A Virtual Machine Supporting Multiple Statechart Extensions

Thomas Huining Feng

Supervisor: Prof. Hans Vangheluwe

MSDL, McGill

http://msdl.cs.mcgill.ca/people/tfeng/

hfeng2@cs.mcgill.ca

Introduction

The goal is to build a generalized simulator capable of executing statechart models.

Design considerations:

- **Interpretation vs Compilation**. SVM is a statechart *interpreter*.
- **Virtual-time Simulation and Real-time Execution**. The same model can be simulated for analysis purpose and executed as a final product.
- **Model Specification**. A statechart model is specified in a text file, which is easy to handle.
- **Portability**. SVM is implemented in Python and Jython. It is portable to many operating systems and architectures.
- **Functionality**. SVM is capable of interpreting a model specified in the extended statechart formalism.

Major Contribution

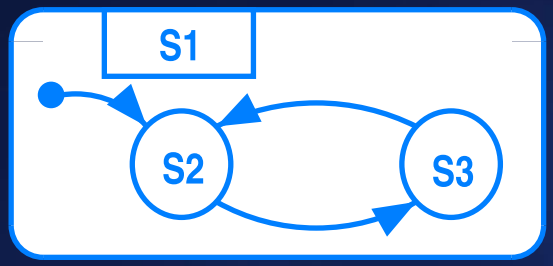
- Extensions are made to enhance the expressiveness of statecharts. The extended formalism is able to describe components and infinite states.
- SVM is built as a simulation/execution tool. It is also an experimental environment to test and analyze semantic elements in the statechart formalism.

Statechart Introduction

Statechart (a discrete-event formalism) is a powerful tool to describe both software systems and physical systems.

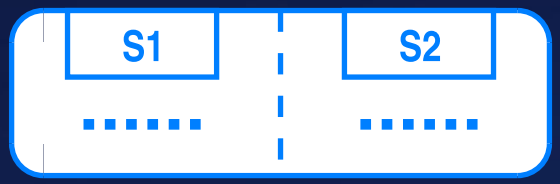
Statechart Elements

finite state automata



hierarchy

history



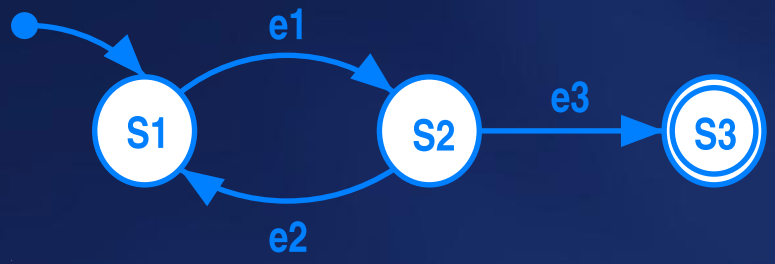
orthogonal components

transition

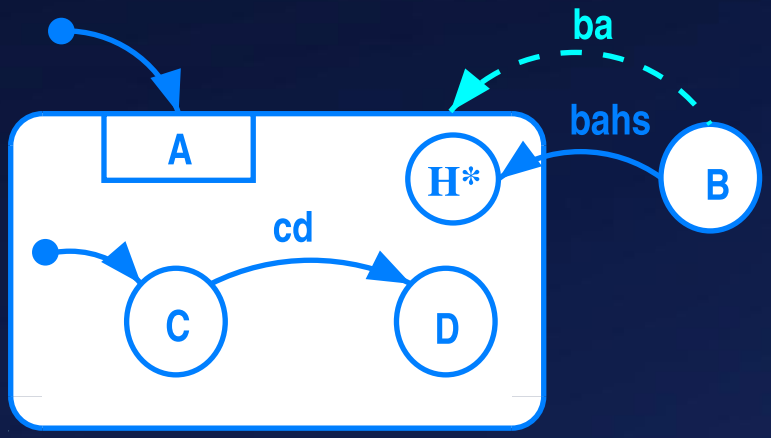
enter/exit actions

Simple Statechart Model

```
STATECHART:
  S1 [DS]
  S2
  S3 [FS]

TRANSITION:
  S: S1
  N: S2
  E: e1
  O: [DUMP("e1 is triggered")]

TRANSITION:
  S: S2
  N: S1
  E: e2
  O: [DUMP("e2 is triggered")]

TRANSITION:
  S: S2
  N: S3
  E: e3
  O: [DUMP("finish")]
```



