Report on SNePS and RTS
SNeRG Technical Note 45 *

Jonathan Bona and Stuart C. Shapiro
Department of Computer Science and Engineering
201 Bell Hall
University at Buffalo, The State University of New York
Buffalo, NY 14260-2000
jpbona@cse.buffalo.edu , shapiro@cse.buffalo.edu

February 27, 2009

Abstract

Electronic health record-keeping systems in the Referent Tracking paradigm use globally unique identifiers to make explicit reference to all relevant particular entities, and structure data by referring to universals and relations defined in external, commonly-used ontologies. SNePS is a general-purpose Knowledge Representation, Reasoning and Acting system with a logical language interface.

We have been exploring the use of SNePS as a reasoning system that can combine (1) instance data stored in a Referent Tracking system with (2) facts from domain ontologies such as FMA, and (3) rules from a domain-independent theory such as BFO-core. This paper outlines and discusses the implementation of three individual components we have developed to interface SNePS with tools from each of these three. Together these components make up a proof-of-concept implementation of a SNePS-based ontology reasoning system for use with referent tracking.

1 Referent Tracking System

Referent tracking (RT) is intended to improve electronic health recordkeeping by allowing unique reference to particular things in the world (e.g. patients, their body parts, disorders), universals (e.g. Human Being, Leg, Fracture), and the relationships that hold between and among these (e.g. 'x is a human', 'y is a left leg', 'y is a part of x', 'Leg is a type of Body Part', and so on).

Each particular entity to which the system can refer has assigned to it a globally unique IUI (Instance Unique Identifer) This IUI is then used whenever reference must be made to the particular. References to particulars appear in the assertions that are stored in the system. Several different types of assertions are possible. These include: that a particular instantiates a given universal; that a particular stands in some relation to one or more other particulars; that a particular is associated with a given concept code; and that a particular has assigned to it a name. These assertions also refer to universals and relations that are defined in external ontologies such as the Foundational Model of Anatomy (FMA) [Rosse and Mejino, 2003]. Similarly, assertions that associate concept codes with particulars can and do use codes from systems such as SNOMED CT [The International Health Terminology Standards Development Organisation, 2008].

The software that is the subject of this paper uses the referent tracking prototype implementation discussed in [Shahid Manzoor and Rudnicki, 2007]. We refer to this system in the rest of this paper as The

*This work is supported by the grant "UB Task Force for ontology-based IT support for large scale field studies in Psychiatry" from the John R. Oishei Foundation. The Foundation's mission is to enhance the quality of life for Buffalo area residents by supporting education, healthcare, scientific research and the cultural, social, civic and other charitable needs of the community. The Foundation was established in 1940 by John R. Oishei, founder of Trico Products Corporation.

RTS, or RTS. The RTS uses an RDF-based [Klyne and Carroll, 2004] representation, with data stored in a relational database.

1.1 Tuple Representation

Entries stored in the RTS relational database can be treated as tuples, each of which represents an assertion that was made to, and is stored in, the system. Generally speaking, each tuple records that some agent with a particular id observed at some time that a particular state of affairs obtained at some time.

1.1.1 Particular-to-Universal

Each Particular-to-Universal (PtoU) tuple represents an instantiation relation that was observed to hold between a particular and some universal at some time. Specifically, a PtoU entry says that the agent with one id said at a certain time that the particular with another id instantiates a universal at a certain time. Both the instantiation relation involved and the universal are specific to an ontology. The PtoU tuple contains the name of the universal, the name of the instantiation relation, and the name of the ontology from which each of them comes.
PtoU: $\langle IUIa, ta, inst, io, IUIp, u, uo, tr\rangle$: The particular with id *IUIa* asserted at time *ta* that the particular with id *IUIp* instantiated (via the *inst* relation from ontology *io*) the universal *u* from ontology *uo* at time tr.

The following is a sample data entry from a running instance of the RTS, displayed in this tuple form:

PtoU: ⟨IUI-0,2004.03.23 21:37:53,instance_of,OBO_REL,000003a9-e0a9-480a-bcf1-adaad0d6c7ee,Back of neck,FMA,2004.03.23 21:37:53⟩

The entry represents the fact that the particular with id *IUI-0* asserted on *2004.03.23* at *21:37:53* that the particular with id *000003a9-e0a9-480a-bcf1-adaad0d6c7ee* instantiated via the the *OBO_REL* relation *instance_of* the *FMA* universal *Back of neck* on *2004.03.23* at *21:37:53*.

