SWB axioms

SWB003+1.ax RDFS Extensional axioms
$\forall p, c$: (iext(uri_rdfs_domain, p, c) $\iff$ (ip(p) and ic(c) and $\forall x, y$: (iext(p, x, y) $\Rightarrow$ icext(c, x)))) fof(owl_rdfsext_domain, a
$\forall p, c$: (iext(uri_rdfs_range, p, c) $\iff$ (ip(p) and ip(c) and $\forall x, y$: (iext(p, x, y) $\Rightarrow$ icext(c, y)))) fof(owl_rdfsext_range, axio
$\forall c_1, c_2$: (iext(uri_rdfs_subClassOf, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: (icext(c_1, x) $\Rightarrow$ icext(c_2, x)))) fof(owl_rdfsext_su
$\forall p_1, p_2$: (iext(uri_rdfs_subPropertyOf, p_1, p_2) $\iff$ (ip(p_1) and ip(p_2) and $\forall x, y$: (iext(p_1, x, y) $\Rightarrow$ iext(p_2, x, y)))) fof(owl_

SWB problems

SWB001+1.p Subgraph Entailment
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_r, uri_owl_Restriction) and iext(uri_owl_onProperty, uri_ex_r, uri_ex_p) fof(testcase_conclusion_full
iext(uri_rdfs_subClassOf, uri_ex_c, uri_ex_r) and iext(uri_rdf_type, uri_ex_r, uri_owl_Restriction) and iext(uri_owl_onProperty, u

SWB001+2.p Subgraph Entailment
iext(uri_rdf_type, uri_ex_r, uri_owl_Restriction) and iext(uri_owl_onProperty, uri_ex_r, uri_ex_p) fof(testcase_conclusion_full
iext(uri_rdfs_subClassOf, uri_ex_c, uri_ex_r) and iext(uri_rdf_type, uri_ex_r, uri_owl_Restriction) and iext(uri_owl_onProperty, u

SWB001+3.p Subgraph Entailment
include('Axioms/SWB002+0.ax')
iext(uri_rdf_type, uri_ex_r, uri_owl_Restriction) and iext(uri_owl_onProperty, uri_ex_r, uri_ex_p) fof(testcase_conclusion_full
iext(uri_rdfs_subClassOf, uri_ex_c, uri_ex_r) and iext(uri_rdf_type, uri_ex_r, uri_owl_Restriction) and iext(uri_owl_onProperty, u

SWB001+4.p Subgraph Entailment
include('Axioms/SWB003+0.ax')
iext(uri_rdf_type, uri_ex_r, uri_owl_Restriction) and iext(uri_owl_onProperty, uri_ex_r, uri_ex_p) fof(testcase_conclusion_full
iext(uri_rdfs_subClassOf, uri_ex_c, uri_ex_r) and iext(uri_rdf_type, uri_ex_r, uri_owl_Restriction) and iext(uri_owl_onProperty, u

SWB002+1.p Existential Blank Nodes
include('Axioms/SWB001+0.ax')
$\exists$bNODE_x, bNODE_y: (iext(uri_ex_p, bNODE_x, bNODE_y) and iext(uri_ex_q, bNODE_y, bNODE_x)) fof(testcase_conclu
$\exists$bNODE_o: (iext(uri_ex_p, uri_ex_s, bNODE_o) and iext(uri_ex_q, bNODE_o, uri_ex_s)) fof(testcase_premise_fullish_002_Ex

SWB002+2.p Existential Blank Nodes
$\exists$bNODE_x, bNODE_y: (iext(uri_ex_p, bNODE_x, bNODE_y) and iext(uri_ex_q, bNODE_y, bNODE_x)) fof(testcase_conclu
$\exists$bNODE_o: (iext(uri_ex_p, uri_ex_s, bNODE_o) and iext(uri_ex_q, bNODE_o, uri_ex_s)) fof(testcase_premise_fullish_002_Ex

SWB002+3.p Existential Blank Nodes
include('Axioms/SWB002+0.ax')
$\exists$bNODE_x, bNODE_y: (iext(uri_ex_p, bNODE_x, bNODE_y) and iext(uri_ex_q, bNODE_y, bNODE_x)) fof(testcase_conclu
$\exists$bNODE_o: (iext(uri_ex_p, uri_ex_s, bNODE_o) and iext(uri_ex_q, bNODE_o, uri_ex_s)) fof(testcase_premise_fullish_002_Ex

SWB002+4.p Existential Blank Nodes
include('Axioms/SWB003+0.ax')
$\exists$bNODE_x, bNODE_y: (iext(uri_ex_p, bNODE_x, bNODE_y) and iext(uri_ex_q, bNODE_y, bNODE_x)) fof(testcase_conclu
$\exists$bNODE_o: (iext(uri_ex_p, uri_ex_s, bNODE_o) and iext(uri_ex_q, bNODE_o, uri_ex_s)) fof(testcase_premise_fullish_002_Ex

SWB003+1.p Blank Nodes for Literals
include('Axioms/SWB001+0.ax')
$\exists$bNODE_x: iext(uri_ex_p, uri_ex_s, bNODE_x) fof(testcase_conclusion_fullish_003_Blank_Nodes_for_Literals, conjecture)
iext(uri_ex_p, uri_ex_s, literal_plain(dat_str_foo)) fof(testcase_premise_fullish_003_Blank_Nodes_for_Literals, axiom)

SWB003+2.p Blank Nodes for Literals
$\exists$bNODE_x: iext(uri_ex_p, uri_ex_s, bNODE_x) fof(testcase_conclusion_fullish_003_Blank_Nodes_for_Literals, conjecture)
iext(uri_ex_p, uri_ex_s, literal_plain(dat_str_foo)) fof(testcase_premise_fullish_003_Blank_Nodes_for_Literals, axiom)

SWB003+3.p Blank Nodes for Literals
include('Axioms/SWB002+0.ax')
$\exists$bNODE_x: iext(uri_ex_p, uri_ex_s, bNODE_x) fof(testcase_conclusion_fullish_003_Blank_Nodes_for_Literals, conjecture)
iext(uri_ex_p, uri_ex_s, literal_plain(dat_str_foo)) fof(testcase_premise_fullish_003_Blank_Nodes_for_Literals, axiom)

SWB003+4.p Blank Nodes for Literals
include('Axioms/SWB003+0.ax')
$\exists$bNODE_x: iext(uri_ex_p, uri_ex_s, bNODE_x) fof(testcase_conclusion_fullish_003_Blank_Nodes_for_Literals, conjecture)
iext(uri_ex_p, uri_ex_s, literal_plain(dat_str_foo)) fof(testcase_premise_fullish_003_Blank_Nodes_for_Literals, axiom)

SWB004+1.p Axiomatic Triples

include('Axioms/SWB001+0.ax')

iext(uri_rdf_type, uri_owl_Class, uri_owl_Thing) and iext(uri_rdf_type, uri_owl_Class, uri_owl_Class) and iext(uri_rdfs_subClassC

SWB004+2.p Axiomatic Triples

$\forall x$: ir(x) fof(simple_ir, axiom)

$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x)) fof(rdfs_cext_def, axiom)

$\forall x$: (idc(x) $\Rightarrow$ ic(x)) fof(owl_parts_idc_cond_set, axiom)

ic(uri_owl_Class) fof(owl_class_classowl_type, axiom)

$\forall x$: (icext(uri_owl_Class, x) $\iff$ ic(x)) fof(owl_class_classowl_ext, axiom)

ic(uri_rdfs_Class) fof(owl_class_classrdfs_type, axiom)

$\forall x$: (icext(uri_rdfs_Class, x) $\iff$ ic(x)) fof(owl_class_classrdfs_ext, axiom)

ic(uri_rdfs_Datatype) fof(owl_class_datatype_type, axiom)

$\forall x$: (icext(uri_rdfs_Datatype, x) $\iff$ idc(x)) fof(owl_class_datatype_ext, axiom)

ic(uri_owl_Thing) fof(owl_class_thing_type, axiom)

$\forall x$: (icext(uri_owl_Thing, x) $\iff$ ir(x)) fof(owl_class_thing_ext, axiom)

$\forall c_1, c_2$: (iext(uri_rdfs_subClassOf, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: (icext(c_1, x) $\Rightarrow$ icext(c_2, x)))) fof(owl_rdfsext_su

$\forall c_1, c_2$: (iext(uri_owl_equivalentClass, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: (icext(c_1, x) $\iff$ icext(c_2, x)))) fof(owl_eqdi

iext(uri_rdf_type, uri_owl_Class, uri_owl_Thing) and iext(uri_rdf_type, uri_owl_Class, uri_owl_Class) and iext(uri_rdfs_subClassC

SWB004+3.p Axiomatic Triples

include('Axioms/SWB002+0.ax')

iext(uri_rdf_type, uri_owl_Class, uri_owl_Thing) and iext(uri_rdf_type, uri_owl_Class, uri_owl_Class) and iext(uri_rdfs_subClassC

SWB004+4.p Axiomatic Triples

include('Axioms/SWB003+0.ax')

iext(uri_rdf_type, uri_owl_Class, uri_owl_Thing) and iext(uri_rdf_type, uri_owl_Class, uri_owl_Class) and iext(uri_rdfs_subClassC

SWB005+1.p Everything is a Resource

include('Axioms/SWB001+0.ax')

iext(uri_rdf_type, uri_ex_s, uri_rdfs_Resource) and iext(uri_rdf_type, uri_ex_s, uri_owl_Thing) and iext(uri_rdf_type, uri_ex_p, uri

iext(uri_ex_p, uri_ex_s, uri_ex_o) fof(testcase_premise_fullish_005_Everything_is_a_Resource, axiom)

SWB005+2.p Everything is a Resource

$\forall x$: ir(x) fof(simple_ir, axiom)

$\forall s, p, o$: (iext(p, s, o) $\Rightarrow$ ip(p)) fof(simple_iext_property, axiom)

$\forall p$: (iext(uri_rdf_type, p, uri_rdf_Property) $\iff$ ip(p)) fof(rdf_type_ip, axiom)

$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x)) fof(rdfs_cext_def, axiom)

$\forall x$: (ir(x) $\iff$ icext(uri_rdfs_Resource, x)) fof(rdfs_ir_def, axiom)

$\forall x$: (icext(uri_owl_Thing, x) $\iff$ ir(x)) fof(owl_class_thing_ext, axiom)

$\forall x$: (icext(uri_owl_ObjectProperty, x) $\iff$ ip(x)) fof(owl_class_objectproperty_ext, axiom)

iext(uri_rdf_type, uri_ex_s, uri_rdfs_Resource) and iext(uri_rdf_type, uri_ex_s, uri_owl_Thing) and iext(uri_rdf_type, uri_ex_p, uri

iext(uri_ex_p, uri_ex_s, uri_ex_o) fof(testcase_premise_fullish_005_Everything_is_a_Resource, axiom)

SWB005+3.p Everything is a Resource

include('Axioms/SWB002+0.ax')

iext(uri_rdf_type, uri_ex_s, uri_rdfs_Resource) and iext(uri_rdf_type, uri_ex_s, uri_owl_Thing) and iext(uri_rdf_type, uri_ex_p, uri

iext(uri_ex_p, uri_ex_s, uri_ex_o) fof(testcase_premise_fullish_005_Everything_is_a_Resource, axiom)

SWB005+4.p Everything is a Resource

include('Axioms/SWB003+0.ax')

iext(uri_rdf_type, uri_ex_s, uri_rdfs_Resource) and iext(uri_rdf_type, uri_ex_s, uri_owl_Thing) and iext(uri_rdf_type, uri_ex_p, uri

iext(uri_ex_p, uri_ex_s, uri_ex_o) fof(testcase_premise_fullish_005_Everything_is_a_Resource, axiom)

SWB006+1.p Literal Values represented by URIs and Blank Nodes

include('Axioms/SWB001+0.ax')

iext(uri_owl_sameAs, uri_ex_u, uri_ex_w) fof(testcase_conclusion_fullish_006_Literal_Values_represented_by_URIs_and_Blank

∃bNODE_x: (iext(uri_owl_sameAs, uri_ex_u, literal_plain(dat_str_abc)) and iext(uri_owl_sameAs, bNODE_x, literal_plain(dat_st

SWB006+2.p Literal Values represented by URIs and Blank Nodes

$\forall x, y$: (iext(uri_owl_sameAs, x, y) $\iff$ $x = y$) fof(owl_eqdis_sameas, axiom)

iext(uri_owl_sameAs, uri_ex_u, uri_ex_w) fof(testcase_conclusion_fullish_006_Literal_Values_represented_by_URIs_and_Blank

∃bNODE_x: (iext(uri_owl_sameAs, uri_ex_u, literal_plain(dat_str_abc)) and iext(uri_owl_sameAs, bNODE_x, literal_plain(dat_s

SWB006+3.p Literal Values represented by URIs and Blank Nodes

include('Axioms/SWB002+0.ax')
iext(uri_owl_sameAs, uri_ex_u, uri_ex_w)   fof(testcase_conclusion_fullish_006_Literal_Values_represented_by_URIs_and_Blank
$\exists$bNODE_x: (iext(uri_owl_sameAs, uri_ex_u, literal_plain(dat_str_abc)) and iext(uri_owl_sameAs, bNODE_x, literal_plain(dat_st

SWB006+4.p Literal Values represented by URIs and Blank Nodes
include('Axioms/SWB003+0.ax')
iext(uri_owl_sameAs, uri_ex_u, uri_ex_w)   fof(testcase_conclusion_fullish_006_Literal_Values_represented_by_URIs_and_Blank
$\exists$bNODE_x: (iext(uri_owl_sameAs, uri_ex_u, literal_plain(dat_str_abc)) and iext(uri_owl_sameAs, bNODE_x, literal_plain(dat_st

SWB007+1.p Equal Classes
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_w, uri_ex_c$_2$) and iext(uri_rdfs_subClassOf, uri_ex_c, uri_ex_c$_2$) and iext(uri_rdfs_range, uri_ex_p, uri_ex
iext(uri_owl_sameAs, uri_ex_c$_1$, uri_ex_c$_2$) and iext(uri_rdf_type, uri_ex_w, uri_ex_c$_1$) and iext(uri_rdfs_subClassOf, uri_ex_c, uri_

SWB007+2.p Equal Classes
$\forall x, y$: (iext(uri_owl_sameAs, x, y) $\iff$ $x = y$)   fof(owl_eqdis_sameas, axiom)
iext(uri_rdf_type, uri_ex_w, uri_ex_c$_2$) and iext(uri_rdfs_subClassOf, uri_ex_c, uri_ex_c$_2$) and iext(uri_rdfs_range, uri_ex_p, uri_ex
iext(uri_owl_sameAs, uri_ex_c$_1$, uri_ex_c$_2$) and iext(uri_rdf_type, uri_ex_w, uri_ex_c$_1$) and iext(uri_rdfs_subClassOf, uri_ex_c, uri_

SWB007+3.p Equal Classes
include('Axioms/SWB002+0.ax')
iext(uri_rdf_type, uri_ex_w, uri_ex_c$_2$) and iext(uri_rdfs_subClassOf, uri_ex_c, uri_ex_c$_2$) and iext(uri_rdfs_range, uri_ex_p, uri_ex
iext(uri_owl_sameAs, uri_ex_c$_1$, uri_ex_c$_2$) and iext(uri_rdf_type, uri_ex_w, uri_ex_c$_1$) and iext(uri_rdfs_subClassOf, uri_ex_c, uri_