Hierarchical Statechart Model




```
STATECHART:
  A [DS] [HS*]
    C [DS]
    D
  B
......
TRANSITION:
  S: A.C
  N: A.D
  E: cd

TRANSITION: [HS]
  S: B
  N: A
  E: bahs
......
```

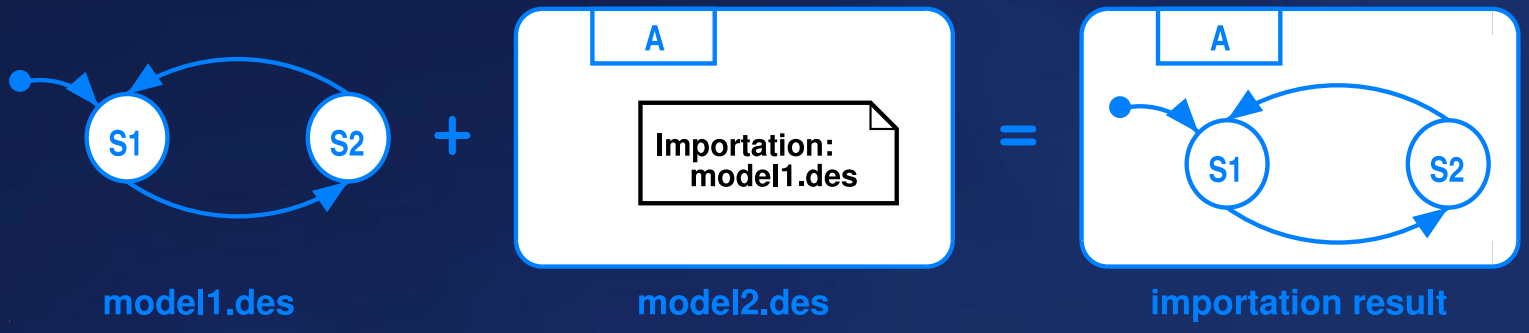
Extension 1: Model Importation

Motive

Statecharts are not modular and thus hard to reuse. The complexity of a statechart model exhibits itself even in solving small problems. It is desirable to divide a large model into smaller parts and assemble them after designing separately. Individual parts can also be reused.

SVM Extension

SVM presents a general idea of model importation. An imported model is a full-function model in its own right. When imported, all its states and transitions are placed in a state of the importing model.



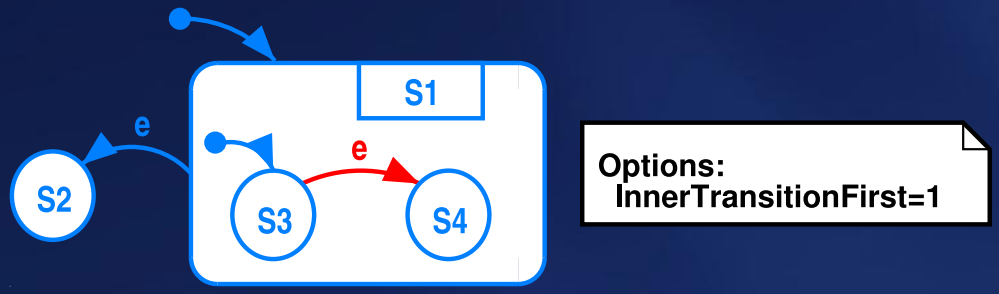
Extension 2: Transition Priorities

Motive

When two or more transitions are enabled by the same event, there is a conflict. In UML, if the source state of a transition is a substate of the source state of the other, it gets higher priority (inner-first); however, in the STATEMATE semantics, it gets lower priority (outer-first). It is desirable to enable both of these schemes.

SVM Extension A.

Every model has a global option: `InnerTransitionFirst`.