1.1.2 Particular-to-Name

Each Particular-to-Name (PtoN) tuple represents the assignment of a name to a particular.

PtoN: $\langle IUIa, ta, iuip, name, nt, tr\rangle$: The particular with id *IUIa* asserted at time *ta* that the particular with id *iuip* had assigned to it the name *n* of name type *nt* at time *tr*.

1.1.3 Particular-to-Concept-Code

Particular-to-Concept-Code (PtoCo) tuples are used to associate particulars in the RTS system with concept codes from various concept code systems.

PtoCo: $\langle IUIa, ta, iuip, code, sys, tr\rangle$: The particular with id *IUIa* asserted at time *ta* that the particular with id *iuip* had associated with it the concept code *code* from system *sys* at time *tr*.

1.1.4 Particular-to-Particular

Each PtoP tuple represents that a relation was observed to hold between particular entities at some time.

PtoP: $\langle IUIa, ta, iuip1, iuip2, rel, ro, tr\rangle$: The particular with id *IUIa* asserted at time *ta* that the particular with id *iuip1* stood in the relation *rel* (from ontology *ro*) to the particular with id *iuip2* at time *tr*.

2 SNePS

SNePS is a general-purpose Knowledge Representation, Reasoning, and Acting system. It has its roots in traditional semantic networks systems, from which description logics are also descended. The current version (SNePS 2.7) combines elements of network-based, frame-based, and logic-based systems, with models of

inference appropriate to each paradigm [Shapiro and The SNePS Implementation Group, 2008]. Discussion and examples in this paper use SNePSLOG, the logical language interface to SNePS.

3 Implementation

3.1 SNePS RTS Interface

A goal of our current project is to use SNePS to represent and reason about ontologies and data that use them. This section discusses the interface we have implemented that allows SNePS to pull information from an online RTS system in realtime, reason about that data, and infer new information based on it.

We refer to this interface as the SNePS/RTS Interface, or SNeRTSi.

The Java-based RTS Server software includes an API for accessing and manipulating records in the system. The current implementation of the SNeRTSi directly accesses records in the system using this API. Future versions of the interface could connnect SNePS to RTS using RTS Web Services, which would allow for a more flexible, distributed system than the current implementation.

The rest of this section discusses the representation of RTS data in SNePS.

3.1.1 Example

The SNePSLOG representation of the information in the PtoU tuple:

⟨IUI-0,2004.03.23 21:37:53,instance_of,OBO_REL,000003a9-e0a9-480a-bcf1-adaad0d6c7ee,Back of neck, FMA,2004.03.23 21:37:53⟩ is as follows:

```
Asserted(IUI-0, time(2004,03,23,21,37,53),
               Inst(000003a9-e0a9-480a-bcf1-adaad0d6c7ee,
                       universal(Back of neck,FMA),
                       relation(OBO_REL, instance_of),
                       time(2004,03,23,21,37,53)))}
```

This representation is described in detail below.

3.1.2 Time

We use functional terms to denote unique seconds in time in SNePSLOG:

`time(y,m,d,h,e,s)`: the year `y`, month `m`, day `d`, hour `h`, minute `e`, and second `s`.

The term `time(2002,04,24,00,12,04)` denotes the second at 00:12:04 on 24 April 2002

We also use functional terms to represent time intervals:

`between(t1, t2)`: the interval from time `t1` to time `t2` (inclusive).

The term `between(time(2002,04,24,00,12,04), time(2003,04,24,12,12,04)` denotes the interval from 00:12:04 on 24 April 2002 to 12:12:04 on 24 April 2003.

3.1.3 Universals

We use `universal` functional terms to represent universals.

`universal(u,o)`: the universal `u` from ontology `o`.

The following term denotes the FMA ontology's 'Back of neck' universal:

`universal(Back of neck,FMA)`

3.1.4 Relations

Similarly, we use `relation` functional terms to denote relations:
`relation(r,o)`: the relation `r` from ontology `o`.

The following term denotes the OBO REL ontology's 'instance_of' relation:
`relation(OBO_REL,instance_of)`

3.1.5 Concepts

The RTS uses concept codes from systems such as SNOMED CT. We use functional terms to denote concepts:
`concept(c,s)`: the concept `c` from system `s`.

