1 Examples

In engineering, Feature Modeling is widely used. In the context of feature modeling, four kinds of features are used: (i) mandatory features (logical AND), (ii) alternative features (logical XOR), (iii) optional features and, (iv) OR-features (logical OR). In analogy to the semantics of feature models and mathematical logical expressions, Sections 1.1, 1.2, and 1.7 can be classified as mandatory (logical AND). Sections 1.3, 1.4, 1.5, and 1.6 can be seen as alternative (logical XOR). For expressing optional like 1.8 and OR features like 1.9, an extension of the System Entity Structure (SES) is used. Sections 1.10, 1.11, and 1.12 are combinations of the previous mentioned basic examples.
Next to these basic and combined examples for creating SES in the Section 1.13 a complex example is shown.

1.1 Example #01: Aspect Node

An SES tree with a single aspect is the simplest example given in Figure 1.1. A coupled system *a* consists of an entity *b* and an entity *c*. For model generation, aspect nodes need to define the special attribute for couplings, while the leaf nodes have the mb-attribute attached referring to a basic model. Note, that the types of the ports of basic models, which the couplings connect, are left away to keep matters simple. Also note, that the derived Pruned Entity Structure (PES) and Flattened Pruned Entity Structure (FPES) are identical to the SES. The resulting abstract, simulator independent model on the right side, is given to illustrate what kind of model structure would result from the FPES. In the following chapters coupling definitions and the FPES are not given.

1.2 Example #02: Multi-Aspect Node

In Figure 1.2, a similar case to 1.1 is given. The system *as* consists of three children of type *b*. Multi-aspect nodes need to define two special attributes, the attributes

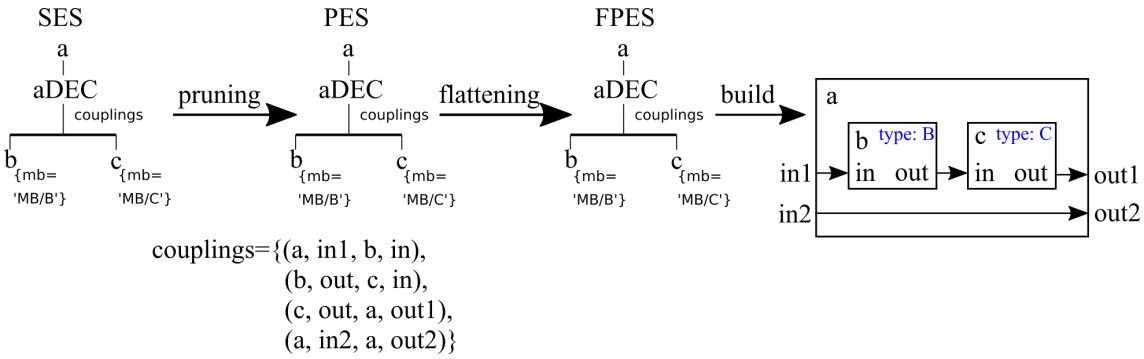


Figure 1.1: Example 01 - Aspect node

number of replications (numRep) and couplings. In this example, the numRep attribute is filled with a hardcoded three, but the value could also be assigned dynamically by using an SES variable (SESvar) or SES function (SESfcn). When pruning a multi-aspect node, the numRep attribute is evaluated and an aspect node with children of the same type is created. The derived PES is similar to the PES in Figure 1.1, except that there is a third brother, *b3*, and all children of *as* refer to the same model B in the model base. Children's names are generated by appending a number to the entity name b to ensure that the resulting siblings fulfill the axiom of valid brothers.

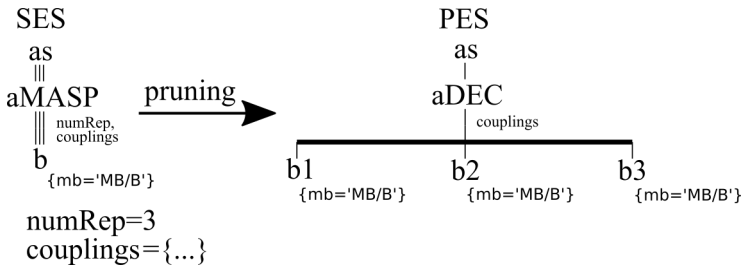


Figure 1.2: Example 02 - Multi-aspect node

1.3 Example #03: Specialization Node

An SES with a specialization node is shown in Figure 1.3. Based on the specialization rule, exactly one child needs to be selected to construct a valid variant. Thus, after pruning, the resulting system can either be of type *b* OR of type *c*. The father inherits the attributes of the child.

