Visually Interacting with a Knowledge Base Using Frames, Logic, and Propositional Graphs

Daniel R. Schlegel and **Stuart C. Shapiro**
University at Buffalo
Buffalo, New York, USA
{drschleg | shapiro}@buffalo.edu

Abstract

The knowledge base of a knowledge representation and reasoning system can simultaneously be thought of as being logic, frame, and graph-based. We present a method for naturally extending this three-fold view to methods for visual interaction with the knowledge base in the context of SNePS 3 and its newly developed user interface. Assertions to, and search of, the knowledge base are tasks well suited to a frame or logical representation. Visualization on the other hand is best done through the use of propositional graphs. We show how these interaction techniques, which are extensions of the underlying knowledge base representation, augment each other to allow knowledge engineers to manipulate and view large knowledge bases.

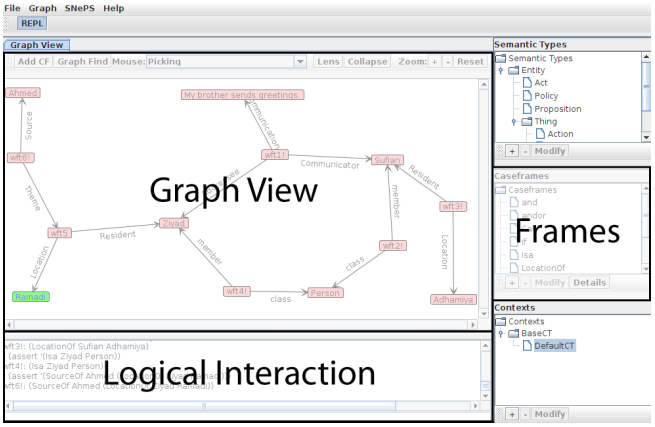

Figure 1: The SNePS 3 GUI supports graph, logic, and frame interactions.

1 Introduction

The knowledge base (KB) of a knowledge representation and reasoning (KR) system can be conceived of as a set of logical expressions, as a set of frames, or as a graph. The choice determines how the user interface presents the KB to the user or knowledge engineer. The KB of the SNePS 3 KR system [Shapiro, 2000] can be thought of as all three simultaneously, and the recently developed SNePS 3 user interface makes use of all three paradigms. In this paper, we focus on the presentation of a SNePS 3 KB as a graph, and on the way that conceiving of it as a set of frames and logical expressions influences the visualization of the graph.

The SNePS 3 graphical user interface (GUI, Figure 1) highlights the visualization of a SNePS 3 KB as a propositional graph. It allows the user to add to the KB using either the logic or frame paradigm, and to select a part of the KB for graphical display using frames in a relational database-esque visual query by example (QBE) way. The user can also hide or show specific instances of a relation through interactions with the graph itself. Binary relations in the graph can be collapsed to provide a less cluttered display without changing the semantics of the graph. In this way, most of the interaction with the knowledge base in terms of assertion and search is carried out in a frame-based way, but visualization of the result is largely graph-based.

In Section 2 we introduce SNePS 3, its three-fold knowledge base representation, and the relationships between those three views. We discuss frame-based interactions with the knowledge base in Section 3 and collapsing binary relations in the graph in Section 4. Finally the usefulness of the system will be discussed along with a comparison with ontology editing software in Section 5.

2 SNePS 3

2.1 Logical Expressions

A SNePS 3 KB may be thought of as a set of logical expressions. For example, the first eight lines of Figure 2 show a user using the logical interaction pane of the SNePS 3 GUI to assert four atomic propositions, and the GUI echoing them back, each labeled with an internal name of the form `wfti!`. The choice of "`wft`" to begin the name will be explained shortly. The "`!`" is appended to the name to indicate that the proposition is asserted (considered to be True in the KB). The syntax used for logical expressions is a version of CLIF [ISO/IEC, 2007], and includes: `not`; `if`; the set-oriented connectives `and`, `or`, `nand`, `nor`, `xor`, `iff`, `andor`, and `thresh` [Shapiro, 2010]; and arbitrary and indefinite terms