The following denotes the SNOMED CT system's 'Marriage, life event' concept (16078700):
`concept(SNOMED CT, 16078700)`

3.1.6 Tuples as Assertions

Each type of tuple is represented in SNePSLOG using a predicate that is unique to that tuple. These are presented in detail below.

It is possible, using the `Asserted` predicate, to reason about the asserting agent and the time at which the assertion was made. We are mostly interested in reasoning about the contents of the assertion itself and will often ignore the agent IUI and the time unless these are essential to the specific reasoning task.

The `Asserted` predicate is defined as:
`Asserted(p,t,a)`: the agent with id `p` asserted at time `t` that `a` was the case.

Currently, SNeRTSi treats as true any assertion that was made by any agent. This amounts to having SNePS believe by default anything that has been entered into the RTS system. This is expressed in SNePS as a simple rule:
`all(a,t,assn)(Asserted(a,t,assn) => assn)`: consider to be true any assertion that was made by any agent at any time.

This allows users of SNeRTSi to write rules and queries that deal directly with the contents of assertions rather than having to account for the asserting agent and the time at which the assertion was made. The meta information about the assertion is not discarded; it also becomes part of the SNePS knowledge base and is available for any reasoning task that should require it. In particular, knowing *who* said that something was the case will be useful in handling contradictions.

3.1.7 Particular-to-Name

We represent PtoN name assignments in SNePSLOG using:
`Named(p, n, nt, t)`: The particular with id `p` had the name `n` (of name type `nt`) at time `t`.

The following says that the particular entity with id 17 had the name 'John' of nametype 'Name' on 24 April 2002 at 00:12:04.
`Named(17, John, Name, time(2002,04,24,00,12,04))`

3.1.8 Particular-to-Universal

The SNePSLOG representation of a PtoU tuple uses the `Inst` predicate, and `universal` and `relation` terms. `Inst` is defined as:
`Inst(p,u,r,t)`: the particular with id `p` instantiated universal `u` via instantiation relation `r` at time `t`.

The following says that the OBO_REL ontology's *instance_of* relation held at 00:12:04 on 24 April 2002 between the particular with id 42 and the FMA ontology's *Back of neck* universal:

`Inst(42,universal(Back of neck,FMA),relation(OBO_REL,instance_of),time(2002,04,24,00,12,04))`.
Or: 'On 24 April 2002 at 00:12:04, the thing with id 42 was a back of neck'.

3.1.9 Particular-to-Concept-Code

The SNePSLOG representation of a PtoCo tuple uses the `PtoCo` predicate:
`PtoCo(p,c,t)`: The particular with id `p` is associated with the concept `c` at time `t`.

The following says that the particular with id 321 is associated with the concept 16078700 ('Marriage, life event') from the SNOMED CT system on 24 April 2002 at 00:12:04:
`PtoCo(321,concept(16078700,SNOMED CT),time(2002,06,02,00,12,04))`

3.1.10 Particular-to-Particular

The SNePSLOG representation of a PtoP tuple uses a `relation` term as well as the `PtoP` predicate:
`PtoP(p1,p2,r,t)`: `p1` was in relation `r` to `p2` at time `t`.

The following says that the 'part' relation in the OBO_REL ontology held between the particular with id hand15 and the particular with id arm375 on 24 April 2002 at 00:12:04:
`PtoP(hand15,arm375,relation(part,OBO_REL),time(2002,06,02,00,12,04))`
Or: 'On 24 April 2002 at 00:12:04, the thing with id hand15 was a part of the thing with id arm375.'

3.2 SNePS OWL Importer

Many ontology editing and reasoning systems import and export entire ontologies as OWL (Web Ontology Language) files, and many existing ontologies are implemented in OWL [McGuinness and van Harmelen, 2004].

We have developed the SNePS OWL Importer (SNOWLi) [Bona, 2007], which allows SNePS to import, represent, and use information stored in OWL [McGuinness and van Harmelen, 2004] files. In particular we have used SNOWLi to read and represent ontologies expressed in OWL DL, including the OWL versions of FMA, BFO, and the OBO Foundry's Human Disease domain ontology.

3.2.1 OWL Example from Human Disease Ontology

The OWL excerpt shown here is taken from a translated OWL version of the the OBO Foundry's Human Disease domain ontology [Smith et al., 2007]. The complete file is available for dowload from `http://www.berkeleybop.org/ontologies/obo-all/disease_ontology/disease_ontology.owl`.