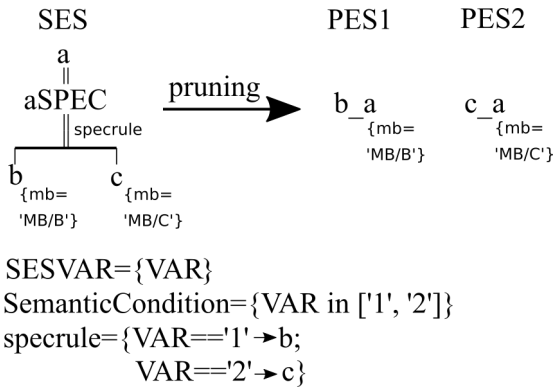


Figure 1.3: Example 03 - Specialization node

1.4 Example #04: Aspect Siblings

If two or more aspect nodes are on the same hierarchy level, exactly one of them has to be selected by evaluating the aspect rules of the aspect brothers. This is presented in Figure 1.4. The system *a* consists either of *b* and *c* or of *d* and *e*. The aspect rules one and two define which branch is selected during pruning.

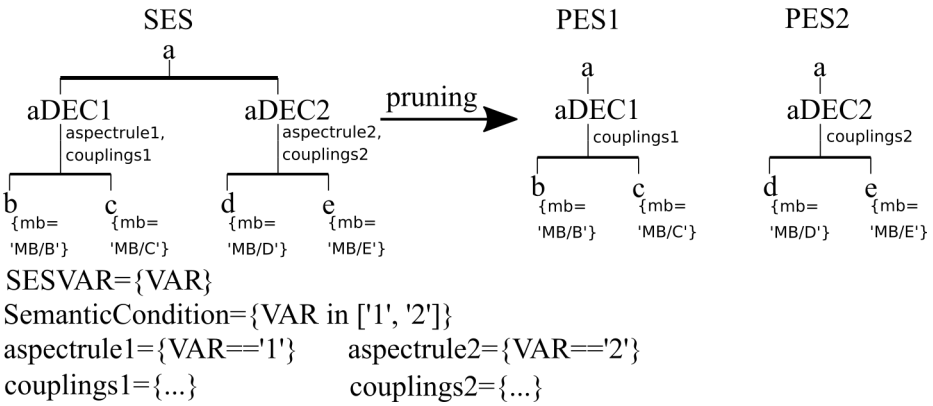


Figure 1.4: Example 04 - Aspect siblings

1.5 Example #05: Multi-Aspect Siblings

In the pattern shown in Figure 1.5, two multi-aspect nodes are on the same layer. In the first pruning step, the multi-aspect nodes are resolved leading to two aspect nodes. After this step, the resulting intermediate PES can be finally resolved as in Section 1.4 for aspect siblings previously described. The system *a* consists either of

b1, *b2*, and *b3* or of *c1* and *c2* depending on which child is selected based on the aspect rules.

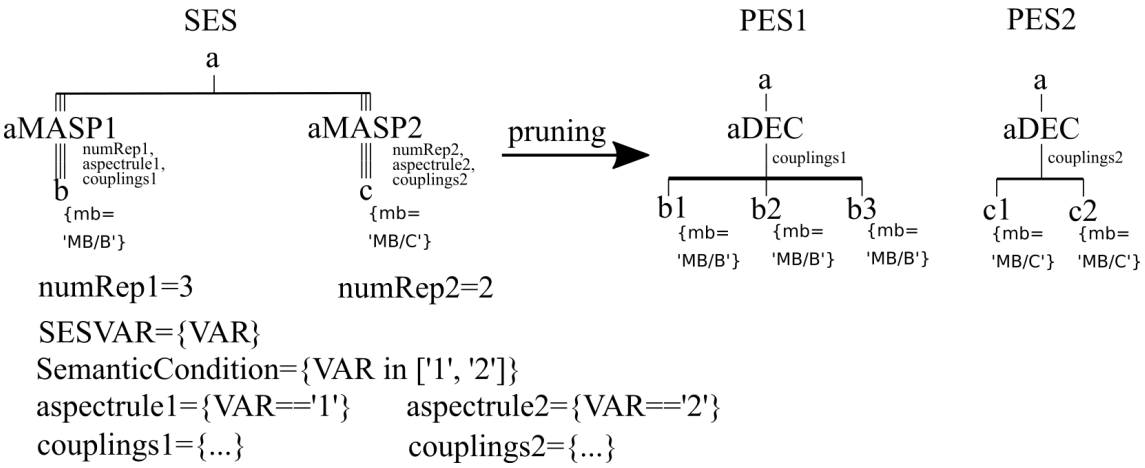


Figure 1.5: Example 05 - Multi-aspect siblings

1.6 Example #06: Aspect and Multi-Aspect Siblings

If there are more than one aspect nodes and multi-aspect nodes on the same hierarchy level, the behavior is like aspect siblings after the multi-aspect node is resolved. The example is shown in Figure 1.6. After pruning, the system is built up as before in Section 1.5.

