Sydney OWL Syntax (SOS)

Authors: Anne Cregan, Rolf Schwitter and Thomas Meyer

Version: 0.1
Date: 2007-03-11
Status: Working Draft

The following notation is used for Sydney OWL Syntax throughout this document:

- `c[N]` denotes a (potentially complex) nominal category in SOS, for example `c[man]` or `c[big,man]`
- `op[N]` denotes a (potentially complex) nominal category in SOS that is part of an object property, for example `has ... op[child]` results in *hasChild* or `has ... op[big,child]` results in *hasBigChild*.
- `op[V]` denotes a verbal category in SOS, for example *likes, married to*
- `iID` denotes a name in SOS, for example *Rona, Aaron*
- `X`, `Y`, `Z` are variables in SOS.

Functional-Style Syntax	SOS Abstract Form	SOS Examples
InverseObjectProperty(op)	having a op[N] is inverse	having a child is inverse
ObjectUnionOf(c_1 ... c_n)	a c[N_1] or ... or a c[N_n]	a man or … or an adult
ObjectIntersectionOf(c_1 ... c_n)	a c[N_1] and ... and a c[N_n]	a man and … and an adult
ObjectComplementOf(c)	is not a c[N]	is not a man
ObjectOneOf(iID_1 ... iID_n)	is one of iID_1or … or iID_n	is one of Tom or … or Harry
ObjectSomeValuesFrom(op c)	has some c[N_1] as a op[N_2] op[V] some c[N]	has some animal as a pet likes some animal
ObjectAllValuesFrom(op c)	has only c[N_1] as a op[N_2] op[V] only c[N]	has only animals as a pet likes only animals
ObjectExistsSelf(op)	op[V] itself	likes itself
ObjectHasValue(op iID)	has iID as a op[N]	has John as a parent
ObjectMinCardinality(n op c)	has at least n c[N_1] as a op[N_2]	has at least 3 cats as an animal
ObjectMaxCardinality(n op c)	has at most n c[N_1] as a op[N_2]	has at most 3 cats as an animal
ObjectExactCardinality(n op c)	has exactly n c[N_1] as a op[N_2]	has exactly 3 cats as an animal
ObjectMinCardinality(n op)	has at least n op[N]	has at least 3 animals
ObjectMaxCardinality(n op)	has at most n op[N]	has at most 3 animals
ObjectExactCardinality(n op)	has exactly n op[N]	has exactly 3 animals
SubClassOf(c_1 c_2)	If X is a c[N_1] then X is a c[N_2].	If X is a car then X is a vehicle.
EquivalentClasses(c_1 ... c_n)	c[N_1] and … and c[N_2] are equivalent.	Aunt and … and auntie are equivalent.
DisjointClasses(c_1 ... c_n)	c[N_1] and … and c[N_2] are mutually exclusive.	Female and … and male are mutually exclusive.
DisjointUnion(cID c_1 ... c_n)	cID[N_1] is equivalent to c[N_2] or … or c[N_n], and c[N_2] and … and c[N_n] are mutually exclusive.	Person is equivalent to female or … or male, and female and … and male are mutually exclusive.
SubObjectPropertyOf(op_1 op_2)	If X has Y as a op[N_1] then X has Y as a op[N_2].	If X has Y as a parent then X has Y as an ancestor.
SubObjectPropertyOf(subObjectPropertyChain(op_1 ... op_n) op)	If X op[V] Y and Y has Z as a op[N_1] and … then X op[V] Z.	If X owns Y and Y has Z as a part and … then X owns Z.
EquivalentObjectProperties(op_1 ... op_n)	op[V_1] and … and op[V_2] are equivalent.	Likes and … and admires are equivalent.
DisjointObjectProperties(op_1 ... op_n)	Being op[V_1] and … and having a op[N_1] are mutually exclusive.	Being married to and … and having an ancestor are mutually exclusive.
ObjectPropertyDomain(op c)	If X has Y as a op[N_1] then X is a c[N_2].	If X has Y as an ancestor then X is a person.
ObjectPropertyRange(op c)	If X has Y as a op[N_1] then Y is	If X has Y as an ancestor then Y is a

	a c[N_2].	person.
InverseObjectProperties(op$_1$ op$_2$)	If X has Y as a op[N_1] then Y has X as a op[N_2].	If X has Y as a parent then Y has X as a child.
TransitiveObjectProperty(op)	If X has Y as a op[N_1] and Y has Z as a op[N_1] then X has Z as a op[N_1].	If X has Y as an ancestor and Y has Z as an ancestor then X has Z as an ancestor.
FunctionalObjectProperty(op)	If X has Y as a op[N_1] then Y is the only op[N_1] of X.	If X has Y as a father then Y is the only father of X.
InverseFunctionalObjectProperty(op)	If X has Y as a op[N_1] then Y is the op[N_1] of the only X.	If X has Y as a son then Y is the son of the only X.
ReflexiveObjectProperty(op)	Each X has itself as a op[N].	Each X has itself as an admirer.
IrreflexiveObjectProperty(op)	Each X does not op[V] itself.	Each X does not like itself.
SymmetricObjectProperty(op)	If X has Y as a op[N_1] then Y has X as a op[N_1].	If X has Y as a sibling then Y has X as a sibling.
AntisymmetricObjectProperty(op)	If X has Y as a op[N_1] and Y has X as a op[N_1] then X is identical to Y.	If X has Y as an ancestor and Y has X as an ancestor then X is identical to Y.
SameIndividual(iID$_1$ ... iID$_n$)	iID$_1$ and … and ID$_1$ are the same individual.	Ron and … and Ronald are the same individual.
DifferentIndividuals(iID$_1$ ... iID$_n$)	iID$_1$ and … and ID$_1$ are different individuals.	Rona and … and Aaron are different individuals.
ClassAssertion(iID c)	iID is a c[N].	Rona is a girl.
ObjectPropertyAssertion(op iID$_1$ iID$_2$)	iID$_1$ op[V] iID$_2$.	Rona likes Aaron.
NegativeObjectPropertyAssertion(op iID$_1$ iID$_2$)	iID$_1$ does not op[V] iID$_2$.	Rona does not like Bart.