Textual State Machine Syntax and Semantics
6.005 Elements of Software Construction

Grammar

state-machine ::= state-decls initializer operation-decl*
state-decls ::= state state-decl
state-decl ::= java-var-decl*
initializer ::= init java-stmt
operation-decl ::= operation*
operation ::= op opname [ ( arg-decl, ) ] cond effect
cond ::= when java-boolean-expr
effect ::= do java-stmt

You can use Java syntax for the declarations of state variables and arguments, for the conditional expressions, and for the initialization and effect statements. If there is no initialization section, you should initialize the state variables in their declarations.

Semantics

Defines a state machine whose states are assignments to the declared variables. The initial state is as given in the initialization section or in the variable declarations. Each declared operation defines an event class. An event can occur when the condition given for the operation is true, and it leads to the state that results from applying the operation's effect. An omitted operation condition is equivalent to TRUE: that is, the operation can occur in any state.

Non-determinism

To express non-determinism, you can use boolean expressions as initializers and operation effects. In this case, the meaning is that the initial states are all those satisfying the initialization, and an operation can result in any of the transitions satisfying the effect. The after value is marked in the effect expression with a prime to distinguish it from the before value.

Example: simple counter

```java
state int i = 0
op incr do i = i + 1
op reset do i = 0
op show(int d) do d = i
```

Example: nonce generator

```java
state Set<Id> ids
init ids.isEmpty()
op generate(Id i)
  do !ids.contains(i) && ids' == ids.add(i)
```

Example: light switch

```java
state boolean on = false
op flipOn
  when !on do on = true
op flipOff
  when on do on = false
```