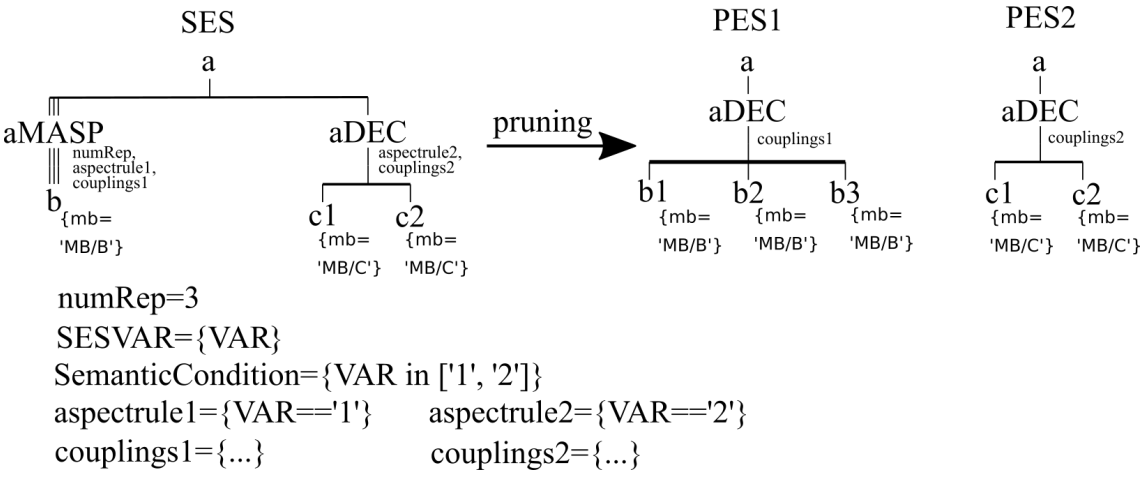


Figure 1.6: Example 6 - Aspect and multi-aspect siblings

1.7 Example #07: Specialization Siblings

If two or more specialization nodes are on the same hierarchy level, they will all be evaluated. For the example depicted in Figure 1.7, this means that a will specialize into one of b or c AND into one of d or e. Which child of the specialization is taken depends on the values of the SESvars and the specialization rules. If, for example, VAR1 is set to 1 and VAR2 is set to 2, at specialization *aSPEC1* the left child b is selected and, at specialization *aSPEC2*, the right child e is selected. During pruning, it depends on the pruning algorithm which of the two specializations is evaluated first. In this software left nodes are evaluated first. Therefore in this example, we assume that the left specialization node *aSPEC1* is evaluated first. The evaluation order influences what is the resulting value of the mb-attribute since, according to the inheritance axiom, attributes with the same name are overwritten in the parent node.

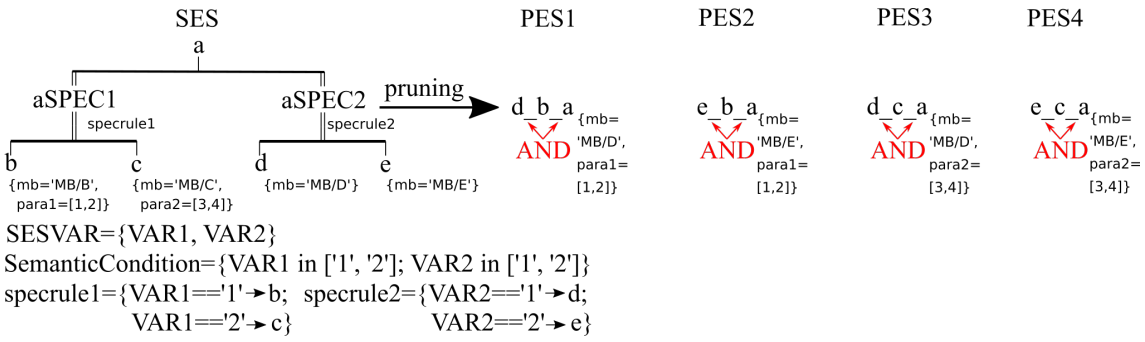


Figure 1.7: Example 07 - Specialization siblings

1.8 Example #08: The NONE Element

The NONE element is an extension of the SES. Nodes can have the name NONE. A NONE element for a leaf entity node means that, if the NONE branch is selected during pruning, the entity is not included at all. In the example shown in Figure 1.8, the specialization has a child which is a NONE element. Hence, this SES can evaluate to NONE during pruning based on the specialization rule. The system a can either be of type b or not existent at all.

