Iris: Higher-Order Concurrent Separation Logic

Lecture 7: Later Modality

Lars Birkedal

Aarhus University, Denmark

November 14, 2017
Overview

Earlier:

- Operational Semantics of $\lambda_{\text{ref,conc}}$
  - $e, (h, e) \leadsto (h, e')$, and $(h, E) \rightarrow (h', E')$
- Basic Logic of Resources
  - $l \hookrightarrow v, P \ast Q, P \rightarrow Q, \Gamma \vdash P \mid Q$
- Basic Separation Logic
  - $\{P\} e \{v.Q\} : \text{Prop}, \text{isList } l \text{ xs, ADTs, foldr}$

Today:

- Later Modality: $\triangleright$
- Necessary for working with invariants (defined later in the course)
- Key Points:
  - Löb rule: $(\triangleright P \Rightarrow P) \Rightarrow P$
  - Guarded recursively defined predicates: $\mu r. P$
Later Modality

- Recall the recursion rule:

\[
\begin{align*}
\text{HT-REC} \\
\Gamma, f : \text{Val} \mid S \land \forall y. \forall v. \{P\} f v \{u.Q\} \vdash \forall y. \forall v. \{P\} e[v/x] \{u.Q\} \\
\Gamma \mid S \vdash \forall y. \forall v. \{P\} (\text{rec } f(x) = e) v \{u.Q\}
\end{align*}
\]

- This rule involves a kind of recursive reasoning.

- We mentioned earlier that this rule is *sound* because function application involves reduction steps, i.e., we will only use the recursive assumption after some reduction steps have taken place.

- Later in the course, when discussing *invariants*, we will want to have other forms of recursive reasoning, where the recursive reasoning steps are not directly tied to corresponding reduction steps.

- But soundness will still hinge on some reduction steps taking place.

- Thus to ensure soundness, we will need some way to express, in the logic, that a property is only supposed to hold *later*, after a reduction step has taken place.

- This is what the later modality \(\triangleright\) achieves: *intuitively*, \(\triangleright P\) holds if \(P\) holds after a reduction step has been taken.
Plan for today

Today

▷ Rules for reasoning about $\triangleright$, including strengthening of earlier Hoare rules.
▷ Example
  ▷ Specification and proof of a fixed point combinator.
  ▷ Proof relies on $\triangleright$.
  ▷ The example is perhaps somewhat contrived — chosen to illustrate expressiveness without being too long.

Later on

▷ the rules we describe today will be important later on, especially when reasoning about invariants.
Löb Rule

- Typing for ▷:
  \[
  \Gamma \vdash P : \text{Prop} \\
  \Gamma \vdash \triangleright P : \text{Prop}
  \]

- Löb Rule:
  \[
  \text{Löb} \\
  S \land \triangleright P \vdash P \\
  S \vdash P
  \]

- Akin to a coinduction proof rule: to show \( P \), it suffices to show \( P \) under the assumption that \( P \) holds later.
 Aside: semantics of propositions

- As suggested by the above, the meaning of Iris proposition is not just a set of resources.
- In more detail, an Iris proposition $P$ is$^1$ a set of pairs $(k, r)$, with $k$ a natural number and $r$ a resource.
- Think of $k$ as a step-index, a natural number which expresses for how many reduction steps we know that $r$ is in $P$.
- If $(k, r) \in P$ and $m \leq k$, then also $(m, r) \in P$.
- The step-indeces are used to interpret $\triangleright$:

$$\triangleright P = \{(m + 1, r) \mid (m, r) \in P\} \cup \{(0, r) \mid r \in R\}$$

- “later” means that the index number is smaller (there are fewer reduction steps left, after we have taken some reduction steps).
- The Löb Rule is proved sound by induction on these step-indeces.

