Logics for Specification

Markus Roggenbach, Swansea (Wales)

Bergen, November 2009

CSP & CASL

Modelling Concurrent Systems: CSP

- Established formalism to describe concurrent systems.
- Still research on foundations; applications in industry, e.g. Train Controllers, Avionics, Security Protocols.

Roscoe. The Theory and Practice of Concurrency. Prentice Hall, 1998.

Abdallah, Jones, Sanders (eds). CSP: The First 25 Years. Springer 2005.

Modelling Data: CASL

- CASL = Common Algebraic Specification Language.
- De-facto standard in algebraic specification

Mosses (ed). CASL Reference Manual, Springer 2004.

Bidoit, Mosses. CASL User Manual, Springer 2004.

A C++ template

Template:

```
template <typename T>
T square(T x) { return x * x; }
```

Instantiation:

```
square <int>
```

Checks:

- Extract the signature required for T: “type T, * ”.
- Check that `<int>` offers this signature.

The same in the specification language CASL

Generic specification:

spec $\textsc{MyTemplate}$ [**sort** T **op** $\_\_*\_\_ : T \times T \rightarrow T$] $=$

  op $square : T \times T \rightarrow T$

  $\bullet\ \forall\ x, y : T \bullet square(x, y) = x * y$

Instantiation: $\textsc{MyTemplate}\ [\textsc{Int}]$

Checks:

- $\textsc{Int}$ is a *refinement* of
  sort T **op** $\_\_*\_\_ : T \times T \rightarrow T$

  (in our example: boils down to a check on signatures only)

Underlying Framework: Institutions

Goguen, Burstall. *Institutions: Abstract model theory for specification and programming.* 1992.

Institutions speak about

- Signatures (e.g.: T is a type, $*$ is an operation)
- Models (e.g.: interpretation of type T by set **Z**)
- Formulae (e.g.: $square(x, y) = x * y$
- Satisfaction (e.g.: **Z** $\models x * y = y * x$)

C++ Concepts with threads?

C++ Concepts with threads? – A CSP study

C++ Concepts with threads? – A CSP study

Generic specification:

```
spec MyTemplate
  [call1 -> call2 -> SKIP [] call2 -> call1 -> SKIP [T= P]
= P; call3 -> Skip
```

Instantiation: `MyTemplate [call1 -> SKIP ||| call2 -> SKIP]`

C++ Concepts with threads? – A CSP study

Generic specification:

```
spec MyTemplate
  [call1 -> call2 -> SKIP [] call2 -> call1 -> SKIP [T= P]
= P; call3 -> Skip
```

Instantiation: `MyTemplate [call1 -> SKIP ||| call2 -> SKIP]`

Check:

- `call1 -> SKIP ||| call2 -> SKIP`

  is a *refinement* of

  `call1 -> call2 -> SKIP [] call2 -> call1 -> SKIP`

Questions

- What properties of threads make sense for C++ Concepts?
- How do we formulate properties of threads?
- What is a useful “refinement” on C++ threads?

Questions

- What properties of threads make sense for C++ Concepts?
- How do we formulate properties of threads?
- What is a useful “refinement” on C++ threads?

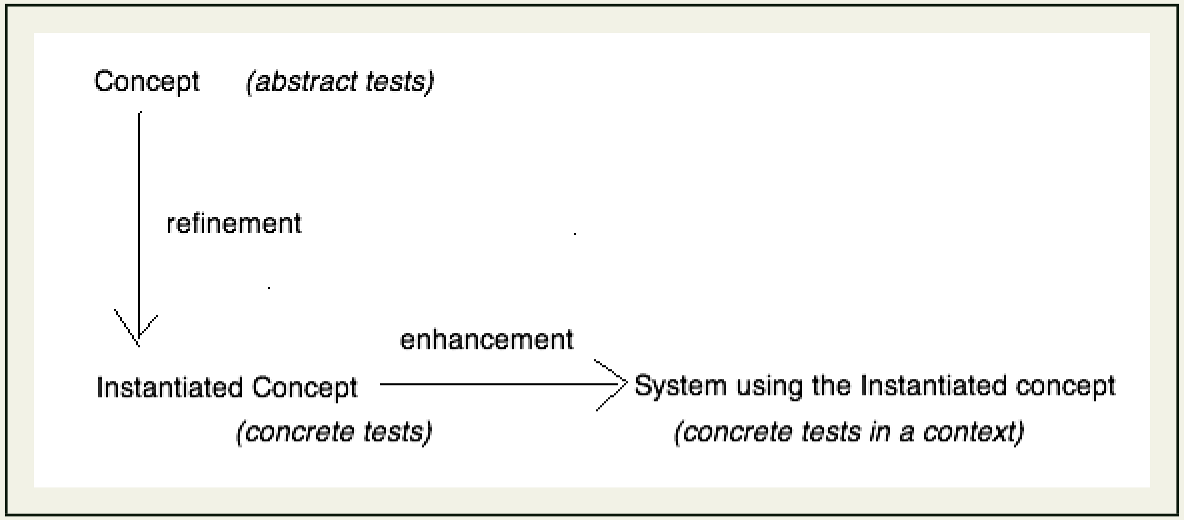
Suggestion:

A “process” algebra of C++ threads

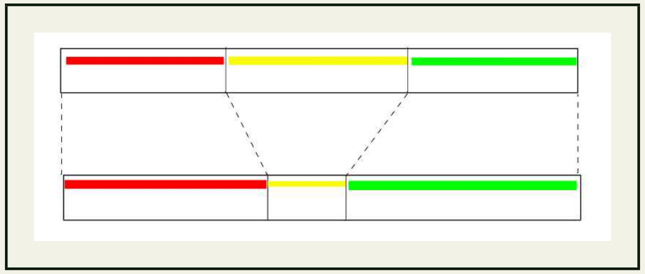
– formulated as an institution.

A testing scenario

Tests on various levels:



Refinement and Tests:



Links to publications

- Mossakowski, Roggenbach: *Structured CSP - A Process Algebra as an Institution*. 2007.
- Mossakowski, Roggenbach: *An institution for processes and data*. 2008.
- Kahsai, Roggenbach, Schlingloff: *Specification-based testing for refinement*. 2007.
- Kahsai, Roggenbach, Schlingloff: *Specification-based testing for Software Product Lines*. 2008.