SWB007+4.p Equal Classes
include('Axioms/SWB003+0.ax')
iext(uri_rdf_type, uri_ex_w, uri_ex_c$_2$) and iext(uri_rdfs_subClassOf, uri_ex_c, uri_ex_c$_2$) and iext(uri_rdfs_range, uri_ex_p, uri_ex
iext(uri_owl_sameAs, uri_ex_c$_1$, uri_ex_c$_2$) and iext(uri_rdf_type, uri_ex_w, uri_ex_c$_1$) and iext(uri_rdfs_subClassOf, uri_ex_c, uri_

SWB008+1.p Inverse Functional Data Properties
include('Axioms/SWB001+0.ax')
iext(uri_owl_sameAs, uri_ex_bob, uri_ex_robert)   fof(testcase_conclusion_fullish_008_Inverse_Functional_Data_Properties, co
iext(uri_rdf_type, uri_foaf_mbox_sha1sum, uri_owl_DatatypeProperty) and iext(uri_rdf_type, uri_foaf_mbox_sha1sum, uri_owl_In

SWB008+2.p Inverse Functional Data Properties
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x))   fof(rdfs_cext_def, axiom)
$\forall x, y$: (iext(uri_owl_sameAs, x, y) $\iff$ $x = y$)   fof(owl_eqdis_sameas, axiom)
$\forall p$: (icext(uri_owl_InverseFunctionalProperty, p) $\iff$ (ip(p) and $\forall x_1, x_2, y$: ((iext(p, x_1, y) and iext(p, x_2, y)) $\Rightarrow$ $x_1 = x_2$)))   fof(owl_char_inversefunctional, axiom)
iext(uri_owl_sameAs, uri_ex_bob, uri_ex_robert)   fof(testcase_conclusion_fullish_008_Inverse_Functional_Data_Properties, co
iext(uri_rdf_type, uri_foaf_mbox_sha1sum, uri_owl_DatatypeProperty) and iext(uri_rdf_type, uri_foaf_mbox_sha1sum, uri_owl_In

SWB008+3.p Inverse Functional Data Properties
include('Axioms/SWB002+0.ax')
iext(uri_owl_sameAs, uri_ex_bob, uri_ex_robert)   fof(testcase_conclusion_fullish_008_Inverse_Functional_Data_Properties, co
iext(uri_rdf_type, uri_foaf_mbox_sha1sum, uri_owl_DatatypeProperty) and iext(uri_rdf_type, uri_foaf_mbox_sha1sum, uri_owl_In

SWB008+4.p Inverse Functional Data Properties
include('Axioms/SWB003+0.ax')
iext(uri_owl_sameAs, uri_ex_bob, uri_ex_robert)   fof(testcase_conclusion_fullish_008_Inverse_Functional_Data_Properties, co
iext(uri_rdf_type, uri_foaf_mbox_sha1sum, uri_owl_DatatypeProperty) and iext(uri_rdf_type, uri_foaf_mbox_sha1sum, uri_owl_In

SWB009+1.p Existential Restriction Entailments
include('Axioms/SWB001+0.ax')
$\exists$bNODE_x: (iext(uri_ex_p, uri_ex_s, bNODE_x) and iext(uri_rdf_type, bNODE_x, uri_ex_c))   fof(testcase_conclusion_fullish_
$\exists$bNODE_z: (iext(uri_rdf_type, uri_ex_p, uri_owl_ObjectProperty) and iext(uri_rdf_type, uri_ex_c, uri_owl_Class) and iext(uri_r

SWB009+2.p Existential Restriction Entailments
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x))   fof(rdfs_cext_def, axiom)
$\forall z, p, c$: ((iext(uri_owl_someValuesFrom, z, c) and iext(uri_owl_onProperty, z, p)) $\Rightarrow$ $\forall x$: (icext(z, x) $\iff$ $\exists y$: (iext(p, x, y) an
$\exists$bNODE_x: (iext(uri_ex_p, uri_ex_s, bNODE_x) and iext(uri_rdf_type, bNODE_x, uri_ex_c))   fof(testcase_conclusion_fullish_
$\exists$bNODE_z: (iext(uri_rdf_type, uri_ex_p, uri_owl_ObjectProperty) and iext(uri_rdf_type, uri_ex_c, uri_owl_Class) and iext(uri_r

SWB009+3.p Existential Restriction Entailments
include('Axioms/SWB002+0.ax')
$\exists$bNODE_x: (iext(uri_ex_p, uri_ex_s, bNODE_x) and iext(uri_rdf_type, bNODE_x, uri_ex_c))   fof(testcase_conclusion_fullish_
$\exists$bNODE_z: (iext(uri_rdf_type, uri_ex_p, uri_owl_ObjectProperty) and iext(uri_rdf_type, uri_ex_c, uri_owl_Class) and iext(uri_r

SWB009+4.p Existential Restriction Entailments
include('Axioms/SWB003+0.ax')
$\exists$bNODE_x: (iext(uri_ex_p, uri_ex_s, bNODE_x) and iext(uri_rdf_type, bNODE_x, uri_ex_c)) fof(testcase_conclusion_fullish_
$\exists$bNODE_z: (iext(uri_rdf_type, uri_ex_p, uri_owl_ObjectProperty) and iext(uri_rdf_type, uri_ex_c, uri_owl_Class) and iext(uri_rd

SWB010+1.p Negative Property Assertions
include('Axioms/SWB001+0.ax')
$\exists$bNODE_z: (iext(uri_rdf_type, bNODE_z, uri_owl_NegativePropertyAssertion) and iext(uri_owl_sourceIndividual, bNODE_z, u
$\exists$bNODE_x$_1$, bNODE_x$_2$, bNODE_x$_3$, bNODE_x$_4$: (iext(uri_rdf_type, uri_ex_p, uri_owl_ObjectProperty) and iext(uri_rdf_type,

SWB010+2.p Negative Property Assertions
$\forall x$: ir(x) fof(simple_ir, axiom)
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x)) fof(rdfs_cext_def, axiom)
$\forall x, y$: (iext(uri_owl_sourceIndividual, x, y) $\Rightarrow$ (icext(uri_owl_NegativePropertyAssertion, x) and ir(y))) fof(owl_prop_sourc
$\forall x, y$: (iext(uri_owl_onProperty, x, y) $\Rightarrow$ (icext(uri_owl_Restriction, x) and ip(y))) fof(owl_prop_onproperty_ext, axiom)
$\forall z, c$: (iext(uri_owl_complementOf, z, c) $\Rightarrow$ (ic(z) and ic(c) and $\forall x$: (icext(z, x) $\iff$ $\neg$ icext(c, x)))) fof(owl_bool_complen
$\forall z, s_1, a_1$: ((iext(uri_rdf_first, s_1, a_1) and iext(uri_rdf_rest, s_1, uri_rdf_nil)) $\Rightarrow$ (iext(uri_owl_oneOf, z, s_1) $\iff$ (ic(z) and $\forall x$: (i
$x = a_1$)))) fof(owl_enum_class$_{001}$, axiom)
$\forall z, p, c$: ((iext(uri_owl_allValuesFrom, z, c) and iext(uri_owl_onProperty, z, p)) $\Rightarrow$ $\forall x$: (icext(z, x) $\iff$ $\forall y$: (iext(p, x, y) $\Rightarrow$
icext(c, y)))) fof(owl_restrict_allvaluesfrom, axiom)
$\forall p, a_1, a_2$: ((ir(a_1) and ip(p) and ir(a_2) and $\neg$ iext(p, a_1, a_2)) $\Rightarrow$ $\exists z$: (iext(uri_owl_sourceIndividual, z, a_1) and iext(uri_owl_ass
$\exists$bNODE_z: (iext(uri_rdf_type, bNODE_z, uri_owl_NegativePropertyAssertion) and iext(uri_owl_sourceIndividual, bNODE_z, u
$\exists$bNODE_x$_1$, bNODE_x$_2$, bNODE_x$_3$, bNODE_x$_4$: (iext(uri_rdf_type, uri_ex_p, uri_owl_ObjectProperty) and iext(uri_rdf_type,

SWB010+3.p Negative Property Assertions
include('Axioms/SWB002+0.ax')
$\exists$bNODE_z: (iext(uri_rdf_type, bNODE_z, uri_owl_NegativePropertyAssertion) and iext(uri_owl_sourceIndividual, bNODE_z, u
$\exists$bNODE_x$_1$, bNODE_x$_2$, bNODE_x$_3$, bNODE_x$_4$: (iext(uri_rdf_type, uri_ex_p, uri_owl_ObjectProperty) and iext(uri_rdf_type,

SWB010+4.p Negative Property Assertions
include('Axioms/SWB003+0.ax')
$\exists$bNODE_z: (iext(uri_rdf_type, bNODE_z, uri_owl_NegativePropertyAssertion) and iext(uri_owl_sourceIndividual, bNODE_z, u
$\exists$bNODE_x$_1$, bNODE_x$_2$, bNODE_x$_3$, bNODE_x$_4$: (iext(uri_rdf_type, uri_ex_p, uri_owl_ObjectProperty) and iext(uri_rdf_type,

SWB011+1.p Entity Types as Classes
include('Axioms/SWB001+0.ax')
iext(uri_owl_disjointWith, uri_owl_Class, uri_owl_ObjectProperty) and iext(uri_rdf_type, uri_ex_x, uri_owl_Class) and iext(uri_r

SWB011+2.p Entity Types as Classes
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x)) fof(rdfs_cext_def, axiom)
$\forall c_1, c_2$: (iext(uri_owl_disjointWith, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: $\neg$ icext(c_1, x) and icext(c_2, x))) fof(owl_eqdis_dis
iext(uri_owl_disjointWith, uri_owl_Class, uri_owl_ObjectProperty) and iext(uri_rdf_type, uri_ex_x, uri_owl_Class) and iext(uri_r

SWB011+3.p Entity Types as Classes
include('Axioms/SWB002+0.ax')
iext(uri_owl_disjointWith, uri_owl_Class, uri_owl_ObjectProperty) and iext(uri_rdf_type, uri_ex_x, uri_owl_Class) and iext(uri_r

SWB011+4.p Entity Types as Classes
include('Axioms/SWB003+0.ax')
iext(uri_owl_disjointWith, uri_owl_Class, uri_owl_ObjectProperty) and iext(uri_rdf_type, uri_ex_x, uri_owl_Class) and iext(uri_r

SWB012+1.p Template Class
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_name, uri_owl_FunctionalProperty) and iext(uri_rdf_type, uri_ex_alice, uri_foaf_Person) fof(testcase
$\exists$bNODE_l$_1$, bNODE_l$_2$, bNODE_l$_3$, bNODE_r: (iext(uri_rdf_type, uri_foaf_Person, uri_owl_Class) and iext(uri_owl_intersection

SWB012+2.p Template Class
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x)) fof(rdfs_cext_def, axiom)
$\forall p, c, x, y$: ((iext(uri_rdfs_domain, p, c) and iext(p, x, y)) $\Rightarrow$ icext(c, x)) fof(rdfs_domain_main, axiom)
$\forall z, s_1, c_1, s_2, c_2, s_3, c_3$: ((iext(uri_rdf_first, s_1, c_1) and iext(uri_rdf_rest, s_1, s_2) and iext(uri_rdf_first, s_2, c_2) and iext(uri_rdf_rest
(iext(uri_owl_intersectionOf, z, s_1) $\iff$ (ic(z) and ic(c_1) and ic(c_2) and ic(c_3) and $\forall x$: (icext(z, x) $\iff$ (icext(c_1, x) and ice
$\forall z, p, a$: ((iext(uri_owl_hasValue, z, a) and iext(uri_owl_onProperty, z, p)) $\Rightarrow$ $\forall x$: (icext(z, x) $\iff$ iext(p, x, a))) fof(owl_re
iext(uri_rdf_type, uri_ex_name, uri_owl_FunctionalProperty) and iext(uri_rdf_type, uri_ex_alice, uri_foaf_Person) fof(testcase
$\exists$bNODE_l$_1$, bNODE_l$_2$, bNODE_l$_3$, bNODE_r: (iext(uri_rdf_type, uri_foaf_Person, uri_owl_Class) and iext(uri_owl_intersection

SWB012+3.p Template Class

include('Axioms/SWB002+0.ax')
iext(uri_rdf_type, uri_ex_name, uri_owl_FunctionalProperty) and iext(uri_rdf_type, uri_ex_alice, uri_foaf_Person)   fof(testcase
$\exists$bNODE_l$_1$, bNODE_l$_2$, bNODE_l$_3$, bNODE_r: (iext(uri_rdf_type, uri_foaf_Person, uri_owl_Class) and iext(uri_owl_intersection

SWB012+4.p Template Class
include('Axioms/SWB003+0.ax')
iext(uri_rdf_type, uri_ex_name, uri_owl_FunctionalProperty) and iext(uri_rdf_type, uri_ex_alice, uri_foaf_Person)   fof(testcase
$\exists$bNODE_l$_1$, bNODE_l$_2$, bNODE_l$_3$, bNODE_r: (iext(uri_rdf_type, uri_foaf_Person, uri_owl_Class) and iext(uri_owl_intersection

SWB013+1.p Cliques
include('Axioms/SWB001+0.ax')
iext(uri_foaf_knows, uri_ex_alice, uri_ex_bob)   fof(testcase_conclusion_fullish_013_Cliques, conjecture)
$\exists$bNODE_r, bNODE_i, bNODE_l$_1$, bNODE_l$_2$, bNODE_l$_3$: (iext(uri_rdf_type, uri_ex_Clique, uri_owl_Class) and iext(uri_rdfs_su

SWB013+2.p Cliques
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x))   fof(rdfs_cext_def, axiom)
$\forall z, p, c$: ((iext(uri_owl_someValuesFrom, z, c) and iext(uri_owl_onProperty, z, p)) $\Rightarrow$ $\forall x$: (icext(z, x) $\iff$ $\exists y$: (iext(p, x, y) an
$\forall c_1, c_2$: (iext(uri_rdfs_subClassOf, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: (icext(c_1, x) $\Rightarrow$ icext(c_2, x))))   fof(owl_rdfsext_su
$\forall p_1, p_2$: (iext(uri_rdfs_subPropertyOf, p_1, p_2) $\iff$ (ip(p_1) and ip(p_2) and $\forall x, y$: (iext(p_1, x, y) $\Rightarrow$ iext(p_2, x, y))))   fof(owl_
$\forall x, y$: (iext(uri_owl_sameAs, x, y) $\iff$ $x = y$)   fof(owl_eqdis_sameas, axiom)
$\forall p, s_1, p_1, s_2, p_2, s_3, p_3$: ((iext(uri_rdf_first, s_1, p_1) and iext(uri_rdf_rest, s_1, s_2) and iext(uri_rdf_first, s_2, p_2) and iext(uri_rdf_res
(iext(uri_owl_propertyChainAxiom, p, s_1) $\iff$ (ip(p) and ip(p_1) and ip(p_2) and ip(p_3) and $\forall y_0, y_1, y_2, y_3$: ((iext($p_1, y_0, y_1$) an
iext(p, y_0, y_3)))))   fof(owl_chain$_{003}$, axiom)
$\forall p_1, p_2$: (iext(uri_owl_inverseOf, p_1, p_2) $\iff$ (ip(p_1) and ip(p_2) and $\forall x, y$: (iext(p_1, x, y) $\iff$ iext(p_2, y, x))))   fof(owl_inv,
iext(uri_foaf_knows, uri_ex_alice, uri_ex_bob)   fof(testcase_conclusion_fullish_013_Cliques, conjecture)
$\exists$bNODE_r, bNODE_i, bNODE_l$_1$, bNODE_l$_2$, bNODE_l$_3$: (iext(uri_rdf_type, uri_ex_Clique, uri_owl_Class) and iext(uri_rdfs_su

