Block structured local state

- Construct for introducing locally scoped locations.
- Involves dynamic creation of fresh locations.
- Essential for properly dealing with recursive procedures.

```java
public class QuickSort {
    public static void sort(Vector array) {
        sort(array, 0, array.size() - 1);
    }
    public static void sort(Vector array, int start, int end) {
        int p;
        if (end > start) {
            p = partition(array, start, end);
            sort(array, start, p - 1);
            sort(array, p + 1, end);
        }
    }
}
```

The language LC<sub>loc</sub>: LC + block structure

```
P ::= C | E | B
C ::= skip | ℓ := E | C ; C | if B then C1 else C2
     begin loc ℓ := E ; C end | while B do C
E ::= n | !ℓ | E iop E
B ::= b | E bop E
```

Evaluation rules for blocks

```
(E, s) ⇓ ⟨n, s′⟩ \quad (C[ℓ′/ℓ], s′[ℓ′ ↦→ n]) \cup (skip, s′′[ℓ′ ↦→ n′])

{begin loc ℓ := E ; C end, s} ⇓ ⟨skip, s′′[ℓ′ ↦→ n′]⟩
```

Provided
- ℓ′ ∉ dom(s′) ∪ dom(s′′)
- ℓ′ does not occur in C
- C[ℓ′/ℓ] is the LC<sub>loc</sub> command obtained from C by replacing all occurrences of ℓ with ℓ′
Intuition for the block rule

In order to evaluate \( ⟨\text{begin loc } \ell \leftarrow E; C \text{ end } , s⟩ \)
- evaluate \( ⟨E,s⟩ \) to \( ⟨n,s'⟩ \)
- replace \( \ell \) everywhere in \( C \) by a new location \( \ell' \) satisfying
  - \( \ell' \) hasn’t been set to any value yet at \( s' \);
  - extend \( s' \) by setting \( \ell' \mapsto n \), yielding \( (C, s'[\ell' \mapsto n]) \) for some state \( s'' \);
- let \( (\text{skip}, s'') \) be the result of evaluating \( ⟨\text{begin loc } \ell := E; C \text{ end } , s⟩ \)

Some questions
- What does \( ⟨\ell := 1; \text{begin loc } \ell :=!\ell + 1; \text{skip end } , s⟩ \) evaluate to?
- Why not \( \text{begin loc } \ell C \text{ end } \) as syntax?
- Can we devise a transition semantics for \( \mathcal{LC}_{loc} \)?
- What do the following two commands evaluate to starting from the state \( \{\ell \mapsto 1, \ell' \mapsto 0\} \)?
  1. \( \text{begin loc } \ell := !\ell + 1; \ell' := !\ell + 1 \text{ end } \)
  2. \( \text{begin loc } \ell := !\ell'; \text{begin loc } \ell'' := !\ell + !\ell'; \ell := !\ell' + !\ell'' \text{ end } \)

Reasons for a formal semantics of a programming language
- form basis for reasoning about program properties and program specifications
- form basis for code optimisation
  - when two program phrases have the same meaning
  - Equations \( P = P' \) and \( P' \) have the same meaning. Allows equational reasoning
  - Especially interested when \( = \) is a congruence
  - if \( P = P' \) then \( C[P] = C[P'] \) where \( C[\ ] \) a program phrase with
  a “hole”

Basic properties of equality

<table>
<thead>
<tr>
<th>Property</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reflexivity</td>
<td>( P = P )</td>
</tr>
<tr>
<td>Symmetry</td>
<td>( P = P' )</td>
</tr>
<tr>
<td>Transitivity</td>
<td>( P = P' \Rightarrow P'' = P'' )</td>
</tr>
<tr>
<td>Congruence</td>
<td>( C[P] = C[P'] )</td>
</tr>
</tbody>
</table>

where \( C[P] \) is a phrase containing an occurrence of \( P \) and \( C[P'] \) is
the same phrase with that occurrence replaced by \( P' \).
Definition of semantic equivalence of LC phrases

Two phrases of the same type are semantically equivalent

\[ P_1 \sim P_2 \]

if and only if for all states \( s \) and all terminal configurations \( (V, s') \)

\[ (P_2, s) \downarrow (V, s') \iff (P_2, s) \downarrow (V, s'). \]

Examples of semantically equivalent LC commands

\[ \begin{align*}
&\text{\qquad \qquad \qquad \qquad \quad \text{\textbf{C}}; \text{\textbf{skip}} \sim \text{\textbf{C}} \sim \text{\textbf{skip}}; \text{\textbf{C}}}
\end{align*} \]

\[ \begin{align*}
&\text{\qquad \qquad \qquad \qquad \quad \text{\textbf{(C ; C') ; C''}} \sim \text{\textbf{C ; (C' ; C'')}}}
\end{align*} \]

\[ \begin{align*}
&\text{\qquad \qquad \qquad \qquad \quad \text{\textbf{if true then}} \text{\textbf{C}} \text{\textbf{else}} \text{\textbf{C'}} \sim \text{\textbf{C}} \sim \text{\textbf{if false then}} \text{\textbf{C'}} \text{\textbf{else}} \text{\textbf{C''}}}
\end{align*} \]

\[ \begin{align*}
&\text{\qquad \qquad \qquad \qquad \quad \text{\textbf{while}} \text{\textbf{B}} \text{\textbf{do}} \text{\textbf{C}} \sim \text{\textbf{if}} \text{\textbf{B}} \text{\textbf{then}} \text{\textbf{C}} \text{\textbf{else}} \text{\textbf{(while B do C)}} \text{\textbf{else}} \text{\textbf{skip}}}
\end{align*} \]

\[ \begin{align*}
&\text{\qquad \qquad \qquad \qquad \quad \text{\textbf{ℓ} := n ; ℓ' := n'}}
\end{align*} \]

\[ \begin{align*}
&\text{\qquad \qquad \qquad \qquad \quad \{ \ell' := n'; \ell := n \text{ if } \ell \neq \ell'}
\end{align*} \]

\[ \begin{align*}
&\text{\qquad \qquad \qquad \qquad \quad \ell := n' \text{ if } \ell = \ell'}
\end{align*} \]

