and then instantiated as a *spin:rule* for the class *owl:Thing*; this will allow the rule to be applied to all possible instances.

It should be noted that SPARQL, and thus SPIN, is a rule *language*, not an implemented rule *system*; for example, CONSTRUCT only expresses one inference step (from one RDF graph to another). A rule system on top of SPARQL (e.g., a SPIN inference engine) would need, among other things, to run CONSTRUCT iteratively and to control recursion in case of recursive rules.

5.9 Nonmonotonic Rules: Motivation and Syntax

5.9.1 Informal Discussion

Now we turn our attention to nonmonotonic rule systems. So far, once the premises of a rule were proved, the rule could be applied, and its head could be derived as a conclusion. In nonmonotonic rule systems, a rule may not be applied even if all premises are known because we have to consider contrary reasoning chains. In general, the rules we consider from now on are called *defeasible* because they can be defeated by other rules. To allow conflicts between rules, *negated atomic formulas* may occur in the head and the body of rules. For example, we may write

$$p(X) \rightarrow q(X)$$

$$r(X) \rightarrow \neg q(X)$$

To distinguish between defeasible rules and standard, monotonic rules, we use a different arrow:

$$p(X) \Rightarrow q(X)$$

$$r(X) \Rightarrow \neg q(X)$$

In this example, given also the facts