LegalRuleML Metamodel

Tara Athan, Harold Boley, Guido Governatori, Monica Palmirani, Adrian Paschke, Adam Wyner

July 13, 2013

RuleML 2013

7th International Web Rule Symposium, Seattle

Contents

Purpose of Metamodel

- Expose LegalRuleML Metadata as Linked Data
- Provide partial semantics by transformation
  - LegalRuleML → RDF + RDFS (+ OWL)
- Establish connections to external ontologies
  - Dublin Core
  - FRBR
  - RDF/RDFS
  - RuleML Metamodel
- Essential Component of LegalRuleML's Language Design Process

LegalRuleML's Cyclic Language Design Process

- Legal Source Examples
- LegalRuleML Metamodel as RDFS Schema
- RDF Instances based on Metamodel
- LegalRuleML Instances from RDF/XML
- XML Schemas Validating Against Instances
- Glossary of XML Elements and Attributes
- Repeat

LegalRuleML Metamodel as RDFS Schema

- rdfs:Class
  - Names for classes of entities
    - Following RDF(S) conventions, UpperCamelCase
  - rdfs:subClassOf hierarchy
    - Connections to external ontologies
- rdf:Property
  - Names for dyadic relations between entities
    - Following RDF(S) conventions, lowerCamelCase
  - rdfs:domain, rdfs:range
  - rdfs:subPropertyOf hierarchy
    - Connections to external ontologies

LegalRuleML Metamodel as (Future) OWL Ontology

- owl:sameAs
  - Used in RDF instances
- rdfs:comment
  - Natural language definitions of classes and properties
  - Describes characteristics that are beyond RDFS expressivity
    - Property Chaining
    - To be implemented

RDF Instances based on Metamodel

- Simplified Samples Extracted from Legal Sources
- Compactification
  - Start with Unnested Triples in any RDF format
  - Nest in Tree Structure using RDF/XML abbreviations to eliminate explicit blank nodes

LegalRuleML Instances from RDF/XML

- Produced by semi-standardized invertible manual transformation
- Design Principles
  - Striping
    - Fully-striped normal form
      - Alternating Node (rdfs:Class) element and edge (rdf:Property) element
      - One child per edge (except for rdfs:Collections)
    - Compact form with redundant stripes removed (stripe-skipping)

LegalRuleML Design Principles (cont.)

- Renaming - shorter element and attribute names, still human readable
  - <Node>Collection → <Nodes>
- Node-skipping
  - Nodes always appearing as blank nodes may be skipped provided no type information is lost
- Leaf Stripes
  - Nodes that often have no content may optionally be skipped, leading to a “leaf-stripe”, provided no type information is lost

LegalRuleML Design Principles (cont.)

- Attributes versus Edges
  - Attributes can lead to more compact syntax
  - However, may inhibit extensibility
  - Only used if, with high confidence,
    - Property will never have cardinality >1
    - Object will never be a blank node
    - Literal Object always has a unique specified datatype