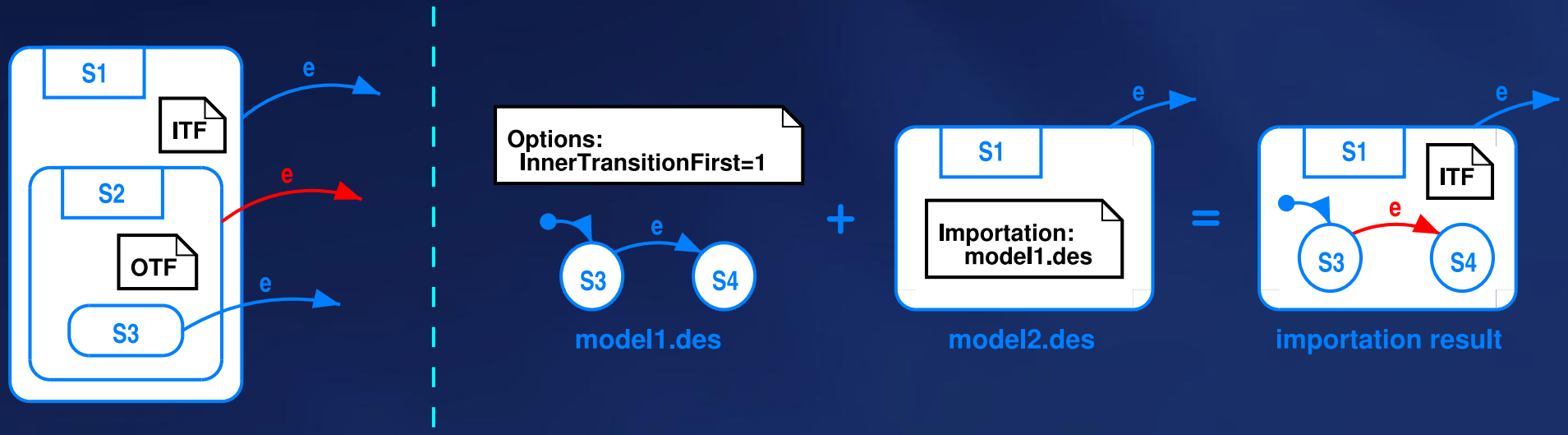
If the current state is S1.S3 and event e occurs, the new state will be S1.S4.

SVM Extension B.

Every state can be associated with one of the following properties:

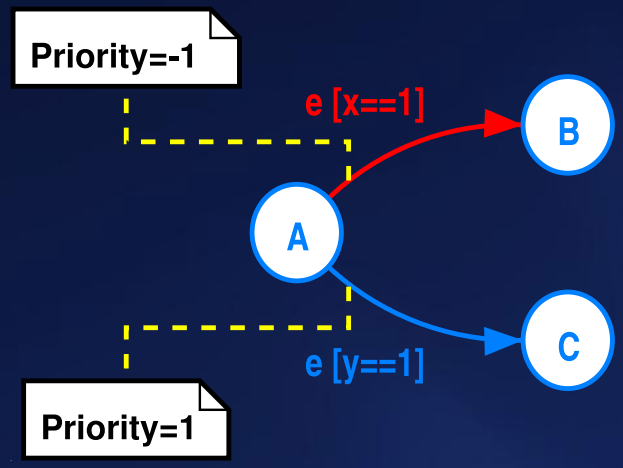
- **ITF**. Inner transition first.
- **OTF**. Outer transition first.
- **RTO**. Reverse transition order. (If its parent state is ITF, it is OTF; vice versa.)

The property of a state override the setting of its parent *in its scope*.



SVM Extension C.

Every transition can be associated with an integer priority number (by default, it is 0). For conflicts which cannot be solved by extensions A and B, a transition with the smallest priority number is fired.



When e occurs, if the model is in state A and both conditions are true,

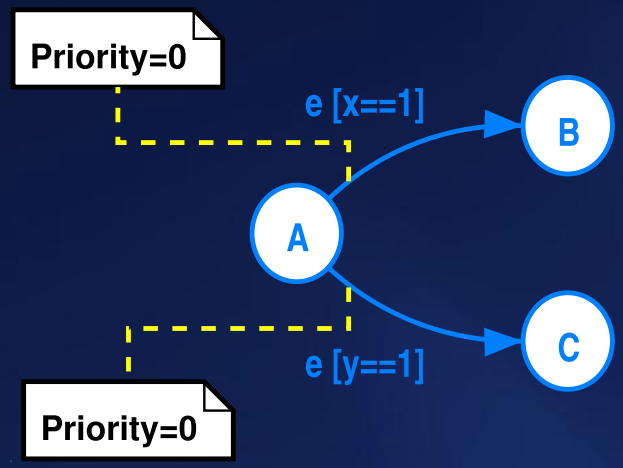
$$\begin{cases} x = 1 \\ y = 1 \end{cases}$$

the state will change to B.

SVM Extension D.

If conflicts still exist at run-time, which cannot be solved by extensions A, B and C, the choice is strictly random according to a uniform distribution.

