A MoDeST plugin for Eclipse

Christophe Boutter

Dependable Systems and Software

30.4.2007

Outline

1. Introduction
   - Eclipse and MoDeST
   - Plugin overview
   - MoDeST introduction
2. The MoDeST Step Simulation
3. Demo

Eclipse

What it is

- open source community
- open development platform
- extensible frameworks, tools and runtimes
- built in Java

Some well known plugins

- JDT - Java Development Toolkit
- CDT - C/C++ Development Toolkit

MoDeST

What it is

- Modeling and description language for stochastic systems
- Models probabilistic non-deterministic systems with realtime constraints
- Can be simulated with the MoToR tool
- Easily understandable syntax

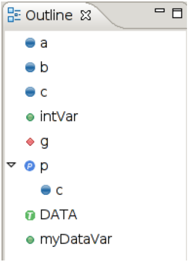
Example

```
action a, b, c;
alt {
    :: a; b
    :: c
}
```

Feature overview

Editor features:

- Syntax highlighting
- Context sensitive word completion
- Syntax error recognition
- An outline



Outline

- a
- b
- c
- intVar
- g
- p
  - c
- DATA
- myDataVar

Feature overview

External programs usable out of Eclipse:

- The compiler
- A dot output of the STA
- The FSNS (*First State Next State*) interface

Feature overview

The MoDeST Step Simulation:

- . . . let’s first have a short intro to MoDeST

Short intro to MoDeST

Clocks

- advance with system time
- can be reset
- cannot be set to a certain value!

Probability distributions

It's possible to sample a variable from a probability distribution.

```
x = Uniform (10,20);
```

Guards

They are blocking, unlike a normal if.

```
when (clock > 3) act
```

Short intro to MoDeST

Alternatives

Non-deterministic alternatives can be declared with an alt.

```
alt{
    :: a
    :: b
}
```

Probabilistic alternatives

Probabilistic alternatives can be declared via a palt.

```
palt{
    :2: a
    :1: b
}
```

Short intro to MoDeST

Synchronized concurrency

- concurrency obtained with a par
- synchronized over actions in the common alphabet

```
par{:: a; b
    :: b; c
}
```

Relabeling of actions

Actions can be relabeled and hidden in a parallel context.

```
par{:: a; b
    :: relabel {a} by {c} a
}
```

Short intro to MoDeST

Features well known from other programming languages:

- Use of variables (int, float, . . .)
- Exception handling (try, catch, throw)
- Process definition (process)
- Loops (do, while)

The MoDeST Step Simulation

Overview

- Step wise simulation of MoDeST code
- Good visualization
- MoDeST semantics conform
- ⇒ FSNS++

Restrictions

- No variable interpretation (no assignments)
- No realtime
- Same restrictions as MoToR

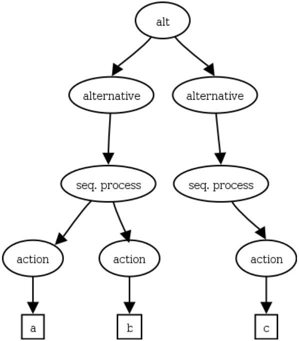
The simulation framework

The simulation framework consists of:

- The SimulationAction button
- The SimulationRoot class
- The SimulationView
- The model tree of SimulationNodes

What is a SimulationNode?

- It is an abstract class, parent of all nodes in the model tree.
- It represents a MoDeST language construct.
- It holds the basic functionality common to all nodes.
- It defines abstract functions that all nodes must implement differently.



Important nodes

- BreakNode: responsible for stopping a do loop
- TryNode and ThrowNode: responsible for the exception handling
- PaltNode: creates the PaltTransition instances
- ParNode: gets the ParTransition instances and merges them.

The typical simulation execution

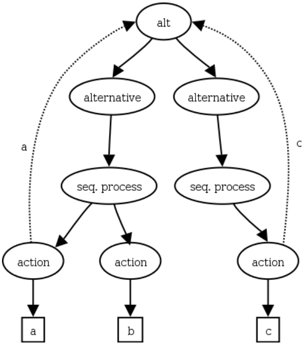
1. The SimulationRoot collects the transitions.
2. The SimulationView displays this transitions.
3. The user selects one of these transitions.
4. The selected SimulationNode is notified.
5. Side effects are handled.
6. The SimulationNode notifies its parent that a transition was taken.

