Annexes

Annex A. Fixed Rules for Method of Transforming an Ontology into Rules

```
(* <#eq-ref> *)
Forall ?p ?o ?s (
  ?s[owl:sameAs->?s] :- ?s[?p->?o])

(* <#eq-ref1> *)
Forall ?p ?o ?s (
  ?p[owl:sameAs->?p] :- ?s[?p->?o])

(* <#eq-ref2> *)
Forall ?p ?o ?s (
  ?o[owl:sameAs->?o] :- ?s[?p->?o])

(* <#eq-sym> *)
Forall ?x ?y (
  ?y[owl:sameAs->?x] :- ?x[owl:sameAs->?y])
```

```
(* <#eq-trans> *)
Forall ?x ?z ?y (
  ?x[owl:sameAs->?z] :- And(
     ?x[owl:sameAs->?y]
     ?y[owl:sameAs->?z]  ))

(* <#eq-rep-s> *)
Forall ?p ?o ?s ?s2 (
  ?s2[?p->?o] :- And(
     ?s[owl:sameAs->?s2]
     ?s[?p->?o]  ))

(* <#eq-rep-p> *)
Forall ?p ?o ?s ?p2 (
  ?s[?p2->?o] :- And(
     ?p[owl:sameAs->?p2]
     ?s[?p->?o]  ))

(* <#eq-rep-o> *)
Forall ?p ?o ?s ?o2 (
  ?s[?p->?o2] :- And(
     ?o[owl:sameAs->?o2]
     ?s[?p->?o]  ))

(* <#eq-diff1> *)
Forall ?x ?y (
  rif:error() :- And(
     ?x[owl:sameAs->?y]
     ?x[owl:differentFrom->?y]  ))

(* <#prp-ap-label> *)
  rdfs:label[rdf:type->owl:AnnotationProperty]

(* <#prp-ap-comment> *)
```

```
  rdfs:comment[rdf:type->owl:AnnotationProperty]

(* <#prp-ap-seeAlso> *)
  rdfs:seeAlso[rdf:type->owl:AnnotationProperty]

(* <#prp-ap-isDefinedBy> *)
  rdfs:isDefinedBy[rdf:type->owl:AnnotationProperty]

(* <#prp-ap-deprecated> *)
  owl:deprecated[rdf:type->owl:AnnotationProperty]

(* <#prp-ap-priorVersion> *)
  owl:priorVersion[rdf:type->owl:AnnotationProperty]

(* <#prp-ap-backwardCompatibleWith> *)
  owl:backwardCompatibleWith[rdf:type->owl:AnnotationProperty]

(* <#prp-ap-incompatibleWith> *)
  owl:incompatibleWith[rdf:type->owl:AnnotationProperty]

(* <#prp-dom> *)
Forall ?p ?c ?x ?y (
  ?x[rdf:type->?c] :- And(
     ?p[rdfs:domain->?c]
     ?x[?p->?y]  ))

(* <#prp-rng> *)
Forall ?p ?c ?x ?y (
  ?y[rdf:type->?c] :- And(
     ?p[rdfs:range->?c]
     ?x[?p->?y]  ))

(* <#cls-thing> *)
  owl:Thing[rdf:type->owl:Class]
```

```
(* <#cls-nothing1> *)
  owl:Nothing[rdf:type->owl:Class]

(* <#cls-nothing2> *)
Forall ?x (
  rif:error() :- ?x[rdf:type->owl:Nothing])

(* <#cax-sco> *)
Forall ?x ?c1 ?c2 (
  ?x[rdf:type->?c2] :- And(
     ?c1[rdfs:subClassOf->?c2]
     ?x[rdf:type->?c1]  ))

(* <#cax-eqc1> *)
Forall ?x ?c1 ?c2 (
  ?x[rdf:type->?c2] :- And(
     ?c1[owl:equivalentClass->?c2]
     ?x[rdf:type->?c1]  ))

(* <#cax-eqc2> *)
Forall ?x ?c1 ?c2 (
  ?x[rdf:type->?c1] :- And(
     ?c1[owl:equivalentClass->?c2]
     ?x[rdf:type->?c2]  ))

(* <#cax-dw> *)
Forall ?x ?c1 ?c2 (
  rif:error() :- And(
     ?c1[owl:disjointWith->?c2]
     ?x[rdf:type->?c1]
     ?x[rdf:type->?c2]  ))

(* <#scm-cls> *)
```

```
Forall ?c (
  ?c[rdfs:subClassOf->?c] :- ?c[rdf:type->owl:Class])

(* <#scm-cls1> *)
Forall ?c (
  ?c[owl:equivalentClass->?c] :- ?c[rdf:type->owl:Class])

(* <#scm-cls2> *)
Forall ?c (
  ?c[rdfs:subClassOf->owl:Thing] :- ?c[rdf:type->owl:Class])

(* <#scm-cls3> *)
Forall ?c (
  owl:Nothing[rdfs:subClassOf->?c] :- ?c[rdf:type->owl:Class])

(* <#scm-sco> *)
Forall ?c1 ?c2 ?c3 (
  ?c1[rdfs:subClassOf->?c3] :- And(
     ?c1[rdfs:subClassOf->?c2]
     ?c2[rdfs:subClassOf->?c3]  ))

(* <#scm-eqc1> *)
Forall ?c1 ?c2 (
  ?c1[rdfs:subClassOf->?c2] :- ?c1[owl:equivalentClass->?c2])