$^1$Not really, but closer to being…
Laws for Later Modality

**Later-Mono**

\[
\frac{Q \vdash P}{\triangleright Q \vdash \triangleright P}
\]

**Later-weak**

\[
\frac{Q \vdash P}{\triangleright Q \vdash \triangleright P}
\]

**Löb**

\[
\frac{Q \land \triangleright P \vdash P}{Q \vdash P}
\]
### Laws for Later Modality

<table>
<thead>
<tr>
<th>Later-Conj</th>
<th>Later-Disj</th>
<th>Later-All</th>
<th>Later-Sep</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R \vdash \triangleright (P \land Q)$</td>
<td>$R \vdash \triangleright (P \lor Q)$</td>
<td>$Q \vdash \forall x. \triangleright P$</td>
<td>$R \vdash \triangleright (P \ast Q)$</td>
</tr>
<tr>
<td>$R \vdash \triangleright P \land \triangleright Q$</td>
<td>$R \vdash \triangleright P \lor \triangleright Q$</td>
<td>$Q \vdash \forall x. \triangleright P$</td>
<td>$R \vdash \triangleright (P \ast Q)$</td>
</tr>
</tbody>
</table>

**Later-sep**

**Later-conj**

```
\triangleright \exists \quad \tau \text{ is inhabited} \\
Q \vdash \exists x : \tau. P \\
Q \vdash \exists x : \tau. \triangleright P
```

**Later-disj**

```
\triangleright \exists \\
Q \vdash \exists x. \triangleright P
```

**Later-all**

```
\triangleright \exists \\
Q \vdash \exists x. \triangleright P
```

**Later-sep**

```
\triangleright \exists \\
Q \vdash \exists x. \triangleright P
```
Stronger rules for Hoare triples

**HT-BETA**
\[
S \vdash \{P\} \ e[v/x] \ {u.Q}
\]
\[
S \vdash \{\triangleright P\} \ (\lambda x.e)\ v \ {u.Q}
\]

**HT-REC**
\[
Q \vdash \{P\} \ e[(\text{rec } f(x) = e)/f, \ v/x] \ {\Phi}
\]
\[
Q \vdash \{\triangleright P\} \ (\text{rec } f(x) = e)\ v \ {\Phi}
\]

**HT-LOAD**
\[
S \vdash \{\triangleright \ell \leftrightarrow u\} \ !\ell \ {v.\ v = u \land \ell \leftrightarrow u}
\]

**HT-STORE**
\[
S \vdash \{\triangleright \exists u. \ell \leftrightarrow u\} \ \ell \leftarrow w \ {v.\ v = () \land \ell \leftrightarrow w}
\]

**HT-MATCH**
\[
S \vdash \{P\} \ e_i [u/x_i] \ {v.Q}
\]
\[
S \vdash \{\triangleright P\} \ \text{match} \ inj_i \ u \ \text{with} \ inj_1 \ x_1 \Rightarrow e_1 \ | \ inj_2 \ x_2 \Rightarrow e_2 \ \text{end} \ {v.Q}
\]
Remark on soundness

Why are the rules $H_T$-load and $H_T$-store sound?

- Difficult to explain intuitively.
- Relies on $\leftrightarrow$ being a timeless predicate together with the definition of Hoare triples (the fact that weakest precondition is “closed wrt. the fancy update modality”).
Stronger derived Hoare triples

\textbf{HT-LET}
\[ S \vdash \{P\} e_1 \{x.\triangleright Q\} \quad S \vdash \forall v. \{Q[v/x]\} e_2 [v/x] \{u.R\} \]
\[ S \vdash \{P\} \text{let } x = e_1 \text{ in } e_2 \{u.R\} \]

\textbf{HT-LET-DET}
\[ S \vdash \{P\} e_1 \{x.\triangleright (x = v) \land \triangleright Q\} \quad S \vdash \{Q[v/x]\} e_2 [v/x] \{u.R\} \]
\[ S \vdash \{P\} \text{let } x = e_1 \text{ in } e_2 \{u.R\} \]

\textbf{HT-SEQ}
\[ S \vdash \{P\} e_1 \{.\triangleright Q\} \quad S \vdash \{R\} e_2 \{u.R\} \]
\[ S \vdash \{P\} e_1; e_2 \{u.R\} \]

\textbf{HT-IF}
\[ S \vdash \{P \ast v = \text{true}\} e_2 \{u.Q\} \quad S \vdash \{P \ast v = \text{false}\} e_3 \{u.Q\} \]
\[ S \vdash \{\triangleright P\} \text{ if } v \text{ then } e_2 \text{ else } e_3 \{u.Q\} \]
Guarded recursively defined predicates

- We extend terms of the logic with

\[ t ::= \cdots | \mu x : \tau. t \]

with the side-condition that the recursive occurrences must be \textit{guarded}: in \( \mu x. t \), the variable \( x \) can only appear under the later \( \triangleright \) modality.