SWB013+3.p Cliques
include('Axioms/SWB002+0.ax')
iext(uri_foaf_knows, uri_ex_alice, uri_ex_bob)   fof(testcase_conclusion_fullish_013_Cliques, conjecture)
$\exists$bNODE_r, bNODE_i, bNODE_l$_1$, bNODE_l$_2$, bNODE_l$_3$: (iext(uri_rdf_type, uri_ex_Clique, uri_owl_Class) and iext(uri_rdfs_su

SWB013+4.p Cliques
include('Axioms/SWB003+0.ax')
iext(uri_foaf_knows, uri_ex_alice, uri_ex_bob)   fof(testcase_conclusion_fullish_013_Cliques, conjecture)
$\exists$bNODE_r, bNODE_i, bNODE_l$_1$, bNODE_l$_2$, bNODE_l$_3$: (iext(uri_rdf_type, uri_ex_Clique, uri_owl_Class) and iext(uri_rdfs_su

SWB014+1.p Harry belongs to some Species
include('Axioms/SWB001+0.ax')
$\exists$bNODE_x: (iext(uri_rdf_type, uri_ex_harry, bNODE_x) and iext(uri_rdf_type, bNODE_x, uri_ex_Species))   fof(testcase_con
$\exists$bNODE_u, bNODE_l$_1$, bNODE_l$_2$: (iext(uri_rdf_type, uri_ex_Eagle, uri_ex_Species) and iext(uri_rdf_type, uri_ex_Falcon, uri_ex

SWB014+2.p Harry belongs to some Species
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x))   fof(rdfs_cext_def, axiom)
$\forall z, s_1, c_1, s_2, c_2$: ((iext(uri_rdf_first, s_1, c_1) and iext(uri_rdf_rest, s_1, s_2) and iext(uri_rdf_first, s_2, c_2) and iext(uri_rdf_rest, s_2, ur
(iext(uri_owl_unionOf, z, s_1) $\iff$ (ic(z) and ic(c_1) and ic(c_2) and $\forall x$: (icext(z, x) $\iff$ (icext(c_1, x) or icext(c_2, x))))))   fo
$\exists$bNODE_x: (iext(uri_rdf_type, uri_ex_harry, bNODE_x) and iext(uri_rdf_type, bNODE_x, uri_ex_Species))   fof(testcase_con
$\exists$bNODE_u, bNODE_l$_1$, bNODE_l$_2$: (iext(uri_rdf_type, uri_ex_Eagle, uri_ex_Species) and iext(uri_rdf_type, uri_ex_Falcon, uri_ex

SWB014+3.p Harry belongs to some Species
include('Axioms/SWB002+0.ax')
$\exists$bNODE_x: (iext(uri_rdf_type, uri_ex_harry, bNODE_x) and iext(uri_rdf_type, bNODE_x, uri_ex_Species))   fof(testcase_con
$\exists$bNODE_u, bNODE_l$_1$, bNODE_l$_2$: (iext(uri_rdf_type, uri_ex_Eagle, uri_ex_Species) and iext(uri_rdf_type, uri_ex_Falcon, uri_ex

SWB014+4.p Harry belongs to some Species
include('Axioms/SWB003+0.ax')
$\exists$bNODE_x: (iext(uri_rdf_type, uri_ex_harry, bNODE_x) and iext(uri_rdf_type, bNODE_x, uri_ex_Species))   fof(testcase_con
$\exists$bNODE_u, bNODE_l$_1$, bNODE_l$_2$: (iext(uri_rdf_type, uri_ex_Eagle, uri_ex_Species) and iext(uri_rdf_type, uri_ex_Falcon, uri_ex

SWB015+1.p Reflective Tautologies I
include('Axioms/SWB001+0.ax')
iext(uri_owl_sameAs, uri_owl_sameAs, uri_owl_sameAs)   fof(testcase_conclusion_fullish_015_Reflective_Tautologies_I, conject

SWB015+2.p Reflective Tautologies I
$\forall x, y$: (iext(uri_owl_sameAs, x, y) $\iff$ $x = y$)   fof(owl_eqdis_sameas, axiom)
iext(uri_owl_sameAs, uri_owl_sameAs, uri_owl_sameAs)   fof(testcase_conclusion_fullish_015_Reflective_Tautologies_I, conject

SWB015+3.p Reflective Tautologies I
include('Axioms/SWB002+0.ax')
iext(uri_owl_sameAs, uri_owl_sameAs, uri_owl_sameAs)   fof(testcase_conclusion_fullish_015_Reflective_Tautologies_I, conject

SWB015+4.p Reflective Tautologies I
include('Axioms/SWB003+0.ax')
iext(uri_owl_sameAs, uri_owl_sameAs, uri_owl_sameAs)   fof(testcase_conclusion_fullish_015_Reflective_Tautologies_I, conject

SWB016+1.p Reflective Tautologies II
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subPropertyOf, uri_owl_equivalentClass, uri_rdfs_subClassOf)   fof(testcase_conclusion_fullish_016_Reflective_T

SWB016+2.p Reflective Tautologies II
$\forall p$: (iext(uri_rdf_type, p, uri_rdf_Property) $\iff$ ip(p))   fof(rdf_type_ip, axiom)
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x))   fof(rdfs_cext_def, axiom)
$\forall p, c, x, y$: ((iext(uri_rdfs_domain, p, c) and iext(p, x, y)) $\Rightarrow$ icext(c, x))   fof(rdfs_domain_main, axiom)
iext(uri_rdfs_domain, uri_rdfs_domain, uri_rdf_Property)   fof(rdfs_domain_domain, axiom)
iext(uri_rdfs_domain, uri_rdfs_subClassOf, uri_rdfs_Class)   fof(rdfs_subclassof_domain, axiom)
ip(uri_owl_equivalentClass)   fof(owl_prop_equivalentclass_type, axiom)
$\forall x, y$: (iext(uri_owl_equivalentClass, x, y) $\Rightarrow$ (ic(x) and ic(y)))   fof(owl_prop_equivalentclass_ext, axiom)
$\forall c_1, c_2$: (iext(uri_rdfs_subClassOf, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: (icext(c_1, x) $\Rightarrow$ icext(c_2, x))))   fof(owl_rdfsext_su
$\forall p_1, p_2$: (iext(uri_rdfs_subPropertyOf, p_1, p_2) $\iff$ (ip(p_1) and ip(p_2) and $\forall x, y$: (iext(p_1, x, y) $\Rightarrow$ iext(p_2, x, y))))   fof(owl_
$\forall c_1, c_2$: (iext(uri_owl_equivalentClass, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: (icext(c_1, x) $\iff$ icext(c_2, x))))   fof(owl_eqdi
iext(uri_rdfs_subPropertyOf, uri_owl_equivalentClass, uri_rdfs_subClassOf)   fof(testcase_conclusion_fullish_016_Reflective_T

SWB016+3.p Reflective Tautologies II
include('Axioms/SWB002+0.ax')
iext(uri_rdfs_subPropertyOf, uri_owl_equivalentClass, uri_rdfs_subClassOf)   fof(testcase_conclusion_fullish_016_Reflective_T

SWB016+4.p Reflective Tautologies II
include('Axioms/SWB003+0.ax')
iext(uri_rdfs_subPropertyOf, uri_owl_equivalentClass, uri_rdfs_subClassOf)   fof(testcase_conclusion_fullish_016_Reflective_T

SWB017+1.p Built-in Based Definitions
include('Axioms/SWB001+0.ax')
iext(uri_owl_differentFrom, uri_ex_w, uri_ex_u)   fof(testcase_conclusion_fullish_017_Built_in_Based_Definitions, conjecture)
iext(uri_owl_propertyDisjointWith, uri_ex_notInstanceOf, uri_rdf_type) and iext(uri_rdf_type, uri_ex_w, uri_ex_c) and iext(uri_e

SWB017+2.p Built-in Based Definitions
$\forall x, y$: (iext(uri_owl_differentFrom, x, y) $\iff$ $x \neq y$)   fof(owl_eqdis_differentfrom, axiom)
$\forall p_1, p_2$: (iext(uri_owl_propertyDisjointWith, p_1, p_2) $\iff$ (ip(p_1) and ip(p_2) and $\forall x, y$: $\neg$ iext(p_1, x, y) and iext(p_2, x, y)))
iext(uri_owl_differentFrom, uri_ex_w, uri_ex_u)   fof(testcase_conclusion_fullish_017_Built_in_Based_Definitions, conjecture)
iext(uri_owl_propertyDisjointWith, uri_ex_notInstanceOf, uri_rdf_type) and iext(uri_rdf_type, uri_ex_w, uri_ex_c) and iext(uri_e

SWB017+3.p Built-in Based Definitions
include('Axioms/SWB002+0.ax')
iext(uri_owl_differentFrom, uri_ex_w, uri_ex_u)   fof(testcase_conclusion_fullish_017_Built_in_Based_Definitions, conjecture)
iext(uri_owl_propertyDisjointWith, uri_ex_notInstanceOf, uri_rdf_type) and iext(uri_rdf_type, uri_ex_w, uri_ex_c) and iext(uri_e

SWB017+4.p Built-in Based Definitions
include('Axioms/SWB003+0.ax')
iext(uri_owl_differentFrom, uri_ex_w, uri_ex_u)   fof(testcase_conclusion_fullish_017_Built_in_Based_Definitions, conjecture)
iext(uri_owl_propertyDisjointWith, uri_ex_notInstanceOf, uri_rdf_type) and iext(uri_rdf_type, uri_ex_w, uri_ex_c) and iext(uri_e

SWB018+1.p Modified Logical Vocabulary Semantics
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_u, uri_ex_Person)   fof(testcase_conclusion_fullish_018_Modified_Logical_Vocabulary_Semantics, conj
iext(uri_rdfs_domain, uri_owl_sameAs, uri_ex_Person) and iext(uri_owl_sameAs, uri_ex_w, uri_ex_u)   fof(testcase_premise_ful

SWB018+2.p Modified Logical Vocabulary Semantics
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x))   fof(rdfs_cext_def, axiom)
$\forall p, c, x, y$: ((iext(uri_rdfs_domain, p, c) and iext(p, x, y)) $\Rightarrow$ icext(c, x))   fof(rdfs_domain_main, axiom)
$\forall x, y$: (iext(uri_owl_sameAs, x, y) $\iff$ $x = y$)   fof(owl_eqdis_sameas, axiom)
iext(uri_rdf_type, uri_ex_u, uri_ex_Person)   fof(testcase_conclusion_fullish_018_Modified_Logical_Vocabulary_Semantics, conj
iext(uri_rdfs_domain, uri_owl_sameAs, uri_ex_Person) and iext(uri_owl_sameAs, uri_ex_w, uri_ex_u)   fof(testcase_premise_ful

SWB018+3.p Modified Logical Vocabulary Semantics

include('Axioms/SWB002+0.ax')
iext(uri_rdf_type, uri_ex_u, uri_ex_Person)   fof(testcase_conclusion_fullish_018_Modified_Logical_Vocabulary_Semantics, conj
iext(uri_rdfs_domain, uri_owl_sameAs, uri_ex_Person) and iext(uri_owl_sameAs, uri_ex_w, uri_ex_u)   fof(testcase_premise_ful

SWB018+4.p Modified Logical Vocabulary Semantics
include('Axioms/SWB003+0.ax')
iext(uri_rdf_type, uri_ex_u, uri_ex_Person)   fof(testcase_conclusion_fullish_018_Modified_Logical_Vocabulary_Semantics, conj
iext(uri_rdfs_domain, uri_owl_sameAs, uri_ex_Person) and iext(uri_owl_sameAs, uri_ex_w, uri_ex_u)   fof(testcase_premise_ful

SWB019+1.p Disjoint Annotation Properties
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_skos_prefLabel, uri_owl_AnnotationProperty) and iext(uri_rdfs_subPropertyOf, uri_skos_prefLabel, uri_rd

SWB019+2.p Disjoint Annotation Properties
$\forall p_1, p_2$: (iext(uri_owl_propertyDisjointWith, p_1, p_2) $\iff$ (ip(p_1) and ip(p_2) and $\forall x, y$: $\neg$ iext(p_1, x, y) and iext(p_2, x, y)))   f
iext(uri_rdf_type, uri_skos_prefLabel, uri_owl_AnnotationProperty) and iext(uri_rdfs_subPropertyOf, uri_skos_prefLabel, uri_rd

SWB019+3.p Disjoint Annotation Properties
include('Axioms/SWB002+0.ax')
iext(uri_rdf_type, uri_skos_prefLabel, uri_owl_AnnotationProperty) and iext(uri_rdfs_subPropertyOf, uri_skos_prefLabel, uri_rd

SWB019+4.p Disjoint Annotation Properties
include('Axioms/SWB003+0.ax')
iext(uri_rdf_type, uri_skos_prefLabel, uri_owl_AnnotationProperty) and iext(uri_rdfs_subPropertyOf, uri_skos_prefLabel, uri_rd

SWB020+1.p Logical Complications
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_d, uri_ex_c$_3$)   fof(testcase_conclusion_fullish_020_Logical_Complications, conjecture)
$\exists$bNODE_xs, bNODE_xc, bNODE_lu$_1$, bNODE_lu$_2$, bNODE_lu$_3$, bNODE_li$_1$, bNODE_li$_2$: (iext(uri_owl_unionOf, uri_ex_c, bNO

SWB020+2.p Logical Complications
$\forall x, y$: (iext(uri_owl_disjointWith, x, y) $\Rightarrow$ (ic(x) and ic(y)))   fof(owl_prop_disjointwith_ext, axiom)
$\forall z, c$: (iext(uri_owl_complementOf, z, c) $\Rightarrow$ (ic(z) and ic(c) and $\forall x$: (icext(z, x) $\iff$ $\neg$ icext(c, x))))   fof(owl_bool_complem
$\forall z, s_1, c_1, s_2, c_2$: ((iext(uri_rdf_first, s_1, c_1) and iext(uri_rdf_rest, s_1, s_2) and iext(uri_rdf_first, s_2, c_2) and iext(uri_rdf_rest, s_2, ur
(iext(uri_owl_intersectionOf, z, s_1) $\iff$ (ic(z) and ic(c_1) and ic(c_2) and $\forall x$: (icext(z, x) $\iff$ (icext(c_1, x) and icext(c_2, x))))
$\forall z, s_1, c_1, s_2, c_2, s_3, c_3$: ((iext(uri_rdf_first, s_1, c_1) and iext(uri_rdf_rest, s_1, s_2) and iext(uri_rdf_first, s_2, c_2) and iext(uri_rdf_rest,
(iext(uri_owl_unionOf, z, s_1) $\iff$ (ic(z) and ic(c_1) and ic(c_2) and ic(c_3) and $\forall x$: (icext(z, x) $\iff$ (icext(c_1, x) or icext(c_2, x
$\forall c_1, c_2$: (iext(uri_rdfs_subClassOf, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: (icext(c_1, x) $\Rightarrow$ icext(c_2, x))))   fof(owl_rdfsext_su
$\forall c_1, c_2$: (iext(uri_owl_disjointWith, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: $\neg$ icext(c_1, x) and icext(c_2, x)))   fof(owl_eqdis_dis
iext(uri_rdfs_subClassOf, uri_ex_d, uri_ex_c$_3$)   fof(testcase_conclusion_fullish_020_Logical_Complications, conjecture)
$\exists$bNODE_xs, bNODE_xc, bNODE_lu$_1$, bNODE_lu$_2$, bNODE_lu$_3$, bNODE_li$_1$, bNODE_li$_2$: (iext(uri_owl_unionOf, uri_ex_c, bNO

