Identifying and Resolving Feature Model Inconsistencies based on DL Reasoning

Project Proposal
CS6795 – Semantic Web Techniques

Esraa Al-Wakel, Alireza Ensan, Mahdi Noorian
{q5v4t, q17k8, j41z9}@unb.ca

Instructor: Dr. Harold Boley

8 November 2010
1. Introduction

The software product line engineering paradigm is amongst the widely used means for capturing and handling the commonalities and variabilities of the many applications of a target domain. Software product lines are often graphically represented through a hierarchical tree structure known as feature models. The main concern in feature model construction is to ensure the defined constraints in feature model can be satisfied at the same time. Identifying and resolving inconsistency in feature model are significant tasks. In this project, we propose a framework to find and fix the inconsistency in feature model based on Description Logic (DL) reasoning. The basic idea of our approach is to first transform feature model, which can be represented in XML format, into DL. The second step is to identify the inconsistency based on reasoning on DL representation of the feature model and then recommend a solution to domain analyst for resolving the corresponding inconsistency.

2. Objectives

The project aims at developing a framework for identifying and resolving inconsistencies in feature model. The main objectives of this research project are as follow:

1. Understanding the correspondences and synergies between feature model and ontology.
2. Formulating and designing transformation rules for translating feature model elements into OWL-DL.
3. Creating methods and using related tools for identifying the inconsistencies in feature model.
4. Developing a strategy for resolving inconsistencies in feature model based on domain analyst satisfaction.

3. Preliminaries

3.1 Feature Model

Features are important distinguishing aspects, qualities, or characteristics of a family of systems [1]. They are widely used for depicting the shared structure and behavior of a set of similar systems. To form a product family, all the features of a set of similar/related systems are composed into a feature model. A feature model represents possible configuration space of all the products of a system product family in terms of its features. Feature models can be represented both formally and graphically; however, the graphical notation depicted through a tree-like structure is more favored due to its visual appeal and easier understanding.

In a feature model, features are hierarchically organized and can be typically classified as: 1) Mandatory: a feature must be included in the description of its parent feature; 2) Optional: a feature may or may not be included in its parent description given the situation; 3) Alternative
**feature group**: one and only one of features from the feature group can be included in the parent description; 4) **Or feature group**: one or more features from a feature group can be included in the description of the parent feature. In some case, the tree structure of feature models falls short at fully representing the complete set of mutual interdependencies of features; thus, additional constraints are often added to feature models and are referred to as **Integrity Constraints**. The two most widely used integrity constraints are: **Includes** – the presence of a given feature (set of features) requires the inclusion of another feature (set of features); and **Excludes** – the presence of a given (set of) feature(s) requires the elimination of another (set of) feature. Moreover, cardinality based feature models (an extension of feature models) define **feature cardinality** and **feature group cardinality**. The former shows the number of instances of a feature in the final products, and the latter shows the minimum and maximum number of sub-features within the grouped feature that can be chosen for the final product.

**Inconsistent Feature Model**

As it mentioned before, to generate a feature model, some constraints like structural constraints, integrity constraints, and cardinality constraints should be considered. When one or more than one contradiction among constraints happens, the feature model would become inconsistent. Inconsistent Feature Model (IFM) is a model that has contradictions among features at the same time. Some approaches exist to check consistency in the feature model. The lack of approaches to solve the inconsistency in a feature model is the motivation if this project.

![Figure 1 Typical Feature Model](image-url)
2.2 Description Logic

Description Logic (DL) is a prominent Knowledge Representation (KR) formalism [2]. It can model concepts, roles (properties), and individuals in a domain. DL can provide reasoning in the domain that highlights itself as strong knowledge representation formalism.

Two main parts of DL representation are T-Box and A-Box. T-Box (Terminology Box) denotes to the description of concepts in the domain and their hierarchy and relations. A-Box (Assertion Box) refers to the relation between individuals and concepts. The key reasoning tasks in the knowledge base are checking satisfiability, instance checking, concept satisfianility, retrieval, subsumption, and equivalency. All of these tasks in a tableau base reasoning algorithm can be reduced to just satisfiability checking of the knowledge base.

4. An Overview of our Approach

To address the challenges that are expressed in objective part; our main goal is to develop a framework to identifying and resolving inconsistencies in the feature model based on description logic reasoning. The general architecture of the framework is illustrated in Figure 2.

The proposed framework consists of following steps:

1. In the first step we will convert the feature model’s topology and constraints into the SXFM format, which is standard XML format of feature model.
2. Then in “Transformation Phase”, by using provided XSLT file we transform SXFM file into OWL-DL. Indeed providing XSLT for transforming SXFM into OWL-DL is a significant task in our project.
3. Now the OWL-DL can be imported to the reasoning engine in order to identify inconsistencies.
4. The last step is devoted to inconsistencies resolving. In this step the domain analyst fix the inconsistent constraints based on prior knowledge that he has about constraints preference.
5. Project Tools

We will use the following tools in our implementation:

2. **Swoop**: is tool for creating, editing, and debugging semantic web ontology, [http://code.google.com/p/swoop/](http://code.google.com/p/swoop/).
References