The following OWL text is a part of the definition of the term *gallbladder disease* in the disease ontology. The definition of *gallbladder disease* in OBO format is much shorter than the OWL version. The full translation of this term into an *owl:Class* is nearly five hundred lines long. A short excerpt with most of the lines omitted is given here:

```
33:  <owl:Class rdf:about="http://purl.org/obo/owl/DOID#DOID_0000000">
34:    <rdfs:label xml:lang="en">gallbladder disease</rdfs:label>
35:    <oboInOwl:hasOBONamespace>mixed_disease_ontology</oboInOwl:hasOBONamespace>
36:    <oboInOwl:hasDbXref>
37:      <oboInOwl:DbXref>
38:        <rdfs:label>GeneRIF:14567398</rdfs:label>
39:        <oboInOwl:hasURI rdf:datatype="http://www.w3.org/2001/XMLSchema#anyURI">
40:            http://purl.org/obo/owl/GeneRIF#GeneRIF_14567398
41:         </oboInOwl:hasURI>
42:      </oboInOwl:DbXref>
43:    </oboInOwl:hasDbXref>
44 - 523:          ....
```

```
524:  </owl:Class>
```

This excerpt defines an OWL class with id `http://purl.org/obo/owl/DOID#DOID_0000000`. The class is labelled "gallbladder disease", and is in the OBO namespace *mixed_disease_ontology*. It is cross-referenced with a term in another OBO ontology. The cross referenced term is labelled "GeneRIF:14567398" and has as its URI `http://purl.org/obo/owl/GeneRIF#GeneRIF_14567398`. The URI's XML Schema datatype is *anyURI*.

The following are the propositions that result when the excerpt shown above of the *gallbladder disease* portion of the human disease OWL file is imported into SNePS.

```
p1:    rdf:type(http://purl.org/obo/owl/DOID#DOID_0000000,owl:Class)
p2:    rdfs:label(http://purl.org/obo/owl/DOID#DOID_0000000,gallbladder disease)
p3:    oboInOwl:hasOBONamespace(http://purl.org/obo/owl/DOID#DOID_0000000,mixed_disease_ontology)
p4:    oboInOwl:hasDbXref(http://purl.org/obo/owl/DOID#DOID_0000000,anon1)
p5:    rdf:type(anon1,oboInOwl:DbXref)
p6:    rdfs:label(anon1,GeneRIF:14567398)
p7:    oboInOwl:hasURI(anon1,http://purl.org/obo/owl/GeneRIF#GeneRIF_14567398)
```

3.3 BFO Axioms

We have implemented a SNePS version of the *BFO core* theory based on Thomas Bittner's Isabelle/FOL implementation of BFO core[1]. This includes definitions and axioms for spatial relations, universals and universal instantiation, parthood relations, and more. SNePS can use some or all of these domain-independent subtheories to reason more effectively about the particulars, universals, and relations to which the RTS instance data refers.

For instance, the BFO core subtheory on Universal Parthood includes axioms and theorems about different parthood relations that can possibly hold between Universals (and between particulars that instantiate them). It implements three distinct such relations, each of which has different properties that are explicated in the implementation.

The first (*UPt1*) holds between any two universals `u1` and `u2` at a time `t` if and only if for every object `o1` that instantiates `u1` at that time, there is some object `o2` that instantiates `u2` at that time, and such that `o1` is part of `o2` at that time.

The second (*UPt2*) holds between any two universals `u1` and `u2` at a time `t` if and only if for every object `o2` that instantiates `u2` at that time, there is some object `o1` that instantiates `u1` at that time, and such that `o1` is part of `o2` at that time.

The third relation (*UPt12*) holds between any two universals `u1` and `u2` at a time `t` if and only if both *UPt1* and *UPt2* hold between `u1` and `u2` at that time.

See Appendix A for a listing of these rules implemented in SNePSLOG.

4 Example Uses

This section shows and discusses SNePS queries that use SNeRTSi to access data in the RTS.

The `Asserted` predicate in SNeRTSi has attached to it a Lisp function that handles connecting to the RTS. When SNePS comes to wonder whether `Asserted(x,y,z)` holds (for any agent `x`, time `y`, and assertion `z`), it calls this function. The function examines its arguments, which may be a combination of ground terms and variables, and passes the appropriate values on to functions in the RTS Java API. A database query is executed, and sequences of strings representing the matching tuples are returned to the SNeRTSi Lisp functions. These results are processed and converted into a list of substitutions each of which, when applied, makes a ground instance of `Asserted(x,y,z)`. When this list of substitutions is returned by `asserted_fn`, the instances are asserted in the SNePS knowledge base.