SWB020+3.p Logical Complications
include('Axioms/SWB002+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_d, uri_ex_c$_3$)   fof(testcase_conclusion_fullish_020_Logical_Complications, conjecture)
$\exists$bNODE_xs, bNODE_xc, bNODE_lu$_1$, bNODE_lu$_2$, bNODE_lu$_3$, bNODE_li$_1$, bNODE_li$_2$: (iext(uri_owl_unionOf, uri_ex_c, bNO

SWB020+4.p Logical Complications
include('Axioms/SWB003+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_d, uri_ex_c$_3$)   fof(testcase_conclusion_fullish_020_Logical_Complications, conjecture)
$\exists$bNODE_xs, bNODE_xc, bNODE_lu$_1$, bNODE_lu$_2$, bNODE_lu$_3$, bNODE_li$_1$, bNODE_li$_2$: (iext(uri_owl_unionOf, uri_ex_c, bNO

SWB021+1.p Composite Enumerations
include('Axioms/SWB001+0.ax')
iext(uri_owl_equivalentClass, uri_ex_c$_3$, uri_ex_c$_4$)   fof(testcase_conclusion_fullish_021_Composite_Enumerations, conjecture)
$\exists$bNODE_l$_{11}$, bNODE_l$_{12}$, bNODE_l$_{21}$, bNODE_l$_{22}$, bNODE_l$_{31}$, bNODE_l$_{32}$, bNODE_l$_{33}$, bNODE_l$_{41}$, bNODE_l$_{42}$: (iext(uri_o

SWB021+2.p Composite Enumerations
$\forall x, y$: (iext(uri_owl_oneOf, x, y) $\Rightarrow$ (ic(x) and icext(uri_rdf_List, y)))   fof(owl_prop_oneof_ext, axiom)
$\forall x, y$: (iext(uri_owl_unionOf, x, y) $\Rightarrow$ (ic(x) and icext(uri_rdf_List, y)))   fof(owl_prop_unionof_ext, axiom)
$\forall z, s_1, c_1, s_2, c_2$: ((iext(uri_rdf_first, s_1, c_1) and iext(uri_rdf_rest, s_1, s_2) and iext(uri_rdf_first, s_2, c_2) and iext(uri_rdf_rest, s_2, ur
(iext(uri_owl_unionOf, z, s_1) $\iff$ (ic(z) and ic(c_1) and ic(c_2) and $\forall x$: (icext(z, x) $\iff$ (icext(c_1, x) or icext(c_2, x))))))   fo
$\forall z, s_1, a_1, s_2, a_2$: ((iext(uri_rdf_first, s_1, a_1) and iext(uri_rdf_rest, s_1, s_2) and iext(uri_rdf_first, s_2, a_2) and iext(uri_rdf_rest, s_2, u
(iext(uri_owl_oneOf, z, s_1) $\iff$ (ic(z) and $\forall x$: (icext(z, x) $\iff$ ($x = a_1$ or $x = a_2$)))))   fof(owl_enum_class$_{002}$, axiom)
$\forall z, s_1, a_1, s_2, a_2, s_3, a_3$: ((iext(uri_rdf_first, s_1, a_1) and iext(uri_rdf_rest, s_1, s_2) and iext(uri_rdf_first, s_2, a_2) and iext(uri_rdf_res
(iext(uri_owl_oneOf, z, s_1) $\iff$ (ic(z) and $\forall x$: (icext(z, x) $\iff$ ($x = a_1$ or $x = a_2$ or $x = a_3$)))))   fof(owl_enum_class$_{003}$, a

$\forall c_1, c_2$: (iext(uri_owl_equivalentClass, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: (icext(c_1, x) $\iff$ icext(c_2, x)))) fof(owl_eqdi
iext(uri_owl_equivalentClass, uri_ex_c$_3$, uri_ex_c$_4$) fof(testcase_conclusion_fullish_021_Composite_Enumerations, conjecture)
$\exists$bNODE_l$_{11}$, bNODE_l$_{12}$, bNODE_l$_{21}$, bNODE_l$_{22}$, bNODE_l$_{31}$, bNODE_l$_{32}$, bNODE_l$_{33}$, bNODE_l$_{41}$, bNODE_l$_{42}$: (iext(uri_o

SWB021+3.p Composite Enumerations
include('Axioms/SWB002+0.ax')
iext(uri_owl_equivalentClass, uri_ex_c$_3$, uri_ex_c$_4$) fof(testcase_conclusion_fullish_021_Composite_Enumerations, conjecture)
$\exists$bNODE_l$_{11}$, bNODE_l$_{12}$, bNODE_l$_{21}$, bNODE_l$_{22}$, bNODE_l$_{31}$, bNODE_l$_{32}$, bNODE_l$_{33}$, bNODE_l$_{41}$, bNODE_l$_{42}$: (iext(uri_o

SWB021+4.p Composite Enumerations
include('Axioms/SWB003+0.ax')
iext(uri_owl_equivalentClass, uri_ex_c$_3$, uri_ex_c$_4$) fof(testcase_conclusion_fullish_021_Composite_Enumerations, conjecture)
$\exists$bNODE_l$_{11}$, bNODE_l$_{12}$, bNODE_l$_{21}$, bNODE_l$_{22}$, bNODE_l$_{31}$, bNODE_l$_{32}$, bNODE_l$_{33}$, bNODE_l$_{41}$, bNODE_l$_{42}$: (iext(uri_o

SWB022+1.p List Member Access
include('Axioms/SWB001+0.ax')
iext(uri_skos_member, uri_ex_MyOrderedCollection, uri_ex_X) and iext(uri_skos_member, uri_ex_MyOrderedCollection, uri_ex_
$\exists$bNODE_pL, bNODE_l$_{11}$, bNODE_l$_{12}$, bNODE_l$_{21}$, bNODE_l$_{22}$, bNODE_l$_{31}$, bNODE_l$_{32}$, bNODE_l$_{33}$: (iext(uri_rdfs_subPrope

SWB022+2.p List Member Access
$\forall p, q$: (iext(uri_rdfs_subPropertyOf, p, q) $\Rightarrow$ (ip(p) and ip(q) and $\forall x, y$: (iext(p, x, y) $\Rightarrow$ iext(q, x, y)))) fof(rdfs_subproper
$\forall p, s_1, p_1, s_2, p_2$: ((iext(uri_rdf_first, s_1, p_1) and iext(uri_rdf_rest, s_1, s_2) and iext(uri_rdf_first, s_2, p_2) and iext(uri_rdf_rest, s_2, ur
(iext(uri_owl_propertyChainAxiom, p, s_1) $\iff$ (ip(p) and ip(p_1) and ip(p_2) and $\forall y_0, y_1, y_2$: ((iext(p_1, y_0, y_1) and iext($p_2, y_1,$
iext(p, y_0, y_2))))) fof(owl_chain$_{002}$, axiom)
iext(uri_skos_member, uri_ex_MyOrderedCollection, uri_ex_X) and iext(uri_skos_member, uri_ex_MyOrderedCollection, uri_ex_
$\exists$bNODE_pL, bNODE_l$_{11}$, bNODE_l$_{12}$, bNODE_l$_{21}$, bNODE_l$_{22}$, bNODE_l$_{31}$, bNODE_l$_{32}$, bNODE_l$_{33}$: (iext(uri_rdfs_subPrope

SWB022+3.p List Member Access
include('Axioms/SWB002+0.ax')
iext(uri_skos_member, uri_ex_MyOrderedCollection, uri_ex_X) and iext(uri_skos_member, uri_ex_MyOrderedCollection, uri_ex_
$\exists$bNODE_pL, bNODE_l$_{11}$, bNODE_l$_{12}$, bNODE_l$_{21}$, bNODE_l$_{22}$, bNODE_l$_{31}$, bNODE_l$_{32}$, bNODE_l$_{33}$: (iext(uri_rdfs_subPrope

SWB022+4.p List Member Access
include('Axioms/SWB003+0.ax')
iext(uri_skos_member, uri_ex_MyOrderedCollection, uri_ex_X) and iext(uri_skos_member, uri_ex_MyOrderedCollection, uri_ex_
$\exists$bNODE_pL, bNODE_l$_{11}$, bNODE_l$_{12}$, bNODE_l$_{21}$, bNODE_l$_{22}$, bNODE_l$_{31}$, bNODE_l$_{32}$, bNODE_l$_{33}$: (iext(uri_rdfs_subPrope

SWB023+1.p Unique List Components
include('Axioms/SWB001+0.ax')
iext(uri_owl_sameAs, uri_ex_w, uri_ex_u) and iext(uri_owl_sameAs, uri_ex_w, uri_ex_v) fof(testcase_conclusion_fullish_023_U
$\exists$bNODE_o, bNODE_l: (iext(uri_rdf_type, uri_rdf_first, uri_owl_FunctionalProperty) and iext(uri_rdf_type, uri_ex_w, bNODE_o)

SWB023+2.p Unique List Components
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x)) fof(rdfs_cext_def, axiom)
$\forall z, s_1, a_1$: ((iext(uri_rdf_first, s_1, a_1) and iext(uri_rdf_rest, s_1, uri_rdf_nil)) $\Rightarrow$ (iext(uri_owl_oneOf, z, s_1) $\iff$ (ic(z) and $\forall x$: (i
$x = a_1$)))) fof(owl_enum_class$_{001}$, axiom)
$\forall p$: (icext(uri_owl_FunctionalProperty, p) $\iff$ (ip(p) and $\forall x, y_1, y_2$: ((iext(p, x, y_1) and iext(p, x, y_2)) $\Rightarrow$ $y_1 =$
y_2))) fof(owl_char_functional, axiom)
$\forall x, y$: (iext(uri_owl_sameAs, x, y) $\iff$ $x = y$) fof(owl_eqdis_sameas, axiom)
iext(uri_owl_sameAs, uri_ex_w, uri_ex_u) and iext(uri_owl_sameAs, uri_ex_w, uri_ex_v) fof(testcase_conclusion_fullish_023_U
$\exists$bNODE_o, bNODE_l: (iext(uri_rdf_type, uri_rdf_first, uri_owl_FunctionalProperty) and iext(uri_rdf_type, uri_ex_w, bNODE_o)

SWB023+3.p Unique List Components
include('Axioms/SWB002+0.ax')
iext(uri_owl_sameAs, uri_ex_w, uri_ex_u) and iext(uri_owl_sameAs, uri_ex_w, uri_ex_v) fof(testcase_conclusion_fullish_023_U
$\exists$bNODE_o, bNODE_l: (iext(uri_rdf_type, uri_rdf_first, uri_owl_FunctionalProperty) and iext(uri_rdf_type, uri_ex_w, bNODE_o)

SWB023+4.p Unique List Components
include('Axioms/SWB003+0.ax')
iext(uri_owl_sameAs, uri_ex_w, uri_ex_u) and iext(uri_owl_sameAs, uri_ex_w, uri_ex_v) fof(testcase_conclusion_fullish_023_U
$\exists$bNODE_o, bNODE_l: (iext(uri_rdf_type, uri_rdf_first, uri_owl_FunctionalProperty) and iext(uri_rdf_type, uri_ex_w, bNODE_o)

SWB024+1.p Cardinality Restrictions on Complex Properties
include('Axioms/SWB001+0.ax')
$\exists$bNODE_x: (iext(uri_ex_hasAncestor, uri_ex_bob, bNODE_x) and iext(uri_ex_hasAncestor, uri_ex_alice, bNODE_x)) fof(te
$\exists$bNODE_z: (iext(uri_rdf_type, uri_ex_hasAncestor, uri_owl_TransitiveProperty) and iext(uri_rdfs_subClassOf, uri_ex_Person, b

SWB024+2.p Cardinality Restrictions on Complex Properties
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x)) fof(rdfs_cext_def, axiom)
$\forall z, p$: ((iext(uri_owl_minCardinality, z, literal_typed(dat_str$_1$, uri_xsd_nonNegativeInteger)) and iext(uri_owl_onProperty, z, p))
$\forall x$: (icext(z, x) $\iff$ $\exists y$: iext(p, x, y))) fof(owl_restrict_mincard$_{001}$, axiom)
$\forall c_1, c_2$: (iext(uri_rdfs_subClassOf, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: (icext(c_1, x) $\Rightarrow$ icext(c_2, x)))) fof(owl_rdfsext_su
$\forall p$: (icext(uri_owl_TransitiveProperty, p) $\iff$ (ip(p) and $\forall x, y, z$: ((iext(p, x, y) and iext(p, y, z)) $\Rightarrow$ iext(p, x, z)))) fof(ow
$\exists$bNODE_x: (iext(uri_ex_hasAncestor, uri_ex_bob, bNODE_x) and iext(uri_ex_hasAncestor, uri_ex_alice, bNODE_x)) fof(te
$\exists$bNODE_z: (iext(uri_rdf_type, uri_ex_hasAncestor, uri_owl_TransitiveProperty) and iext(uri_rdfs_subClassOf, uri_ex_Person, b

SWB024+3.p Cardinality Restrictions on Complex Properties
include('Axioms/SWB002+0.ax')
$\exists$bNODE_x: (iext(uri_ex_hasAncestor, uri_ex_bob, bNODE_x) and iext(uri_ex_hasAncestor, uri_ex_alice, bNODE_x)) fof(te
$\exists$bNODE_z: (iext(uri_rdf_type, uri_ex_hasAncestor, uri_owl_TransitiveProperty) and iext(uri_rdfs_subClassOf, uri_ex_Person, b

SWB024+4.p Cardinality Restrictions on Complex Properties
include('Axioms/SWB003+0.ax')
$\exists$bNODE_x: (iext(uri_ex_hasAncestor, uri_ex_bob, bNODE_x) and iext(uri_ex_hasAncestor, uri_ex_alice, bNODE_x)) fof(te
$\exists$bNODE_z: (iext(uri_rdf_type, uri_ex_hasAncestor, uri_owl_TransitiveProperty) and iext(uri_rdfs_subClassOf, uri_ex_Person, b

SWB025+1.p Cyclic Dependencies between Complex Properties
include('Axioms/SWB001+0.ax')
iext(uri_ex_hasUncle, uri_ex_alice, uri_ex_charly) and iext(uri_ex_hasCousin, uri_ex_bob, uri_ex_alice) fof(testcase_conclusio
$\exists$bNODE_l$_{11}$, bNODE_l$_{12}$, bNODE_l$_{21}$, bNODE_l$_{22}$, bNODE_l$_3$: (iext(uri_owl_propertyChainAxiom, uri_ex_hasUncle, bNODE_