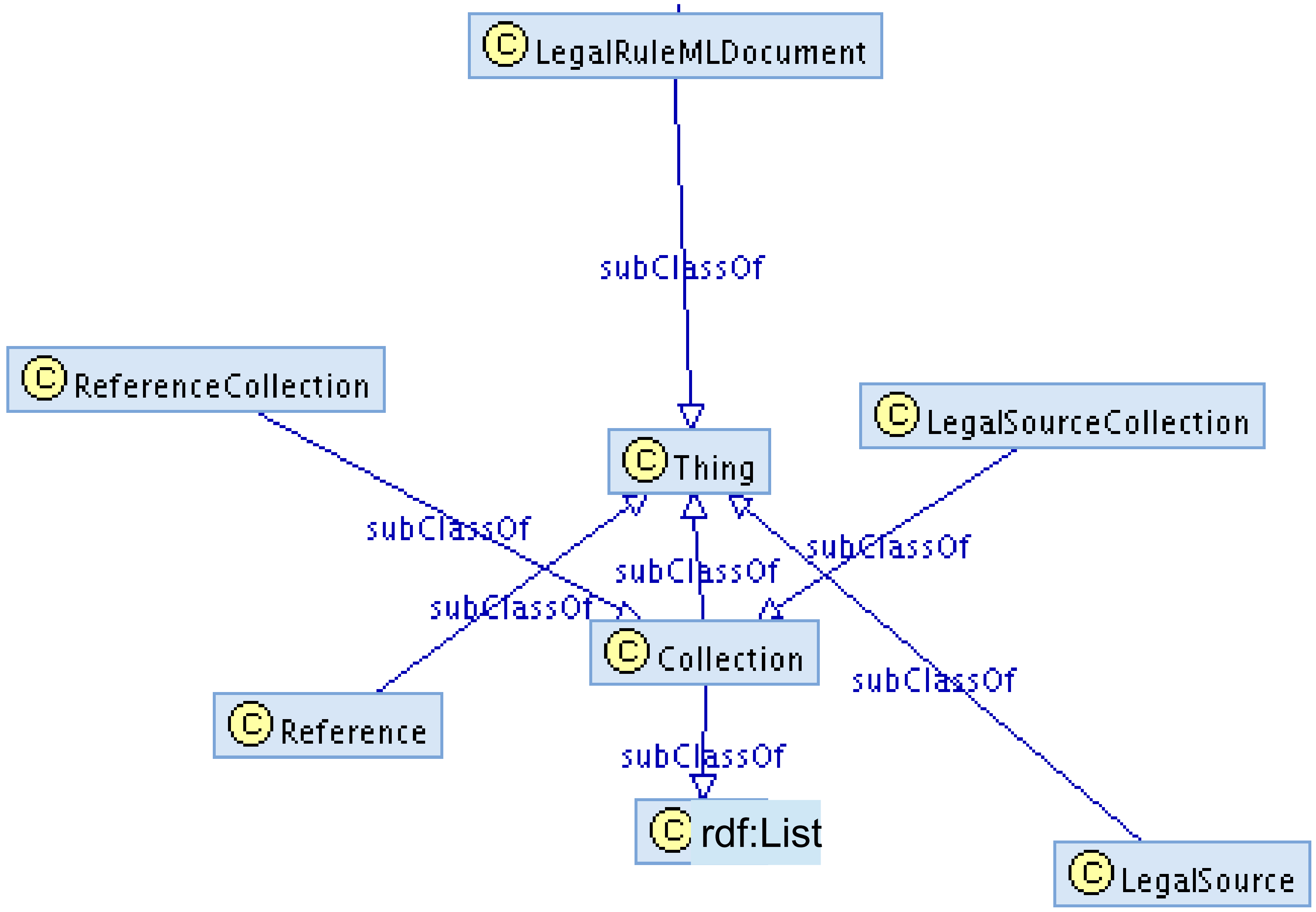
XML Schemas Validating Against Instances

- Modular Relax NG schemas
  - Customization by selection of a subset of the modules
  - Extension by including additional modules
- Generated Monolithic XSD schemas
- Schema Validation using various engines (Saxon EE, XMLSpy, …)
- Instance Validation as Requirements Testing