[1] `http://www.acsu.buffalo.edu/~bittner3/Theories/BFO/`

4.1 Finding Concept Associations

What concepts are associated with the particular whose IUI is 00005cc5-7fd2-461e-9866-72af31c147f1?

```
: PtoCo(00005cc5-7fd2-461e-9866-72af31c147f1, ?concept, ?t)?
  wff67!:  PtoCo(00005cc5-7fd2-461e-9866-72af31c147f1,
                           concept(248152002,SNOMED CT),
                           time(2002,06,02,00,12,04))
  wff66!:  PtoCo(00005cc5-7fd2-461e-9866-72af31c147f1,
                           concept(116154003,SNOMED CT),
                           time(2002,06,02,00,12,04))
```

The concept codes associated with this particular are `116154003 (patient)` and `248153002 (female)` . This indicates that the particular in question is a female patient.

4.2 Finding Parts of Individuals

Queries and rules that use the full 'Pto$X$' and other tuple predicates can be abbreviated by defining predicates in SNePS for the type of facts we're interested in.

Consider the SNePS query to ask which things a particular has as its parts (at any time):

`PtoP(00005cc5-7fd2-461e-9866-72af31c147f1, ?x, relation(part, OBO_REL), ?t)?`

This can be simplified by defining a `Part-of` predicate, and writing a rule to allow SNePS to infer formulas that use `Part-of` from the corresponding `PtoP` formula:

`Part-of(p1,p2,t)`: `p1` is part of `p2` at time `t`.

Rule: for any two particulars `p1` and `p2`, if `p1` stands in a relation called `part` (from any ontology) to `p2` at a time `t`, then `p1` is a part of `p2` at `t`:

`all(p1,p2,ont,t)(PtoP(p1,p2,relation(part,ont), t) => Part-of(p2,p1,t)).`

The `Part-of` predicate can then be used to make queries: what are the parts of the individual with IUI `00005cc5-7fd2-461e-9866-72af31c147f1`?

```
: Part-of(?x,00005cc5-7fd2-461e-9866-72af31c147f1,?t)?
  wff49!:  Part-of(0e5742fe-c361-49c1-903c-f46b39afabfe,
                           00005cc5-7fd2-461e-9866-72af31c147f1,
                           time(2004,10,25,00,12,04))
  wff50!:  Part-of(db162622-3349-43d2-ac92-5df653e75c2c,
                           00005cc5-7fd2-461e-9866-72af31c147f1,
                           time(2002,04,24,00,12,04))
  wff51!:  Part-of(ea2c7c41-8e4f-4e4c-b0d0-5a2ef1d15c6a,
                           00005cc5-7fd2-461e-9866-72af31c147f1,
                           time(2004,09,26,00,12,04))
 ...
```

In this case the system found the IUIs of sixteen particulars that have asserted of them in the RTS that they are part of the particular with IUI `00005cc5-7fd2-461e-9866-72af31c147f1`. Only three are shown here as SNePS well-formed-formulae (wffs).

`wff49` says that the particular with id `0e5742fe-c361-49c1-903c-f46b39afabfe` is a part of the particular with id `00005cc5-7fd2-461e-9866-72af31c147f1` at time 00:12.04 on November 25, 2004.

`wff50` says that the particular with id `db162622-3349-43d2-ac92-5df653e75c2c` is a part of the particular with id `00005cc5-7fd2-461e-9866-72af31c147f1` at time 00:12.04 on April 04, 2002.

`wff49` says that the particular with id `0e5742fe-c361-49c1-903c-f46b39afabfe` is a part of the particular with id `00005cc5-7fd2-461e-9866-72af31c147f1` at time 00:12.04 on September 26, 2004.