SWB025+2.p Cyclic Dependencies between Complex Properties
$\forall p, s_1, p_1, s_2, p_2$: ((iext(uri_rdf_first, s_1, p_1) and iext(uri_rdf_rest, s_1, s_2) and iext(uri_rdf_first, s_2, p_2) and iext(uri_rdf_rest, s_2, u
(iext(uri_owl_propertyChainAxiom, p, s_1) $\iff$ (ip(p) and ip(p_1) and ip(p_2) and $\forall y_0, y_1, y_2$: ((iext(p_1, y_0, y_1) and iext(p_2, y_1,
iext(p, y_0, y_2))))) fof(owl_chain$_{002}$, axiom)
$\forall p_1, p_2$: (iext(uri_owl_inverseOf, p_1, p_2) $\iff$ (ip(p_1) and ip(p_2) and $\forall x, y$: (iext(p_1, x, y) $\iff$ iext(p_2, y, x)))) fof(owl_inv,
iext(uri_ex_hasUncle, uri_ex_alice, uri_ex_charly) and iext(uri_ex_hasCousin, uri_ex_bob, uri_ex_alice) fof(testcase_conclusio
$\exists$bNODE_l$_{11}$, bNODE_l$_{12}$, bNODE_l$_{21}$, bNODE_l$_{22}$, bNODE_l$_3$: (iext(uri_owl_propertyChainAxiom, uri_ex_hasUncle, bNODE_

SWB025+3.p Cyclic Dependencies between Complex Properties
include('Axioms/SWB002+0.ax')
iext(uri_ex_hasUncle, uri_ex_alice, uri_ex_charly) and iext(uri_ex_hasCousin, uri_ex_bob, uri_ex_alice) fof(testcase_conclusio
$\exists$bNODE_l$_{11}$, bNODE_l$_{12}$, bNODE_l$_{21}$, bNODE_l$_{22}$, bNODE_l$_3$: (iext(uri_owl_propertyChainAxiom, uri_ex_hasUncle, bNODE_

SWB025+4.p Cyclic Dependencies between Complex Properties
include('Axioms/SWB003+0.ax')
iext(uri_ex_hasUncle, uri_ex_alice, uri_ex_charly) and iext(uri_ex_hasCousin, uri_ex_bob, uri_ex_alice) fof(testcase_conclusio
$\exists$bNODE_l$_{11}$, bNODE_l$_{12}$, bNODE_l$_{21}$, bNODE_l$_{22}$, bNODE_l$_3$: (iext(uri_owl_propertyChainAxiom, uri_ex_hasUncle, bNODE_

SWB026+1.p Inferred Property Characteristics I
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_p, uri_owl_InverseFunctionalProperty) fof(testcase_conclusion_fullish_026_Inferred_Property_Chara
$\exists$bNODE_x$_1$, bNODE_x$_2$, bNODE_l$_1$, bNODE_l$_2$: (iext(uri_rdfs_domain, uri_ex_p, bNODE_x$_1$) and iext(uri_owl_oneOf, bNODE

SWB026+2.p Inferred Property Characteristics I
$\forall p$: (iext(uri_rdf_type, p, uri_rdf_Property) $\iff$ ip(p)) fof(rdf_type_ip, axiom)
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x)) fof(rdfs_cext_def, axiom)
$\forall p, c, x, y$: ((iext(uri_rdfs_domain, p, c) and iext(p, x, y)) $\Rightarrow$ icext(c, x)) fof(rdfs_domain_main, axiom)
iext(uri_rdfs_domain, uri_rdfs_domain, uri_rdf_Property) fof(rdfs_domain_domain, axiom)
$\forall z, s_1, a_1$: ((iext(uri_rdf_first, s_1, a_1) and iext(uri_rdf_rest, s_1, uri_rdf_nil)) $\Rightarrow$ (iext(uri_owl_oneOf, z, s_1) $\iff$ (ic(z) and $\forall x$: (i
$x = a_1$)))) fof(owl_enum_class$_{001}$, axiom)
$\forall p$: (icext(uri_owl_InverseFunctionalProperty, p) $\iff$ (ip(p) and $\forall x_1, x_2, y$: ((iext(p, x_1, y) and iext(p, x_2, y)) $\Rightarrow$
$x_1 = x_2$))) fof(owl_char_inversefunctional, axiom)
iext(uri_rdf_type, uri_ex_p, uri_owl_InverseFunctionalProperty) fof(testcase_conclusion_fullish_026_Inferred_Property_Chara
$\exists$bNODE_x$_1$, bNODE_x$_2$, bNODE_l$_1$, bNODE_l$_2$: (iext(uri_rdfs_domain, uri_ex_p, bNODE_x$_1$) and iext(uri_owl_oneOf, bNODE

SWB026+3.p Inferred Property Characteristics I
include('Axioms/SWB002+0.ax')
iext(uri_rdf_type, uri_ex_p, uri_owl_InverseFunctionalProperty) fof(testcase_conclusion_fullish_026_Inferred_Property_Chara
$\exists$bNODE_x$_1$, bNODE_x$_2$, bNODE_l$_1$, bNODE_l$_2$: (iext(uri_rdfs_domain, uri_ex_p, bNODE_x$_1$) and iext(uri_owl_oneOf, bNODE

SWB026+4.p Inferred Property Characteristics I

include(’Axioms/SWB003+0.ax’)
iext(uri_rdf_type, uri_ex_p, uri_owl_InverseFunctionalProperty)   fof(testcase_conclusion_fullish_026_Inferred_Property_Chara
$\exists$bNODE_x$_1$, bNODE_x$_2$, bNODE_l$_1$, bNODE_l$_2$: (iext(uri_rdfs_domain, uri_ex_p, bNODE_x$_1$) and iext(uri_owl_oneOf, bNODE

SWB027+1.p Inferred Property Characteristics II
include(’Axioms/SWB001+0.ax’)
iext(uri_rdf_type, uri_ex_p, uri_owl_InverseFunctionalProperty)   fof(testcase_conclusion_fullish_027_Inferred_Property_Chara
$\exists$bNODE_l$_1$, bNODE_l$_2$, bNODE_v: (iext(uri_owl_propertyChainAxiom, uri_owl_sameAs, bNODE_l$_1$) and iext(uri_rdf_first, bN

SWB027+2.p Inferred Property Characteristics II
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x))   fof(rdfs_cext_def, axiom)
$\forall x, y$: (iext(uri_owl_sameAs, x, y) $\iff$ $x = y$)   fof(owl_eqdis_sameas, axiom)
$\forall p, s_1, p_1, s_2, p_2$: ((iext(uri_rdf_first, s_1, p_1) and iext(uri_rdf_rest, s_1, s_2) and iext(uri_rdf_first, s_2, p_2) and iext(uri_rdf_rest, s_2, ur
(iext(uri_owl_propertyChainAxiom, p, s_1) $\iff$ (ip(p) and ip(p_1) and ip(p_2) and $\forall y_0, y_1, y_2$: ((iext(p_1, y_0, y_1) and iext(p_2, y_1, y
iext(p, y_0, y_2)))))   fof(owl_chain$_{002}$, axiom)
$\forall p$: (icext(uri_owl_InverseFunctionalProperty, p) $\iff$ (ip(p) and $\forall x_1, x_2, y$: ((iext(p, x_1, y) and iext(p, x_2, y)) $\Rightarrow$ $x_1 = x_2$)))   fof(owl_char_inversefunctional, axiom)
$\forall p_1, p_2$: (iext(uri_owl_inverseOf, p_1, p_2) $\iff$ (ip(p_1) and ip(p_2) and $\forall x, y$: (iext(p_1, x, y) $\iff$ iext(p_2, y, x))))   fof(owl_inv,
iext(uri_rdf_type, uri_ex_p, uri_owl_InverseFunctionalProperty)   fof(testcase_conclusion_fullish_027_Inferred_Property_Chara
$\exists$bNODE_l$_1$, bNODE_l$_2$, bNODE_v: (iext(uri_owl_propertyChainAxiom, uri_owl_sameAs, bNODE_l$_1$) and iext(uri_rdf_first, bN

SWB027+3.p Inferred Property Characteristics II
include(’Axioms/SWB002+0.ax’)
iext(uri_rdf_type, uri_ex_p, uri_owl_InverseFunctionalProperty)   fof(testcase_conclusion_fullish_027_Inferred_Property_Chara
$\exists$bNODE_l$_1$, bNODE_l$_2$, bNODE_v: (iext(uri_owl_propertyChainAxiom, uri_owl_sameAs, bNODE_l$_1$) and iext(uri_rdf_first, bN

SWB027+4.p Inferred Property Characteristics II
include(’Axioms/SWB003+0.ax’)
iext(uri_rdf_type, uri_ex_p, uri_owl_InverseFunctionalProperty)   fof(testcase_conclusion_fullish_027_Inferred_Property_Chara
$\exists$bNODE_l$_1$, bNODE_l$_2$, bNODE_v: (iext(uri_owl_propertyChainAxiom, uri_owl_sameAs, bNODE_l$_1$) and iext(uri_rdf_first, bN

SWB028+1.p Inferred Property Characteristics III
include(’Axioms/SWB001+0.ax’)
iext(uri_rdfs_subClassOf, uri_ex_InversesOfFunctionalProperties, uri_owl_InverseFunctionalProperty)   fof(testcase_conclusio
$\exists$bNODE_z: (iext(uri_owl_equivalentClass, uri_ex_InversesOfFunctionalProperties, bNODE_z) and iext(uri_rdf_type, bNODE_z

SWB028+2.p Inferred Property Characteristics III
ic(uri_owl_InverseFunctionalProperty)   fof(owl_class_inversefunctionalproperty_type, axiom)
$\forall x, y$: (iext(uri_owl_inverseOf, x, y) $\Rightarrow$ (ip(x) and ip(y)))   fof(owl_prop_inverseof_ext, axiom)
$\forall x, y$: (iext(uri_owl_equivalentClass, x, y) $\Rightarrow$ (ic(x) and ic(y)))   fof(owl_prop_equivalentclass_ext, axiom)
$\forall p$: (icext(uri_owl_FunctionalProperty, p) $\iff$ (ip(p) and $\forall x, y_1, y_2$: ((iext(p, x, y_1) and iext(p, x, y_2)) $\Rightarrow$ $y_1 = y_2$)))   fof(owl_char_functional, axiom)
$\forall p$: (icext(uri_owl_InverseFunctionalProperty, p) $\iff$ (ip(p) and $\forall x_1, x_2, y$: ((iext(p, x_1, y) and iext(p, x_2, y)) $\Rightarrow$ $x_1 = x_2$)))   fof(owl_char_inversefunctional, axiom)
$\forall c_1, c_2$: (iext(uri_rdfs_subClassOf, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: (icext(c_1, x) $\Rightarrow$ icext(c_2, x))))   fof(owl_rdfsext_su
$\forall c_1, c_2$: (iext(uri_owl_equivalentClass, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: (icext(c_1, x) $\iff$ icext(c_2, x))))   fof(owl_eqdi
$\forall z, p, c$: ((iext(uri_owl_someValuesFrom, z, c) and iext(uri_owl_onProperty, z, p)) $\Rightarrow$ $\forall x$: (icext(z, x) $\iff$ $\exists y$: (iext(p, x, y) an
$\forall p_1, p_2$: (iext(uri_owl_inverseOf, p_1, p_2) $\iff$ (ip(p_1) and ip(p_2) and $\forall x, y$: (iext(p_1, x, y) $\iff$ iext(p_2, y, x))))   fof(owl_inv,
iext(uri_rdfs_subClassOf, uri_ex_InversesOfFunctionalProperties, uri_owl_InverseFunctionalProperty)   fof(testcase_conclusio
$\exists$bNODE_z: (iext(uri_owl_equivalentClass, uri_ex_InversesOfFunctionalProperties, bNODE_z) and iext(uri_rdf_type, bNODE_z

SWB028+3.p Inferred Property Characteristics III
include(’Axioms/SWB002+0.ax’)
iext(uri_rdfs_subClassOf, uri_ex_InversesOfFunctionalProperties, uri_owl_InverseFunctionalProperty)   fof(testcase_conclusio
$\exists$bNODE_z: (iext(uri_owl_equivalentClass, uri_ex_InversesOfFunctionalProperties, bNODE_z) and iext(uri_rdf_type, bNODE_z

SWB028+4.p Inferred Property Characteristics III
include(’Axioms/SWB003+0.ax’)
iext(uri_rdfs_subClassOf, uri_ex_InversesOfFunctionalProperties, uri_owl_InverseFunctionalProperty)   fof(testcase_conclusio
$\exists$bNODE_z: (iext(uri_owl_equivalentClass, uri_ex_InversesOfFunctionalProperties, bNODE_z) and iext(uri_rdf_type, bNODE_z

SWB029+1.p Ex Falso Quodlibet
include(’Axioms/SWB001+0.ax’)
iext(uri_rdf_type, uri_ex_w, uri_ex_B)   fof(testcase_conclusion_fullish_029_Ex_Falso_Quodlibet, conjecture)
$\exists$bNODE_x, bNODE_y, bNODE_l$_1$, bNODE_l$_2$: (iext(uri_rdf_type, uri_ex_A, uri_owl_Class) and iext(uri_rdf_type, uri_ex_B, uri_

SWB029+2.p Ex Falso Quodlibet
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x)) fof(rdfs_cext_def, axiom)
$\forall z, c$: (iext(uri_owl_complementOf, z, c) $\Rightarrow$ (ic(z) and ic(c) and $\forall x$: (icext(z, x) $\iff$ $\neg$ icext(c, x)))) fof(owl_bool_comple
$\forall z, s_1, c_1, s_2, c_2$: ((iext(uri_rdf_first, s_1, c_1) and iext(uri_rdf_rest, s_1, s_2) and iext(uri_rdf_first, s_2, c_2) and iext(uri_rdf_rest, s_2, ur
(iext(uri_owl_intersectionOf, z, s_1) $\iff$ (ic(z) and ic(c_1) and ic(c_2) and $\forall x$: (icext(z, x) $\iff$ (icext(c_1, x) and icext(c_2, x))))
iext(uri_rdf_type, uri_ex_w, uri_ex_B) fof(testcase_conclusion_fullish_029_Ex_Falso_Quodlibet, conjecture)
$\exists$bNODE_x, bNODE_y, bNODE_l$_1$, bNODE_l$_2$: (iext(uri_rdf_type, uri_ex_A, uri_owl_Class) and iext(uri_rdf_type, uri_ex_B, uri_

SWB029+3.p Ex Falso Quodlibet
include('Axioms/SWB002+0.ax')
iext(uri_rdf_type, uri_ex_w, uri_ex_B) fof(testcase_conclusion_fullish_029_Ex_Falso_Quodlibet, conjecture)
$\exists$bNODE_x, bNODE_y, bNODE_l$_1$, bNODE_l$_2$: (iext(uri_rdf_type, uri_ex_A, uri_owl_Class) and iext(uri_rdf_type, uri_ex_B, uri_

SWB029+4.p Ex Falso Quodlibet
include('Axioms/SWB003+0.ax')
iext(uri_rdf_type, uri_ex_w, uri_ex_B) fof(testcase_conclusion_fullish_029_Ex_Falso_Quodlibet, conjecture)
$\exists$bNODE_x, bNODE_y, bNODE_l$_1$, bNODE_l$_2$: (iext(uri_rdf_type, uri_ex_A, uri_owl_Class) and iext(uri_rdf_type, uri_ex_B, uri_

SWB030+1.p Bad Class
include('Axioms/SWB001+0.ax')
$\exists$bNODE_x: (iext(uri_rdf_type, uri_ex_c, uri_owl_Class) and iext(uri_owl_complementOf, uri_ex_c, bNODE_x) and iext(uri_rdf_t