- Fixed-point property expressed by the following rule:

\[ \text{Mu-fixed} \]

\[ Q \vdash \mu x : \tau. t =_\tau t[\mu x : \tau. t/x] \]
Guarded recursively defined predicates

- Example: using a stream (infinite list) as model of linked list:

  \[\mu\text{isStream} : \text{Val} \to \text{stream Val} \to \text{Prop}. \quad \lambda l. \lambda xs : \text{stream Val}.
  \begin{align*}
  (xs = [] \land l = \text{inj}_1()) \lor \\
  (\exists x, xs'. xs = x : xs' \land \exists hd, l'. l = \text{inj}_2(hd) \ast hd \hookrightarrow (x, l') \ast \triangleright (\text{isStream} l'xs))
  \end{align*}\]

- Note that \(xs\) is a stream (infinite list). Therefore we cannot define the predicate by induction on \(xs\).

- Above, the recursion variable occurs positively.

- In Iris, Hoare triples are defined in terms of weakest-preconditions, which are defined by means of a guarded recursive definition for a positive definition to give a partial correctness interpretation.

- One can also define mixed-variance recursive predicates.
Guarded recursively defined predicates

- Mixed-variance guarded recursive predicates are useful for
  - Interpreting recursive types in a typed programming language by Iris predicates / relations (see ipm-paper).
  - Defining models of untyped / unityped languages (e.g., for object capabilities).
  - Specifying and reasoning about libraries that can call themself recursively, e.g., an event loop library (see iCap-paper).
    - M.Sc. Project idea: Formalize event loop library in Iris in Coq, if ambitious, consider library for asynchronous IO.
Example: Fixed-point combinator $\Theta_F$

- Given a value $F$, the call-by-value Turing fixed-point combinator $\Theta_F$ is:

\[
\Omega_F = \lambda r. F(\lambda x. rrx)
\]
\[
\Theta_F = \Omega_F \Omega_F
\]

- For any values $F$ and $v$,

\[
\Theta_F v \leadsto F(\lambda x. \Theta_F x) v
\]

- Thus, if $F = \lambda f x. e$ then one should think of $\Theta_F$ as $rec \ f(x) = e$. 
Proof Rule for $\Theta_F$

- Now we wish to derive proof rule for $\Theta_F$, similar to the recursion rule.

\[
\frac{\text{HT-TURING-FP}}{\Gamma \mid S \wedge \forall v. \{P\} \Theta_F v \{u.Q\} \vdash \forall v. \{P\} F(\lambda x. \Theta_F x) v \{u.Q\}}
\]

- We will use the Löb rule.

- We will also use that if $P$ is persistent, then $\triangleright P$ is persistent, which means that it can be moved in and out of preconditions.
Proof

- We proceed by the Löb rule and hence we assume

\[ \triangledown \forall v. \{ P \} \Theta_F v \{ u. Q \} \]  

(\*)

- and we are to show

\[ \forall v. \{ P \} \Theta_F v \{ u. Q \} \]

- Let \( v \) be a value.

- By Later-weak and the rule of consequence SFTS

\[ \{ \triangledown P \} \Theta_F v \{ u. Q \} \]
Proof

- Since Hoare triples are persistent, we can move our assumption $(\ast)$ into the precondition, and thus SFTS:

\[
\{\triangleright (\forall v. \{ P \} \Theta_F v \{ u.Q \} \land P) \} \Theta_F v \{ u.Q \}
\]

- By the bind rule and the stronger rule HT-BETA introduced above SFTS

\[
\{\forall v. \{ P \} \Theta_F v \{ u.Q \} \land P \} F(\lambda x. \Theta_F x) v \{ u.Q \}
\]

- We again use persistence and move the triple $\forall v. \{ P \} \Theta_F v \{ u.Q \}$ into the context and then SFTS

\[
\{ P \} F(\lambda x. \Theta_F x) v \{ u.Q \}
\]

- But this is exactly the premise of the rule HT-TURING-FP, and thus the proof is concluded.