A Sound Reasoning System for Asynchronous Communication with Shared Futures

Crystal Chang Din, Olaf Owe

Precise Modeling and Analysis group (PMA), University of Oslo, Norway

Outline

1. Introduction

2. Operational semantics vs. Reasoning system

3. Soundness proof

4. Summary
We are here now...

1. Introduction

2. Operational semantics vs. Reasoning system

3. Soundness proof

4. Summary
Introduction

- Concurrent and distributed systems
- Difficulties of reasoning about concurrent and distributed systems
- Necessity of compositional reasoning
- Object orientation vs. distributed systems
- Weakness of the existing paradigms for concurrent and distributed systems, ex. Java RMI.
What do we consider?

- Concurrent objects
- Asynchronous method calls
- Shared futures
Motivation of this work

Focus

- **ABS (Abstract Behavioral Specification Language)**
- Connecting reasoning system with operational semantics

Challenges

- The program logic: local reasoning for each object.
- Operational semantics define the global state transitions.
- Futures are global entities which can be shared between objects.

The contribution in this work:

- To find a sound reasoning system with respect to the operational semantics.
- To enable local reasoning about shared futures.
The syntax of a kernel language

\begin{verbatim}
skip          skip statement
v := e        assignment statement
fr := v!m(e*) asynchronous method call (fr: future identifier)
put e         generating a future unit with return value e
v := get e    query statement (block until e contains value)

await e?      suspend until e contains value
\end{verbatim}
Asynchronous method calls and shared futures

Definition (Communication History/Trace)

The sequence of observable communication events.

A wellformed global history satisfies a certain communication order.
Remark: The objects have disjoint alphabets!
We are here now...

1. Introduction

2. Operational semantics vs. Reasoning system

3. Soundness proof

4. Summary
### Global configuration

#### Definition (Configuration Mappings)

<table>
<thead>
<tr>
<th>Mapping</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$o \mapsto \text{ob}(b, \bar{s})$</td>
<td>an object. Let $b=(a/l)$, $a$: attributes, $l$: local variable.</td>
</tr>
<tr>
<td>$u \mapsto \text{msg}(o, m, \bar{p})$</td>
<td>a message reflecting a method invocation</td>
</tr>
<tr>
<td>$u \mapsto \text{fut}(v)$</td>
<td>a future unit containing a value $v$</td>
</tr>
</tbody>
</table>

For example:

$$o \mapsto \text{ob}(b, \text{skip}; \bar{s}) \xrightarrow{\text{empty}} o \mapsto \text{ob}(b, \bar{s})$$

For disjoint sub-states $g_1$ and $g_2$ (i.e. mappings with disjoint domains), $g_1$ will involve exactly one object, $o$, plus possibly messages and futures.

$$g_1 \xrightarrow{\alpha} g_1' \quad \quad g_1 \parallel g_2 \xrightarrow{\alpha} g_1' \parallel g_2$$
Operational semantics: $g_1 \xrightarrow{\alpha} g'_1$

**assign:**

\[
o \mapsto \text{ob}(b, v := e; s)
\]

\[
o \mapsto \text{ob}(b[v := e], s)
\]

**call:**

\[
o \mapsto \text{ob}(b, fr := v!m(e); s),
\]

\[
b[\text{destiny}] \mapsto \text{msg}(b[v], m, b[\bar{e}])
\]

**method:**

\[
u \mapsto \text{msg}(o, m, d)
\]

\[
o \mapsto \text{ob}((a|l), \text{empty})
\]

\[
o \mapsto \text{ob}((a|(l[p \mapsto d, \text{future} \mapsto u])), s)
\]

**return:**

\[
o \mapsto \text{ob}(b, \text{put} e)
\]

\[
o \mapsto \text{ob}(b, \text{empty})
\]

\[
b[\text{future}] \mapsto \text{fut}(b[e])
\]

**query:**

\[
u \mapsto \text{fut}(d)
\]

\[
o \mapsto \text{ob}(b, v := \text{get} e; s)
\]

\[
\text{if } b[e] = u
\]

\[
o \mapsto \text{ob}(b[v := d], s)
\]

\[
o \mapsto \text{ob}(b[\text{future}], b[e])
\]
Reasoning rules \{P\}s\{Q\}

skip
\{Q\} skip \{Q\}

assign
\{Q^v_e\} v := e \{Q\}

call
\{\forall fr' . Q^{fr,H}_{fr',H} \langle \text{this} \rightarrow o,fr',m,e \rangle\} fr := o!m(e) \{Q\}

query
\{\forall v' . Q^{v,H}_{v',H} \langle \text{this} \leftarrow e,v' \rangle\} v := \text{get} e \{Q\}

method
\{P^y_{y'}\} \mathcal{H} := \mathcal{H} \cdot \langle \rightarrow \text{this}, u, m, x \rangle; \bar{s}; \mathcal{H} := \mathcal{H} \cdot \langle \leftarrow \text{this}, u, e \rangle \{Q^y_{y'}\}
\{P\} (m(x)\{\text{var} y; \bar{s}; \text{put} e\}) \{Q\}
We are here now...