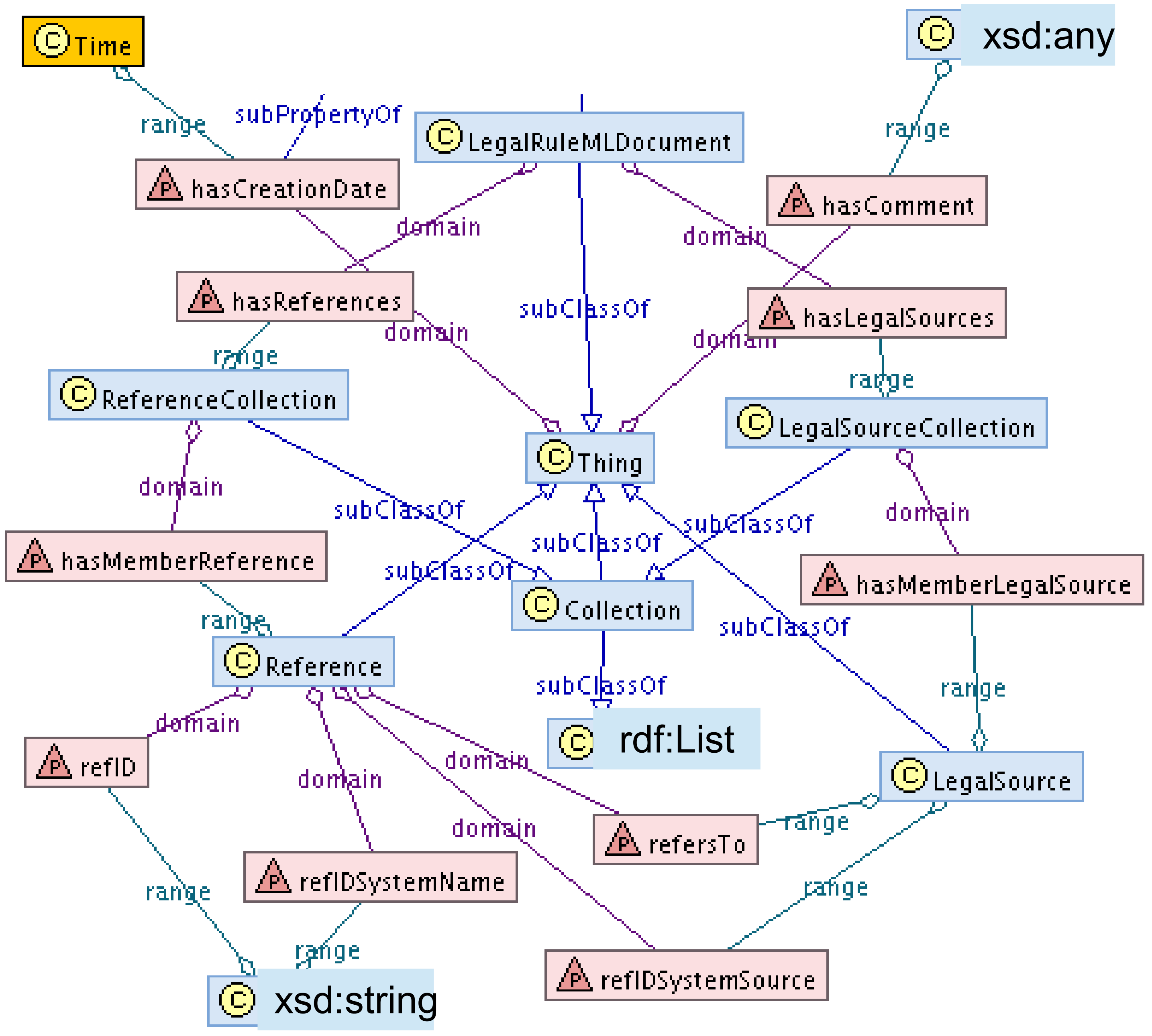
Glossary of XML Elements and Attributes

- Definitions for XML elements and attributes