4.3 Universal Instantiation

At this point, we know that the particular with IUI `00005cc5-7fd2-461e-9866-72af31c147f1` has some parts, and that three of them are the particulars with the IUIs `ea2c7c41-8e4f-4e4c-b0d0-5a2ef1d15c6a`, `db162622-3349-43d2-ac92-5df653e75c2c`, and `0e5742fe-c361-49c1-903c-f46b39afabfe`. It may be important to know what sorts of things these parts are. This can be determined by asking which universals they instantiate. The following shows three queries, one for each of the particular parts seen above. Each query asks 'Which universal(s) does the particular instantiate, and via which relation, and at what time?'. The answers reveal that these three particulars instantiate `FMA:Left hip`, `FMA:Nose`, and `FMA:Right thumb`.

```
: Inst(db162622-3349-43d2-ac92-5df653e75c2c, ?u, ?r, ?t)?
  wff62!:  Inst(db162622-3349-43d2-ac92-5df653e75c2c,
                        universal(Nose,FMA),
                        relation(instance_of,OBO_RE),
                        time(2002,04,24,00,12,04))
: Inst(ea2c7c41-8e4f-4e4c-b0d0-5a2ef1d15c6a, ?u, ?r, ?t)?
  wff64!:  Inst(ea2c7c41-8e4f-4e4c-b0d0-5a2ef1d15c6a,
                        universal(Left hip,FMA),
                        relation(instance_of,OBO_RE),
                        time(2004,09,26,00,12,04))
: Inst(0e5742fe-c361-49c1-903c-f46b39afabfe, ?u, ?r, ?t)?
  wff66!:  Inst(0e5742fe-c361-49c1-903c-f46b39afabfe,
                        universal(Right thumb,FMA),
                        relation(instance_of,OBO_RE),
                        time(2004,10,25,00,12,04))
```

`wff62` says that the particular with id `db162622-3349-43d2-ac92-5df653e75c2c` instantiates the `Nose` universal from the `FMA` ontology via the `OBO_REL instance_of` relation at 00:12.04 on April 24, 2002, or "db162622-3349-43d2-ac92-5df653e75c2c is a Nose."
`wff64` says that the particular with id `ea2c7c41-8e4f-4e4c-b0d0-5a2ef1d15c6a` instantiates the `Left hip` universal from the `FMA` ontology via the `OBO_REL instance_of` relation at 00:12.04 on September 26, 2004, or "ea2c7c41-8e4f-4e4c-b0d0-5a2ef1d15c6a is a Left hip."
`wff66` says that the particular with id `0e5742fe-c361-49c1-903c-f46b39afabfe` instantiates the `Right thumb` universal from the `FMA` ontology via the `OBO_REL instance_of` relation at 00:12.04 on September 26, 2004, or "0e5742fe-c361-49c1-903c-f46b39afabfe is a Right thumb."

5 Conclusions

We have presented here the SNePS/RTS Interface, which connects SNePS to an instance of the Referent Tracking System. This interface allows SNePS to access instance data in the system *as needed* in the course of answering a query or performing other reasoning tasks. It provides a simple way of querying the RTS system from SNePS, a logical system with a syntax similar to standard first-order logics.

This interface may also be used, in combination with SNePS' native reasoning and acting capabilities, to perform advanced reasoning tasks with patient data in a referent tracking system. A SNePS agent with access to data in the system can perform reasoning on that data, along with information from other sources (ontologies, concept systems, etc), and derive new information that follows from what was already known. Such an agent could even assert derived information back into the RTS database when appropriate.

For instance, we could implement in SNePS an online referent tracking reasoning agent tasked with performing quality assurance checks on patient data. As data is added to the system, the agent would check it for consistency in real time using SNePS' built-in contradiction handling system, and take appropriate action when any contradiction is detected.

6 Future Work

6.1 Subsystems Integration

SNeRTSi will allow SNePS to combine instance data from the RTS with facts from ontologies such as *OBO REL* and *FMA*, and with axioms and definitions from more domain-independent ontological theories such as *BFO core* [Bittner, T., M. Donnelly, and B. Smith, 2007] . By combining these elements, SNePS will be able to handle more complicated queries, the answers to which are not asserted in the RTS system.

For example, the query: 'What particulars are part of `x`'s Face?' (where `x` is the IUI of a particular that is a patient) may have true answers that are not asserted in the RTS database. Some of these answers can be inferred based on information about the `Face` universal in FMA: that there is a parthood relation between `FMA:Nose` and `FMA:Face`, `FMA:Eye` and `FMA:Face`, etc. In order to do this, the system will also need to know the properties of the universal parthood relation in question and have axioms for reasoning about the relation and the consequences for particulars that instantiate the universals involved.