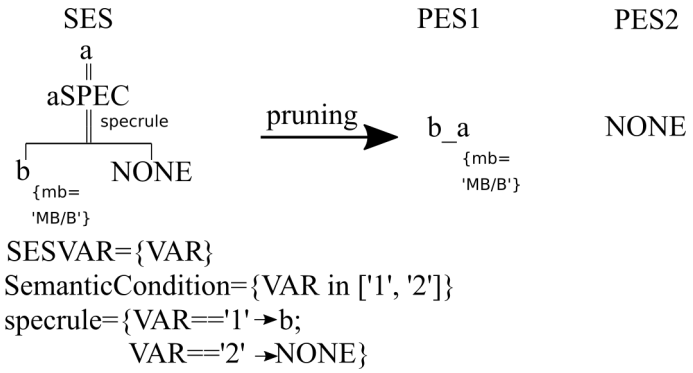


Figure 1.8: Example 08 - The NONE element

1.9 Example #09: Express OR in the SES

The example in Figure 1.9 shows an aspect node whose children are followed by specializations. Each specialization contains a NONE element (see Section 1.8) as one child. For expressing a logical OR, at least one specialization has to evaluate to a node not being NONE, as coded in the semantic condition. By defining the specialization rules reasonably, the user has to ensure, that this is guaranteed and the semantic conditions are met. This example is composed by the example in Section 1.1 in combination with the example in Section 1.8. After pruning, the system a consists of *b* and *c*. System *b*, in turn, is of type *bs* or not existent while *c* is of type *cs* or not existent. Couplings have to be justified when the tree changes by evaluating nodes. Since the couplings are defined at the aspect node *aDEC*, it is obvious, that there is a need for the possibility to define variable couplings. Variable couplings can be defined by SES Functions.

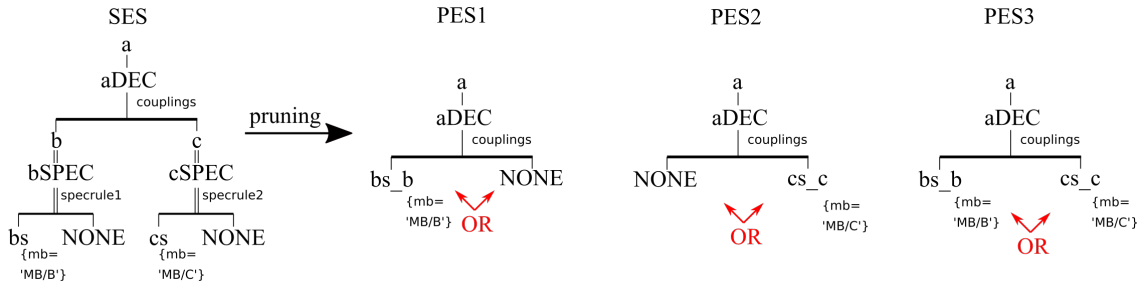
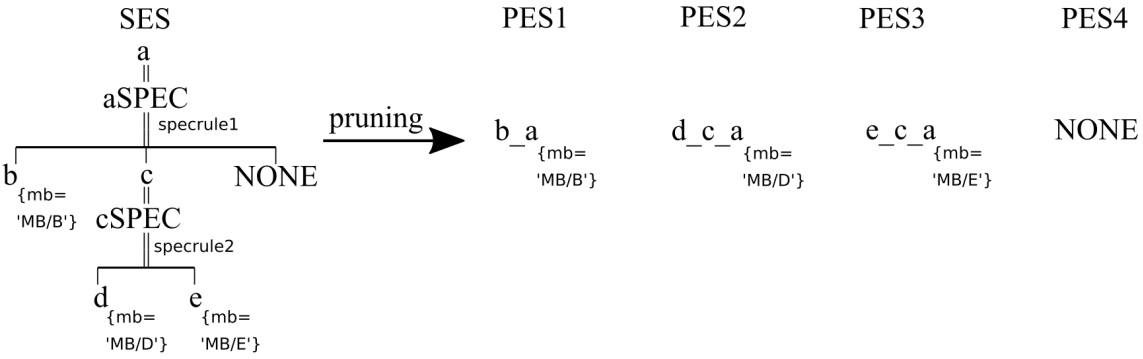