- Synchronization with natural language comments in RDFS metamodel
  - Initiates update of metamodel

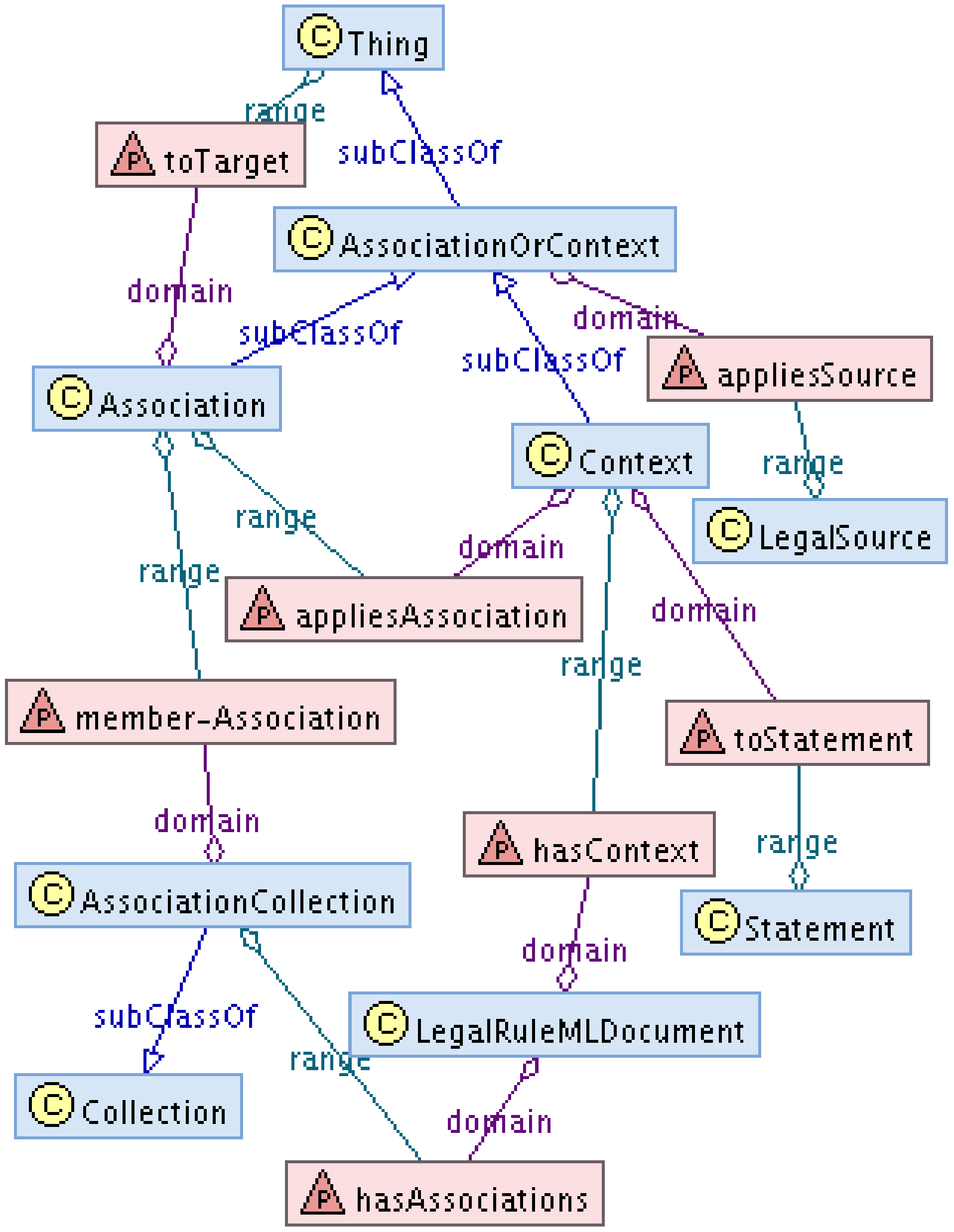
Upper Metamodel (Classes)



