The PSOATransRun 1.0 System for Object-Relational Reasoning in RuleML

Gen Zou, Faculty of Computer Science, University of New Brunswick, Canada

The 6th Atlantic Workshop on Semantics and Services (AWoSS 2015)
Faculty of Computer Science, University of New Brunswick, Fredericton, NB, Canada, December 9, 2015

The relational and object-centered modeling paradigms are widely used for knowledge representation. The Web rule language Positional-Slotted, Object-Applicative (PSOA) RuleML integrates the two paradigms by permitting the application of a predicate (acting as a relation) to be [in an *oidless*/*oidful* dimension] without or with an Object IDentifier (OID) – typed by the predicate (acting as a class) – and the predicate's arguments to be [in an orthogonal dimension] *positional*, *slotted*, or *combined*.

In order to support reasoning in PSOA RuleML, we have developed the PSOATransRun 1.0 system, including a composition of our translator PSOA2Prolog, from PSOA RuleML to a subset of the logic programming language ISO Prolog, with the well-known efficient XSB Prolog engine. The system supports KBs and queries employing all major PSOA features, e.g. all kinds of psoa atoms (including relationships, shelves, pairships, and frames), head-existential rules, subclass formulas for 'ABox' reasoning, and equality in the body restricted to unification and external-function evaluation.

The Java- and ANTLR-based PSOA2Prolog translator consists of a multi-step source-to-source normalizer followed by a mapper to the pure (Horn) subset of ISO Prolog. The mapper recursively transforms normalization results to Prolog clauses. The normalizer transforms the PSOA input into an objectified, existential-free normalized form, where atoms and rules cannot be further decomposed (into conjunctions). It performs five (sequential) transformation steps, namely objectification, Skolemization, slotribution/tupribution, flattening, as well as rule splitting.

In particular, objectification refines the earlier static approach, which generates OIDs for all of the KB's oidless atoms, by a novel static/dynamic approach, which leaves unchanged as many of the KB's oidless atoms as possible, instead constructing virtual OIDs as query variable bindings. It first partitions the set of KB predicates into two disjoint subsets: non-relational (at least one occurrence in a multi-tuple, oidful, or slotted atom, or in a subclass formula) and relational (no such occurrence). It then statically generates OIDs only for the KB's oidless psoa atoms with non-relational predicates while dynamically performing OID virtualization for queries with OID variables corresponding to the KB's psoa atoms with relational predicates: Query atoms using the KB's relational predicates with OID variables are rewritten via equalities that unify an OID variable with a (Skolem-like) OID-constructor function application. This allows users to pose queries using OIDs regardless of whether the underlying KB clauses have OIDs or not, as well as making maximum use of the underlying Prolog engine for efficient inference on the KB clauses with relational predicates.

The PSOATransRun 1.0 system has been released in Java source form and as an executable jar file. It can be downloaded from http://wiki.ruleml.org/index.php/PSOA_RuleML#PSOATransRun and executed on any operating system with a Java environment and an XSB engine.