This is usually caused by a design flaw, in which case the designer cannot foresee a potential conflict in the model.



Extension 3: Parametrized Model Templates

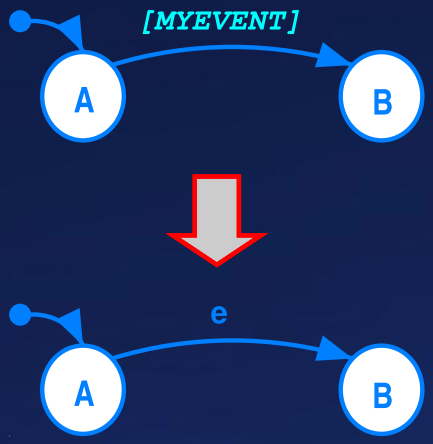
Motive

Usually a design cannot be reused in a new system without any change or customization. The importing model should be able to customize the imported model before placing it in one of its states.

The customization should be restricted and modular.

SVM Extension

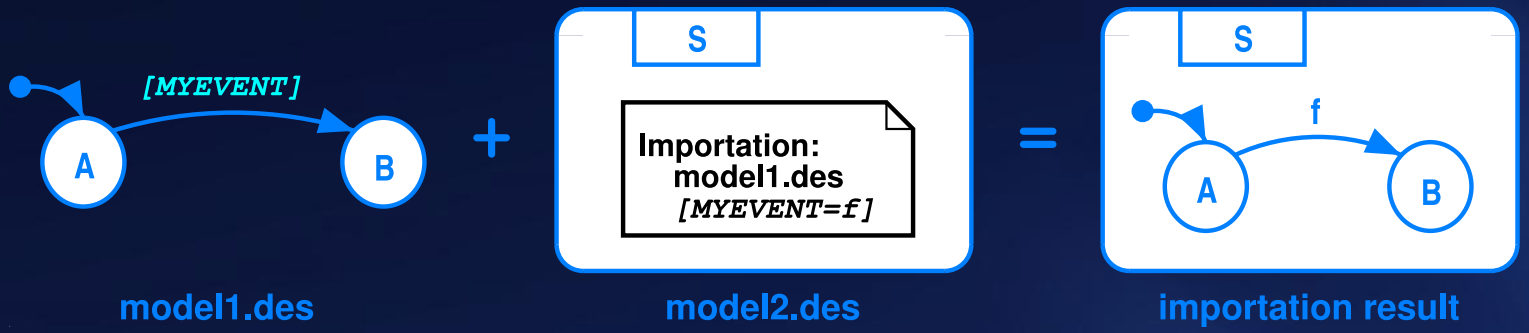
Macros can be defined in a macro and used anywhere in its description.




```
MACRO:
  MYEVENT = e
......
TRANSITION:
  S: A
  N: B
  E: [MYEVENT]
......
```

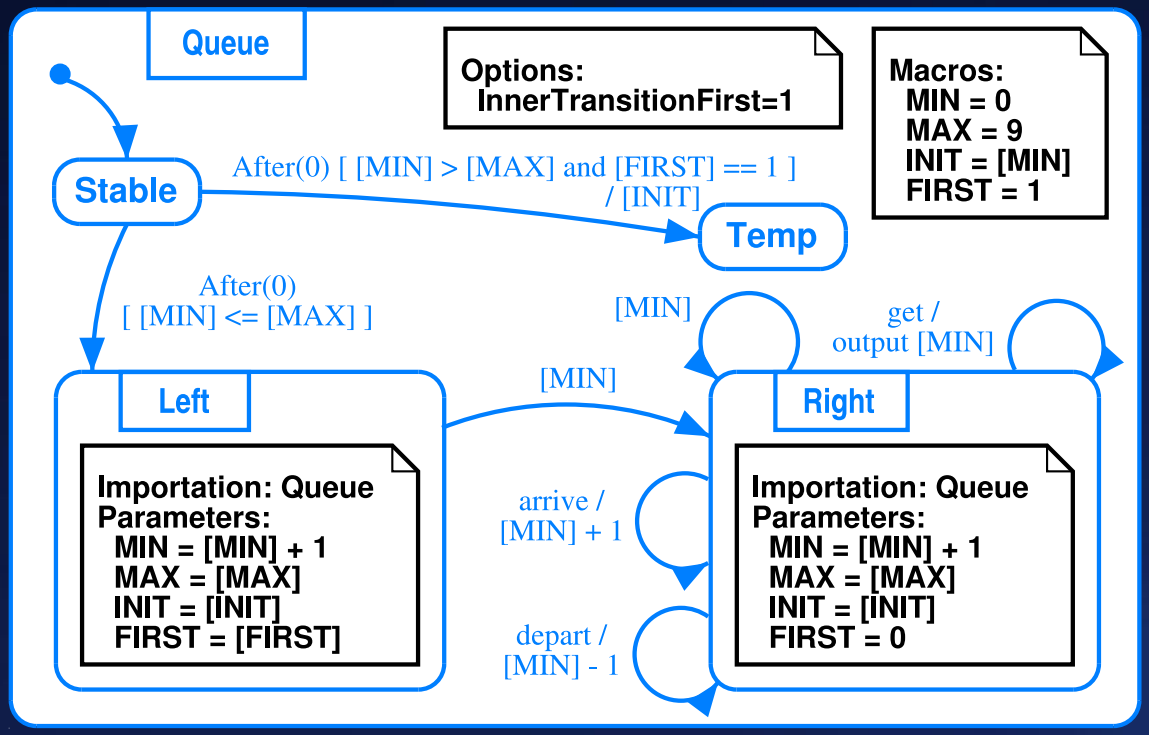
SVM Extension (Continued)