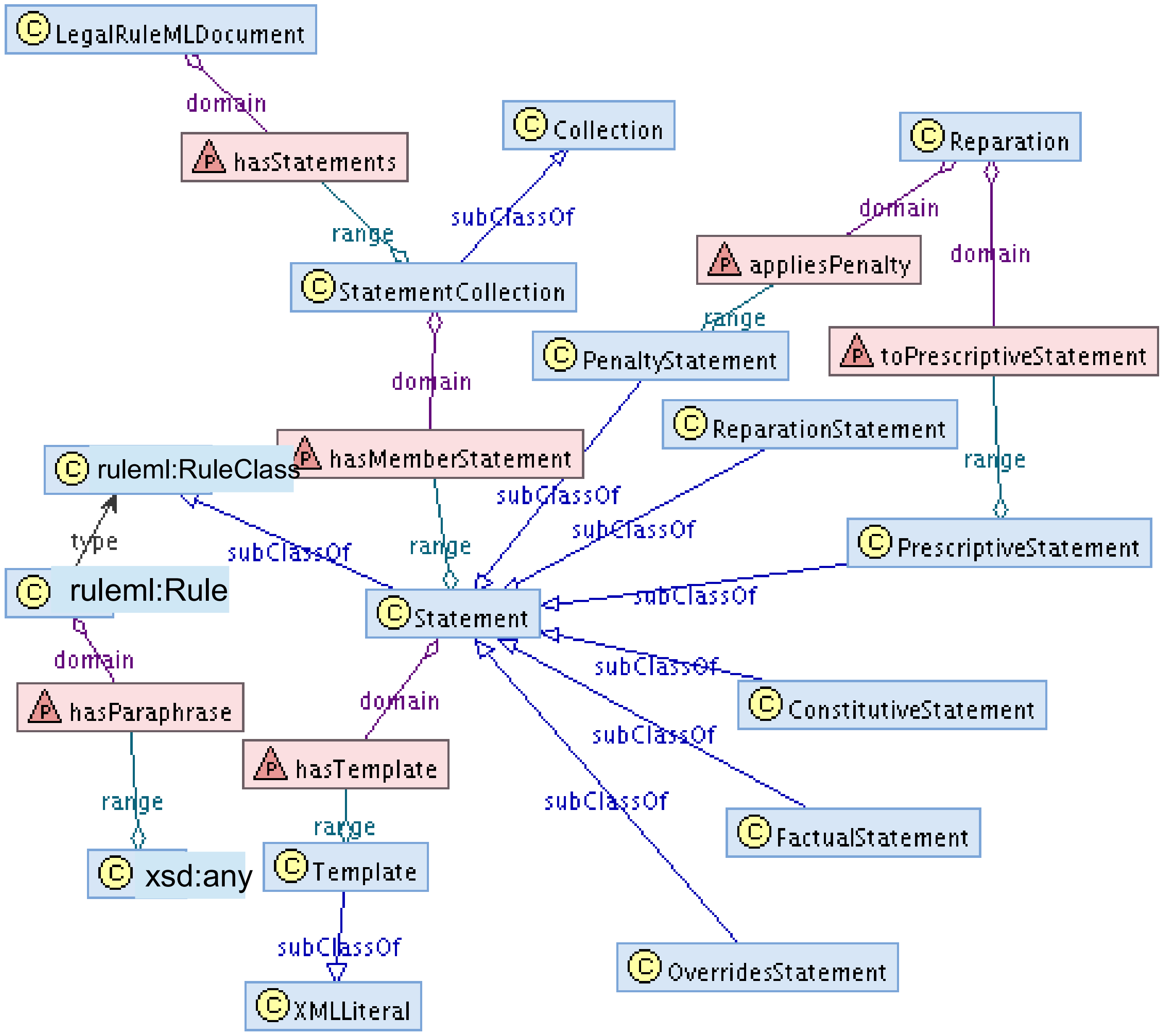
Upper Metamodel (Properties)



