Assignment 8 SPL

Ivo Melse s1088677 & Madelief Slaats s1032615

November 2024

Task 1: Analysis of feature models and configurations

a) How can one determine if a given feature selection (configuration) is valid with regard to this feature model? Are the following to configurations valid?

We can determine whether a given feature selection is valid with regard to this feature model by ensuring that it follows all constraints of the feature model. This involves checking dependencies, mutual exclusions, mandatory features and required parent features for each selected feature.

We have the following configurations:
C1 = {DataStructures, Algorithms, Structures, Array, Tree, Visualisation, Simulation}
C2 = {DataStructures, Algorithms, QuickSort, LinearSearch, Structures, Array, Visualisation, Simulation}

C1 is not valid as we have Algorithms, but neither Quicksort or LinearSearch. As the 'or'-relation states that at least one of them should be present, C1 cannot be valid. C2 is valid; it has all mandatory features and the Algorithms 'or' is correct. Also, the implication of "QuickSort $\vee$ LinearSearch $\Rightarrow$ Array" is satisfied.

b) Is the feature model consistent? How can one use a SAT solver to answer this question?

A feature model is consistent if there is at least one valid configuration that satisfies all constraints and as we have seen in a) that C2 is a valid configuration, the feature model is consistent.

We can use a SAT solver by encoding all the feature model constraints as propositional logic formulas and give it to the SAT solver. If the SAT solver finds a solution, the model is consistent.

c) Does the feature model contain any dead features? Which features always have to be activated? How can one use a SAT solver to answer both questions?

Dead features are features that cannot be selected in any valid configuration. This feature model does not have any dead features, because there is a valid configuration possible for all features.

Features that always have to be activated are mandatory features. This feature model has mandatory features: Visualisation and Simulation. As Data-Structures is at the top, it is also a mandatory feature.

A SAT solver can identify dead features by checking for configurations that include each feature; if a feature cannot appear in any configuration, it is dead. Similarly, a feature is mandatory if it appears in all possible configurations that satisfy the constraints.

Task 2: Comparison of feature models

a) Which categories of changes of feature models exist? What does each category mean for the set of products arising from the feature model?

There are four categories:

1. **Refactoring**: No products are deleted and no products are added.
2. **Generalization**: No products are deleted, but there are products that are added.
3. **Specialization**: No products are added, but there are products that are deleted.
4. **Arbitrary Edit**: Products are added and products are deleted.

b) Assume that the feature model from task 1 is changed so that the following feature model shows the result of the change. To what category does the change belong?

The change belongs to the Refactoring category, there are a few changes, but those do not affect the products that can be constructed. As can be seen as follows:

- The Algorithm ‘branch’ stays the same, thus there is no change there.
- The Structure ‘branch’ does change, but the products that can be constructed are the same with respect to the features in this branch:

  - In the feature model from task 1, we have the following possible combinations:
    {-}, {Structures}, {Structures, Array}, {Structures, Tree}, {Structures, Array, Tree}
  - In the feature model from task 2, Array and Tree, do not depend on Structure anymore in the graph of the model, but there is an added constraint, namely: "Array $\vee$ Tree $\Rightarrow$ Structures". This constraint adds that dependency back in. We have the following possible combinations:
    {-}, {Structures}, {Array, Structures}, {Tree, Structures}, {Array, Tree, Structures}

  We can see that these are the same sets.

- The Visualisation 'branch' also changes. There is an extra feature Console. At first it seems that this 'xor'-relation adds the combination {Visualisation, Console} to the products. However, there is also a constraint involving Console, namely: "Console $\Rightarrow$ Simulation", which means that if Console is Selected, then also Simulation has to be selected. As, Console and Simulation are in a 'xor'-relation, this is not possible. Thus, Console can never be selected and therefore the combinations that are constructed by the Visualisation branch stay the same.

c) Give an example for a generalization of the feature model from task 2b.

You could remove the "Console $\Rightarrow$ Simulation" constraint. The set of products is then extended to also include all configurations with the combination 'Visualisation, Console' except from only the option 'Visualisation, Simulation'.

Task 3: Analysis of code

a) How can one determine if a preprocessor product line contains any dead code or unnecessary annotations?

Dead code in a preprocessor product line is code that is never included in any valid configuration and unnecessary annotations are annotations without a function. To determine if a preprocessor product line contains any dead code or unnecessary annotations, one could analyse all configurations to identify code sections that remain unused across all configurations.

b) How can one use the feature model during this analysis?

The feature model helps determine which code sections are relevant for each configuration, allowing for dead code identification by analysing feature dependencies and constraints.

c) Which of these problems occur in the following code excerpt, which is based on the feature model from task 2b?

We determine the presence condition for each relevant line of code. We also add a column for "Presence correct if". By this we mean that the line should be included iff this condition holds (according to the feature model).

Please include line numbers for this assignment next year to make our life easier!

Line	Presence conditions	Presence correct if
Declare array	Array	Array
Print array	Array	Array
Print Structures	Array $\wedge$ Structures	Structures
Print QuickSort array	QuickSort	QuickSort $\wedge$ Array
Declare sort	QuickSort	QuickSort
Print statistics (within sort)	Simulation $\wedge$ QuickSort	Simulation $\wedge$ QuickSort

There are two discrepancies.

1. The Print Structures line. It should always be printed if Structures is enabled, however it only works if Array is also enabled. There is no condition that Structures implies Array in the feature model. This also means that the annotations for Structures are unnecessary here.

2. The Print QuickSort array line. It makes no sense to execute this line if Array is not enabled, but this is actually possible in the program. However, according to the feature model, Quicksort implies Array. This means that there are no problems here.