KreYol

Maximilian Dylla

Chalmers University of Technology

8th KeY Symposium 2009, Speyer

Creol

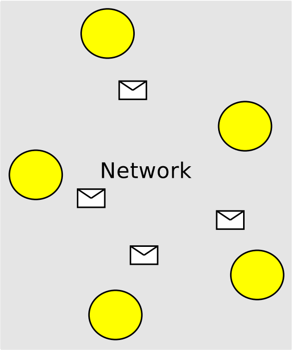
- executable OO modelling language
- verifyable by design
- developed at University of Oslo

1st Level of Parallelism: System

- distributed system of objects
- message passing
- communication via (co)interfaces
- asynchronous communication:

```
label ! obj.meth(x,y);
...;
label ? (y)
```

- only assumption on network: Messages are delivered eventually



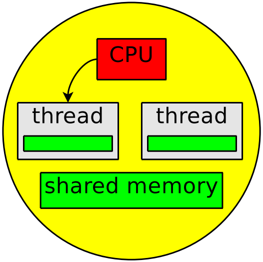
2nd Level of Parallelism: Inside an Object

- thread creation on method invocation
- at most one active thread at the time
- communication via shared variables
- cooperative scheduling
  $\Rightarrow$ release points:

```
release
```

```
await exp
```

- no assumptions on scheduling strategy



Verifying Release Points

Class Invariant

- ensures properties of class attributes
- must hold on a thread switch
  $\Rightarrow$ must hold at release points

$$\frac{\Longrightarrow \mathit{Inv}_{class} \qquad \Longrightarrow U_A(\bar{v}_{attr})(\mathit{Inv}_{class} \rightarrow \langle\omega\rangle\phi)}{\Longrightarrow \langle \texttt{release;}\ \omega\rangle\phi}$$

- no other threads considered

Verifying a Method

- interface contains contract
- class invariant must hold before and after

```
op meth(in a:Int; out b:Bool) == body
```

$$\Longrightarrow Pre_{meth}(a) \land Inv_{class} \rightarrow \langle \texttt{body} \rangle Post_{meth}(b) \land Inv_{class}$$

Verifying Method Calls

History $\mathcal{H}$

- system wide communication log containing:
  - invocation messages
  - completion messages
  - new object messages
- ordered by sending time (the only known order)

Verifying a Class

- local history $\mathcal{H}/\mathit{this}$ as a ghost class attribute
  $\Rightarrow$ class invariant talks about local history
- ensure well formedness of history (similar to reachable state)

Verifying the System

- given $\mathcal{H}/o$ for all classes, show $\exists\mathcal{H}$

Local History

Problem: Arriving messages

- sending time unknown
- sending order unknown
- number unknown

Solution

- model the history with uncertainty
- assert existance of seen messages

Local History: Example

$$
\begin{array}{ccccccc}
h_0 & \leq & h_1 & \leq & h_2 & \leq & h_3 \\
\neg Invoc(h_0, l) & & Invoc(h_1, l) & & Invoc(h_2, l) & & Invoc(h_3, l) \\
\neg Comp(h_0, l) & & \neg Comp(h_1, l) & & & & Comp(h_3, l) \\
\pi; & & \text{l!o.m(x);} & & \ldots; & & \text{l?(y);}
\end{array}
$$

$$
\frac{\begin{array}{l}
\Longrightarrow o \neq null \wedge Wf(h_{pre}) \\
\Longrightarrow \left\{ \mathcal{H} := some\ h.\ \begin{array}{l} Wf(h) \wedge h_{pre} \leq h \wedge Invoc(h_{pre}, l) \\ \wedge \neg Invoc(h, l) \wedge \neg Comp(h, l) \end{array} \right\} (Pre_m(x) \wedge \langle \omega \rangle \phi)
\end{array}}{\Longrightarrow \langle \texttt{l!o.m(x);}\ \omega \rangle \phi}
$$

$$
\frac{\begin{array}{l}
\Longrightarrow l \neq null \wedge Wf(h_{pre}) \wedge Invoc(h_{pre}, l) \\
\Longrightarrow \{\mathcal{H} := some\ h.Wf(h) \wedge h_{pre} \leq h \wedge Comp(h, l)\} U_A(y)(Post(y) \rightarrow \langle \omega \rangle \phi)
\end{array}}{\Longrightarrow \langle \texttt{l?(y);}\ \omega \rangle \phi}
$$

Hybrid History

Problem

- only assertions about existance of messages possible

Solution

- order of sending is known
  $\Rightarrow$ divide into: $\mathcal{H}_{obj}$ and $\mathcal{H}_{send} := \mathcal{H}_{obj}/this_{\rightarrow}$
- keep $\mathcal{H}_{send}$ as a list of messages (between release points)
- drawback: consistency checks

Hybrid History: Example

$$
\begin{array}{ccccc}
ho_0 & \leq & ho_1 & \leq & ho_2 \\
\neg Invoc(ho_0, l) & & Invoc(ho_1, l) & & Invoc(ho_2, l) \\
\neg Comp(ho_0, l) & & \neg Comp(ho_1, l) & & Comp(ho_2, l) \\
\hline
hs & & hs := hs \vdash [this \rightarrow o.m(x)] & & \\
\hline
\pi & & \mathrm{l!o.m(x)} & & \mathrm{l?(y)}
\end{array}
$$

Case study

```
class Buffer
begin
 var cell : Any;
 with Any
  op put(in a :Any) == await cell=null;
                          cell:=a
  op get(out b :Any) == await cell!=null;
                          b:=cell; cell:=null
end
```

- $Pre_{put}(a) := a \neq null$
- $Post_{get}(b) := b \neq null$
- $Inv_C := \left(\begin{array}{l} \neg cell \doteq null \leftrightarrow (\mathcal{H} \vdash [caller \leftarrow this.put()]) \\ \wedge cell \doteq null \leftrightarrow (\mathcal{H} \vdash [caller \leftarrow this.get()]) \end{array} \right) \wedge Prefix(\mathcal{H})$

KeYCreol

Prototype Version

- standart rules working
- rules involving histories need adaptions to specific example
- no support for program loading

Implementation

package	lines of code (without strategy)
key.lang.clang	25k
key.lang.creol	3k
key.java	50k

- data structures created on startup
  $\Rightarrow$ configurable, but slower
- one class for AST
  $\Rightarrow$ e.g. ifThenElse.getCondition() impossible
- pushdown automaton for AST creation

Good luck with HATS!