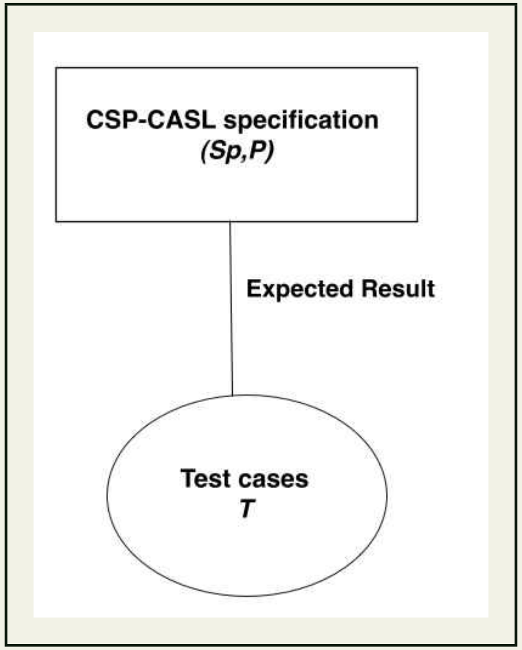
iff

1. $(Sp', check_F(T, P)) =_{\mathcal{T}} (Sp', OK \to Stop)$ syntactic characterization
2. $(Sp'', check_F(T, P) =_{\mathcal{F}} (Sp'', Div)$

Proof in CSP-CASL Prover of condition 1 for
$Button!0 \to Display!0 \to Button!1 \to Display!1 \to Stop$ w.r.t. BCALC3

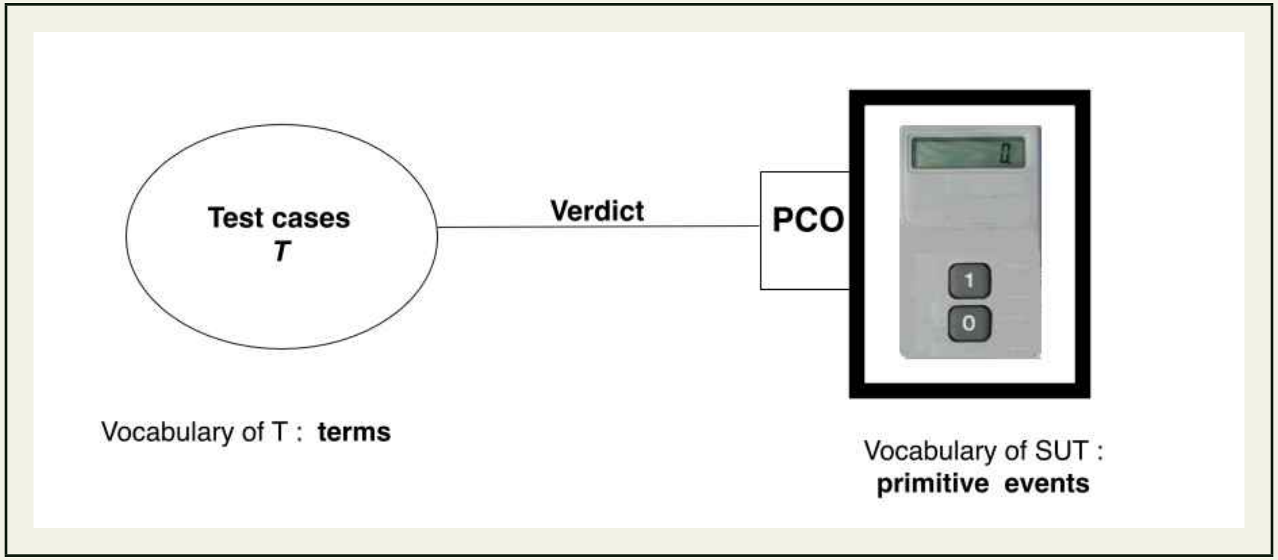
So far . . .

expected result of a test case with respect (Sp, P)



Now:

How to execute a test case w.r.t. a particular SUT?



Point of control and observation (PCO)

Given: System under Test (SUT) and specification (Sp, P)

A PCO $\mathcal{P} = (\mathcal{A}, \|...\|, \mathcal{D})$ of an SUT consists of:

- an alphabet $\mathcal{A}$ of primitive events
- a mapping $\|...\| : \mathcal{A} \longrightarrow T_{\Sigma}$
- a direction $D : \mathcal{A} \longrightarrow \{ts2sut, sut2ts\}$.

A PCO for the calculator example

$$\mathcal{A} = \{button_0,\ button_1,\ display_0,\ display_1\}$$

$$\begin{aligned} \|button_0\| &= Button.0 \\ \|button_1\| &= Button.1 \\ \|display_0\| &= Display.(0+0) \\ \|display_1\| &= Display.1 \end{aligned}$$

$ts2sut$ – button events

$sut2ts$ – display events

Matching test case and PCO

A test case

$$T \text{ is executable}$$

w.r.t.

