$$Type(?cp, ConstraintProperty) \longleftrightarrow$$

$$(Type(?cp, ConstraintResource) \wedge Type(?cp, Property))$$

domain and range are constraint properties:

$$Type(domain, ConstraintProperty)$$

$$Type(range, ConstraintProperty)$$

domain and range define, respectively, the domain and range of a property. Recall that the domain of a property P is the set of all objects to which P applies. If the domain of P is D, then for every $P(x, y)$, $x \in D$.

$$PropVal(domain, ?p, ?d) \longrightarrow$$

$$\forall ?x \forall ?y (PropVal(?p, ?x, ?y) \longrightarrow Type(?x, ?d))$$

The range of a property P is the set of all values P can take. If the range of P is R, then for every $P(x, y)$, $y \in R$.

$PropVal(range, ?p, ?r) \longrightarrow$

$\forall ?x \forall ?y (PropVal(?p, ?x, ?y) \longrightarrow Type(?y, ?r))$

Formulas that can be inferred from the preceding ones:

$$PropVal(domain, range, Property)$$

$$PropVal(range, range, Class)$$

$$PropVal(domain, domain, Property)$$

$$PropVal(range, domain, Class)$$

Thus we have formalized the semantics of RDF and RDFS. Software equipped with this knowledge is able to draw interesting conclusions. For example, given that the range of *rents* is *ResidentialUnit*, that *ResidentialUnit* is a subclass of *Unit*, and that *rents(JeffMeyer, BaronWayApartment)*, the agent can automatically deduce *Unit(BaronWayApartment)* using the predicate logic semantics or one of the predicate logic proof systems.