Logics for Specification

Markus Roggenbach, Swansea (Wales)

Bergen, November 2009
Modelling Concurrent Systems: CSP

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

Modelling Data: CASL

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

A C++ template

Template:

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

Instantiation:

```cpp
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:

\[
\text{spec } \text{MyTemplate} \left[ \text{sort } T \text{ op } \text{___*___} : T \times T \to T \right] = \\
\text{op square : } T \times T \to T \\
\forall x, y : T \bullet \text{square}(x, y) = x \ast y
\]

Instantiation: \text{MyTemplate} \left[ \text{int} \right]

Checks:

\bullet \ \text{INT is a refinement of} \\
\text{sort } T \text{ op } \text{___*___} : T \times T \to T

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


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.: $\text{square}(x, y) = x \times y$)
- Satisfaction (e.g.: $Z \models x \times y = y \times x$)
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