ConEditor: Tool to Input and Maintain Constraints

Suraj Ajit[1], Derek Sleeman[1], David W. Fowler[1] and David Knott[2]

[1]Department of Computing Science, University of Aberdeen, Scotland, AB24 3UE, UK
{sajit, sleeman, dfowler}@csd.abdn.ac.uk
[2]Rolls Royce plc, Derby, UK
david.knott@rolls-royce.com

Abstract. We present a tool which helps domain experts capture and maintain constraints. The tool displays parts of an ontology (as classes, sub-classes and properties) in the form of a tree. A number of keywords and operators from a constraint language are also listed. The tool helps a user to create a constraint expression. Additionally, the tool has a facility which allows the user to input tabular data. The expressed constraints can be converted into a standard format, making them portable. It is planned to integrate this tool, ConEditor, with Designers' Workbench, a system that supports human designers.

1 Motivation

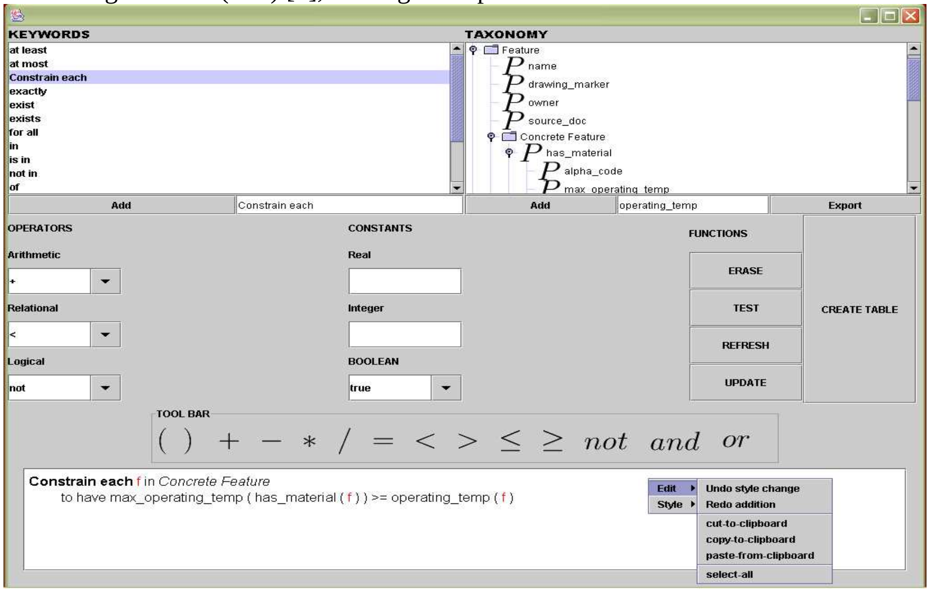
Designers in Rolls Royce, as in many large organizations, work in teams. Thus it is important when a group of designers are working on aspects of a common project that the subcomponent designed by one engineer is consistent with the overall specification, and with that designed by other members of the team. Additionally, all designs have to be consistent with the company's design rule book(s). Making sure that these various constraints are complied with is a complicated process, and so the AKT consortium has developed a Designers' Workbench [1], which seeks to support these activities. As noted above, the Designers' Workbench needs access to the various constraints, including those inherent in the company's design rule book(s). Currently, to capture this information, an engineer works with a knowledge engineer to identify the constraints, and it is then the task of the knowledge engineer to encode these, currently in Prolog, in the workbench's knowledge base. The purpose of ConEditor is to allow engineers (particularly those responsible for design standards) to capture and maintain these constraints themselves.

2. ConEditor

Here, we consider an example of a simple constraint and see how it can be input using ConEditor's GUI (Figure 1). A simple constraint expressed in Colan [2] is as follows:

Constrain each f in Concrete Feature
to have max_operating_temp (has_material (f)) >= operating_temp (f)

This constraint states that “For every instance of the class Concrete Feature, the value of the maximum operating temperature of its material must be greater than or equal to the environmental operating temperature.” ConEditor’s GUI mainly consists of five components, namely, keywords panel, taxonomy panel, central panel, tool bar and result panel. The **keywords panel** consists of a list of keywords, e.g. “Constrain each”, “in”, “to have”. The **taxonomy panel** displays a taxonomy of classes, subclasses and properties, extracted from the domain ontology (ontology used by Designers’ Workbench). The entities “Concrete Feature”, “max_operating_temp”, “has_material”, “operating_temp” are selected from this panel. Clicking the “Add” button appends the selected entity to the result panel. The **central panel** lists operators, boolean values, function buttons and textfields (for entering constants). The **tool bar** displays the operators (arithmetic, relational and logical) and delimiters. The operator ‘>=’ and the delimiters ‘(’, ‘)’ are chosen from the tool bar. The **result panel** consists of a text area, displaying the expression input by the user. These constraints input using ConEditor can then be converted into a standard format, Constraint Interchange Format (CIF) [3], making them portable.



Constrain each f in *Concrete Feature*
to have max_operating_temp (has_material (f)) >= operating_temp (f)

Fig. 1. A screenshot showing the GUI of ConEditor

References

1. D. Fowler, D. Sleeman, Gary Wills, Terry Lyon, David Knott, *Designers’ Workbench*, Departmental Technical Report, University of Aberdeen, 2004.
2. N. Bassiliades, P. Gray, *CoLan: A Functional Constraint Language and its Implementation*, Data Knowledge Engineering, 14(3):203-249, 1995.
3. P. Gray, K. Hui, A. Preece, *An Expressive Constraint Language for Semantic Web Applications* in A.Preece & D.O’Leary (eds), E-Business and the Intelligent Web: Papers from the IJCAI-01 Workshop, AAAI Press, pp. 46-53, 2001.