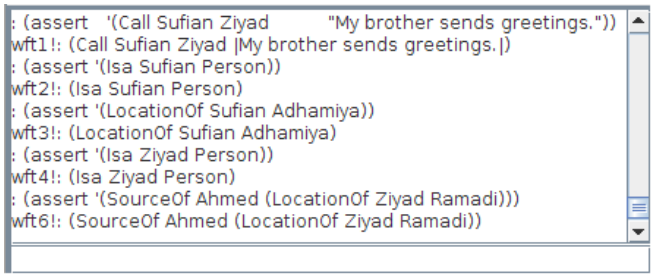

Figure 2: A set of assertions in the text-based logical interface to SNePS 3.

[Shapiro, 2004].[1]

To facilitate metaknowledge—knowledge about knowledge [Shapiro *et al.*, 2007], propositions are considered to be first-class objects in the semantic domain, and all well-formed SNePS logical expressions, including those that look like formulas, are terms [Shapiro, 1993]. So what look like predicates (for example, "`Call`", "`Isa`", "`LocationOf`", and "`SourceOf`" in Figure 2) are actually proposition-valued functions. Even "logical connectives" are function symbols. The internal names of SNePS 3 expressions start with "`wft`" as a reminder that they are "well-formed terms." This is illustrated in the last assertion of Figure 2, which is intended to represent the proposition that Ahmed is the source of the proposition that the location of Ziyad is Ramadi. Taken together, the assertions of Figure 2 are intended to mean that *"Sufian, a person in Adhamiya, called Ziyad, a person who, according to Ahmed, is in Ramadi, saying 'My brother sends greetings.'"*

2.2 Frames

Before a function symbol can be used, it must be associated with a caseframe. A caseframe has a *semantic type* t, specifies a *slot* s_i for each argument x_i of the function, and is associated with one or more function symbols. `Isa` is predefined in SNePS 3 as associated with a caseframe whose slots are `member` and `class` and whose semantic type is `Proposition`. The slots in a caseframe are ordered as they are upon its definition, corresponding to the order of the arguments of the function it is meant to represent.

The user interface we have developed allows the graphical definition of such a caseframe. Figure 3 shows the graphical definition of the `Isa` caseframe. The user selects the slots they are interested in and moves them to the right column. The order they are in from top to bottom represents the ordering of the slots in the caseframe. This graphical view is automatically converted to its appropriate representation in the logical interface to SNePS 3 and displayed in the user interface (see Figure 4). Also displayed in the user interface is the current listing of defined caseframes, which in Figure 5 now includes the `Isa` caseframe.

A *frame* is an instance of a caseframe. That is, the term $\mathrm{F}(x_1, ..., x_n)$ is represented by the frame of type t whose

[1] SNePS 3 examples shown in this paper will be limited to ground atomic propositions.

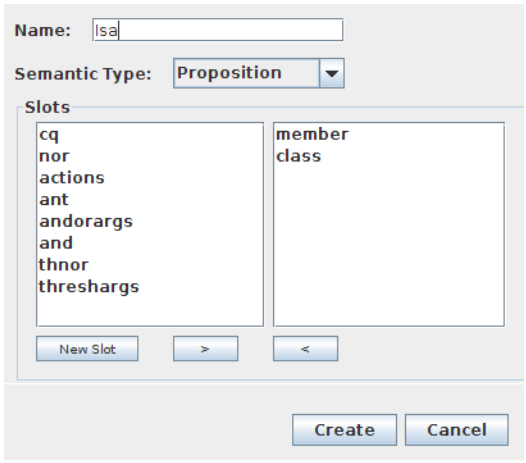

Figure 3: The user interface for defining a caseframe, populated with the slots `member` and `class` for defining the `Isa` caseframe.

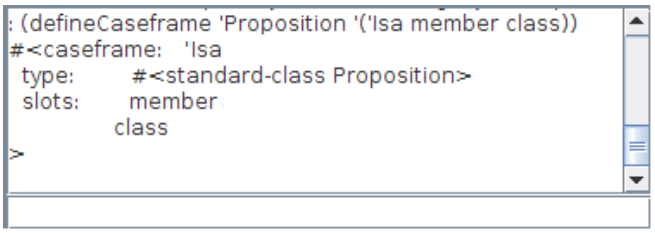

Figure 4: The logical interface displaying the execution of the input in Figure 3 in the logical language.

