Attributed Feature Models in Clafer

Kacper Bąk[1]

Generative Software Development Lab, University of Waterloo, Canada,
kbak@gsd.uwaterloo.ca

Abstract. tutorial; explains how to read and specify feature models in Clafer.

1 What Is Clafer?

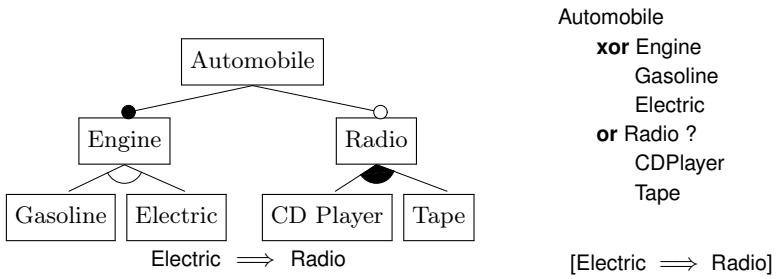
Clafer [3] is a textual language for concept modeling and specification of software product lines. In this document we are concerned with the latter. In particular we show how to encode feature models with attributes. Those models are expressive enough to represent real-world variability models, such as the Linux kernel and eCos.

Clafer aims at providing a common infrastructure for analyses of feature and meta-models. The need for analyses was recognized, for example, in the operating systems domain by the Linux Kernel and eCos developers [2]. They both provide variability models of system kernels that are supported by configuration tools. The tools guide the configuration process, so that end-users are less likely to build incorrect kernels. Many non-trivial analyses of variability models are reducible to the NP-hard problem of finding model instances by combinatorial solvers. At present, Clafer translates input models to Alloy [4]. Alloy uses SAT solvers to do model checking, or to find model instances.

2 Feature Models in Clafer

Feature models [5] are tree-like structures that specify commonalities and variabilities within a software family. Figure 1a shows a feature model of variabilities found in an automobile product line.

Each box represents a *feature*, that is a user-relevant product characteristic, such as Engine or Radio. Mandatory features are marked with filled circles; optional features with hollow circles. In our example, every car must have an Engine, but not every car must have a Radio. Feature models also have *alternative groups* (marked with empty arcs) and *or-groups* (marked with filled arcs). An *alternative group* allows to select only one subfeature, e.g. a car must have either Gasoline or Electric engine. An *or-group* requires at least one subfeature to be selected, e.g. a valid car configuration has either CD Player, or Tape player, or both of them. Finally, some constraints cannot be encoded in the tree structure. In that case we use cross-tree constraints specified as propositional formulas to



a) FODA notation b) Clafer notation

Fig. 1. Feature model of automobile product line

restrict the feature model. For instance, the constraint below the tree says that cars with Electric engine must have some Radio player on board.

Figure 1b shows a Clafer model that corresponds to the automobile feature model from Fig. 1a. The hierarchy of model elements is established by code indentation. Feature group (*alternative* and *or*) specification precedes the name (e.g. xor before Engine). In contrast with FODA feature models, each feature in Clafer may have specified group cardinality that restricts the number of subfeatures that can be selected. For instance, for Engine exactly one of its subfeatures can be selected; for Radio at least one. Optionality of elements is marked by the question mark following the name (e.g. Radio).

Cross tree constraints are specified in square brackets in Fig. 1b. FODA feature models make use only of propositional constraints. Those constraints involve feature names and a standard set of logical operators, i.e. equivalence (<=>), exclusive-or (xor), implication (=>), conjunction (&&), disjunction (||), negation (∼).

3 Feature Models with Attributes

Several extensions to feature models have been proposed; among them feature *attributes* [1]. An attribute allows feature to store a value of primitive type (such as integer, string, enumeration). Let us extend the automobile example with two attributed features: MaxSpeed and Model (see Fig. 2). The former specifies car's maximum speed in km/h, the latter car model name.

In Clafer, feature attributes follow feature name. The name is followed by colon, and then by type of the attribute. Therefore, MaxSpeed is an integer number, and Model a textual string. Furthermore, both features are constrained (in the last two lines), so that values of attributes are well-defined. We set car's maximum speed to 240, and model name to "Supercar". Please not that both equations are specified within the same square brackets. Such a specification is interpreted simply as conjunction of logical expressions. We could have defined the two equations in separate sets of brackets.

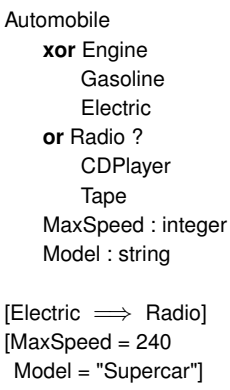


Fig. 2. Attributed feature model of automobile product line

Integer and string attributes can participate in relational expressions that compare values: equality (=), inequality (/=), less than (<), less than or equal (<=), greater than (>), greater than or equal (>=). Integers also participate in arithmetic expressions: addition (+), substraction (-), and multiplication ($*$).

References

1. Benavides, D., Trinidad, P., Ruiz-Cortés, A.: Automated reasoning on feature models. In: CAiSE'05 (2005)
2. Berger, T., She, S., Lotufo, R., Wąsowski, A., Czarnecki, K.: Variability modeling in the real: a perspective from the operating systems domain. In: ASE'10 (2010)
3. Bąk, K., Czarnecki, K., Wąsowski, A.: Feature and meta-models in Clafer: mixed, specialized, and coupled. SLE'10 (2010)
4. Jackson, D.: Software Abstractions: Logic, Language, and Analysis. The MIT Press (2006)
5. Kang, K., Cohen, S., Hess, J., Nowak, W., Peterson, S.: Feature-oriented domain analysis (FODA) feasibility study. Tech. Rep. CMU/SEI-90-TR-21, CMU (1990)