Combination of fuzzy sets with the Object Constraint Language (OCL)

Dagi Troegner
Institute of Systems Engineering, Department of Simulation,
Leibniz Universität, Welfengarten 1, 30167 Hannover
Dagi.Troegner@dlr.de

Abstract: In domain-specific modeling languages a variety of constraints originating from different resources have to be considered and integrated into a verification process to ensure a model's correctness. The basis of common model checking methods are constraints derived from meta-model specifications. Besides the meta-model, these constraints often derive from a domain's knowledge. In principle, the heuristic rules and data formulated by domain experts are not precise. To cope with vagueness in the domain knowledge, the concept of fuzzy sets is used. This work therefore describes an approach to integrate fuzzy sets and constraints expressed in the Object Constraint Language (OCL) in a combined constraint reasoning process. The approach is exemplified by introducing a domain-specific modeling language (DSL) for the scope of arrival management.

1 Introduction

A verification process for domain-specific modeling languages has to consider a variety of constraints that may originate from different resources. The basis of common model verification is constraints derived from the meta-model specifications. Correctness of data types of attributes, inheritance or association cardinalities in models are examples of these constraints, which can be added to a meta-model specification in terms of declarative languages (e.g. OCL, Prolog). Furthermore, modeling languages reflect the knowledge of their domain. A domain's knowledge is, besides common basic information (e.g. specifications, legal requirements), based on the experiential knowledge of the domain's experts (e.g. heuristic data and procedures, preferences of application users). The constraints derived from this knowledge can be referred to as domain constraints. In complex domains they can improve the correctness and reasonability of model states in large solution spaces as they enhance the available knowledge in the constraint base. A combination of OCL with fuzzy sets therefore contributes to an enrichment of model checking processes by adding further requirements representing a domain's knowledge and indicating a correct model. In [KPP06] an alignment of OCL with domain-specific languages is presented. This approach is adopted in this paper and enriched by a concept to transform the elements of a DSL to according class concepts of an object-oriented programming language. The set of constraints that is applicable in the verification process is considered as a constraint hierarchy, which labels each constraint as either required or preferred at different strengths. A constraint satisfaction algorithm for the hierarchical constraints is formulated according to an approach presented in [BFW92].

2 Motivating Example

The domain-specific modeling language AMAN-ML (Arrival Management Modeling Language) has been developed with the purpose of designing and implementing an arrival management system that is adaptable to airport-specific requirements and the heuristic knowledge of air traffic controllers. One of the components of AMAN-ML is the *RunwayConfiguration*-model that is based on the layout of an airport's runway system. In addition to the runway layout the model enables a visualization of the dependence-relationships between runways (parallel-, crossing- or V-layout) and resulting separation requirements for aircraft operations. Figure 1 shows an exemplary model for the airport of Hamburg-Fuhlsbüttel in Germany.