- PCO $\mathcal{P}$ and
- specification (Sp, P)

if the primitive events of the PCO 'uniquely cover' the terms of T.

Remark: 'covarage' includes the test oracle problem of Alg Spec.

Test verdict

The execution of a test T at a particular SUT yields a verdict in

$$\{pass, fail, inconclusive\}$$

w.r.t. to a specification (Sp, P).

- Pass – increased confidence in SUT w.r.t. (SP, P)
- Fail – violation of the intentions described in (Sp, P)
- Inconclusive – neither increased nor destroyed confidence

This test verdict is defined algorithmically.

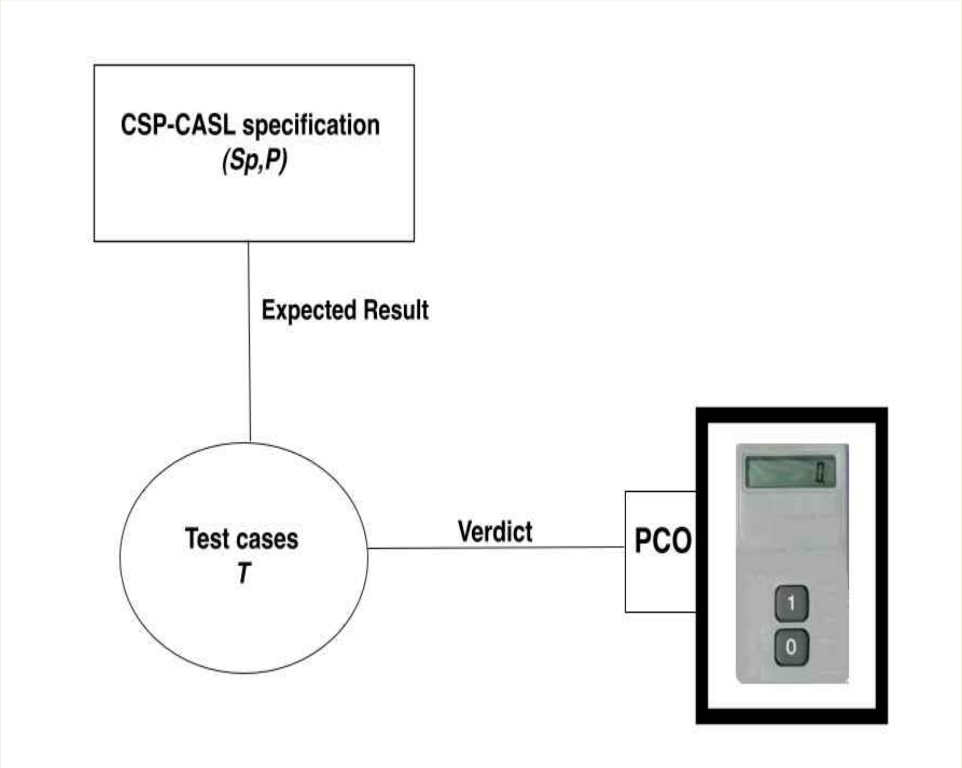
Testing the calculator using Java Reflection

- SUT: correct (!) binary calculator implemented in Java (Java swing)
- Test environment (Java abbot)
  - uses reflection to find the components (button, display) of SUT
  - instantiates robot to stimulate SUT/check for output from SUT
- Test (green w.r.t. $\textsc{Bcalc3}$):

$$\mathit{Button}!0 \rightarrow \mathit{Display}!0 \rightarrow \mathit{Button}!1 \rightarrow \mathit{Display}!1 \rightarrow \mathit{Stop}$$