Exercise

\[ \begin{align*}
&\text{\qquad \qquad \qquad \qquad \quad \text{Is the following true?}}
\end{align*} \]

\[ \begin{align*}
&\text{\qquad \qquad \qquad \qquad \quad (\text{if } B \text{ then } C \text{ else } C'); C'' \sim (\text{if } B \text{ then } C; C'' \text{ else } C; C'')}
\end{align*} \]

\[ \begin{align*}
&\text{\qquad \qquad \qquad \qquad \quad \text{What about?}}
\end{align*} \]

\[ \begin{align*}
&\text{\qquad \qquad \qquad \qquad \quad \text{\textbf{if}} \text{\textbf{B}} \text{\textbf{ then}} \text{\textbf{C}}; \text{\textbf{C'}} \text{\textbf{ else}} \text{\textbf{C''}} \sim \text{\textbf{if}} \text{\textbf{B}} \text{\textbf{ then}} \text{\textbf{C'; C else C''}}}
\end{align*} \]

Properties of \( \sim = \)

\[ \begin{align*}
&\text{\qquad \qquad \qquad \qquad \quad \text{\textbf{Does}} \sim = \text{\textbf{obey}} \text{\textbf{equality properties?}}} \quad \text{\textbf{Reflexivity, Symmetry, Transitivity ?}}
\end{align*} \]

\[ \begin{align*}
&\text{\qquad \qquad \qquad \qquad \quad \text{\textbf{Congruence? How to prove this?}}}
\end{align*} \]
Semantic equivalence is a congruence: the while case

Need to prove

if \( C_1 \sim= C_2 \),
then while \( B \) do \( C_1 \sim= \) while \( B \) do \( C_2 \)

By the equivalence of the LC transition and evaluation semantics, it suffices to prove

if \( C_1 \sim= C_2 \),
then for all states \( s, s' \),
\( \langle \text{while } B \text{ do } C_1, s \rangle \rightarrow^* \langle \text{skip}, s' \rangle \)
if and only if
\( \langle \text{while } B \text{ do } C_2, s \rangle \rightarrow^* \langle \text{skip}, s' \rangle \)

Exercise: compiler optimization

▶ Does the following hold?
\[ i := E; j := E \sim= i := E; j :=!i \]

▶ Are the following pair equivalent?
while \( (1 \leq i \text{ and } i \leq n) \) do
\[ \begin{align*}
    s &:= s + (i \times E), \\
i &:= i + 1
\end{align*} \]

begin loc \( k := E; \)
while \( (1 \leq i \text{ and } i \leq n) \) do
\[ \begin{align*}
    s &:= s + (i \times (k)), \\
i &:= i + 1
\end{align*} \]
end

Reasons for a formal semantics of a programming language

▶ form basis for code optimization
▶ form basis for reasoning about program properties and program specifications
An application of semantics to correctness

- The idea of partial correctness
- \{ formula \} P \{ formula’ \} means if formula is true before execution of P and P terminates then formula’ is true after execution
- We can make this precise as follows
- if formula is true in state \( s \) and \( \langle P, s \rangle \Downarrow \langle \text{skip}, s' \rangle \), then formula’ is true in state \( s' \)
- Example: \{!x = 3\} x :=!x + 5; x :=!x∗!x{!x = 64}
- More than this: use the semantics to justify proof system

Proof rules of Hoare Logic

\[
\{ p[E/x] \} x := E \{ p \}
\]

\( p[E/x] \) is \( p \) when all (free) occurrences of \!x are replaced by \( E \)

\[
\frac{(p) C_1 (q) \quad (q) C_2 (r)}{(p) C_1; C_2 (r)}
\]

\[
\frac{(B \land p) C_1 (q) \quad (\neg B \land p) C_2 (q)}{(p) \text{ if } B \text{ then } C_1 \text{ else } C_2 (q)}
\]

\[
\frac{(B \land p) C (p)}{(p) \text{ while } B \text{ do } C \{ p \land \neg B \}}
\]

\[
\frac{p \Rightarrow p'}{(p') C \{ q' \}}
\]

And \( p \Rightarrow p' \) means \( p \) implies \( p' \)

Hoare Logic continued

- Example of axiom (+ final rule)
  \{x = 3\} x := x + 5 \{x = 8\}
- Because \( \{x = 3\} [x := x + 5] \{x = 8\} \) is \( \{x = 8\} \)
- Example: use LC evaluation semantics to show soundness of the logic. Show that the axiom is true and that the rules preserve truth.
- That is prove: if \( p[E/x] \) is true at \( s \) and \( \{x := E, s\} \Downarrow \langle \text{skip}, s' \rangle \) then \( p \) is true at state \( s' \)
- And, for example, for the while rule:
- Assume for any \( s \) if \( B \land p \) is true at \( s \) and \( \langle C, s \rangle \Downarrow \langle \text{skip}, s' \rangle \) then \( p \) is true at \( s' \).
- Then show: for any \( s \), if \( p \) is true at \( s \) and \( \langle \text{while } B \text{ do } C, s \rangle \Downarrow \langle \text{skip}, s' \rangle \) then \( p \land \neg B \) is true at \( s' \).
- And similarly for the other rules