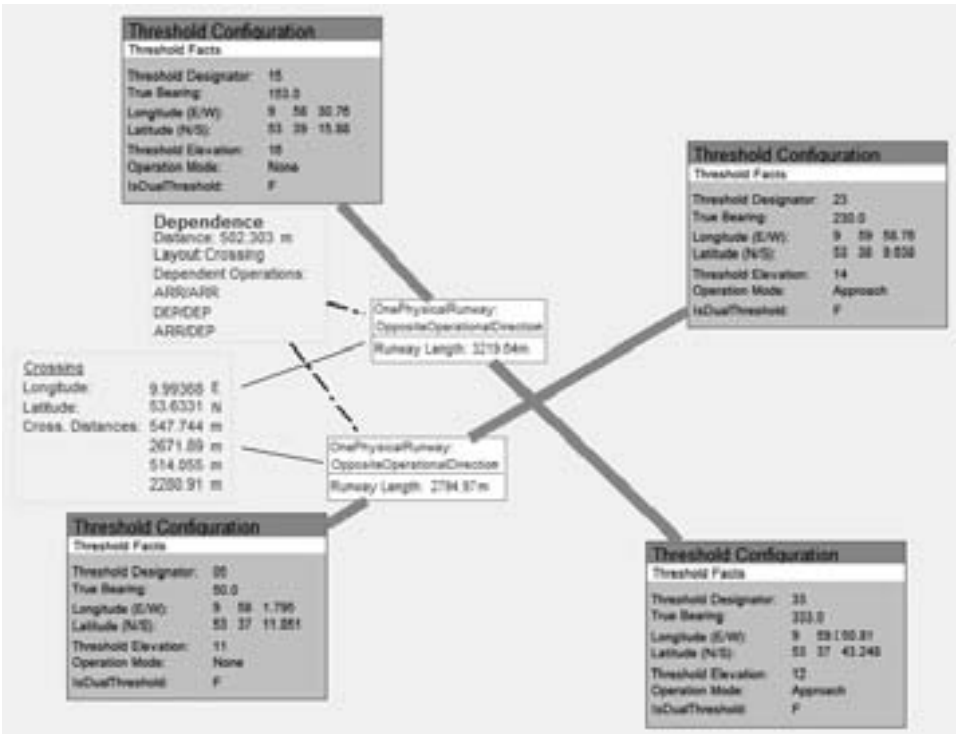

Figure 1: *RunwayConfiguration*-model for Hamburg-Fuhlsbüttel, Germany

The basic elements in the *RunwayConfiguration*-model for Hamburg-Fuhlsbüttel are the runway thresholds containing runway designator, bearing, geographic coordinates and operation mode (arrival, departure or multi mode) as attributes. The created runway thresholds can be related to each other using different relationship types to enhance the level of information for the modeler and consequently for the subsequent code generation. Two opposing thresholds of one runway can be related by a relationship of type 'OnePhysicalRunway' in the category 'OppositeOperationalDirections'. This relationship has an attribute specifying the length of a runway, calculated as the geographical distance between the paired thresholds.

The two runways in Figure 1 (15/33 and 05/23) have intersecting centerlines, which can be specified by a relationship of type 'Crossing'. The 'Crossing'-relationship provides further information regarding the crossing coordinates and the distances from the thresholds to the intersection point. A crossing implies dependencies for all operation modes for the runways involved. These dependencies are defined in a referring 'Dependence'-relationship, containing all operation mode procedures leading to separation requirements for arriving or departing aircraft.

A variety of domain-inherent constraints prescribe the correctness of a model instance. Referring to the *RunwayConfiguration*-example, e.g. legal requirements have to be considered that define the degree of dependence between runways based on their distance, potential intersection and intended operation modes (departure, approach or mixed operations). The dependence in turn has an impact on allowed operation modes and separation requirements between aircraft. Additional domain constraints are considered in form of vague concepts or heuristic procedures to enhance the correctness of the model. Particularly an *OnePhysicalRunway*-relationship can be constrained by verifying the calculated distance between the included runway thresholds against the heuristic concept of runway length. Reasonable values for the length of runways can be derived from heuristic considerations and statistics in airport development.

3 Combination of OCL and fuzzy sets

Model checking is a method for verifying the correctness of a given model state on the basis of a given specification. A variety of constraint languages can be used for this purpose as constraint language specifications differ particularly in their coverage. Therefore, different types of constraints may be necessary to represent their requirements appropriately. In this paper an approach for the integration of fuzzy constraints with meta-model based constraints is presented. Thus, model verification methods are enhanced with requirements representing descriptions heuristically and the rules of domain experts can be formulated in combination with requirements based on a model's specification.

3.1 OCL (Object Constraint Language)

The Object Constraint Language (OCL) is a declarative language for capturing constraints that can be applied to model specifications that are conform to Meta Object Facility (MOF) meta-modeling architectures. OCL provides a set of powerful facilities for navigating and querying models meeting these requirements. An alignment specification is needed for mapping the OCL concept to domain-specific modeling languages (DSL-OCL), as presented e.g. in [KPP06]. In Figure 2 an exemplary constraint is illustrated. The context specifies the correlation of a constraint to a model element. In this case, the constraint restricts relationships of type 'OnePhysicalRunway' with the category 'OppositeOperationalDirections' in a model state. The keyword *inv* implies that an invariant is applied to the given model element determining that the following condition has to be satisfied for each instantiated model element at any time.

```
[context OnePhysicalRunway.OppositeOperationalDirections inv:
 self.length >= 0 AND
 self.length =  fuzzy (RunwayLength)];
```

