MEANS TO CONTROL THE SELECTION OF STATES

1. Introduction

Dymola supports automatic state selection according to the specification of Modelica 2.0. For a long time there was a discussion within the Modelica Design Group on how to separate the state selection problem from the specification of initial conditions. The approach taken is to introduce another attribute, stateSelect, for variables being subtypes of Real to allow a library developer or a user to give hints or even imperatively control the selection of variables to use as continuous time state variables. The state selection is separated from the specification of initial conditions. The fixed attribute should exclusively be used for specifying start conditions and it should not influence the selection of states at all.

2. Motivation

The general view is that selection of states ought to be done automatically. This is also possible and unproblematic in most models, and we thus clearly understand that manual state selection can easily be overused. However, there are several reasons for allowing model library developers as well as users to influence or control the state selection:

- **Accuracy:** There are often many sets of state variables that will work from a pure mathematical point of view. However, they may have drastically different numerical properties. For mechanical systems it is favorable to use relative positions as state variables. If absolute coordinates are used then accuracy is lost when taking differences to calculate relative positions. The effect is drastic in rotating machinery systems and power systems where angular positions are increasing with time, but relative positions are rather constant, at least in normal operation. Say that two rotating bodies are connected by a spring such that the relative distance between them are 1 and that their angular speed is 1000. If the positions are calculated with a relative accuracy of 0.001, after one second there is hardly any accuracy in calculating the distance by taking the difference. The difference behaves irregularly and gives an irregular torque. The simulation stops. It is very difficult for a tool to find this out without actually doing simulation runs. Model developers for mechanical systems and power systems know it very well. It would be easy for them to indicate that absolute positions are bad choices when selecting states.
- **Efficiency by avoiding inverting functions**: The relations between possible sets of state variables may be non-linear. For some choices it may be necessary to invert non-linear functions, while for another set it is straightforward to calculate others. A typical example is thermodynamic problems, where you have property functions. They often assume two variables to be inputs (for example pressure and enthalpy) and calculate other properties (such as temperature, density etc). Thus, if such variables are selected as state variables it is "simply" calling property functions to calculate other need variables. If not it is necessary to solve equation systems to calculate the input variables. A model library developer knows this and it is straightforward to him to indicate good choices when selecting dynamic states.
- **Selecting a less nonlinear representation**: Different sets, x, of states gives an ODE, der(x) = f(x) where the right hand side f have different properties. In general, the problem is simpler to solve if f is a less nonlinear problem. The Park transformation for three-phase power systems is a classical way of transforming a nonlinear time-varying ODE into a time-invariant linear ODE. For control design it is very favorable to have linear time-invariant models, because there are lot of analysis and design methods and tools for such models. When using linearized versions of Modelica models it is important that the set of state variables is insensitive to minor changes in the model.
- **Avoiding dynamic state selection**: When selecting states the problem consists of a set of algebraic state constraints that relate dynamic variables. It may be remarked that these constraints are equations that are differentiated by Pantelides's algorithm. The task when selecting states is actually to use the algebraic constraints to solve for some of the variables, which thus are deselected as states and the remaining dynamic variables become state variables. A subset of dynamic variables can be deselected locally if its Jacobian is non-singular. In the general case the state selection must be made dynamic, but in many real

applications it is possible to make a static selection of states. If the Jacobian has constant elements it is straightforward to make it automatically. However, for non-linear problems such as closed kinematics loops it is difficult to establish that a time-varying Jacobian always is non-singular. For reasons of efficiency it would be favorable to avoid the overhead of dynamic state selection and allow a user to inform that a certain selection of states will always work. Tools can support such an explicit control. Using dynamic state selection and making off-line simulations one can find a fixed choice that will work for real-time simulation, where efficiency is really needed.

- **The reinit construct:** The construct reinit(x) requires that x is state.
- **Use auxiliary variables as states:** To avoid unnecessary differentiation, it is useful to consider only variables appearing differentiated in a model as candidates when selecting states. It means that if a user would like to see an auxiliary variable, v, as a state variable, he has today to introduce another variable, say derv and an equation derv = der(v) to make the derivative der(v) appear in the model. It would be convenient to have a simpler way to introduce a variable as a state candidate
- **Sensors:** A sensor for measuring speed, v, makes a variable differentiated, v = der(r) and in most cases it is not desirable to have the variable of the sensor model as a state variable. Introduction of variable for just plotting should not influence the state selection.

3. The state select attribute

A variable being subtype of Real variable has an attribute stateSelect to indicate its possible use as state variable. Its value can be

- *never*: Do not use as a state at all.
- *avoid:* Avoid it as state in favour of those having the *default value*
- *default*: If the variable does not appear differentiated in the model this means *no*.
- *prefer*: Prefer it as state over those having the *default value.*
- *always*: Do use it as a state.

The values of the stateSelect attribute are to given as

```
Real y(stateSelect = StateSelect.never);
Real y(stateSelect = StateSelect.avoid);
Real y(stateSelect = StateSelect.default);
Real y(stateSelect = StateSelect.prefer);
Real y(stateSelect = StateSelect.always);
```

The two extreme values *never* and *always* have clear and context independent meanings. If stateSelect is *always*, the variable will be a state. If such a variable does not appear differentiated in the model, the index reduction procedure will differentiate equations in order to be able to calculate the derivative. A model with two variables, x and y, with attribute *stateSelect* being *always* and being algebraically constrained, is thus erroneous. It is compulsory for variables appearing as arguments in reinit expressions. It supports explicit control of the selection of states and gives the user full control. It eliminates use of dynamic state selection. A dynamic state selection problem should only include variables having *stateSelect* being *prefer, default* or *avoid.*

The value *never* forbids the variable to be used as a state and it solves the sensor problem:

```
Real r(stateSelect = StateStateSelect.never);
Real v = der(r);
```

The value prefer indicates that the variable should be used as a state when possible. The ambiguity lies in that there may be several candidates with prefer when selecting states. It solves the problem of giving preference to relative positions in mechanical problems. It is also useful for thermodynamic problems to avoid nonlinear equation systems. However, here the value *never* may be useful to rule out other candidates as well.

The value *default* means *never* for algebraic variables of the model. The index reduction procedure may introduce derivatives of algebraic variables when differentiating equations. However, this should not make them candidates for being state variables. Neither should higher order derivatives make derivatives candidates for being state variables. For example in mechanics we have

der(r) = v;
m*__der__(v) = F;

The index procedure may introduce the second order derivative of r, but we should then not consider der(r) as candidate for being state variable.

The priorities for state selection are thus *always*, *prefer*, *default and avoid* for variables appearing differentiated.