Figure 1.9: Example 09 - Express OR in the SES

1.10 Example #10: Two Specialization Nodes in One Path

During pruning, specialization nodes inherit the attributes of the selected child and append them to the father's attributes, as previously described. In Figure 1.10, there are two specialization nodes in one path. Additionally, the NONE element is used. This shall clarify the axiom for attribute inheritance. During pruning, firstly *aSPEC* is evaluated. In case the child *c* is selected, another decision for child *d* or child *e* is taken. In that case, the attributes of the child node of *cSPEC* are inherited to the first node.




```
SESVAR={VAR1, VAR2}
SemanticCondition={VAR1 in ['1', '2', '3']; VAR2 in ['1', '2']}
specrule1={VAR1=='1'→b;          specrule2={VAR2=='1'→d;
           VAR1=='2'→c;                     VAR2=='2'→e}
           VAR1=='3'→NONE}
```

Figure 1.10: Example 10 - Two specialization nodes in one path

1.11 Example #11: Specialization with Succeeding Aspect

Figure 1.11 depicts a specialization node with a succeeding aspect node. This example is a combination of a specialization node followed by a single aspect node. The system *a* can be of type *b* or *c*, the *b* can be decomposed in *d* and *e*.

1.12 Example #12: Specialization and Aspect Siblings

When aspect nodes and specialization nodes are brothers, the specialization node has to be resolved first during pruning. If, additionally, an aspect node is below the specialization node, during pruning two aspect nodes will become siblings. Since, in

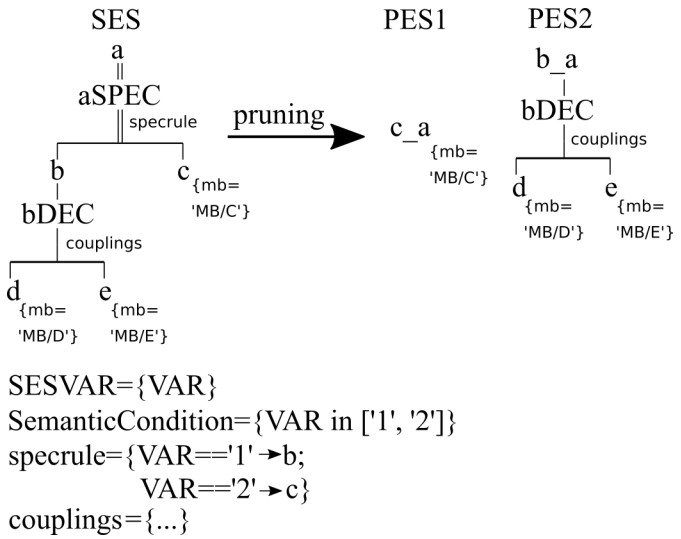
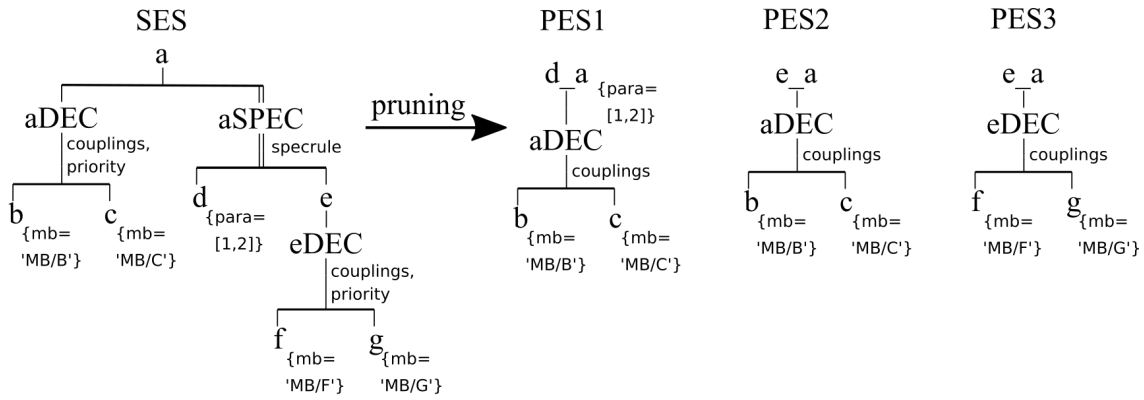


Figure 1.11: Example 11 - Specialization with succeeding aspect

this case, the occurrence of aspect siblings is not known until the first pruning step, aspect rules could not be formulated beforehand. In order to tackle this, the priority attribute for aspect nodes was introduced. Throughout the whole SES, aspect nodes get a unique number. If, during pruning, aspect nodes become brothers, a decision as to which to choose can be found. Nodes with higher priority numbers are priorized over those with lower numbers. In Figure 1.12, an example is given. The system a consists of b and c, if it is of type d. If it is of type e, it can consist of b and c or f and g. In case that the system is of type e, a decision as to which decomposition to take is made by the priority attribute.




```
SESVAR={VAR}
SemanticCondition={VAR in ['1', '2']}
specrule={VAR=='1'➔d;
          VAR=='2'➔e}
couplings={...}
```