We have previously developed a SNePS OWL Importer (SNOWLi)[Bona, 2007], which allows SNePS to access ontologies expressed in OWL. We have used SNOWLi to import and represent in SNePS ontologies expressed in OWL DL, including OWL versions of FMA, BFO [Stenzhorn et al., 2007], and the OBO Foundry's Human Disease domain ontology[2] [Smith et al., 2007]. We have also implemented a SNePS version of the *BFO core* theory based on Thomas Bittner's Isabelle/FOL implementation of BFO core[3]. This includes definitions and axioms for different universal parthood relations. These will help provide SNePS with information about, and rules for reasoning about, the universals and relations to which the RTS instance data refers.

We have started work on integrating these distinct subsystems into a single system that, when completed, will be able derive new facts using instance data available in the RTS, information about universals and relations implemented in external ontologies, and domain-independent theories such as BFO core.

6.2 Online Ontology Use

The original version of SNOWLi works by importing and representing in SNePS an entire OWL ontology at a time. This approach has proven to be inefficient for large ontologies, as it uses significant amounts of time and other computational resources, and much of the information stored in any large ontology is irrelevant to any particular reasoning task. We are currently modifying SNOWLi to import portions of ontologies *as needed* during reasoning, much as instance data is imported from the RTS system only when required within SNeRTSi.

The original version also is configured only to access ontologies that are stored as OWL files on the local filesystem. We will modify SNOWLi to directly access ontologies on the web when available..

6.3 Improve Interface Between SNePS and RTS

Our current proof-of-concept implementation connects SNePS to RTS by directly accessing tuples in the RTS database via a Java API. This arrangement requires SNePS to make calls out to a Java program that has direct access to the database. This is not distributable, scalable, efficient, or robust. The full implementation of SNeRTSi will connect to RTS using RTS Web Services, which should solve these issues.

6.4 Natural Language

A subgroup of the SNePS Research Group is engaged in providing a full natural language (English) processing system for SNePS. When completed, this interface will allow SNePS to accept as inputs, and reason about, free text such as occurs in existing medical records.

[2] `http://www.berkeleybop.org/ontologies/obo-all/disease_ontology/disease_ontology.owl`
[3] `http://www.acsu.buffalo.edu/~bittner3/Theories/BFO/`

References

[Bittner, T., M. Donnelly, and B. Smith, 2007] Bittner, T., M. Donnelly, and B. Smith (2007). A Spatio-Temporal Top-Level Ontology for Geographic Information Processing. *Technical report, Department of Philosophy, SUNY Buffalo*.

[Bona, 2007] Bona, J. (2007). OWL Ontologies in SNePS. *SNeRG Technical Note 41, Department of Computer Science and Engineering, SUNY at Buffalo*.

[Klyne and Carroll, 2004] Klyne, G. and Carroll, J. J. (2004). *Resource Description Framework (RDF): Concepts and Abstract Syntax*. Published online on February 10th, 2004 at `http://www.w3.org/TR/2004/REC-rdf-concepts-20040210/`.

[McGuinness and van Harmelen, 2004] McGuinness, D. L. and van Harmelen, F. (2004). *OWL Web Ontology Language Overview*. Published online on February 10th, 2004 at `http://www.w3.org/TR/2004/REC-owl-features-20040210/`.

[Rosse and Mejino, 2003] Rosse, C. and Mejino, J. (2003). A reference ontology for biomedical informatics: the Foundational Model of Anatomy. *Journal of Biomedical Informatics*, 36(6):478–500.

[Shahid Manzoor and Rudnicki, 2007] Shahid Manzoor, W. C. and Rudnicki, R. (2007). Implementation of a referent tracking system. *International Journal of Healthcare Information Systems and Informatics*, 2(4):41–58.

[Shapiro and The SNePS Implementation Group, 2008] Shapiro, S. C. and The SNePS Implementation Group (2008). *SNePS 2.7 User's Manual*. Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY. Available as `http://www.cse.buffalo.edu/sneps/Manuals/manual27.pdf`.

[Smith et al., 2007] Smith, B., Ashburner, M., Rosse, C., Bard, J., Bug, W., Ceusters, W., Goldberg, L. J., Eilbeck, K., Ireland, A., Mungall, C. J., the OBI Consortium, Leontis, N., Rocca-Serra, P., Ruttenberg, A., Sansone, S.-A., H, R., Scheuermann, Shah, N., Whetzel, P. L., and Lewis, S. (2007). The obo foundry: Coordinated evolution of ontologies to support biomedical data integration. *Nature Biotechnology*, 25(11).

