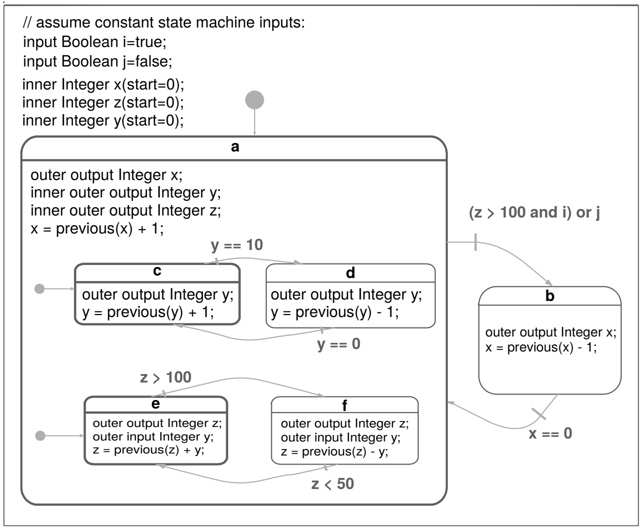
Flattening of Modelica State Machines: A Practical Symbolic Representation

Bernhard Thiele[1] Adrian Pop[1] Peter Fritzson[1]

[1]PELAB, Linköping University, Sweden,
{bernhard.thiele,adrian.pop,peter.fritzson}@liu.se

Modelica 3.3 introduced dedicated built-in language support for *state machines* that was inspired by semantics known from *Statechart* and *mode automata* formalisms. The specification describes the semantics of these constructs in terms of data-flow equations that allows it to be related to the Modelica DAE representation which is the conceptual intermediate format of Modelica code after instance creation (flattening). However, a complete transformation of state machine constructs into data-flow equations at the stage of flattening requires an early commitment to implementation details that potentially hinders model optimizations at subsequent translation phases. Also, due to the required substantial model transformation the semantic distance between the original source model and the *flattened* representation is rather large. Hence, this paper proposes a more versatile symbolic representation for flattened state machine constructs that preserves the state machine's composition structure and allows postponing optimizations to subsequent compiler phases.

The proposed approach has been implemented for the OpenModelica compiler. Figure **(a)** shows an example of a Modelica state machine using hierarchical and parallel composition of state machines. Figure **(b)** illustrates the compilation process using the intermediate representation for state machines.



```
// assume constant state machine inputs:
input Boolean i=true;
input Boolean j=false;
inner Integer x(start=0);
inner Integer z(start=0);
inner Integer y(start=0);
```

```
outer output Integer x;
inner outer output Integer y;
inner outer output Integer z;
x = previous(x) + 1;
```

```
outer output Integer y;
y = previous(y) + 1;
```

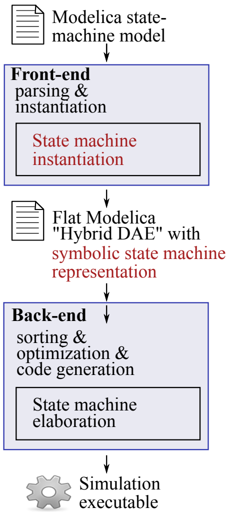
```
outer output Integer y;
y = previous(y) - 1;
```

```
outer output Integer x;
x = previous(x) - 1;
```

```
outer output Integer z;
outer input Integer y;
z = previous(z) + y;
```

```
outer output Integer z;
outer input Integer y;
z = previous(z) - y;
```

(a) Modelica state machine using hierarchical and parallel composition of state machines[a].



(b) Outline of the state machine compilation process.

[a] Motivated by the example shown in F. Maraninchi and Y. Rémond: Mode-Automata: a new domain-specific construct for the development of safe critical systems, *Science of Computer Programming*, 46:219–254, 2003.