Figure 1.12: Example 12 - Specialization and aspect siblings

1.13 Example Describing a Feedback Control System with Optional Feedforward Control

An example describing a feedback control system with optional feedforward control is given in this section.
A feedback control system can be modeled using transfer functions describing the behavior of the components in frequency domain. Controlled variables in a feedback control system are usually influenced by disturbances. A common approach for minimizing the influence of predictable disturbances is adding a feedforward control. How such a system can be designed and tested using the extended SES/MB (eSES/MB) infrastructure and the introduced tools is described in the following paragraphs.

1.13.1 Problem Description

A process unit with a *PT1* behavior shall be controlled using a PID controller. A disturbance with a *PT1* behavior affects the output of the process unit. There are two structure variants, either with a feedforward control or without a feedforward control. Figure 1.13 depicts a schematic representation of the application.

The system's behavior follows the *PT1* transfer function in equation 1.1 and the step-shaped disturbance affects the output of the process unit with a *PT1* behavior

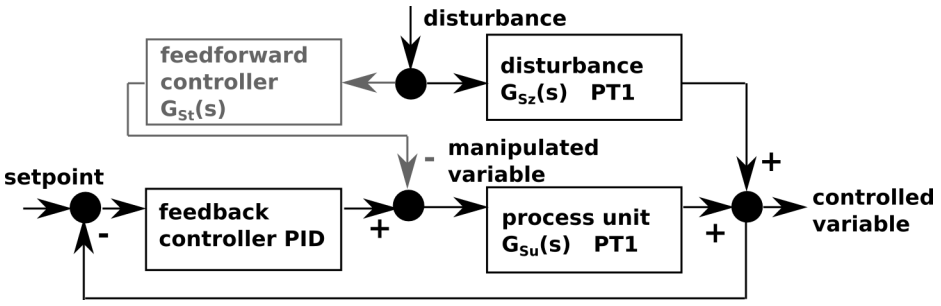


Figure 1.13: Structure of the feedback control system with optional feedforward control.

according to equation 1.2. The optional feedforward control is realized by subtracting the disturbing signal calculated by equation 1.3 from the manipulated variable.

$$G_{Su}(s) = \frac{1}{20 \cdot s + 1} \tag{1.1}$$

$$G_{Sz}(s) = \frac{1}{10 \cdot s + 1} \tag{1.2}$$

$$G_{St}(s) = \frac{G_{Sz}(s)}{G_{Su}(s)} = \frac{20 \cdot s + 1}{10 \cdot s + 1} \tag{1.3}$$

1.13.2 Variant Modeling

Figure 1.14 depicts the SES.

The subtree of the root entity node *ctrlSys* specifies the two system structures, i.e. a variant with and a variant without feedforward controller. The aspect *ctrlSysDEC* describes that each system variant consists of the entities: *sourceSys*, *feedbackSys*, *ctrlPIDSys*, *procUnitSys*, *sourceDist*, *tfDist* and *addDist*. They are mandatory system elements. The optional feedforward control is specified by the subtree of entity *feedforwardCtrl*.

Optional parts in an SES are expressed by a specialization node where one of its children is a NONE element (see example before). A NONE element means that the entity is not included at all. The selection at a specialization is defined by the specrule. The specrule of the specialization *feedforwardCtrlSpec* defines that either the entity *fc* or *NONE* is selected during pruning. The result of evaluating