[Stenzhorn et al., 2007] Stenzhorn, H., Spear, A., Grenon, P., and Ruttenberg, A. (2007). *Basic Formal Ontology (BFO)*. Published online on September 18th, 2007 at `http://www.ifomis.org/bfo/1.1`.

[The International Health Terminology Standards Development Organisation, 2008] The International Health Terminology Standards Development Organisation (2008). *SNOMED Clinical Terms Technical Reference Guide*.

Appendix A: BFO Core Universal Parthood in SNePS

Representing Existentials in SNePS

SNePS 2.7 implements the universal quantifier but does not use the existential quantifier. A formula that contains an existentially-quantified variable can be converted to an equivalent formula that uses universals and skolem functions. This is the approach used here. For instance, in the following section, the skolem function *upt1DFN* with arguments (*c,d,t,x*) plays the role of an existentially-quantified variable within the scope of *c,d,t* and *x*.

Representation

The following are the predicates used in the SNePS representation of universal parthood relations.

`Un(x)` : `x` is a universal
`Ti(x)` : `x` is a time
`Ob(x)` : `x` is an object
`P(x,y,z)` : `x` is a part of `y` at time `t`
`Inst(x,y,z)` : `x` istantiates `y` at time `t`
`UPt1(x,y,z)` : `x` stands in the UniversalParthood1 relation to `y` at time `z`
`UPt2(x,y,z)` : `x` stands in the UniversalParthood2 relation to `y` at time `z`
`UPt12(x,y,z)` : `x` stands in the UniversalParthood12 relation to `y` at time `z`

Universal Parthood Relation UPt1

For any two universals, *c, d* and time *t*:

If *UPt1* holds between *c* and *d* at *t*, then every object *x* that instantantiates *c* at *t*, is part of some object (denoted here by the skolem functional term *upt1DFN(c,d,t,x)*) that instantiates *d* at *t*.

```
all(c,d,t)( {Un({c,d}), Ti(t)} &=>
    (UPt1(c,d,t) =>
        all(x)({Ob(x), Inst(x,c,t)} &=>
            andor(3,3){ Ob(upt1DFN(c,d,t,x)),
                        Inst(upt1DFN(c,d,t,x),d,t),
                        P(x,upt1DFN(c,d,t,x),t)}))).
```

If every object *x* that instantiates *c* at *t* is part of an object *y* that instantiates *d* at *t*, then *UPt1* holds between *c* and *d* at *t*.

```
all(c,d,t)( {Un({c,d}), Ti(t)} &=>
    ( all(x,y)({Ob({x,y}), Inst(x,c,t)} &=>
        ( Inst(y,d,t) and P(x,y,t))) => UPt1(c,d,t))).
```

Universal Parthood Relation UPt2

For any two universals, *c, d* and time *t*:

If *UPt2* holds between *c* and *d* at *t*, then every object *y* that instantiates *d* at *t* has as its part at *t* some object (denoted here by the skolem functional term *upt2DNF(c,d,t,y)*) that instantiates *c* at *t* .

```
all(c,d,t)( {Un({c,d}), Ti(t)} &=>
    (UPt2(c,d,t) =>
        all(y)( {Ob(y), Inst(y,d,t)} &=>
            andor(3,3){ Ob(upt2DNF(c,d,t,y)),
                Inst(upt2DNF(c,d,t,y),c,t),
                P(upt2DNF(c,d,t,y),y,t)}))).
```

If every object y that instantiates d at t has as its part an object x that isntantiates c at t, then *UPt2* holds between c and d at t.

```
all(c,d,t)( {Un({c,d}), Ti(t)} &=>
    all(x,y)( {Ob({x,y}), Inst(y,d,t)} &=>
        ({Inst(x,c,t), P(x,y,t)} &=> UPt2(c,d,t)))).
```

Universal Parthood Relation UPt12

For any two universals, *c, d* and time t:

UPt12 holds between c and d at t if and only if *UPt1* holds between c and d at t, and *UPt2* holds between c and d at t,.

```
all(c,d,t)({Un({c,d}), Ti(t)} &=> (UPt12(c,d,t) <=> (UPt1(c,d,t) and UPt2(c,d,t)))).
```