Establishing safety of X10 programs (v 0.1)
Vijay Saraswat

1 Outline

1. An *action* is a function from variables to variables. A sequential program executes a *totally ordered* sequence of actions. A program with `async` executes a *partially ordered* sequence of actions. We call the partial order the *happens before* relation since in every possible execution of the program, f must happen before g.

2. For two actions a, b if neither $a \leq b$ nor $b \leq a$ then we say that a and b *May Happen in Parallel* (MHP), and write $a\#b$.

3. The rules for HB are simple.

   With each statement S we associate a *process*. A process is a triple $P = (X, \leq, Z)$ where X is a (finite) set of actions, $\leq$ is a partial order on X and $Z \subseteq X$ marks the subset of actions of X that must execute before any process Q that follows P can start executing.

   We can define operators on processes to mimic sequential execution, `async` and `finish`. Before that some preliminaries.

   - For sets A and B, the set $A \times B$ is just the set of pairs whose first element is from A and second element from B.
   - For a binary relation R on a set A, let $R\star$ represent the transitive closure of R.
   - For a partially ordered set $(U, \leq)$, let $\min(U)$ stand for the minimal elements of U (i.e. all elements $x \in U$ such that there is no other element y such that $y \leq x$. Similarly for $\max(U)$.
   - If $P = (X, \leq, Z)$ then we define X_P to be X, $\leq_P$ to be $\leq$ and Z_P to be Z, $\min(P)$ to be $\min_{\leq_P}(X_P)$, and $\max(P)$ to be $\max_{\leq_P}(X_P)$.

4. Now we can provide the definitions. Let $\leq' = (\leq_P \cup \leq_Q \cup (Z_P \times \min(Q)))\star$. Let f be an action representing a single statement. Let R be a process that is totally ordered, and let g represent the action obtained by composing the actios of R in the order specified by the

given total order.

$$
\begin{aligned}
f &= (\{f\}, \emptyset, \{f\}, \{f\}) & (1)\\
\texttt{atomic}R &= (\{g\}, \emptyset, \{g\}, \{g\}) & (2)\\
P;Q &= (X_P \cup X_Q, \leq', \max_{\leq'}(Z_P \cup Z_Q)) & (3)\\
\texttt{async}\, P &= (X_P, \leq_P, \emptyset) & (4)\\
\texttt{finish}\, P &= (X_P, \leq_P, \max(P)) & (5)
\end{aligned}
$$

5. An *execution* of a process $P = (X, \leq, Z)$ is obtained by running the actions of X in any total order that extends $\leq$, from an initial heap. The *result* of the execution is the final heap.

6. Show: programs with `async` may have an HB order that is not total, hence may have multiple distinct executions, and therefore, multiple distinct results.

7. A program S is said to have a *concrete data race* if the associated process has two actions a, b such that they operate on the same location, at least one of them is a write and it is not the case that $a \leq b$ or $b \leq a$.

8. Programs with no data races are *scheduler-determinate*: on every execution they will yield the same result.