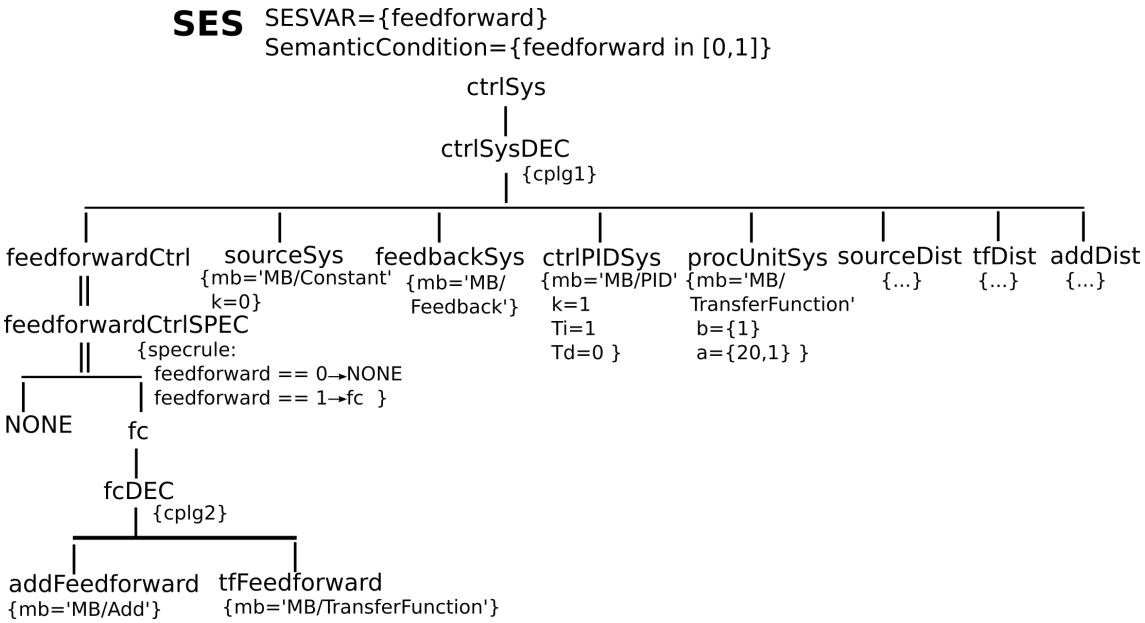


Figure 1.14: SES specifying the feedback control system.

the specrule at node *feedforwardCtrlSPEC* depends on the value of the SESvar *feedforward*. The SESvar codes the two possible structure variants as values 1 or 0. Therefore, the semantic condition $feedforward \in [0, 1]$ applies to the SESvar. The entity *fc* and its subsequent aspect *fcDEC* specifies the feedforward control structure as a composition of the two entities *tfFeedforward* and *addFeedforward*.

The coupling relations are abbreviated with *cplg* in Figure 1.14. Due to the varying system structures specified in the SES, the couplings in attribute *cplg1* of aspect node *ctrlSysDEC* are defined using an SESfcn. The coupling definitions in *cplg2* at node *fcDEC* are invariable and can therefore be defined without using anSESfcn. The coupling definitions are not listed separately since the SES is given as example.

Each leaf node defines an mb-attribute referring to a basic model in an Model Base (MB). The other attributes of the leaf nodes define properties to configure the linked basic models.

1.13.3 PES and FPES of the Feedback Control System

As described before, two different structure variants can be derived as PES. These are presented in Figures 1.15 and 1.16.

Figures 1.17 and 1.18 show the structure variants of both FPES.

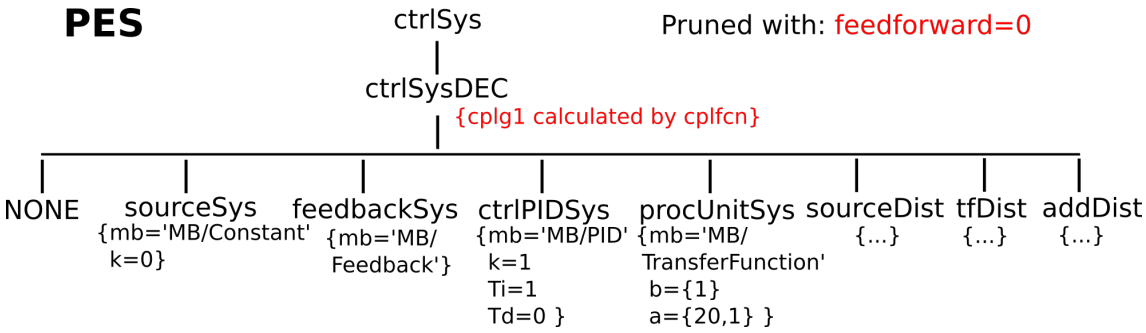


Figure 1.15: PES of the feedback control system (without feedforward control).

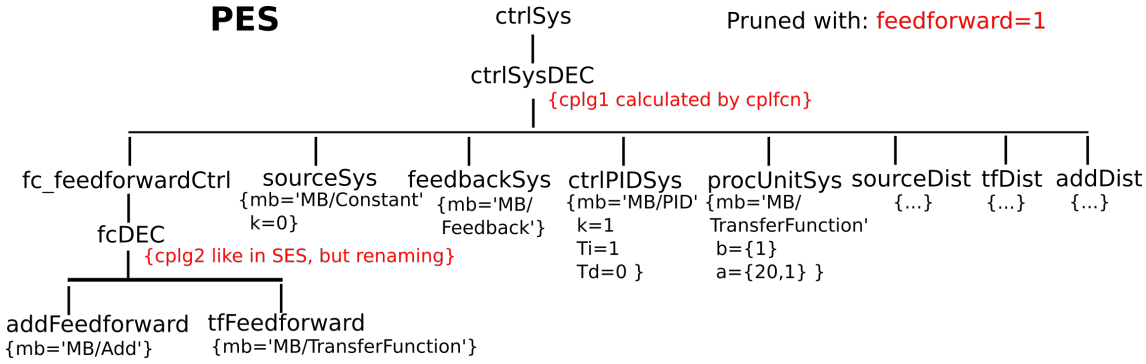


Figure 1.16: PES of the feedback control system (with feedforward control).