(* <#scm-eqc11> *)
Forall ?c1 ?c2 (
  ?c2[rdfs:subClassOf->?c1] :- ?c1[owl:equivalentClass->?c2])

(* <#scm-eqc2> *)
Forall ?c1 ?c2 (
  ?c1[owl:equivalentClass->?c2] :- And(
     ?c1[rdfs:subClassOf->?c2]
     ?c2[rdfs:subClassOf->?c1]  ))
```

```
(* <#scm-op> *)
Forall ?p (
  ?p[rdfs:subPropertyOf->?p] :- ?p[rdf:type->owl:ObjectProperty])

(* <#scm-op1> *)
Forall ?p (
  ?p[owl:equivalentProperty->?p] :- ?p[rdf:type->owl:ObjectProperty])

(* <#scm-dp> *)
Forall ?p (
  ?p[rdfs:subPropertyOf->?p] :- ?p[rdf:type->owl:DatatypeProperty])

(* <#scm-dp1> *)
Forall ?p (
  ?p[owl:equivalentProperty->?p] :- ?p[rdf:type->owl:DatatypeProperty])

(* <#scm-spo> *)
Forall ?p3 ?p2 ?p1 (
  ?p1[rdfs:subPropertyOf->?p3] :- And(
     ?p1[rdfs:subPropertyOf->?p2]
     ?p2[rdfs:subPropertyOf->?p3]  ))

(* <#scm-eqp1> *)
Forall ?p2 ?p1 (
  ?p1[rdfs:subPropertyOf->?p2] :- ?p1[owl:equivalentProperty->?p2])

(* <#scm-eqp11> *)
Forall ?p2 ?p1 (
  ?p2[rdfs:subPropertyOf->?p1] :- ?p1[owl:equivalentProperty->?p2])

(* <#scm-eqp2> *)
Forall ?p2 ?p1 (
  ?p1[owl:equivalentProperty->?p2] :- And(
```

```
      ?p1[rdfs:subPropertyOf->?p2]
      ?p2[rdfs:subPropertyOf->?p1]  ))

(* <#scm-dom1> *)
Forall ?p ?c1 ?c2 (
  ?p[rdfs:domain->?c2] :- And(
      ?p[rdfs:domain->?c1]
      ?c1[rdfs:subClassOf->?c2]  ))

(* <#scm-dom2> *)
Forall ?c ?p2 ?p1 (
  ?p1[rdfs:domain->?c] :- And(
      ?p2[rdfs:domain->?c]
      ?p1[rdfs:subPropertyOf->?p2]  ))

(* <#scm-rng1> *)
Forall ?p ?c1 ?c2 (
  ?p[rdfs:range->?c2] :- And(
      ?p[rdfs:range->?c1]
      ?c1[rdfs:subClassOf->?c2]  ))

(* <#scm-rng2> *)
Forall ?c ?p2 ?p1 (
  ?p1[rdfs:range->?c] :- And(
      ?p2[rdfs:range->?c]
      ?p1[rdfs:subPropertyOf->?p2]  ))

 (* <#dt-type2> *)
 Group (
    Forall ?s ?p ?lt ( ?lt[rdf:type->rdf:PlainLiteral rdf:type->rdfs:Literal]
                 :- And( ?s[?p->?lt] External( pred:is-literal-PlainLiteral( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->rdf:XMLLiteral rdf:type->rdfs:Literal]
                 :- And( ?s[?p->?lt] External( pred:is-literal-XMLLiteral( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:decimal rdf:type->rdfs:Literal]
```

```
                :- And( ?s[?p->?lt] External( pred:is-literal-decimal( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:integer rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-integer( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:nonNegativeInteger rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-nonNegativeInteger( ?lt
)) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:nonPositiveInteger rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-nonPositiveInteger( ?lt ))
))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:positiveInteger rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-positiveInteger( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:negativeInteger rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-negativeInteger( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:long rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-long( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:int rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-int( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:short rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-short( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:byte rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-byte( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:unsignedLong rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-unsignedLong( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:unsignedInt rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-unsignedInt( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:unsignedShort rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-unsignedShort( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:unsignedByte rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-unsignedByte( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:float rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-float( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:double rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-double( ?lt )) ))
    Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:string rdf:type->rdfs:Literal]
```

```
                :- And( ?s[?p->?lt] External( pred:is-literal-string( ?lt )) ))
   Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:normalizedString rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-normalizedString( ?lt ))
))
   Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:token rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-token( ?lt )) ))
   Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:language rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-language( ?lt )) ))
   Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:Name rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-Name( ?lt )) ))
   Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:NCName rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-NCName( ?lt )) ))
   Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:NMTOKEN rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-NMTOKEN( ?lt )) ))
   Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:boolean rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-boolean( ?lt )) ))
   Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:hexBinary rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-hexBinary( ?lt )) ))
   Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:base64Binary rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-base64Binary( ?lt )) ))
   Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:anyURI rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-anyURI( ?lt )) ))
   Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:dateTime rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-dateTime( ?lt )) ))
   Forall ?s ?p ?lt ( ?lt[rdf:type->xsd:dateTimeStamp  rdf:type->rdfs:Literal]
                :- And( ?s[?p->?lt] External( pred:is-literal-dateTimeStamp ( ?lt )) ))
 )

(* <#dt-not-type> *)
Group (

  Forall ?lt (
   rif:error() :- And (
     ?lt[rdf:type->rdf:PlainLiteral] External(pred:is-literal-not-PlainLiteral( ?lt )) ))
```

```
  Forall ?lt (
   rif:error() :- And (
