

```

@prefix : <http://www.csd.abdn.ac.uk/~ggrimnes/dev/smeagol/ontology#>.
@prefix ac: <http://www.csd.abdn.ac.uk/~ggrimnes/dev/smeagol/actions#>.
@prefix math: <http://www.w3.org/2000/10/swap/math#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix ql: <http://www.w3.org/2004/12/ql#>.

```

```

:MagicProperty a rdf:Property ;
  rdfs:label "Smeagol MagicProperty".

```

```

:AlwaysSucceedsProperty a rdf:Property ;
  rdfs:label "Smeagol always succeeds Property" ;
  rdfs:subPropertyOf :MagicProperty.

```

```

:triples a :AlwaysSucceedsProperty.

```

```

ac:read a :Action ;
  rdfs:label "read" ;
  :in ( ?u ) ;
  :out () ;
  :preconditions { ?u :source ?source . ?u :triples ?t1 } ;
  :effects { ?u :triples ?t2 . ?t2 math:greaterThan ?t1 } .

```

```

ac:findSource a :Action ;
  rdfs:label "find source" ;
  :in ( ?u ) ;
  :out () ;
  :preconditions { } ;
  :effects { ?u :source ?s }.

```

```

ac:readOntology a :Action ;
  rdfs:label "read ontology" ;
  :description "[ ?u ]" ;
  :in ( ?u ) ;
  :preconditions {
    ?u :triples ?t .
    ?t math:greaterThan 0
  } ;
  :effects {
    ?o :triples ?ot .
    ?ot math:greaterThan 0 ;
    :ontologyOf ?u } .

```

```

ac:readOntology2 a :Action ;

```

```

rdfs:label "read ontology for class" ;
rdfs:comment "This is hoping that we can read the ontology from the base a URI " ;
:description "[ ?c ]";
:in ( ?c ) ;
:preconditions { } ;
:effects {
    ?o :definingOntologyOf ?c
}.

ac:rdfsInference a :NoAction ;
    rdfs:label "rdfs inference" ;
    :preconditions { ?u :triples ?t1 . ?t1 math:greaterThan 0 . ?ot :ontologyOf ?u } ;
    :effects { ?u :triples ?t2 . ?t2 math:greaterThan ?t1 }.

ac:cluster a :Action ;
    rdfs:label "clustering" ;
    rdfs:comment "cluster the instances in list ?l. " ;
    :preconditions { ?s a :Set . ?u rdfs:member ?s. } ;
    :effects { ?u :inCluster ?c }.

ac:generaliseFromCluster a :Action ;
    rdfs:label "generalise from cluster" ;
    rdfs:comment "find a value for a given property based on the values of similar resources." ;
    :in ( ?u ?p ) ;
    :preconditions { ?u :inCluster ?c } ;
    :effects { ?u ?p ?v . }.

ac:classifyInstance a :Action ;
    rdfs:label "classify some instances" ;
    :in ( ?p ?v ) ;
    :preconditions {
        ?r a :RuleSet ;
        :describedProperty ?p ;
        :describedValue ?v .
    } ;
    :effects { ?c ?p ?v } . # this is ok - ?c is existentially quantified...

ac:listFromCluster a :Action ;
    rdfs:label "list the members of a cluster" ;
    :preconditions { ?i :inCluster ?c } ;
    :effects { ?s a rdf:List ; rdfs:member ?i }.

ac:describeSet a :Action ;
    rdfs:label "describe a set" ;
    :in ( ?p ?v ) ;
    :preconditions {

```

```

    ?s a :Set ;
    :describedProperty ?p ;
    :describedValue ?v .

};
:effects {
    ?r a :RuleSet ;
    :describedProperty ?p ;
    :describedValue ?v .
}.

ac:findSet a :Action ;
rdfs:label "find a set based on a property" ;
:description " [ ?p, ?v ]" ;
:in ( ?p ?v ) ;
# :preconditions { } ;
:effects {
    ?s a :Set ;
    :describedProperty ?p ;
    :describedValue ?v .
}.

ac:findSetType a :Action ;
rdfs:label "find a set based on type" ;
rdfs:comment "find a set of things of the same type as the parameter ?i" ;
:description " [ ?i ]" ;
:in ( ?i ) ;
# :preconditions { } ;
:effects {
    ?s a :Set ;
    :describedProperty rdf:type ;
    :describedValue ?t.
    ?i rdfs:member ?s .
}.

ac:rabbiSetDifference a :Action ;
rdfs:label "learn rules differentiating two sets" ;
:preconditions {
    ?s1 a rdf:List .
    ?s2 a rdf:List .
    ?t1 :triples ?s1 .
    ?t2 :triples ?s2 .
    ?t1 math:greaterThan 0 .
    ?t2 math:greaterThan 0 . } ;

```

```

:effects {
  ?r a :RuleSet .
  ?r :ruleType :Difference .
  ?r :set1 ?s1 .
  ?r :set2 ?s2. } .

ac:newProperty a :noAction ;
  rdfs:label "" ;
  :in ( ?a ) ;
  :preconditions { ?r a :RuleSet ; :ruleType :Difference } ;

  :effects { ?a ?p ?b. ?p a rdf:Property. } .

ac:followSeeAlso a :noAction .

ac:query a :Action ;
  rdfs:label "Query " ;
  :in ( ?q ) ;
  :preconditions { ?u :anchor ?t ; :triples ?tr . ?tr math:greaterThan 0 } ;
  :effects { ?q ql:select ?v ; ql:where ?t ; ql:results ?res } .

ac:subClassInference a :Action ;
  rdfs:label "RDFS Subclass Inference" ;
  :description "[ ?x ]" ;
  :in ( ?x ) ;
  :preconditions {
    ?ot :ontologyOf ?x .
  } ;
#   :preconditions {
#     ?c a rdfs:Class ;
#     :subClassOf ?sc .
#     ?x rdf:type ?sc
#   } ;
  :effects { ?x rdf:type ?c } .

ac:subClassInference2 a :Action ;
  rdfs:label "RDFS Subclass Inference (reverse)" ;
  :description "[ ?c ]" ;
  :in ( ?c ) ;
  :preconditions {
    ?ot :definingOntologyOf ?c .
  } ;
  :effects { ?x rdf:type ?c } .

```

