

## Position Paper on Rule Languages for Interoperability

Kurt Godden  
General Motors R&D  
[kurt.godden@gm.com](mailto:kurt.godden@gm.com)

General Motors is moving towards increasing automation of business processes and manufacturing operations under the rubric of eManufacturing. To enable eManufacturing, we will need the capability to represent and process arbitrary business rules and constraints in a wide variety of domains such as: manufacturing plant operations, contract management, process monitoring, engineering management, internal communications, and so forth.

Rules and constraints that are of interest primarily fall into those that can be represented in first-order predicate calculus. For example,

- If robot 1 is busy, send pallet to robot 2
- Hole diameter of punch must not exceed 3mm
- Piston rings will be shipped in lot sizes of 1,000

However, it is also noted that higher-order rules would also be of interest. For example, “If there are any conditions on the delivery date, then notify purchasing manager.”

Concurrent with a need to represent and process business rules, the web is emerging as the ideal computing environment in which to implement these rules due to its independence from any particular platform or operating system. In order to avoid writing custom web-based applications for each domain and rule set cited above, we need a common representation language for rules and constraints such that a general-purpose rule processor can be built and deployed in a web-based environment with API's for hooks to external systems.

When considering how to represent arbitrary rules and constraints in RDF, for example, it becomes necessary to distinguish URI's that represent names from those that represent rule variables. This is a more specific problem of the generalized problem of how to distinguish an RDF triple that represents a rule from an RDF triple that represents a claim about the domain. Once this distinction is made, then the rule processor can be invoked to interpret the rule or enforce the constraint, as opposed to invoking an inference engine to derive a conclusion.

Fortunately, RDF provides a mechanism that can be used to make just this distinction between rules/constraints and assertions. It is the vocabulary of reification that can be used for this purpose. For example, in the triple ‘var:x rdf:type gm:transmission .’ we can use reification to provide the meta-statement to the effect that the URI ‘var:x’ is a

variable, not a named entity and that the triple is not a statement that the thing called 'var:x' is a transmission, but that 'var:x' refers to other things that are transmissions.

Toward this end we have been developing an RDF language called Resource Constraint Framework (RCF) that leverages RDF's reification for the purpose of representing arbitrary rules and constraints.