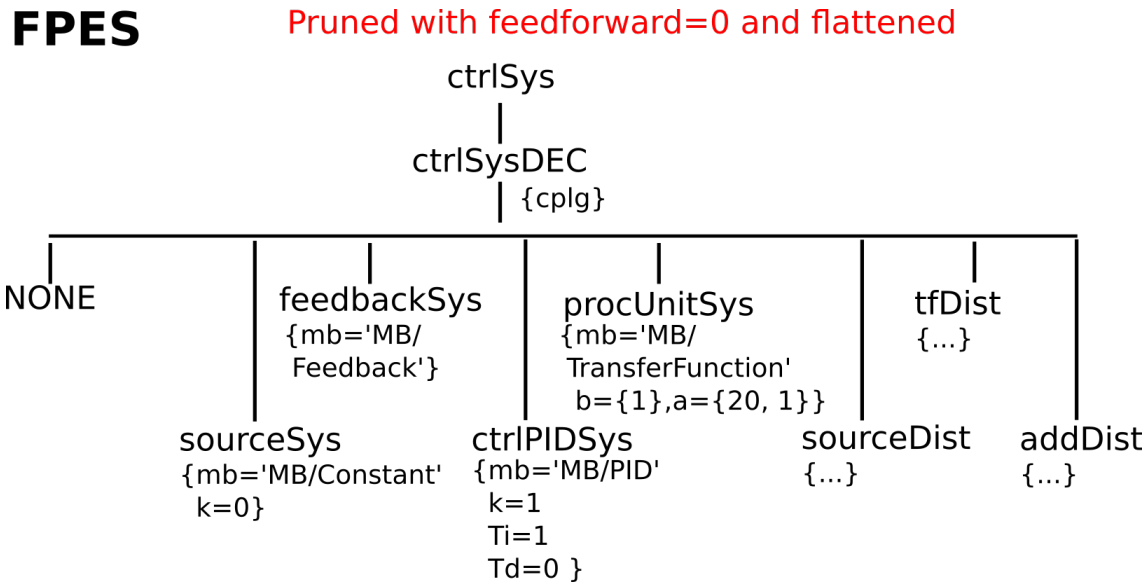


Figure 1.17: FPES of the feedback control system (without feedforward control).

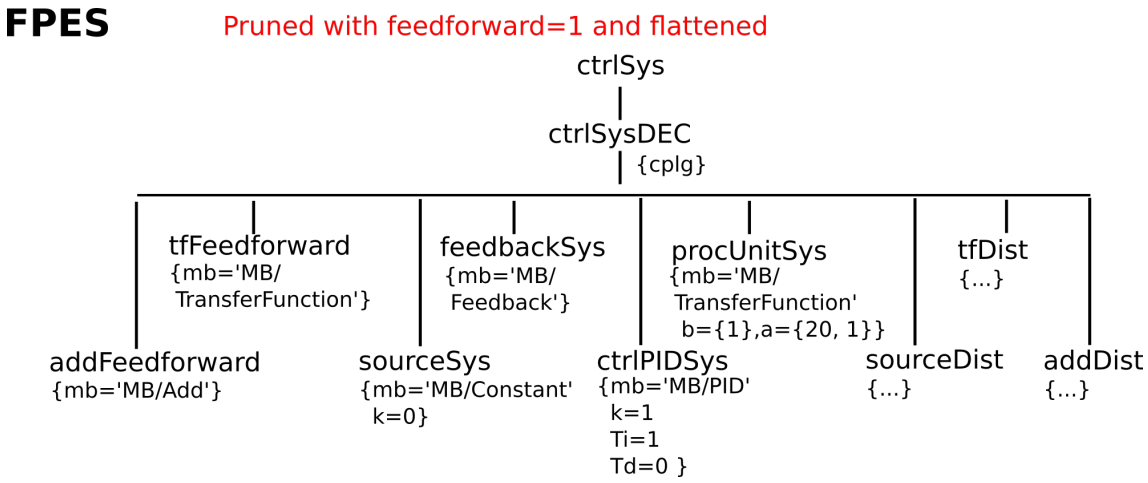


Figure 1.18: FPES of the feedback control system (with feedforward control).