xterm: test.sh

The conformance relation

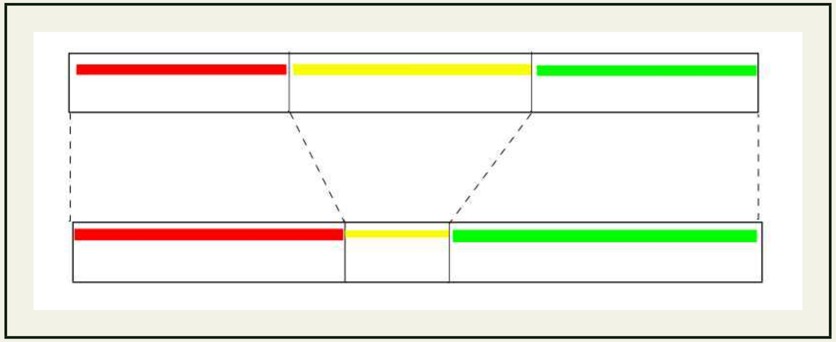


Relating Refinement and Conformance

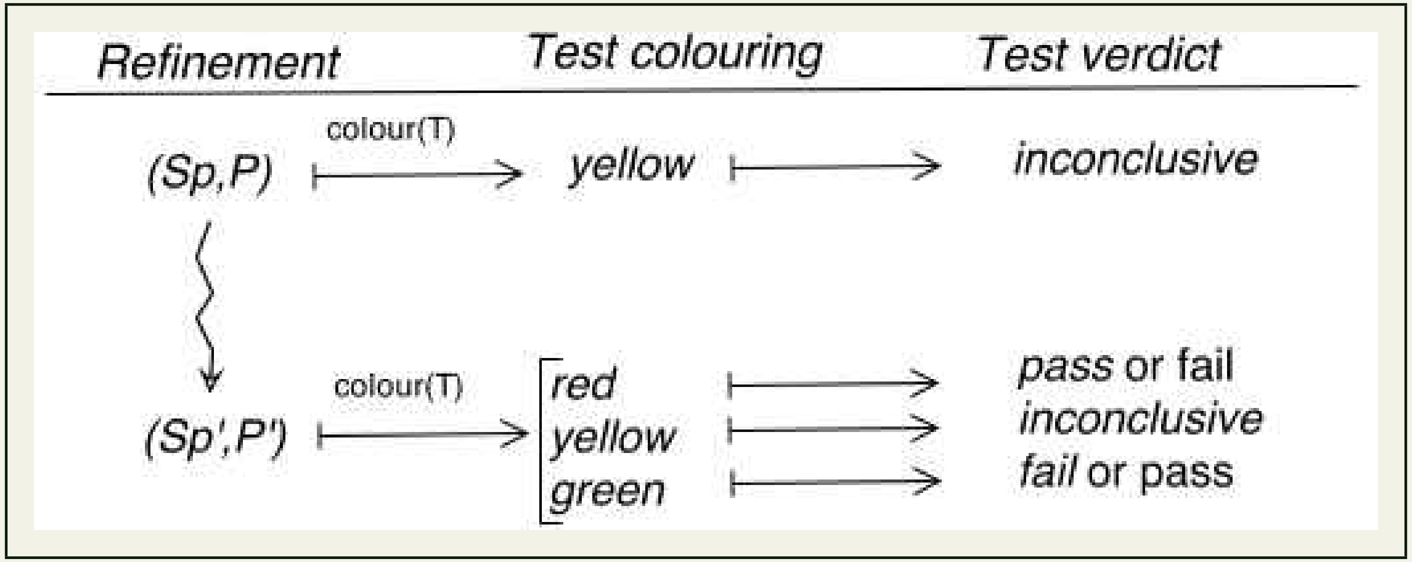
Well-behaved refinement

A refinement relation $\leadsto$ is called well-behaved (w-b) if – given $(Sp, P) \leadsto (Sp', P')$ for consistent Sp and Sp' – for all tests T:

1. $colour(T) = green$ with respect to (Sp, P) implies $colour(T) = green$ with respect to (Sp', P'), and
2. $colour(T) = red$ with respect to (Sp, P) implies $colour(T) = red$ with respect to (Sp', P').



Colouring and verdict under w-b refinement



Well-behaved refinement relations

Refinement relation	Well behaved	
Data refinement	$\surd$	
Process refinement over $\mathcal{T}$	X	
Process refinement over $\mathcal{F}$	$\surd$	$(*)$
Process refinement over $\mathcal{N}$	$\surd$	$(*)$
Process refinement over $\mathcal{R}$	$\surd$	$(*)$

* for divergence-free processes.

Counter example over $\mathcal{T}$

ccspec $\textsc{DoOneA} =$	**ccspec** $\textsc{DoNothing} =$
data **sort** s	**data** **sort** s
op $a : s$	**op** $a : s$
process	**process**
$a \rightarrow Stop$	$Stop$
end	**end**

$$\textsc{DoOneA} \rightsquigarrow_{\mathcal{T}}^{process} \textsc{DoNothing}$$

$colour(a \rightarrow Stop)$ w.r.t. $\textsc{DoOneA}$:	green
$colour(a \rightarrow Stop)$ w.r.t. $\textsc{DoNothing}$:	red

Summary and Future Work

Summary

- Conformance relation for Csp-Casl
  - Separation of expected result and evaluation of a test
  - Three valued expected result: green, red, yellow
  - Three valued verict: pass, fail, inconclusive
- Relation of conformance & refinement
- Tool support / automated test execution possible

Future Work

- Test selection, Test generation
- Study coding rule $(\|...\| : \mathcal{A} \longrightarrow T_\Sigma)$ in the framework of a Csp-Casl institution
- Enhancement (horizontal development)
- Apply Csp-Casl testing to
  - EP2
  - Air plane engines (project with Rolls Royce)