The simulation end

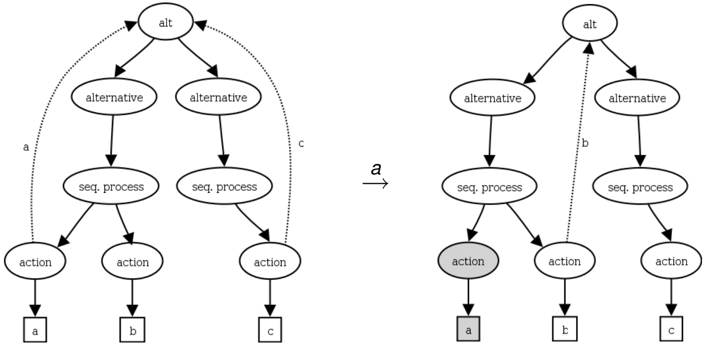
The execution steps are repeated until:

- No transition is left and all nodes are finished.
- No transition is left and a deadlock occurred.
- An exception was thrown and not catched.

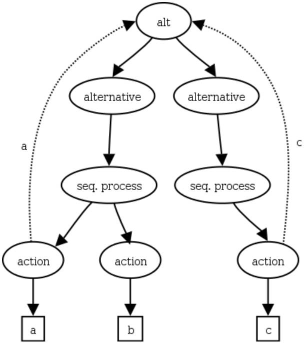
The progression of the simulation



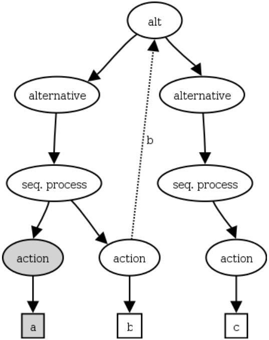
The progression of the simulation



The progression of the simulation



$\xrightarrow{a}$



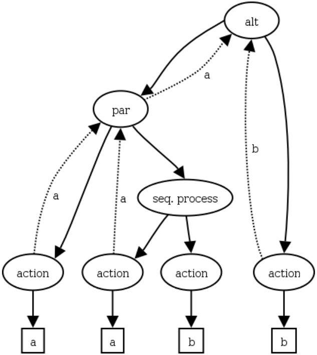
$\xrightarrow{b} \quad \surd$

The progression of the simulation

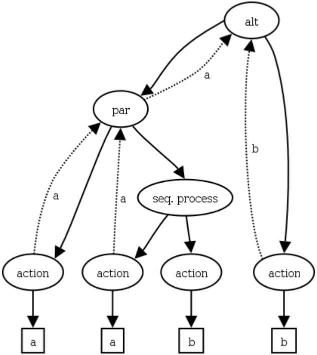
Example

```
alt {
   :: par {:: a
            :: a; b
       }
   :: b
}
```

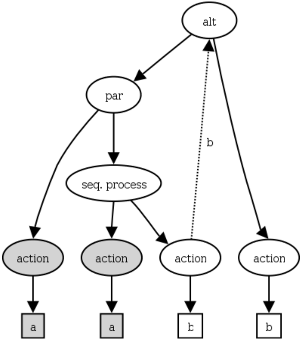
The progression of the simulation



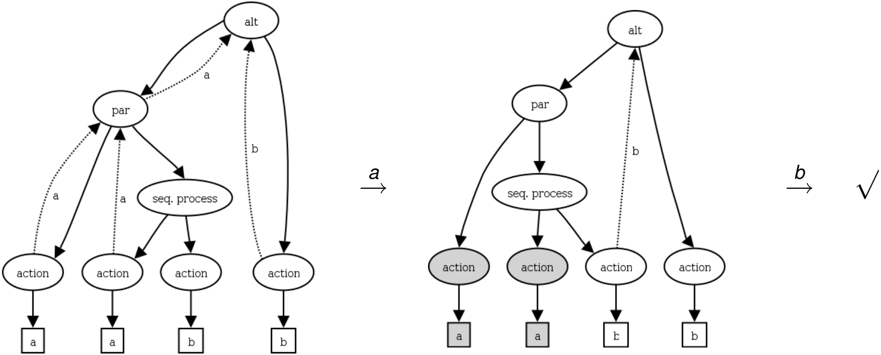
The progression of the simulation



$\xrightarrow{a}$



The progression of the simulation



Demo