CS181b 2008/09 Lecture 1

Instructor:
   Alain Martin <alain@async> 254 Jorg.

Teaching Assistant:
   Chris Moore <cc@async> 160B Jorg.

Course Secretary:
   Diane Goodefellow <diane@cs> 266 Jorg.

System Administrators:
   David LeBlanc <help@cs> 272 Jorg.

Physical Resources

Class will meet in Jorgensen 287 Mondays and Wednesdays from 1:00 to 2:00 P.M.; exceptions will be announced well in advance.

You will be using computers in the UNIX lab on the first floor of Jorgensen.
Course Requirements

This term consists of lectures, homework, and another final project. The homework will consist of a mixture of short answer questions and laboratory work.

Collaboration Policy

Students are expected to do their own work, but discussion of problem sets among classmates is allowed.

Late Policy

Unless you receive prior permission or have a verified illness, late homeworks will be penalized by weighting errors on those problems that were completed past the due date by the number of days past the due date plus an extra 5% per day.
Communicating Hardware Processes (CHP)

Variables
  booleans, records, arrays, integers, symbols

Assignments
  \( x := \text{expr} \)
  \( x \uparrow, x \downarrow \)
  \( x, y, z := e1, e2, e3 \)

Composition Operators
  Sequential \( S1; S2 \)
  Parallel \( S1 \parallel S2 \)
  Bullet \( S1 \bullet S2 \)

Control Structures
  Selection
    \[
    [B_1 \rightarrow S_1 \parallel B_2 \rightarrow S_2 \parallel \ldots]
    \]
    \[
    [B_1 \rightarrow S_1 \mid B_2 \rightarrow S_2 \mid \ldots]
    \]
  Repetition
    \[
    *[B_1 \rightarrow S_1 \parallel B_2 \rightarrow S_2 \parallel \ldots]
    \]
    \[
    *[B_1 \rightarrow S_1 \mid B_2 \rightarrow S_2 \mid \ldots]
    \]

Procedures and functions

Replication (syntax)

Processes, ports and channels
  \text{input}(X? a), \text{output}(Y! \text{expr}), \text{probe}(\overline{X})
Data Types

Three *generic* types: boolean, integer, symbol

*Specific* types: generic types restricted to a finite range.

\[
\text{type byte} = \{0\ldots255\}
\]

\[
\text{type color} = \{\text{red,white,blue}\}
\]

Arrays:

\[
\text{array} [0..5] \text{ of color}
\]

\[
\text{array} [0..5,1..10] \text{ of byte}
\]

\[
\text{var a: array} [0..5] \text{ of byte}
\]

\[
\text{var a: array} [0..5] \text{ of } \{\text{red,green,blue}\}
\]

Records:

\[
\text{var inst: record op,par1,par2:byte}
\]

Constants:

\[
\text{const n: integer = 16}
\]
Assignments

\[ x := \text{expr} \]

Only action that changes the state of the computation.
\[ x := \text{true}, x\uparrow, x\downarrow, x := y \lor z \]
\[ y := y + 1 \]

Operational Semantics:

i. Evaluate the right-hand-side expression.

ii. Assign the result to the left-hand-side variable.

Multiple Assignments:
\[ x_1, x_2, x_3 := expr_1, expr_2, expr_3 \]

i. Evaluate all right-hand-side expressions.

ii. Assign the results to the corresponding left-hand-side variables.
\[ x, y := x + 1, x \]

\textbf{different} from the concurrent assignment:
\[ x := x + 1 \parallel y := x \]
Assertions

\{P\} S \{Q\}

S: statement
P: precondition
Q: postcondition

P and Q are comments on the state of the computation.

The strongest assertion in a state of the computation is the value of all defined variables in the state.

Assertions may be any predicate that holds in the state.
Composition

Sequential composition
\( S_1; S_2 \)

Parallel composition
\( S_1 \parallel S_2 \) or \( S_1, S_2 \)

Bullet operator
\( S_1 \bullet S_2 \)
Control Structures - Selection

Deterministic Selection:

\[ G_1 \rightarrow S_1 \land \ldots \land G_n \rightarrow S_n \]

Each \( G_i \) is called a guard.
At any time, at most one guard holds.
Executes the \( S_i \) for which \( G_i \) holds. If no guards hold, execution is suspended until a guard evaluates to \text{true}.

Non-deterministic Selection:

\[ G_1 \rightarrow S_1 \mid \ldots \mid G_n \rightarrow S_n \]

Same as the deterministic selection except multiple guards may be \text{true} at the same time.
If multiple guards hold, an arbitrary true guard is selected.
Control Structures - Repetition

Deterministic Repetition:

\[ * [\begin{array}{c}
G_1 
\rightarrow
S_1 \\
\vdots \\
G_n 
\rightarrow 
S_n
\end{array}] \]

At any time, at most one guard holds. Repeatedly executes the \( S_i \) for which \( G_i \) holds. If none of the \( G_i \) hold, then the repetition terminates.

Non-deterministic Repetition:

\[ * [\begin{array}{c}
G_1 
\rightarrow
S_1 \\
\vdots \\
G_n 
\rightarrow 
S_n
\end{array}] \]

Same as the deterministic repetition except multiple guards may be \texttt{true} at the same time. If multiple guards hold, an arbitrary true guard is selected.
Common Abbreviations

- \([B]\) stands for \([B \rightarrow \text{skip}]\)
- \(*[S]\) stands for \(*[\text{true} \rightarrow S]\)