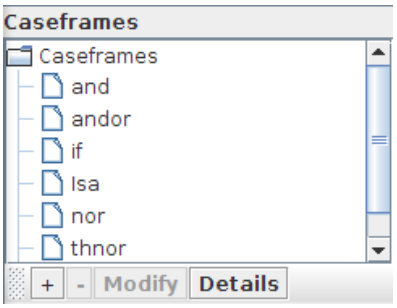

Figure 5: The list of defined caseframes, including the `Isa` caseframe.

slots are $s_1, ..., s_n$ filled by the representations of $x_1, ..., x_n$, respectively. So `wft2` is a frame whose `member` slot contains the *filler* `Sufian`, and whose `class` slot contains the filler `Person`. In this example, `Call` is associated with a frame whose slots are `Communicator`, `Addressee`, and `Communication` (eg. `wft1`), `LocationOf` is associated with a frame whose slots are `Resident` and `Location` (eg. `wft3`), and `SourceOf` is associated with a frame whose slots are `Source` and `Theme` (eg. `wft6`).[2]

[2] The `Call`, `LocationOf` and `SourceOf` frames are based

Caseframes, motivated by Fillmore's case theory [Fillmore, 1968], are comparable to relational database schemas, with slot names corresponding to attributes, and frames to rows of a relational database table. There are several differences, however: a SNePS frame slot may be filled with another frame; and a SNePS frame slot may have multiple fillers. Multiple fillers in a slot are interpreted conjunctively. If one frame has two fillers in one slot and three in another, the frame is semantically equivalent to six frames with only one filler in each of the two slots.

2.3 Propositional Graphs

A SNePS 3 KB may be conceived of as a graph. Every term, whether an individual constant, an arbitrary or indefinite term,[3] or a functional term (whether or not denoting a proposition), corresponds to a node in the graph. Slot names label directed arcs that go from nodes corresponding to frames (functional terms) to the nodes that correspond to the fillers of that slot in that frame (arguments of the function). This is illustrated in Figure 6, which shows the same KB created in Figure 2. Recall that an exclamation mark, "`!`", is appended to the `wft` names indicating those propositions that are asserted in the KB. For example, `wft6!` is asserted to indicate that it is true that Ahmed was the source of the information that Ziyad is in Ramadi, but `wft5` is not asserted indicating that Ahmed is not (yet?) believed. The graph is rendered within the GUI by the JUNG (Java Universal Network/Graph) system [The JUNG Framework Development Team, 2010].

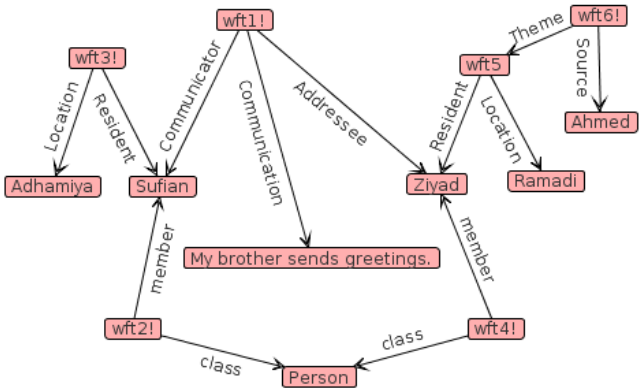

Figure 6: A graphical view of the SNePS 3 KB created in Figure 2, meaning *"Sufian, a person in Adhamiya, called Ziyad, a person who, according to Ahmed, is in Ramadi, saying 'My brother sends greetings.' "*

A SNePS 3 graph may also be considered a hypergraph with labeled nodes and edges. The `wft` nodes correspond to hyperedges, with the benefit, however, that they can have arcs pointing into them (for example, `wft5` of Figure 6). We believe that visualizing the graphs as in Figure 6 is clearer than replacing the `wft` nodes with contours. However, a simplification is discussed and illustrated in §4 below.