Figure 2: Example of fuzzy variables in OCL expressions

The body definition of the adapted OCL-constraint contains the set of conditions that have to be satisfied for the given model element. The conditions can be connected by standard Boolean operators. The resulting DSL-OCL is further adapted in this approach to the requirements of integrating fuzzy logic variables into an OCL expression. In the example DSL, AMAN-ML, attribute values can be constrained by approximate degrees of truth. This can be helpful when a values' margin can't be limited exactly, for example when determining the correspondent pair of thresholds to define a runway. The length of a runway may lie between 700m and 4000m with an average of 2918m for Germany. This can be expressed using a fuzzy variable named *RunwayLength* with a Gaussian membership function centering the average value (as can be seen in Figure 3). An integration of fuzzy logic variables into OCL expressions is enabled in the constraint model by invoking an implicit evaluation over the keyword *fuzzy* (Figure 2). A fuzzy inference system is implemented as a separated process calculating the degree of truth and returning the result to the invoking process.

3.2 Fuzzy Constraints

Fuzzy logic can be defined as a form of multi-valued logic based on the paradigm of inference under vagueness [SE10]. This vagueness may result from applying linguistic variables to express natural language concepts or from impreciseness implicated with computational methods. In this approach fuzzy constraint reasoning is used to address constraints expressing a domain's knowledge. Heuristic procedures and values for a domain are mapped to corresponding rules and facts using fuzzy sets. The model verification processes invoke constraint reasoning procedures over selected sets of constraints that may include fuzzy sets. These can be simple fuzzy constraints or OCL constraints using fuzzy variables. In case of rules based on simple fuzzy constraints, a fuzzy reasoning is initiated that follows a standard Mamdani's fuzzy inference method [Ma76]. In Figure 3, a fuzzy reasoning for the correctness of a new relationship of type *OnePhysicalRunway* is shown.

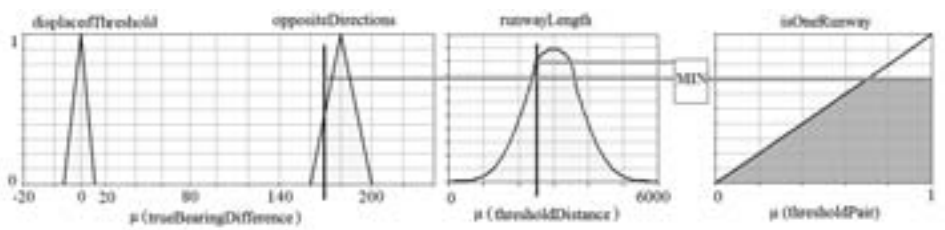

Figure 3: Fuzzy Reasoning example

The correctness of a relationship of type *OnePhysicalRunway* between two objects *Threshold* is dependent to variables that can be expressed most appropriate using fuzzy sets. A pair of thresholds is most likely to be lying on opposite directions of one runway, when their true bearings differ about 180 degrees and the calculated distance between them wouldn't exceed the interval mentioned in the section before. Therefore the *thresholdDistance* and the *trueBearingDifference* are defined as fuzzy variables. According membership functions denote their degrees of membership to the vague sets of *runwayLength* and *oppositeDirections*. The fuzzy operation resolving the AND-operator of the rule defined, determines the resulting output fuzzy set. Further aggregation methods and a defuzzification of the output set (in this case centroid) are used to identify the satisfaction of a fuzzy constraint. In the example shown, a threshold pair with 2600m distance and a bearing distance of 170° is likely to be part of one runway with a vague value of 0.75. Together with an assumption of a marginal value of 0.7 for the result variable, the object pair would be evaluated as being one physical runway. A second case is given when a constraint includes a combination of fuzzy constraints and crisp constraints (e.g. OCL). The constraint reasoning initiated in this situation applies an intermediate step mapping the included crisp variables to corresponding fuzzy variables before passing them to the fuzzy inference process. Crisp constraints can be mapped to fuzzy constraints using balk membership functions (for single values) or sharp-edged trapezoidal membership functions (for value intervals). In the example given, the condition 'self.length>=0' would be mapped to an according fuzzy variable with membership values defined as 1 for all values greater than 0.

