

Reflections on promise theory using prototyping

Kyrre Begnum, kyrre@iu.hio.no
Oslo University College

Motivation

- A proposed theory and notation for promises among hosts
- How can we facilitate discussion and evaluation of this theory?
- How can we investigate/prove properties of the theory?
- How can we identify hidden assumptions that are critical?

Approach

- Build a executable prototype of the promise theory which can generate all possible promise expressions from a finite list of nodes and constraints
- Investigate the divide between all syntactically (logically) correct and all *sensible (semantically correct)* promise expressions
- Look for interesting promise expressions

Maude – A language for executable formal specifications

- A powerful and flexible interpreter
- Based on rewriting theory
 - A term can be reduced to new terms using the rules in the specification.
 - Maude can search amongst all possible outcomes

A single rule: $\text{eq } f(x) = g(x,a) .$

We can reduce the following term in one step:

$f(f(b)) \rightsquigarrow g(f(b),a)$

But also:

$f(f(b)) \rightsquigarrow f(g(b,a))$

Translating promise theory into a formal specification

- A subset of the notation:
 - \oplus, \otimes for concatenation of promises
 - Three basic promises: $n_1 \xrightarrow{C/\pi} n_2$ $n_1 \xrightarrow{C} n_2$ $n_1 \xrightarrow{U(C)} n_2$
- Maude equivalents:

op `__-__->__` : Host Constraint Host \rightarrow Promise .

op `__+__` : SerialPromiseList SerialPromiseList \rightarrow SerialPromiseList .

op `__X__` : PromiseList PromiseList \rightarrow PromiseList .

Semantically meaningful expressions

- We define a set of rewriting rules which build promise expressions based on certain requirements:
 - The same promise cannot be present more than once between the same nodes
 - A regular promise and a conditional promise cannot exist at the same time between the same hosts and same condition
 - A usage promise can only be present if there is a promise that offers the constraint to be used

Some results

- One of the random executions yielded this expression:

“h₁” - “db” -> “h₂” + “h₁” - “web” / “db” -> “h₂”

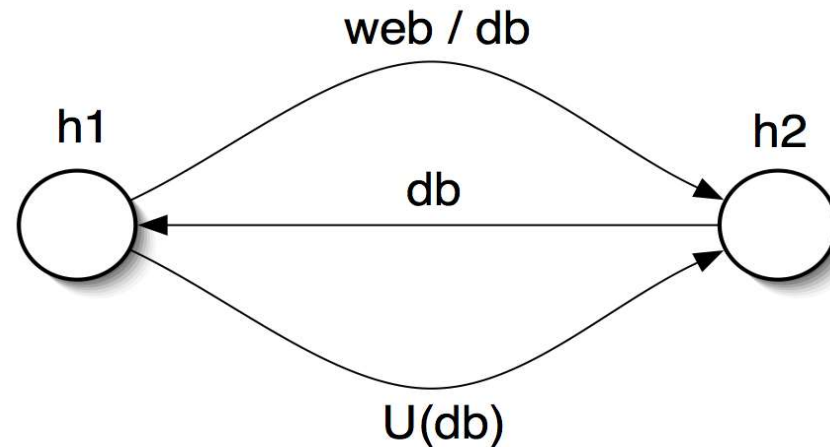
$$h_1 \xrightarrow{db} h_2 \oplus h_1 \xrightarrow{web/db} h_2$$

This does not make much sense, so we need to revise our rules and add the following:

- A host cannot promise a constraint and at the same time indicate the need for the same constraint in a conditional promise

Other Results

- Given two nodes and two constraints, search for all expressions that involve a usage promise and a conditional. (76 different results)
 - Including:



Two interpretations: 1. Negotiation
2. Continuation of promise

Discussion

- Will this approach help in the development and evaluation of promise theory
- Other interesting questions we want to answer or properties we would like to prove?
- Other tools (theorem provers, simulations ... etc)?
- How can promise theory approach a communication protocol?