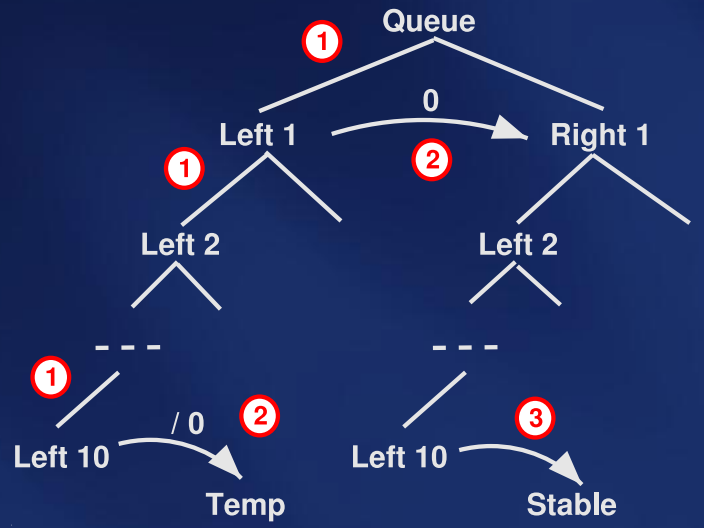
The designer is allowed to redefine the macros when reusing a model.