     ?lt[rdf:type->rdf:XMLLiteral]  External(pred:is-literal-not-XMLLiteral( ?lt ))
))
  Forall ?lt (
   rif:error() :- And (
     ?lt[rdf:type->xsd:decimal] External(pred:is-literal-not-decimal( ?lt )) ))
  Forall ?lt (
   rif:error() :- And (
     ?lt[rdf:type->xsd:integer] External(pred:is-literal-not-integer( ?lt )) ))
  Forall ?lt (
   rif:error() :- And (
     ?lt[rdf:type->xsd:nonNegativeInteger]   External(pred:is-literal-not-nonNega-
tiveInteger( ?lt )) ))
  Forall ?lt (
   rif:error() :- And (
     ?lt[rdf:type->xsd:nonPositiveInteger]      External(pred:is-literal-not-nonPosi-
tiveInteger( ?lt )) ))
  Forall ?lt (
   rif:error() :- And (
     ?lt[rdf:type->xsd:positiveInteger]  External(pred:is-literal-not-positiveInteger(
?lt )) ))
  Forall ?lt (
   rif:error() :- And (
     ?lt[rdf:type->xsd:negativeInteger]     External(pred:is-literal-not-negativeInte-
ger( ?lt )) ))
  Forall ?lt (
   rif:error() :- And (
     ?lt[rdf:type->xsd:long] External(pred:is-literal-not-long( ?lt )) ))
  Forall ?lt (
   rif:error() :- And (
     ?lt[rdf:type->xsd:int] External(pred:is-literal-not-int( ?lt )) ))
  Forall ?lt (
   rif:error() :- And (
```

```
    ?lt[rdf:type->xsd:short] External(pred:is-literal-not-short( ?lt )) ))
  Forall ?lt (
   rif:error() :- And (
    ?lt[rdf:type->xsd:byte] External(pred:is-literal-not-byte( ?lt )) ))
  Forall ?lt (
   rif:error() :- And (
    ?lt[rdf:type->xsd:unsignedLong]    External(pred:is-literal-not-unsignedLong(
?lt )) ))
  Forall ?lt (
   rif:error() :- And (
    ?lt[rdf:type->xsd:unsignedInt]  External(pred:is-literal-not-unsignedInt(  ?lt  ))
))
  Forall ?lt (
   rif:error() :- And (
    ?lt[rdf:type->xsd:unsignedShort]    External(pred:is-literal-not-unsignedShort(
?lt )) ))
  Forall ?lt (
   rif:error() :- And (
    ?lt[rdf:type->xsd:unsignedByte] External(pred:is-literal-not-unsignedByte( ?lt
)) ))
  Forall ?lt (
   rif:error() :- And (
    ?lt[rdf:type->xsd:float] External(pred:is-literal-not-float( ?lt )) ))
  Forall ?lt (
   rif:error() :- And (
    ?lt[rdf:type->xsd:double] External(pred:is-literal-not-double( ?lt )) ))
  Forall ?lt (
   rif:error() :- And (
    ?lt[rdf:type->xsd:string] External(pred:is-literal-not-string( ?lt )) ))
  Forall ?lt (
   rif:error() :- And (
    ?lt[rdf:type->xsd:normalizedString]          External(pred:is-literal-not-normal-
izedString( ?lt )) ))
  Forall ?lt (
```

```
   rif:error() :- And (
     ?lt[rdf:type->xsd:token] External(pred:is-literal-not-token( ?lt )) ))
   Forall ?lt (
    rif:error() :- And (
     ?lt[rdf:type->xsd:language] External(pred:is-literal-not-language( ?lt )) ))
   Forall ?lt (
    rif:error() :- And (
     ?lt[rdf:type->xsd:Name] External(pred:is-literal-not-Name( ?lt )) ))
   Forall ?lt (
    rif:error() :- And (
     ?lt[rdf:type->xsd:NCName] External(pred:is-literal-not-NCName( ?lt )) ))
   Forall ?lt (
    rif:error() :- And (
     ?lt[rdf:type->xsd:NMTOKEN] External(pred:is-literal-not-NMTOKEN( ?lt ))
))
   Forall ?lt (
    rif:error() :- And (
     ?lt[rdf:type->xsd:boolean] External(pred:is-literal-not-boolean( ?lt )) ))
   Forall ?lt (
    rif:error() :- And (