SWB030+2.p Bad Class
$\forall x, c$: (iext(uri_rdf_type, x, c) $\iff$ icext(c, x)) fof(rdfs_cext_def, axiom)
$\forall z, p, v$: ((iext(uri_owl_hasSelf, z, v) and iext(uri_owl_onProperty, z, p)) $\Rightarrow$ $\forall x$: (icext(z, x) $\iff$ iext(p, x, x))) fof(owl_rest
$\forall z, c$: (iext(uri_owl_complementOf, z, c) $\Rightarrow$ (ic(z) and ic(c) and $\forall x$: (icext(z, x) $\iff$ $\neg$ icext(c, x)))) fof(owl_bool_comple
$\exists$bNODE_x: (iext(uri_rdf_type, uri_ex_c, uri_owl_Class) and iext(uri_owl_complementOf, uri_ex_c, bNODE_x) and iext(uri_rdf_t

SWB030+3.p Bad Class
include('Axioms/SWB002+0.ax')
$\exists$bNODE_x: (iext(uri_rdf_type, uri_ex_c, uri_owl_Class) and iext(uri_owl_complementOf, uri_ex_c, bNODE_x) and iext(uri_rdf_t

SWB030+4.p Bad Class
include('Axioms/SWB003+0.ax')
$\exists$bNODE_x: (iext(uri_rdf_type, uri_ex_c, uri_owl_Class) and iext(uri_owl_complementOf, uri_ex_c, bNODE_x) and iext(uri_rdf_t

SWB031+1.p Large Universe
include('Axioms/SWB001+0.ax')
$\exists$bNODE_x, bNODE_l: (iext(uri_owl_equivalentClass, uri_owl_Thing, bNODE_x) and iext(uri_owl_oneOf, bNODE_x, bNODE_l,

SWB031+2.p Large Universe
$\forall x$: ir(x) fof(simple_ir, axiom)
$\forall x$: (icext(uri_owl_Thing, x) $\iff$ ir(x)) fof(owl_class_thing_ext, axiom)
$\forall x$: $\neg$ icext(uri_owl_Nothing, x) fof(owl_class_nothing_ext, axiom)
$\forall z, s_1, a_1$: ((iext(uri_rdf_first, s_1, a_1) and iext(uri_rdf_rest, s_1, uri_rdf_nil)) $\Rightarrow$ (iext(uri_owl_oneOf, z, s_1) $\iff$ (ic(z) and $\forall x$: (i
$x = a_1$)))) fof(owl_enum_class$_{001}$, axiom)
$\forall c_1, c_2$: (iext(uri_owl_equivalentClass, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: (icext(c_1, x) $\iff$ icext(c_2, x)))) fof(owl_eqdi
$\exists$bNODE_x, bNODE_l: (iext(uri_owl_equivalentClass, uri_owl_Thing, bNODE_x) and iext(uri_owl_oneOf, bNODE_x, bNODE_l,

SWB031+3.p Large Universe
include('Axioms/SWB002+0.ax')
$\exists$bNODE_x, bNODE_l: (iext(uri_owl_equivalentClass, uri_owl_Thing, bNODE_x) and iext(uri_owl_oneOf, bNODE_x, bNODE_l,

SWB031+4.p Large Universe
include('Axioms/SWB003+0.ax')
$\exists$bNODE_x, bNODE_l: (iext(uri_owl_equivalentClass, uri_owl_Thing, bNODE_x) and iext(uri_owl_oneOf, bNODE_x, bNODE_l,

SWB032+1.p Datatype Relationships
include('Axioms/SWB001+0.ax')
iext(uri_owl_disjointWith, uri_xsd_decimal, uri_xsd_string) and iext(uri_rdfs_subClassOf, uri_xsd_integer, uri_xsd_decimal) f

SWB032+2.p Datatype Relationships
idc(uri_xsd_string) fof(owl_dat_dtype_string_type, axiom)
idc(uri_xsd_decimal) fof(owl_dat_dtype_decimal_type, axiom)
idc(uri_xsd_integer) fof(owl_dat_dtype_integer_type, axiom)
$\forall x$: $\neg$ icext(uri_rdf_PlainLiteral, x) and icext(uri_owl_real, x) fof(owl_dat_dtype_relation_disjoint_plainliteral_real, axiom)
$\forall x$: (icext(uri_xsd_string, x) $\Rightarrow$ icext(uri_rdf_PlainLiteral, x)) fof(owl_dat_dtype_relation_subtype_string_plainliteral, axion

$\forall x$: (icext(uri_owl_rational, x) $\Rightarrow$ icext(uri_owl_real, x)) fof(owl_dat_dtype_relation_subtype_rational_real, axiom)

$\forall x$: (icext(uri_xsd_decimal, x) $\Rightarrow$ icext(uri_owl_rational, x)) fof(owl_dat_dtype_relation_subtype_decimal_rational, axiom)

$\forall x$: (icext(uri_xsd_integer, x) $\Rightarrow$ icext(uri_xsd_decimal, x)) fof(owl_dat_dtype_relation_subtype_integer_decimal, axiom)

$\forall x$: (idc(x) $\Rightarrow$ ic(x)) fof(owl_parts_idc_cond_set, axiom)

$\forall c_1, c_2$: (iext(uri_rdfs_subClassOf, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: (icext(c_1, x) $\Rightarrow$ icext(c_2, x)))) fof(owl_rdfsext_su

$\forall c_1, c_2$: (iext(uri_owl_disjointWith, c_1, c_2) $\iff$ (ic(c_1) and ic(c_2) and $\forall x$: $\neg$ icext(c_1, x) and icext(c_2, x))) fof(owl_eqdis_dis

iext(uri_owl_disjointWith, uri_xsd_decimal, uri_xsd_string) and iext(uri_rdfs_subClassOf, uri_xsd_integer, uri_xsd_decimal) f

SWB032+3.p Datatype Relationships

include('Axioms/SWB002+0.ax')

iext(uri_owl_disjointWith, uri_xsd_decimal, uri_xsd_string) and iext(uri_rdfs_subClassOf, uri_xsd_integer, uri_xsd_decimal) f

SWB032+4.p Datatype Relationships

include('Axioms/SWB003+0.ax')

iext(uri_owl_disjointWith, uri_xsd_decimal, uri_xsd_string) and iext(uri_rdfs_subClassOf, uri_xsd_integer, uri_xsd_decimal) f

SWB033+1.p Datatype Relationships

include('Axioms/SWB001+0.ax')

SWB034+1.p Datatype Relationships

include('Axioms/SWB002+0.ax')

SWB035+1.p Datatype Relationships

include('Axioms/SWB003+0.ax')

SWB037+1.p Class Complement Extensional

If a class is an equivalent class to the complement of another class, then the former class defines the complement of the other class. This test checks that class complement is realized as an iff-condition.

include('Axioms/SWB001+0.ax')

iext(uri_owl_complementOf, uri_ex_c$_1$, uri_ex_c$_2$) fof(conclusion_rdfbased_sem_bool_complement_ext, conjecture)

$\exists x_0$: (iext(uri_owl_complementOf, x_0, uri_ex_c$_2$) and iext(uri_owl_equivalentClass, uri_ex_c$_1$, x_0)) fof(premise_rdfbased_sem

SWB038+1.p Class De Morgan

The complement of the union of two classes is an equivalent class to the intersection of the complements of the two classes.

include('Axioms/SWB001+0.ax')

iext(uri_owl_equivalentClass, uri_ex_c$_1$, uri_ex_c$_2$) fof(conclusion_rdfbased_sem_bool_demorgan, conjecture)

$\exists x_1, x_4, x_0, x_2, x_5, x_3, x_6$: (iext(uri_owl_intersectionOf, uri_ex_c$_2$, x_0) and iext(uri_owl_complementOf, x_1, uri_ex_x) and iext(uri

SWB039+1.p Class Intersection Extensional

If a class is an equivalent class to the intersection of other classes, then the former class defines the intersection of the other classes. This test checks that class intersection is realized as an iff-condition.

include('Axioms/SWB001+0.ax')

$\exists x_1, x_0$: (iext(uri_rdf_first, x_0, uri_ex_x) and iext(uri_rdf_rest, x_0, x_1) and iext(uri_rdf_first, x_1, uri_ex_y) and iext(uri_rdf_rest, x

$\exists x_0, x_1, x_2$: (iext(uri_rdf_first, x_0, uri_ex_y) and iext(uri_rdf_rest, x_0, uri_rdf_nil) and iext(uri_owl_intersectionOf, x_1, x_2) and iex

SWB040+1.p Class Modus Tollens

If a class is subsumed by another class, then the complement of the latter class is subsumed by the complement of the former class.

include('Axioms/SWB001+0.ax')

iext(uri_rdfs_subClassOf, uri_ex_n$_2$, uri_ex_n$_1$) fof(conclusion_rdfbased_sem_bool_tollens, conjecture)

iext(uri_owl_complementOf, uri_ex_n$_2$, uri_ex_c$_2$) and iext(uri_owl_complementOf, uri_ex_n$_1$, uri_ex_c$_1$) and iext(uri_rdfs_subCl

SWB041+1.p Class Union Extensional

If a class is an equivalent class to the union of two other classes, then the former class defines the union of the other classes. This test checks that class union is realized as an iff-condition.

include('Axioms/SWB001+0.ax')

$\exists x_0, x_1$: (iext(uri_rdf_first, x_0, uri_ex_x) and iext(uri_rdf_rest, x_0, x_1) and iext(uri_rdf_first, x_1, uri_ex_y) and iext(uri_rdf_rest, x

$\exists x_2, x_1, x_0$: (iext(uri_owl_unionOf, x_0, x_1) and iext(uri_rdf_first, x_1, uri_ex_x) and iext(uri_rdf_rest, x_1, x_2) and iext(uri_rdf_first

SWB042+1.p Property Chain Extensional

If the chain of extensions of two properties p1 and p2 is a subset of the extension of a property p, then a sub property chain axiom is entailed for p and the chain properties p1 and p2.

include('Axioms/SWB001+0.ax')

$\exists x_0, x_1$: (iext(uri_owl_propertyChainAxiom, uri_ex_p, x_0) and iext(uri_rdf_first, x_1, uri_ex_p$_2$) and iext(uri_rdf_rest, x_1, uri_rdf_

$\exists x_2, x_1, x_4, x_5, x_0, x_3$: (iext(uri_rdfs_domain, uri_ex_p$_1$, x_0) and iext(uri_ex_p$_2$, uri_ex_z, uri_ex_y) and iext(uri_rdf_first, x_1, uri_e

SWB043+1.p Singleton Property Chain As Subsumption
A sub property chain axiom with a single chain property corresponds to a sub property axiom.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subPropertyOf, uri_ex_p$_1$, uri_ex_p) fof(conclusion_rdfbased_sem_chain_subprop, conjecture)
$\exists x_0$: (iext(uri_owl_propertyChainAxiom, uri_ex_p, x_0) and iext(uri_rdf_first, x_0, uri_ex_p$_1$) and iext(uri_rdf_rest, x_0, uri_rdf_nil)

SWB044+1.p Asymmetric Property Extensional
If the extension of a property is asymmetric, then the property itself is.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_p, uri_owl_AsymmetricProperty) fof(conclusion_rdfbased_sem_char_asymmetric_ext, conjecture)
$\exists x_2, x_3, x_0, x_1$: (iext(uri_owl_oneOf, x_0, x_1) and iext(uri_rdfs_domain, uri_ex_p, x_2) and iext(uri_rdfs_range, uri_ex_p, x_0) and ie

SWB045+1.p Functional Property Extensional
If the extension of a property is functional, then the property itself is.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_p, uri_owl_FunctionalProperty) fof(conclusion_rdfbased_sem_char_functional_ext, conjecture)
$\exists x_0, x_1$: (iext(uri_rdfs_range, uri_ex_p, x_0) and iext(uri_ex_p, uri_ex_x, uri_ex_y) and iext(uri_rdf_first, x_1, uri_ex_y) and iext(uri

SWB046+1.p Inverse-Functional Property Extensional
If the extension of a property is inverse functional, then the property itself is.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_p, uri_owl_InverseFunctionalProperty) fof(conclusion_rdfbased_sem_char_inversefunc_ext, conjectu
$\exists x_1, x_0$: (iext(uri_rdf_first, x_0, uri_ex_x) and iext(uri_rdf_rest, x_0, uri_rdf_nil) and iext(uri_owl_oneOf, x_1, x_0) and iext(uri_ex_p,

SWB047+1.p Functional Inverse Property As Inverse-Functional
The inverse of a functional property is an inverse functional property.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_p$_2$, uri_owl_InverseFunctionalProperty) fof(conclusion_rdfbased_sem_char_inversefunc_term, conjec
iext(uri_owl_inverseOf, uri_ex_p$_2$, uri_ex_p$_1$) and iext(uri_rdf_type, uri_ex_p$_1$, uri_owl_FunctionalProperty) fof(premise_rdfba

SWB048+1.p Irreflexive Property Extensional
If the extension of a property is irreflexive, then the property itself is.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_p, uri_owl_IrreflexiveProperty) fof(conclusion_rdfbased_sem_char_irreflexive_ext, conjecture)
$\exists x_0, x_3, x_1, x_2$: (iext(uri_rdfs_domain, uri_ex_p, x_0) and iext(uri_rdfs_range, uri_ex_p, x_1) and iext(uri_owl_oneOf, x_1, x_2) and ie

SWB049+1.p Reflexive Property Extensional
If the extension of a property is reflexive, then the property itself is. Note that reflexivity is globally defined on the whole universe, which has infinite cardinality under the RDF-Based Semantics. Therefore, instead of explicitly defining the extension of some custom property, the extension of the built-in property owl:topObjectProperty is referred to.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_p, uri_owl_ReflexiveProperty) fof(conclusion_rdfbased_sem_char_reflexive_ext, conjecture)
iext(uri_owl_equivalentProperty, uri_ex_p, uri_owl_topObjectProperty) fof(premise_rdfbased_sem_char_reflexive_ext, axiom)

SWB050+1.p Symmetric Property Extensional
If the extension of a property is symmetric, then the property itself is.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_p, uri_owl_SymmetricProperty) fof(conclusion_rdfbased_sem_char_symmetric_ext, conjecture)
$\exists x_0, x_2, x_3, x_5, x_1, x_4$: (iext(uri_rdfs_domain, uri_ex_p, x_0) and iext(uri_rdfs_range, uri_ex_p, x_1) and iext(uri_owl_oneOf, x_0, x_2)

SWB051+1.p Transitive Property Extensional
If the extension of a property is transitive, then the property itself is.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_p, uri_owl_TransitiveProperty) fof(conclusion_rdfbased_sem_char_transitive_ext, conjecture)
$\exists x_5, x_0, x_3, x_1, x_2, x_4$: (iext(uri_owl_oneOf, x_0, x_1) and iext(uri_rdfs_domain, uri_ex_p, x_0) and iext(uri_rdfs_range, uri_ex_p, x_2)

SWB052+1.p Transitive Irreflexive Property As Asymmetric
A transitive and irreflexive property is an asymmetric property.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_p, uri_owl_AsymmetricProperty) fof(conclusion_rdfbased_sem_char_transitive_term, conjecture)
iext(uri_rdf_type, uri_ex_p, uri_owl_TransitiveProperty) and iext(uri_rdf_type, uri_ex_p, uri_owl_IrreflexiveProperty) fof(pre

SWB053+1.p Empty Class As Sub-Class
Every OWL class is a super class of the vocabulary class owl:Nothing.

include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_owl_Nothing, uri_ex_c)   fof(conclusion_rdfbased_sem_class_nothing_term, conjecture)
iext(uri_rdf_type, uri_ex_c, uri_owl_Class)   fof(premise_rdfbased_sem_class_nothing_term, axiom)