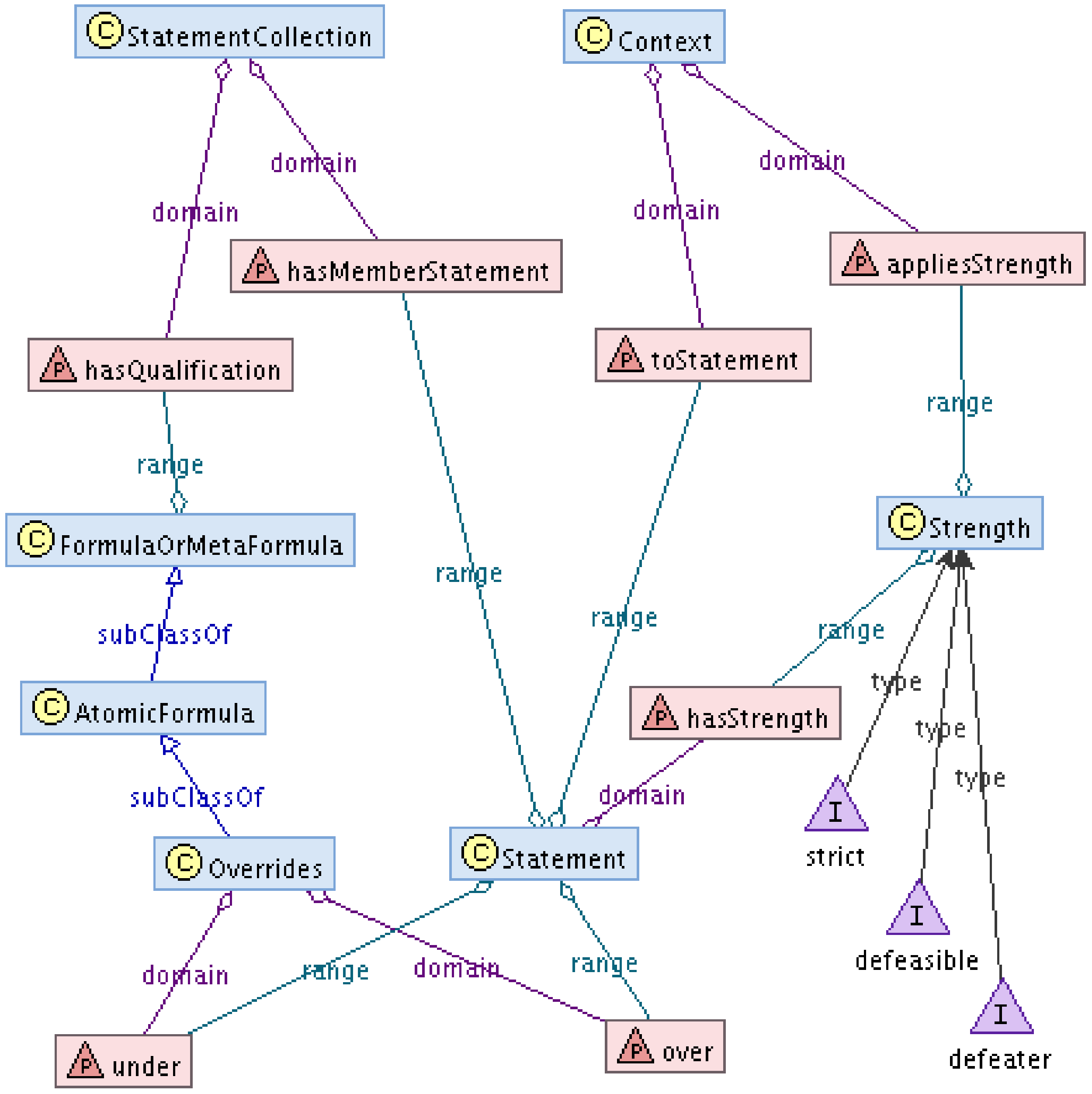
Context Metamodel



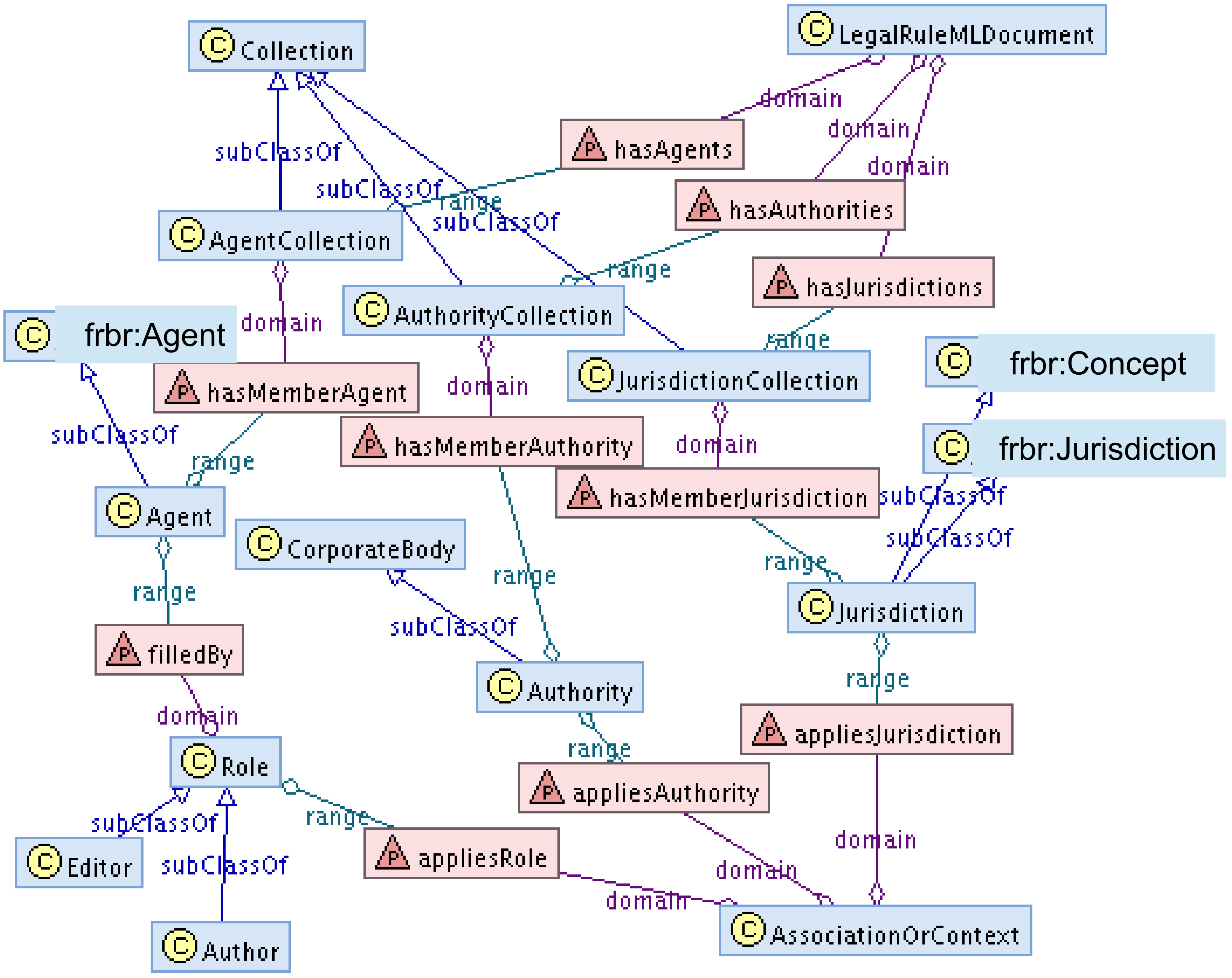
Statement Metamodel



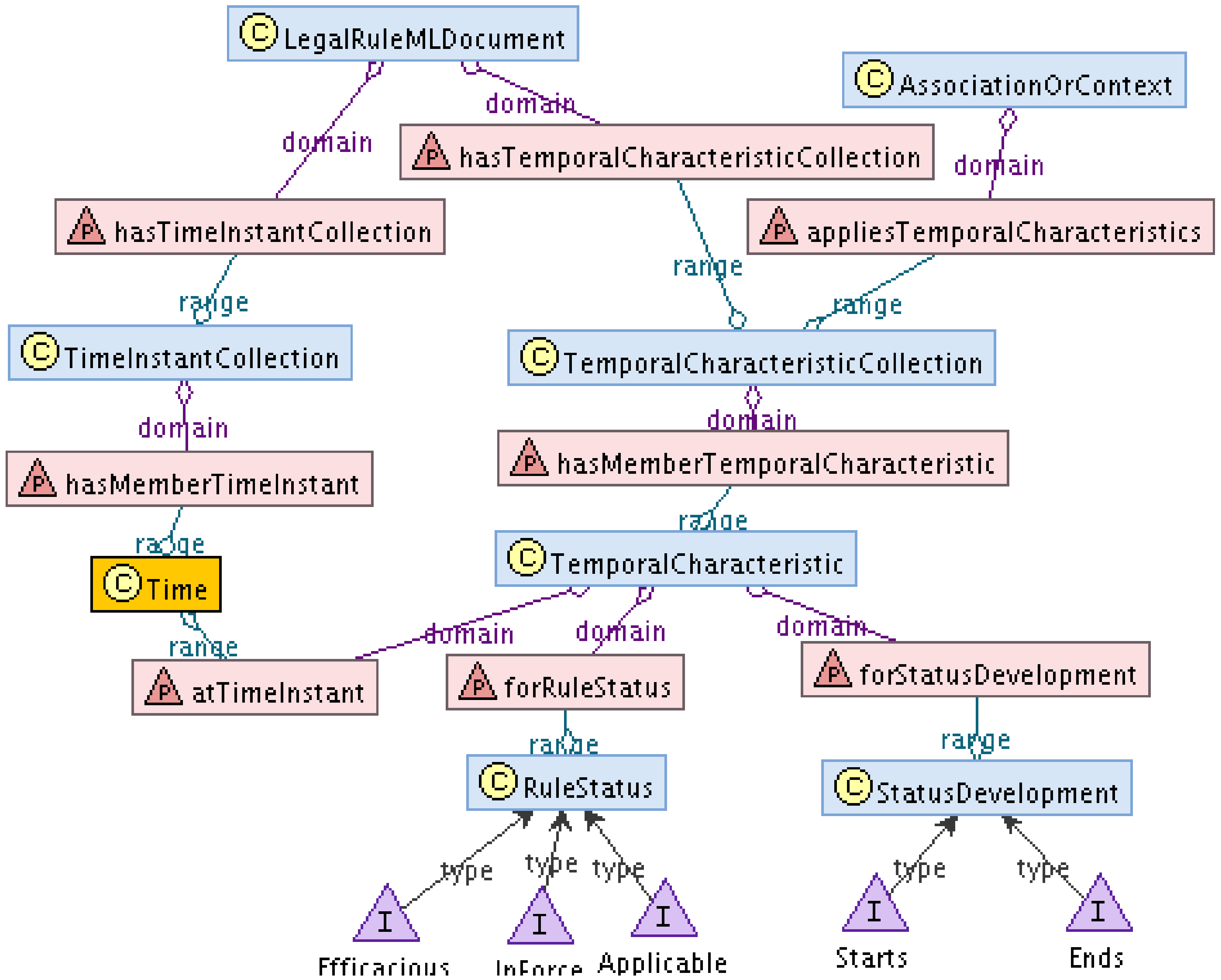