1. Introduction

2. Operational semantics vs. Reasoning system

3. Soundness proof

4. Summary
Soundness

We say that a reasoning system is sound if any provable property is valid, i.e.,

\[ \vdash \{P\} s \{Q\} \implies \models \{P\} s \{Q\} \]

To prove that a reasoning system is sound, we need to show that all axioms of the system are valid and that all inference rules are sound, in the sense that they preserve validity.

\[ \models \{P\} s \{Q\} \triangleq ? \]
Operational Semantics and Reasoning Rules for the query statement

The reasoning rule $\{P\}s\{Q\}$ of query statement is given by

$$\{\forall v'. Q_{v',\mathcal{H}}^{v',\mathcal{H}} \} v := \text{get fr} \{Q\}$$

The transition rule $g \xrightarrow{\alpha} g'$ of query statement is given by

$$u \mapsto \text{fut}(d)$$
$$o \mapsto (b, v := \text{get e}; \bar{s})$$
$$\text{if } b[e] = u$$

$$\langle o\leftarrow, u, d \rangle \xrightarrow{\alpha} u \mapsto \text{fut}(d)$$
$$o \mapsto (b[v := d], \bar{s})$$
Including history in the global state transitions

The given operational semantics does not explicitly include a history. We may include the history $H$ explicitly by transforming each transition rule

$$g_1 \xrightarrow{\alpha} g_2$$

to

$$g_1 + [H \mapsto h] \rightarrow g_2 + [H \mapsto h \cdot \alpha]$$

where $h$ is the history of the prestate and $h \cdot \alpha$ the history of the poststate, letting $h \cdot \alpha$ denote $h$ when $\alpha$ is empty.
From global state to local state

\[ g_1 \xrightarrow{o:s} g_2 \] which expresses a transition from \( g_1 \) to \( g_2 \) (including \( \mathcal{H} \)) due to an execution step of statement \( s \) by object \( o \).

\[ o \leftrightarrow \text{ob}(\textbf{State}, \textbf{Code}) \] is in the global configuration.

\[ \text{loc}(g, o) \triangleq g[o].\textbf{State} + [\mathcal{H} \mapsto g[\mathcal{H}]/o] \]
### Definition of validity

**Definition (Validity of pre/post conditions over execution steps)**

\[
P \xrightarrow{o'}_o Q \triangleq \forall g, g'. g \xrightarrow{o'} g' \land \text{loc}(g, o)[P] \Rightarrow \text{loc}(g', o)[Q]
\]

**Lemma (Non-interference)**

\[
o \neq o' \Rightarrow P \xrightarrow{o'}_o P
\]

**Definition (Validity of Hoare triples)**

\[
\models \{P\} s \{Q\} \triangleq \forall o. P \xrightarrow{o\cdot s}_o Q
\]
The operational semantics of query statement is given by

\[ u \mapsto \text{fut}(d) \]
\[ o \mapsto (b, v := \text{get} \ e; \bar{s}) \]
\[ \text{if } b[e] = u \]
\[ o \mapsto (b[v := d], \bar{s}) \]

defining a prestate \( g \) and a poststate \( g' \) containing \( \mathcal{H} \).

Now \( \models \{ P \} \ v := \text{get} \ e \ \{ Q \} \) gives (for all \( o, d, \) and \( h \))

\[ \text{loc}(g, o)[P] \Rightarrow \text{loc}(g', o)[Q] \]

assuming \( b[e] = u \), which reduces to

\[ P \Rightarrow \forall d \cdot Q_{d, \mathcal{H}, \langle \text{this}\leftarrow,e,d \rangle}^{v, \mathcal{H}} \]

The given Hoare rule for query \( \{ \forall v'. \ Q_{v', \mathcal{H}, \langle \text{this}\leftarrow,e,v' \rangle}^{v, \mathcal{H}} \} \ v := \text{get} \ e \ \{ Q \} \)
expresses the weakest precondition, and is therefore sound and complete with respect to the semantics of query.
Compositional reasoning

The history invariant $I_S(\mathcal{H})$ for a system $S$ is then given by combining the history invariants of the composed objects:

$$I_S(\mathcal{H}) \triangleq wf(\mathcal{H}) \bigwedge_{(o:C(\overline{e})) \in S} \exists \overline{w} . I_{C_o, \overline{e}, \overline{h}}$$
We are here now…

1. Introduction

2. Operational semantics vs. Reasoning system

3. Soundness proof

4. Summary
Conclusion

- A sound compositional reasoning system for concurrent objects and futures, which is proved with respect to the official operational semantics
- A sound reasoning system for the ABS language
- Futures are handled by events appearing in the histories
- Classes can be specified independently from the surroundings
- Modularity is achieved
- Global specification is realized by the composition rule
- ABS interpreter
Ongoing and Future Work

- Completeness
- The reasoning system is currently being implemented within the KeY framework at Technical University Darmstadt.
- Semi-automatic verification of large case studies using KeY
  - An elevator system has been implemented, simulated and tested according to the generated histories.
Thank You