SWB054+1.p Universal Class As Super-Class
Every OWL class is a sub class of the vocabulary class owl:Thing.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_c, uri_owl_Thing)   fof(conclusion_rdfbased_sem_class_thing_term, conjecture)
iext(uri_rdf_type, uri_ex_c, uri_owl_Class)   fof(premise_rdfbased_sem_class_thing_term, axiom)

SWB055+1.p Individual Enumeration Extensional
If a class is an equivalent class to the enumeration of two individuals, then the class defines the enumeration of the two individuals. This test checks that enumeration is realized as an iff-condition.
include('Axioms/SWB001+0.ax')
$\exists x_0, x_1$: (iext(uri_rdf_first, x_0, uri_ex_y) and iext(uri_rdf_rest, x_0, uri_rdf_nil) and iext(uri_owl_oneOf, uri_ex_e, x_1) and iext(uri_
$\exists x_0, x_2, x_1$: (iext(uri_owl_oneOf, x_0, x_1) and iext(uri_owl_equivalentClass, uri_ex_e, x_0) and iext(uri_rdf_first, x_1, uri_ex_x) and i

SWB056+1.p Individual Enumeration Closed
If a class defines an enumeration class expression from two individuals, and if an individual is an instance of the class but is different from one of the two component individuals, then the individual equals the remaining component individual.
include('Axioms/SWB001+0.ax')
iext(uri_owl_sameAs, uri_ex_z, uri_ex_y)   fof(conclusion_rdfbased_sem_enum_inst_closed, conjecture)
$\exists x_0, x_1$: (iext(uri_rdf_type, uri_ex_z, uri_ex_e) and iext(uri_owl_differentFrom, uri_ex_z, uri_ex_x) and iext(uri_owl_oneOf, uri_ex

SWB057+1.p Individual Difference Extensional
If two individuals are distinct, then the owl:differentFrom relation holds between them.
include('Axioms/SWB001+0.ax')
iext(uri_owl_differentFrom, uri_ex_w, uri_ex_u)   fof(conclusion_rdfbased_sem_eqdis_different_ext, conjecture)
$\exists x_0$: (iext(uri_rdf_type, uri_ex_u, x_0) and iext(uri_owl_complementOf, x_0, uri_ex_c) and iext(uri_rdf_type, uri_ex_w, uri_ex_c))

SWB058+1.p Class Disjointness Extensional
If the non-empty extensions of two classes are disjoint, then the classes themselves are disjoint.
include('Axioms/SWB001+0.ax')
iext(uri_owl_disjointWith, uri_ex_c$_1$, uri_ex_c$_2$)   fof(conclusion_rdfbased_sem_eqdis_disclass_ext, conjecture)
$\exists x_1, x_0$: (iext(uri_owl_oneOf, uri_ex_c$_2$, x_0) and iext(uri_rdf_first, x_1, uri_ex_x) and iext(uri_rdf_rest, x_1, uri_rdf_nil) and iext(uri

SWB059+1.p Disjoint Class Union Composite
If a class expresses the union of two disjoint component classes, then the class expresses a disjoint union class axiom for the component classes.
include('Axioms/SWB001+0.ax')
$\exists x_1, x_0$: (iext(uri_owl_disjointUnionOf, uri_ex_c$_3$, x_0) and iext(uri_rdf_first, x_0, uri_ex_c$_1$) and iext(uri_rdf_rest, x_0, x_1) and iext
$\exists x_0, x_1$: (iext(uri_owl_unionOf, uri_ex_c$_3$, x_0) and iext(uri_owl_disjointWith, uri_ex_c$_1$, uri_ex_c$_2$) and iext(uri_rdf_first, x_1, uri_e

SWB060+1.p Disjoint Class Union As Disjointness
The component classes on the right hand side of a disjoint-union axiom are pairwise disjoint.
include('Axioms/SWB001+0.ax')
iext(uri_owl_disjointWith, uri_ex_c$_1$, uri_ex_c$_2$)   fof(conclusion_rdfbased_sem_eqdis_disjointunion_disjoint, conjecture)
$\exists x_1, x_0$: (iext(uri_owl_disjointUnionOf, uri_ex_c$_3$, x_0) and iext(uri_rdf_first, x_1, uri_ex_c$_2$) and iext(uri_rdf_rest, x_1, uri_rdf_nil) a

SWB061+1.p Disjoint Class Union As Union
If a class is defined by a disjoint-union axiom for two component classes, then the class expresses the union of the two classes.
include('Axioms/SWB001+0.ax')
$\exists x_0, x_1$: (iext(uri_rdf_first, x_0, uri_ex_c$_1$) and iext(uri_rdf_rest, x_0, x_1) and iext(uri_rdf_first, x_1, uri_ex_c$_2$) and iext(uri_rdf_rest,
$\exists x_0, x_1$: (iext(uri_owl_disjointUnionOf, uri_ex_c$_3$, x_0) and iext(uri_rdf_first, x_1, uri_ex_c$_2$) and iext(uri_rdf_rest, x_1, uri_rdf_nil) a

SWB062+1.p Property Disjointness Extensional
If the non-empty extensions of two properties are disjoint, then the properties themselves are disjoint.
include('Axioms/SWB001+0.ax')
iext(uri_owl_propertyDisjointWith, uri_ex_p$_1$, uri_ex_p$_2$)   fof(conclusion_rdfbased_sem_eqdis_disprop_ext, conjecture)
$\exists x_1, x_2, x_0, x_3$: (iext(uri_rdfs_domain, uri_ex_p$_1$, uri_ex_x$_1$) and iext(uri_rdfs_range, uri_ex_p$_1$, uri_ex_y$_1$) and iext(uri_ex_p$_2$, uri

SWB063+1.p Class Equivalence Extensional
If the extensions of two classes are equal, then the classes are themselves equivalent.

include('Axioms/SWB001+0.ax')
iext(uri_owl_equivalentClass, uri_ex_c$_1$, uri_ex_c$_2$) fof(conclusion_rdfbased_sem_eqdis_eqclass_ext, conjecture)
$\exists x_0, x_1$: (iext(uri_owl_oneOf, uri_ex_c$_2$, x_0) and iext(uri_owl_oneOf, uri_ex_c$_1$, x_1) and iext(uri_rdf_first, x_0, uri_ex_x) and iext(u

SWB064+1.p Property Equivalence Extensional
If the extensions of two properties are equal, then the properties themselves are equivalent.
include('Axioms/SWB001+0.ax')
iext(uri_owl_equivalentProperty, uri_ex_p$_1$, uri_ex_p$_2$) fof(conclusion_rdfbased_sem_eqdis_eqprop_ext, conjecture)
$\exists x_1, x_0$: (iext(uri_rdfs_domain, uri_ex_p$_1$, uri_ex_x) and iext(uri_rdfs_range, uri_ex_p$_1$, uri_ex_y) and iext(uri_ex_p$_1$, uri_ex_s, uri_

SWB065+1.p Individual Equality Extensional
If two individuals are equal, then the owl:sameAs relation holds between them.
include('Axioms/SWB001+0.ax')
iext(uri_owl_sameAs, uri_ex_w, uri_ex_u) fof(conclusion_rdfbased_sem_eqdis_sameas_ext, conjecture)
$\exists x_0, x_1$: (iext(uri_rdf_first, x_0, uri_ex_u) and iext(uri_rdf_rest, x_0, uri_rdf_nil) and iext(uri_owl_oneOf, x_1, x_0) and iext(uri_rdf_ty

SWB066+1.p Inverse Property Extensional
If the extensions of two properties are inverse, then the properties themselves are.
include('Axioms/SWB001+0.ax')
iext(uri_owl_inverseOf, uri_ex_q, uri_ex_p) fof(conclusion_rdfbased_sem_inv_ext, conjecture)
$\exists x_3, x_7, x_0, x_6, x_4, x_5, x_1, x_2$: (iext(uri_owl_oneOf, x_0, x_1) and iext(uri_rdfs_domain, uri_ex_p, x_2) and iext(uri_rdfs_range, uri_ex

SWB067+1.p Double Inverse Property As Equivalence
Transitively related inverse properties are equivalent.
include('Axioms/SWB001+0.ax')
iext(uri_owl_equivalentProperty, uri_ex_p$_3$, uri_ex_p$_1$) fof(conclusion_rdfbased_sem_inv_trans, conjecture)
iext(uri_owl_inverseOf, uri_ex_p$_3$, uri_ex_p$_2$) and iext(uri_owl_inverseOf, uri_ex_p$_2$, uri_ex_p$_1$) fof(premise_rdfbased_sem_inv

SWB068+1.p N-Ary Individual Difference Extensional
For a group of mutually different individuals, the corresponding owl:AllDifferent construct exist.
include('Axioms/SWB001+0.ax')
$\exists x_2, x_0, x_1, x_3$: (iext(uri_rdf_first, x_0, uri_ex_w$_1$) and iext(uri_rdf_rest, x_0, x_1) and iext(uri_rdf_first, x_1, uri_ex_w$_2$) and iext(uri_
$\exists x_0, x_2, x_1$: (iext(uri_owl_differentFrom, uri_ex_w$_1$, uri_ex_w$_2$) and iext(uri_owl_differentFrom, uri_ex_w$_1$, uri_ex_w$_3$) and iext(u

SWB069+1.p N-Ary Class Disjointness Extensional
For a group of mutually disjoint classes, the corresponding owl:AllDisjointClasses construct exists.
include('Axioms/SWB001+0.ax')
$\exists x_2, x_3, x_0, x_1$: (iext(uri_rdf_first, x_0, uri_ex_c$_2$) and iext(uri_rdf_rest, x_0, x_1) and iext(uri_rdf_first, x_1, uri_ex_c$_3$) and iext(uri_r
$\exists x_1, x_2, x_0$: (iext(uri_owl_disjointWith, uri_ex_c$_2$, uri_ex_c$_3$) and iext(uri_rdf_first, x_0, uri_ex_c$_2$) and iext(uri_rdf_rest, x_0, x_1) an

SWB070+1.p N-Ary Property Disjointness Extensional
For a group of mutually disjoint properties the corresponding owl:AllDisjointProperties construct exists.
include('Axioms/SWB001+0.ax')
$\exists x_1, x_0, x_3, x_2$: (iext(uri_rdf_first, x_0, uri_ex_p$_2$) and iext(uri_rdf_rest, x_0, x_1) and iext(uri_rdf_first, x_1, uri_ex_p$_3$) and iext(uri_r
$\exists x_2, x_0, x_1$: (iext(uri_owl_propertyDisjointWith, uri_ex_p$_1$, uri_ex_p$_2$) and iext(uri_owl_propertyDisjointWith, uri_ex_p$_1$, uri_ex_

SWB071+1.p Negative Individual Property Assertion Extensional
If a triple 's p o' does not hold, then the negative property assertion NPA(s p o) is entailed.
include('Axioms/SWB001+0.ax')
$\exists x_0$: (iext(uri_owl_sourceIndividual, x_0, uri_ex_s) and iext(uri_owl_assertionProperty, x_0, uri_ex_p) and iext(uri_owl_targetIndiv
iext(uri_rdf_type, uri_ex_p, uri_owl_FunctionalProperty) and iext(uri_ex_p, uri_ex_s, uri_ex_o$_1$) and iext(uri_owl_differentFrom, u

SWB072+1.p Empty Data Property Extensional Low
The extension of the vocabulary property owl:bottomDataProperty is empty. See also rdfbased-sem-prop-bottomdataproperty-ext-hi for a weaker variant of this test case.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_domain, uri_owl_bottomDataProperty, uri_owl_Nothing) and iext(uri_rdfs_range, uri_owl_bottomDataProperty, u
tautology or $\neg$ tautology fof(premise_rdfbased_sem_prop_bottomdataproperty_ext_lo, axiom)

SWB073+1.p Empty Data Property As Sub-Property
Every data property is a super property of the vocabulary property owl:bottomDataProperty.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subPropertyOf, uri_owl_bottomDataProperty, uri_ex_p) fof(conclusion_rdfbased_sem_prop_bottomdataproper
iext(uri_rdf_type, uri_ex_p, uri_owl_DatatypeProperty) fof(premise_rdfbased_sem_prop_bottomdataproperty_term, axiom)

SWB074+1.p Universal Data Property Extensional Low

The extension of the vocabulary property owl:topDataProperty equals the class product of OWL individuals and data values, and thus subsumes that class product. See also rdfbased-sem-prop-topdataproperty-ext-hi for a test case checking the exact upper bounds of the domain and range.
include('Axioms/SWB001+0.ax')
iext(uri_owl_topDataProperty, uri_ex_x, uri_ex_y)   fof(conclusion_rdfbased_sem_prop_topdataproperty_ext_lo, conjecture)
iext(uri_rdf_type, uri_ex_y, uri_rdfs_Literal) and iext(uri_rdf_type, uri_ex_x, uri_owl_Thing)   fof(premise_rdfbased_sem_prop_

SWB075+1.p Universal Data Property As Super-Property
Every data property is a sub property of the vocabulary property owl:topDataProperty.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subPropertyOf, uri_ex_p, uri_owl_topDataProperty)   fof(conclusion_rdfbased_sem_prop_topdataproperty_term
iext(uri_rdf_type, uri_ex_p, uri_owl_DatatypeProperty)   fof(premise_rdfbased_sem_prop_topdataproperty_term, axiom)

SWB076+1.p Property Range Extensional OWL
If the extension of a given class is a range for the extension of a given property, then the class is a range for the property.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_range, uri_ex_p, uri_ex_c)   fof(conclusion_rdfbased_sem_rdfsext_range_ext, conjecture)
$\exists x_0$: (iext(uri_owl_allValuesFrom, x_0, uri_ex_c) and iext(uri_owl_onProperty, x_0, uri_ex_p) and iext(uri_rdfs_subClassOf, uri_ow

SWB077+1.p Property Range Sub-Property OWL
Every sub property of a given property with a given range also has this range.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_range, uri_ex_p$_1$, uri_ex_c)   fof(conclusion_rdfbased_sem_rdfsext_range_subprop, conjecture)
iext(uri_rdfs_range, uri_ex_p$_2$, uri_ex_c) and iext(uri_rdfs_subPropertyOf, uri_ex_p$_1$, uri_ex_p$_2$)   fof(premise_rdfbased_sem_rd

SWB078+1.p Property Range Super-Class OWL
Every super class of a range for a given property is itself a range for that property.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_range, uri_ex_p, uri_ex_c$_2$)   fof(conclusion_rdfbased_sem_rdfsext_range_superclass, conjecture)
iext(uri_rdfs_range, uri_ex_p, uri_ex_c$_1$) and iext(uri_rdfs_subClassOf, uri_ex_c$_1$, uri_ex_c$_2$)   fof(premise_rdfbased_sem_rdfsex

SWB079+1.p Class Subsumption Extensional OWL
If the extension of a given class is subsumed by the extension of a second class, then the first class is a subclass of the second class.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_c$_1$, uri_ex_c$_2$)   fof(conclusion_rdfbased_sem_rdfsext_subclass_ext, conjecture)
$\exists x_0, x_1, x_2$: (iext(uri_owl_oneOf, uri_ex_c$_2$, x_0) and iext(uri_rdf_first, x_0, uri_ex_w) and iext(uri_rdf_rest, x_0, x_1) and iext(uri_ow

SWB080+1.p Property Subsumption Extensional OWL
If the extension of a given property is subsumed by the extension of a second property, then the first property is a subproperty of the second property.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subPropertyOf, uri_ex_p$_1$, uri_ex_p$_2$)   fof(conclusion_rdfbased_sem_rdfsext_subprop_ext, conjecture)
$\exists x_8, x_4, x_2, x_5, x_1, x_6, x_3, x_7, x_0$: (iext(uri_rdfs_domain, uri_ex_p$_1$, x_0) and iext(uri_rdfs_range, uri_ex_p$_1$, x_1) and iext(uri_rdf_fir