Defeasible Metamodel



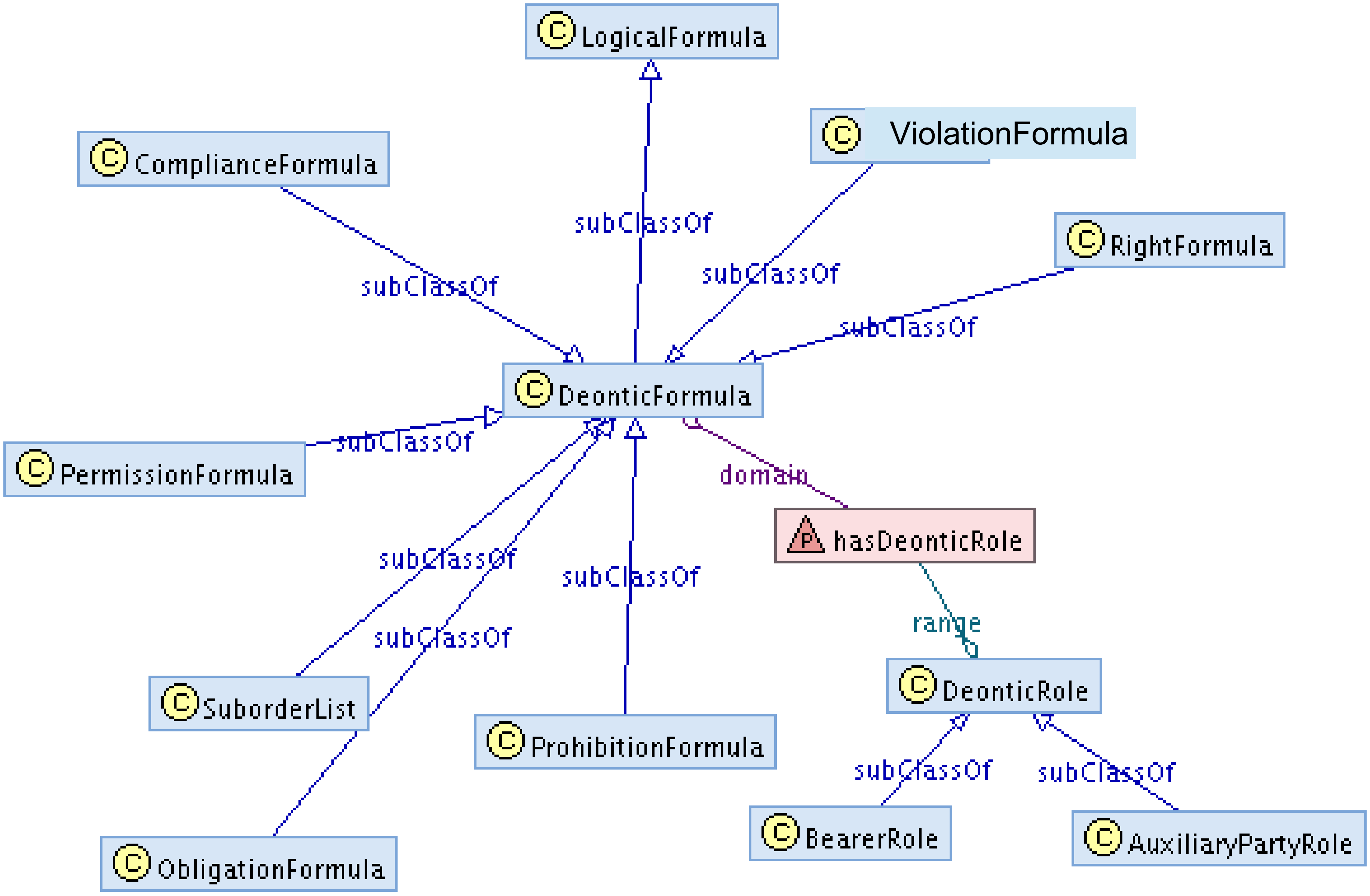
Metadata Metamodel



Legal Temporal Metamodel



Deontic Metamodel


LogicalFormula
ComplianceFormula
ViolationFormula
RightFormula
DeonticFormula
PermissionFormula
hasDeonticRole
SuborderList
ProhibitionFormula
DeonticRole
ObligationFormula
BearerRole
AuxiliaryPartyRole
subClassOf
domain
range