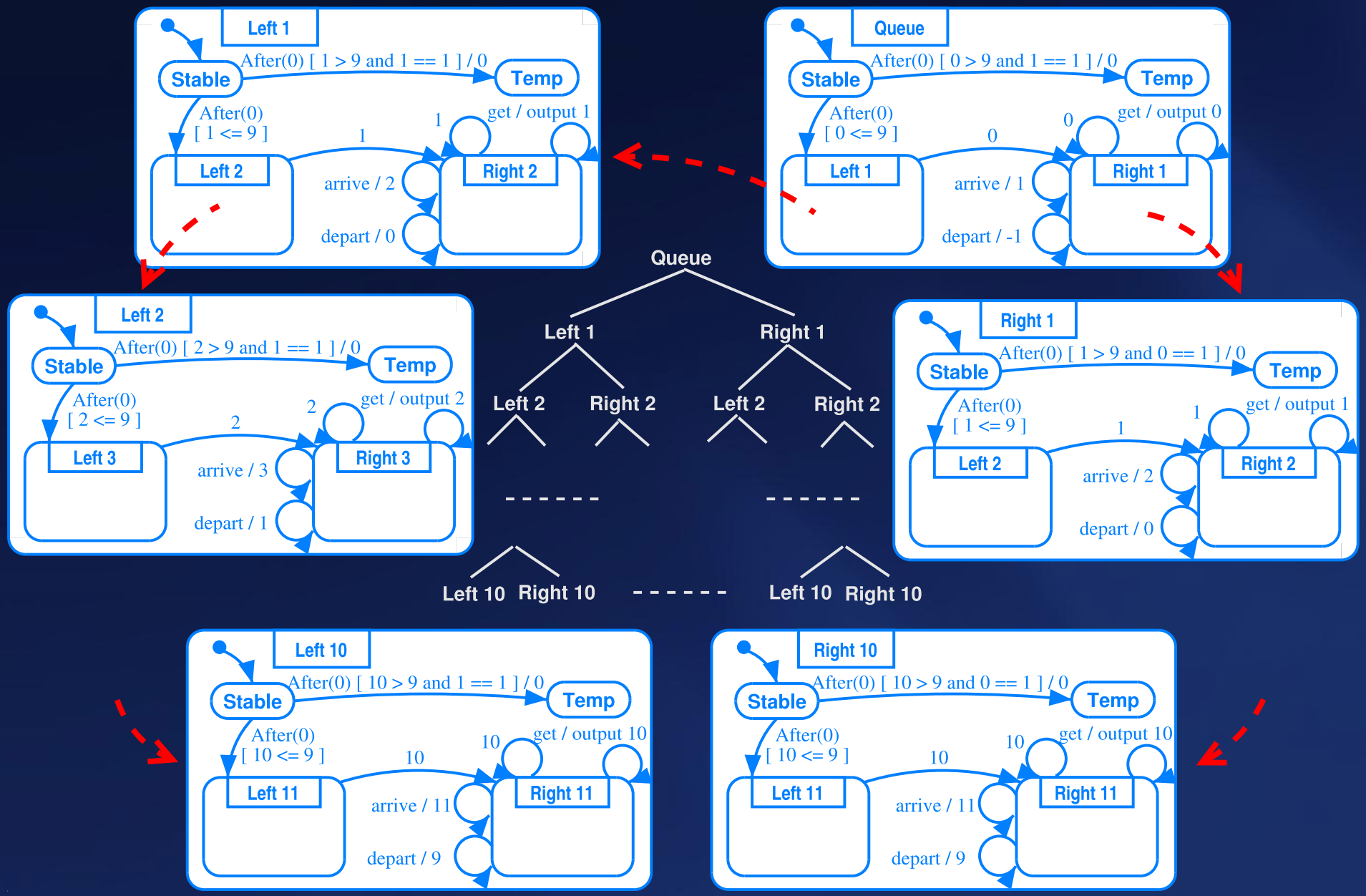
The outside world is able to modify the behavior of a model only by parameters, which is defined in the model with its consent. There is no way to modify its hard-coded parts.

An Example: Queue





- **Create structure**. Import the same model 10 times and reach state `LEFT.LEFT.LEFT.LEFT.LEFT.LEFT.LEFT.LEFT.LEFT.LEFT.TEMP`
- **Initialize**. Event `[INIT]` is raised. For `[INIT]==0`, new state will be `RIGHT.LEFT.LEFT.LEFT.LEFT.LEFT.LEFT.LEFT.LEFT.LEFT.STABLE`
- **Ready**. The model is ready to receive events 0 $\sim$ 9, arrive, depart and get


Left 1
Stable
After(0) [1 > 9 and 1 == 1] / 0
Temp
After(0)
[1 <= 9]
1
1
get / output 1
Left 2
Right 2
arrive / 2
depart / 0
Queue
Stable
After(0) [0 > 9 and 1 == 1] / 0
Temp
After(0)
[0 <= 9]
0
0
get / output 0
Left 1
Right 1
arrive / 1
depart / -1
Left 2
Stable
After(0) [2 > 9 and 1 == 1] / 0
Temp
After(0)
[2 <= 9]
2
2
get / output 2
Left 3
Right 3
arrive / 3
depart / 1
Queue
Left 1
Right 1
Left 2
Right 2
Left 2
Right 2
Left 10
Right 10
Left 10
Right 10
Right 1
Stable
After(0) [1 > 9 and 0 == 1] / 0
Temp
After(0)
[1 <= 9]
1
1
get / output 1
Left 2
Right 2
arrive / 2
depart / 0
Left 10
Stable
After(0) [10 > 9 and 1 == 1] / 0
Temp
After(0)
[10 <= 9]
10
10
get / output 10
Left 11
Right 11
arrive / 11
depart / 9
Right 10
Stable
After(0) [10 > 9 and 0 == 1] / 0
Temp
After(0)
[10 <= 9]
10
10
get / output 10
Left 11
Right 11
arrive / 11
depart / 9

MSDL

Conclusion

SVM is a flexible and modular tool for the extended statechart formalism.

Future work will focus on built-in support for distributed simulation, model analysis and model checking.

Thank you for your attendance!