SWB081+1.p Universal Restriction Comparison By Class
A universal restriction on some property and some class is a sub class of another universal restriction on the same property but on a super class.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_x$_1$, uri_ex_x$_2$)   fof(conclusion_rdfbased_sem_restrict_allvalues_cmp_class, conjecture)
iext(uri_owl_allValuesFrom, uri_ex_x$_2$, uri_ex_c$_2$) and iext(uri_owl_onProperty, uri_ex_x$_2$, uri_ex_p) and iext(uri_owl_allValuesF

SWB082+1.p Universal Restriction Comparison By Property
A universal restriction on some property and some class is a sub class of another universal restriction on the same class but on a sub property.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_x$_2$, uri_ex_x$_1$)   fof(conclusion_rdfbased_sem_restrict_allvalues_cmp_prop, conjecture)
iext(uri_owl_allValuesFrom, uri_ex_x$_2$, uri_ex_c) and iext(uri_owl_onProperty, uri_ex_x$_2$, uri_ex_p$_2$) and iext(uri_owl_allValuesF

SWB083+1.p Universal Restriction Extensional
If the range of property p is defined to be class c, then every individual is an instance of the universal restriction on p to c.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_w, uri_ex_z)   fof(conclusion_rdfbased_sem_restrict_allvalues_inst_subj, conjecture)

iext(uri_owl_allValuesFrom, uri_ex_z, uri_ex_c) and iext(uri_owl_onProperty, uri_ex_z, uri_ex_p) and iext(uri_rdfs_range, uri_ex_p

SWB084+1.p Exact-2-QCR Intensional
If an individual w is an instance of the exact-2-QCR on property p to class c, then two distinct individuals x1 and x2 exist in c with w p x1 and w p x2
include('Axioms/SWB001+0.ax')
$\exists x_1, x_0$: (iext(uri_rdf_type, x_0, uri_ex_c) and iext(uri_owl_differentFrom, x_0, x_1) and iext(uri_rdf_type, x_1, uri_ex_c) and iext(uri
iext(uri_owl_qualifiedCardinality, uri_ex_z, literal_typed(dat_str$_2$, uri_xsd_nonNegativeInteger)) and iext(uri_owl_onProperty, u

SWB085+1.p Exact-2-QCR Extensional
For an individual w, if there are exactly two triples w p x1 and w p x2 with x1 and x2 being instances of class c, then w is an instance of the exact-2-QCR on p to c.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_w, uri_ex_z)   fof(conclusion_rdfbased_sem_restrict_exactqcr_inst_subj_two, conjecture)
$\exists x_3, x_4, x_0, x_1, x_2$: (iext(uri_rdfs_range, uri_ex_p, x_0) and iext(uri_owl_complementOf, x_1, uri_ex_c) and iext(uri_owl_qualifiedCa

SWB086+1.p Self-Restriction Comparison By Property
A self restriction on some property is a sub class of another self restriction on a super property.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_x$_1$, uri_ex_x$_2$)   fof(conclusion_rdfbased_sem_restrict_hasself_cmp_prop, conjecture)
iext(uri_owl_hasSelf, uri_ex_x$_2$, literal_typed(dat_str_true, uri_xsd_boolean)) and iext(uri_owl_onProperty, uri_ex_x$_2$, uri_ex_p$_2$)

SWB087+1.p Self-Restriction Extensional
For a triple w p w, the individual w is an instance of the self restriction on p.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_w, uri_ex_z)   fof(conclusion_rdfbased_sem_restrict_hasself_inst_subj, conjecture)
iext(uri_owl_hasSelf, uri_ex_z, literal_typed(dat_str_true, uri_xsd_boolean)) and iext(uri_owl_onProperty, uri_ex_z, uri_ex_p) and

SWB088+1.p Max-QCR Comparison By Cardinality
A max-QCR for some cardinality on some property and to some class is a subclass of another max-QCR on the same property to the same class and for some larger cardinality.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_x$_1$, uri_ex_x$_2$)   fof(conclusion_rdfbased_sem_restrict_maxqcr_cmp_card, conjecture)
iext(uri_owl_maxQualifiedCardinality, uri_ex_x$_2$, literal_typed(dat_str$_2$, uri_xsd_nonNegativeInteger)) and iext(uri_owl_onProp

SWB089+1.p Max-QCR Comparison By Class
A max-QCR for some cardinality on some property and to some class is a subclass of another max-QCR for the same cardinality on the same property and to some sub class.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_x$_1$, uri_ex_x$_2$)   fof(conclusion_rdfbased_sem_restrict_maxqcr_cmp_class, conjecture)
iext(uri_owl_maxQualifiedCardinality, uri_ex_x$_2$, literal_typed(dat_str$_1$, uri_xsd_nonNegativeInteger)) and iext(uri_owl_onProp

SWB090+1.p Max-QCR Comparison By Property
A max-QCR for some cardinality on some property and to some class is a subclass of another max-QCR for the same cardinality to the same class and on a sub property.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_x$_1$, uri_ex_x$_2$)   fof(conclusion_rdfbased_sem_restrict_maxqcr_cmp_prop, conjecture)
iext(uri_owl_maxQualifiedCardinality, uri_ex_x$_2$, literal_typed(dat_str$_1$, uri_xsd_nonNegativeInteger)) and iext(uri_owl_onProp

SWB091+1.p Max-1-QCR Extensional
For an individual w, if there is at most one triple w p x with x in class c, then w is an instance of the max-1-QCR on p to c.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_w, uri_ex_z)   fof(conclusion_rdfbased_sem_restrict_maxqcr_inst_subj_one, conjecture)
$\exists x_1, x_3, x_2, x_0$: (iext(uri_rdfs_range, uri_ex_p, x_0) and iext(uri_owl_maxQualifiedCardinality, uri_ex_z, literal_typed(dat_str$_1$, uri

SWB092+1.p Max-0-QCR Extensional
For an individual w, if there is no triple w p x, with x being an instance of class c, then w is an instance of the max-0-QCR on p to c.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_w, uri_ex_z)   fof(conclusion_rdfbased_sem_restrict_maxqcr_inst_subj_zero, conjecture)
$\exists x_2, x_4, x_0, x_3, x_1$: (iext(uri_owl_complementOf, x_0, uri_ex_c) and iext(uri_rdfs_range, uri_ex_p, x_1) and iext(uri_owl_maxQualifi

SWB093+1.p Min-QCR Comparison By Cardinality

A min-QCR for some cardinality on some property and to some class is a subclass of another min-QCR on the same property to the same class and for some smaller cardinality.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_x$_1$, uri_ex_x$_2$) fof(conclusion_rdfbased_sem_restrict_minqcr_cmp_card, conjecture)
iext(uri_owl_minQualifiedCardinality, uri_ex_x$_2$, literal_typed(dat_str$_1$, uri_xsd_nonNegativeInteger)) and iext(uri_owl_onPrope

SWB094+1.p Min-QCR Comparison By Class
A min-QCR for some cardinality on some property and to some class is a subclass of another min-QCR for the same cardinality on the same property and to some super class.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_x$_1$, uri_ex_x$_2$) fof(conclusion_rdfbased_sem_restrict_minqcr_cmp_class, conjecture)
iext(uri_owl_minQualifiedCardinality, uri_ex_x$_2$, literal_typed(dat_str$_1$, uri_xsd_nonNegativeInteger)) and iext(uri_owl_onPrope

SWB095+1.p Min-QCR Comparison By Property
A min-QCR for some cardinality on some property and to some class is a subclass of another min-QCR for the same cardinality to the same class and on a super property.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_x$_1$, uri_ex_x$_2$) fof(conclusion_rdfbased_sem_restrict_minqcr_cmp_prop, conjecture)
iext(uri_owl_minQualifiedCardinality, uri_ex_x$_2$, literal_typed(dat_str$_1$, uri_xsd_nonNegativeInteger)) and iext(uri_owl_onPrope

SWB096+1.p Min-1-QCR Intensional
If an individual w is an instance of the min-1-QCR on property p to class c, then an individual x exists with w p x and x in c.
include('Axioms/SWB001+0.ax')
$\exists x_0$: (iext(uri_rdf_type, x_0, uri_ex_c) and iext(uri_ex_p, uri_ex_w, x_0)) fof(conclusion_rdfbased_sem_restrict_minqcr_inst_obj
iext(uri_owl_minQualifiedCardinality, uri_ex_z, literal_typed(dat_str$_1$, uri_xsd_nonNegativeInteger)) and iext(uri_owl_onProper

SWB097+1.p Min-1-QCR Extensional
For a triple w p x, with x being an instance of the class c, the individual w is an instance of the min-1-QCR on p to c.
include('Axioms/SWB001+0.ax')
iext(uri_rdf_type, uri_ex_w, uri_ex_z) fof(conclusion_rdfbased_sem_restrict_minqcr_inst_subj_one, conjecture)
iext(uri_owl_minQualifiedCardinality, uri_ex_z, literal_typed(dat_str$_1$, uri_xsd_nonNegativeInteger)) and iext(uri_owl_onProper

SWB098+1.p Existential Restriction Comparison By Class
An existential restriction on some property and some class is a sub class of another existential restriction on the same property but on a super class.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_x$_1$, uri_ex_x$_2$) fof(conclusion_rdfbased_sem_restrict_somevalues_cmp_class, conjecture)
iext(uri_owl_someValuesFrom, uri_ex_x$_2$, uri_ex_c$_2$) and iext(uri_owl_onProperty, uri_ex_x$_2$, uri_ex_p) and iext(uri_owl_someVal

SWB099+1.p Existential Restriction Comparison By Property
An existential restriction on some property and some class is a sub class of another existential restriction on the same class but on a super property.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_x$_1$, uri_ex_x$_2$) fof(conclusion_rdfbased_sem_restrict_somevalues_cmp_prop, conjecture)
iext(uri_owl_someValuesFrom, uri_ex_x$_2$, uri_ex_c) and iext(uri_owl_onProperty, uri_ex_x$_2$, uri_ex_p$_2$) and iext(uri_owl_someVal

SWB100+1.p Existential Restriction Intensional
If an individual w is an instance of the existential restriction on property p and class c, then an individual x exists with w p x and x in c.
include('Axioms/SWB001+0.ax')
$\exists x_0$: (iext(uri_rdf_type, x_0, uri_ex_c) and iext(uri_ex_p, uri_ex_w, x_0)) fof(conclusion_rdfbased_sem_restrict_somevalues_inst
iext(uri_owl_someValuesFrom, uri_ex_z, uri_ex_c) and iext(uri_owl_onProperty, uri_ex_z, uri_ex_p) and iext(uri_rdf_type, uri_ex_

SWB101+1.p Cardinality Restriction As QCR
A cardinality restriction can be written as a QCR to class owl:Thing.
include('Axioms/SWB001+0.ax')
iext(uri_owl_equivalentClass, uri_ex_z$_1$, uri_ex_z$_2$) fof(conclusion_rdfbased_sem_restrict_term_cardqcr, conjecture)
iext(uri_owl_qualifiedCardinality, uri_ex_z$_2$, literal_typed(dat_str$_1$, uri_xsd_nonNegativeInteger)) and iext(uri_owl_onProperty,

SWB102+1.p Data-QCR As Object-QCR
An object-QCR on a data property to a datatype is a data-QCR. Also, every data-QCR is an object-QCR.
include('Axioms/SWB001+0.ax')
iext(uri_owl_equivalentClass, uri_ex_z$_1$, uri_ex_z$_2$) fof(conclusion_rdfbased_sem_restrict_term_dataqcr, conjecture)

iext(uri_rdf_type, uri_ex_p, uri_owl_DatatypeProperty) and iext(uri_owl_qualifiedCardinality, uri_ex_z$_2$, literal_typed(dat_str$_1$, u

SWB103+1.p Exact-QCR As Min-QCR Max-QCR Intersection
The intersection of a min- and a max-QCR for the same cardinality, property and class is equivalent to an exact-QCR for the same cardinality, property and class.
include('Axioms/SWB001+0.ax')
iext(uri_owl_equivalentClass, uri_ex_z$_4$, uri_ex_z$_1$)   fof(conclusion_rdfbased_sem_restrict_term_minmaxexact, conjecture)
$\exists x_0, x_1$: (iext(uri_owl_minQualifiedCardinality, uri_ex_z$_2$, literal_typed(dat_str$_1$, uri_xsd_nonNegativeInteger)) and iext(uri_owl

SWB104+1.p Universal Class As Min-QCR Max-QCR Union
The union of a min- and a max-QCR for the same cardinality, property and class covers the whole universe.
include('Axioms/SWB001+0.ax')
iext(uri_owl_equivalentClass, uri_ex_z$_3$, uri_owl_Thing)   fof(conclusion_rdfbased_sem_restrict_term_minmaxthing, conjecture
$\exists x_0, x_1$: (iext(uri_owl_maxQualifiedCardinality, uri_ex_z$_2$, literal_typed(dat_str$_1$, uri_xsd_nonNegativeInteger)) and iext(uri_ow

SWB105+1.p Universal Existential Restriction Duality
Universal and existential restrictions are dual: The universal restriction on property p to the complement of class c is equivalent to the complement of the existential restriction on p to c.
include('Axioms/SWB001+0.ax')
iext(uri_owl_equivalentClass, uri_ex_z$_1$, uri_ex_nz$_2$)   fof(conclusion_rdfbased_sem_restrict_term_sameall, conjecture)
iext(uri_owl_complementOf, uri_ex_nz$_2$, uri_ex_z$_2$) and iext(uri_owl_complementOf, uri_ex_nc, uri_ex_c) and iext(uri_owl_someV

SWB106+1.p Self-Restriction As Existential Restriction
Self restrictions are more specific than unconstrained existential restrictions.
include('Axioms/SWB001+0.ax')
iext(uri_rdfs_subClassOf, uri_ex_z$_1$, uri_ex_z$_2$)   fof(conclusion_rdfbased_sem_restrict_term_selfsome, conjecture)
iext(uri_owl_someValuesFrom, uri_ex_z$_2$, uri_owl_Thing) and iext(uri_owl_onProperty, uri_ex_z$_2$, uri_ex_p) and iext(uri_owl_has

SWB107+1.p Has-Restriction As Existential Restriction
A has-value restriction for a value $\vee$ can be written as an existential restriction to a singleton class containing v.
include('Axioms/SWB001+0.ax')
iext(uri_owl_equivalentClass, uri_ex_z$_1$, uri_ex_z$_2$)   fof(conclusion_rdfbased_sem_restrict_term_somehas, conjecture)
$\exists x_1, x_0$: (iext(uri_owl_hasValue, uri_ex_z$_2$, uri_ex_u) and iext(uri_owl_onProperty, uri_ex_z$_2$, uri_ex_p) and iext(uri_owl_oneOf, x

SWB108+1.p Existential Restriction As Min-QCR
An existential restriction can be written as a min-1-QCR.
include('Axioms/SWB001+0.ax')
iext(uri_owl_equivalentClass, uri_ex_z$_1$, uri_ex_z$_2$)   fof(conclusion_rdfbased_sem_restrict_term_someqcr, conjecture)
iext(uri_owl_minQualifiedCardinality, uri_ex_z$_2$, literal_typed(dat_str$_1$, uri_xsd_nonNegativeInteger)) and iext(uri_owl_onPrope