     ?lt[rdf:type->xsd:hexBinary] External(pred:is-literal-not-hexBinary( ?lt )) ))
   Forall ?lt (
    rif:error() :- And (
     ?lt[rdf:type->xsd:base64Binary] External(pred:is-literal-not-base64Binary( ?lt
)) ))
   Forall ?lt (
    rif:error() :- And (
     ?lt[rdf:type->xsd:anyURI] External(pred:is-literal-not-anyURI( ?lt )) ))
   Forall ?lt (
    rif:error() :- And (
     ?lt[rdf:type->xsd:dateTime] External(pred:is-literal-not-dateTime( ?lt )) ))
   Forall ?lt (
    rif:error() :- And (
```

```
    ?lt[rdf:type->xsd:dateTimeStamp ] External(pred:is-literal-not-
dateTimeStamp ( ?lt )) ))
)

(* <#eq-diff1-literal1> *)
Forall ?x ?y ?s1 ?s2 ?p1 ?p2 (
  rif:error() :- And(
     ?s1[?p1->?x]  ?s2[?p2->?y]
     ?x[owl:sameAs->?y]
     External(pred:literal-not-identical(?x ?y))  ))

(* <#eq-diff1-literal2> *)
Forall ?x ?y ?s1 ?s2 ?p1 ?p2 (
  rif:error() :- And(
     ?s1[?p1->?x]  ?s2[?p2->?y]
     ?x = ?y
     ?x[owl:differentFrom->?y]  ))

(* <#dt-type1-PlainLiteral> *) rdf:PlainLiteral[rdf:type -> rdfs:Datatype]
(* <#dt-type1-decimal> *) xsd:decimal[rdf:type -> rdfs:Datatype]
(* <#dt-type1-integer> *) xsd:integer[rdf:type -> rdfs:Datatype]
(* <#dt-type1-double> *) xsd:double[rdf:type -> rdfs:Datatype]
(* <#dt-type1-string> *) xsd:string[rdf:type -> rdfs:Datatype]
(* <#dt-type1-dateTime> *) xsd:dateTime[rdf:type -> rdfs:Datatype]
(* <#dt-type1-XMLLiteral> *) rdf:XMLLiteral[rdf:type -> rdfs:Datatype]
(* <#dt-type1-Literal> *) rdfs:Literal[rdf:type -> rdfs:Datatype]

(* <#dt-type1-nonNegativeInteger> *) xsd:nonNegativeInteger[rdf:type ->
rdfs:Datatype]
(* <#dt-type1-nonPositiveInteger> *) xsd:nonPositiveInteger[rdf:type ->
rdfs:Datatype]
(* <#dt-type1-positiveInteger> *) xsd:positiveInteger[rdf:type -> rdfs:Datatype]
(* <#dt-type1-negativeInteger> *) xsd:negativeInteger[rdf:type -> rdfs:Datatype]
(* <#dt-type1-long> *) xsd:long[rdf:type -> rdfs:Datatype]
```

(* <#dt-type1-int> *) xsd:int[rdf:type -> rdfs:Datatype]
(* <#dt-type1-short> *) xsd:short[rdf:type -> rdfs:Datatype]
(* <#dt-type1-byte> *) xsd:byte[rdf:type -> rdfs:Datatype]
(* <#dt-type1-unsignedLong> *) xsd:unsignedLong[rdf:type -> rdfs:Datatype]
(* <#dt-type1-unsignedInt> *) xsd:unsignedInt[rdf:type -> rdfs:Datatype]
(* <#dt-type1-unsignedShort> *) xsd:unsignedShort[rdf:type -> rdfs:Datatype]
(* <#dt-type1-unsignedByte> *) xsd:unsignedByte[rdf:type -> rdfs:Datatype]
(* <#dt-type1-normalizedString> *) xsd:normalizedString[rdf:type -> rdfs:Datatype]
(* <#dt-type1-token> *) xsd:token[rdf:type -> rdfs:Datatype]
(* <#dt-type1-language> *) xsd:language[rdf:type -> rdfs:Datatype]
(* <#dt-type1-Name> *) xsd:Name[rdf:type -> rdfs:Datatype]
(* <#dt-type1-NCName> *) xsd:NCName[rdf:type -> rdfs:Datatype]
(* <#dt-type1-NMTOKEN> *) xsd:NMTOKEN[rdf:type -> rdfs:Datatype]

(* <#dt-type1-float> *) xsd:float[rdf:type -> rdfs:Datatype]
(* <#dt-type1-boolean> *) xsd:boolean[rdf:type -> rdfs:Datatype]
(* <#dt-type1-hexBinary> *) xsd:hexBinary[rdf:type -> rdfs:Datatype]
(* <#dt-type1-base64Binary> *) xsd:base64Binary[rdf:type -> rdfs:Datatype]
(* <#dt-type1-anyURI> *) xsd:anyURI[rdf:type -> rdfs:Datatype]
(* <#dt-type1-dateTimeStamp> *) xsd:dateTimeStamp [rdf:type -> rdfs:Datatype]