4 Constraint Reasoning

The difference regarding constraint verification based on crisp constraints and constraints including fuzzy sets lies basically in the result of the constraint satisfaction problem. Constraint verification over a set of crisp constraints always has a return value of Boolean satisfaction. In this approach, a domain-specific constraint solver is used to cope with satisfiability in combination with vague values. The constraint solver is based on a knowledge base for the specified modeling language. In the context of AMAN-ML, an actual model state is transformed into corresponding constructs of a programming language with instantiations of their classes representing the actual elements of a model state. The relationships of a model instance are represented using self-contained classes with pointers to any objects they are bound to in the actual model instance. The constraints defined in the constraint base are directly related to the class definitions of a model and can be assigned to them through a defined context. A model verification process is invoked either by a modeler's initiation or automatically by modifications to model elements. In the first mentioned scenario all elements of the regarded model will be verified, whereas in model alterations only the elements concerned are going to be checked for maintaining the correctness. When a model verification is initiated, like the creation of a relationship, the corresponding model checking process invokes a constraint verification for the elements included. The set of objects and the relationship created are given to a verification method (e.g. *addOppositeoperationalDirection (*objectSource, *objectTarget, *relationship)*). The model checking process determines the set of applicable constraints through their defined contexts. Boolean satisfaction

results are expected to identify constraints and corresponding model elements not satisfying a given requirement. Invalid model elements are highlighted in the model and stated in an error report with a reference to the constraint violated. The constraint verification uses three sets of constraints: simple OCL, simple fuzzy and combined OCL-Fuzzy. In each set, the required constraints are labelled with degrees of strength to enable a hierarchical constraint solving. The constraints referring to the objects and their attributes are labelled with the highest degrees of strength, followed by the roles and relationships of the model with lower values. This assures the correctness of the bound elements first before verifying a relationship's correctness. Due to the interrelation of the correctness of *Dependence*-relationships to other relationship types, these element types are labelled with the lowest strength value in the constraint hierarchy. In the *RunwayConfiguration*-example, all constraints referring to the context of *Thresholds* will be verified, followed by *OnePhysicalRunway*, *Crossing* and the *Dependence* at last. For each of the three constraint sets this hierarchy is applied. The constraint sets with combined OCL-fuzzy include an implicit fuzzy reasoning process after transforming the included crisp variables to according fuzzy sets. The fuzzy result values are compared to marginal values that are predefined for each fuzzy variable and retransformed to according Boolean values, like in simple fuzzy sets. Finally, after evaluating each constraint set separately, the results are combined by standard Boolean AND-operators.

5 Conclusion

In domain-specific modeling a variety of constraints have to be taken into account representing not only meta-model specifications but also the requirements derived from a domain's knowledge. This knowledge can often be expressed by domain experts most appropriately using vague concepts. Thus, the presented constraint model in this paper enables the usage of fuzzy variables within OCL constraints to combine requirements on meta-model level with domain heuristics. A combined constraint verification process is used for model checking purposes to assist a modeler in creating correct model instances. The verification process described in this paper defines different groups of constraints (simple OCL, simple fuzzy and OCL-Fuzzy) and solves them separately using hierarchical constraint solving. For model checking purpose, basic Boolean operations are used to determine the final result over the subsets. Fuzzy result values are mapped to according Boolean satisfaction values by assigning a marginal value to each defined fuzzy variable.

References

[SE10] Stanford Encyclopedia of Philosophy, http://plato.stanford.edu/entries/logic-fuzzy

[KPP06] Kolovos, D.S.; Paig, R.F.; Polack, F.A.: Aligning OCL with DSLs to Support Instance-Level Queries. In: Journal of the Electronic Communications EASST, 2006.

[BFW92] Borning , A.; Feldman-Benson, B.; Wilson, M.: Constraint Hierarchies. In: LISP and Symbolic Computation. Kluwer Academic Publishers, p. 223-270, 1992.

[Ma76] Mamdani, E.H.: Application of fuzzy logic to approximate reasoning using linguistic synthesis. In: Proceedings of the sixth international symposium on Multiple-valued logic. IEEE Computer Society Press, p. 196 – 202, 1976.