3 Frame-Based Interaction

The addition of new knowledge to the KB requires that all slots of the desired caseframe are filled with the proper number of fillers, specified during the definition of the slots themselves. This number can vary between instances of a single caseframe. This is difficult to enforce graphically; the caseframe would have to be instantiated one piece at a time, leading to periods where the graph being edited would be syntactically invalid and inconsistent with the KB that supports it. Relaying in what ways the graph is incorrect and how the issues should be resolved to the user is very difficult. Our solution to this problem is for the user to add knowledge using either the traditional logic based interface or a frame based interface. As you can see in the logical interface as described above, information is added to the KB using logical representations of frames, so the visual interface we have developed is a conceptual extension of the logical language. In fact, when assertions are added to the KB using the frame-based user interface, they are converted to the logical language first and displayed in the logical interface portion of the GUI.

The frame-based interface we have devised for assertions can be seen in Figure 7. The user selects the caseframe which they wish to add an instance of to the KB. This automatically provides a frame based view of the slots which require fillers. Conceptually frames are often seen as tables of slots and fillers, and this is the metaphor we use here. As previously mentioned, the slots in a SNePS 3 caseframe may be configured such that they may exist in the frame a variable number of times. For example, a single instance of the caseframe Isa can be used to say that both Toto and Fido are instances of both Dog and Pet. In the interface the user may add instances of slots to the frame which then must be assigned values by the user. In using an interface such as this, the inputs required by the user become obvious and easily machine verifiable. No changes to the graph, and hence KB, are made until the change is guaranteed to result in a valid assertion to the KB.

Queries can be performed on the KB and the results displayed in the graph using Query By Example [Zloof, 1975]. This is the same as the visual QBE interface designed for database systems wherein the user is shown an empty row for a table and can enter values into fields they are interested in in order to filter results. This speaks to the general likeness of a KB with a database system wherein frames are like rows in a database table and the caseframe defines the table itself.[4]

Using our method, the knowledge engineer can display only the parts of a graph matching certain parts of a relation. In Figure 8 the interface we use for the QBE process is shown. The engineer first chooses the relation (caseframe) they are interested in and chooses to provide fillers for none, some, or all of the slots shown. All areas not completed in

on the Contacting, Residence, and Source_of_getting frames of FrameNet [Fillmore, 1976; Ruppenhofer *et al.*, 2006], but simplified for this paper.

[3] Arbitrary and indefinite terms are not illustrated in this paper.

[4] The analogy begins to break down when we consider two abilities which differentiate a caseframe from a database table: the ability to have a variable number of instances of a slot act as the fillers of a frame, and the ability to fill a slot with an instance of another frame.

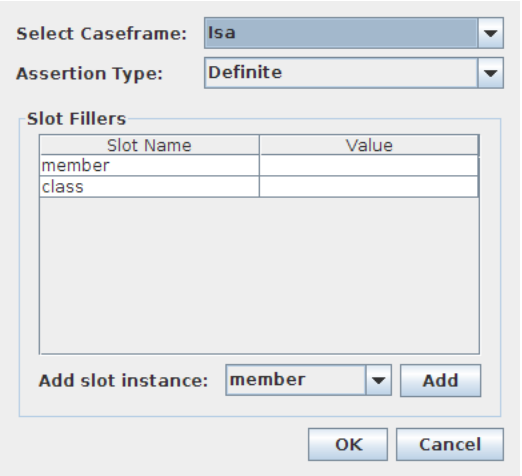

Figure 7: A frame-based interface for assertions to the KB.

the query interface will automatically be converted to variables $\{?w_1, .., ?w_n\}$. The system converts the text entered into the logical language and queries the KB. The resulting matches are displayed on the graph and all others are hidden.

In this example, the user is interested in only instances of the `Isa` caseframe in which the slot for `class` has filler `Person`. We will apply this frame-based query to the KB containing the knowledge about the call Sufian made to Ziyad discussed earlier. The system filters the graph, as seen in Figure 9, to contain only the graphical representations of `Isa` frames in which the slot for `class` has the filler `Person`.

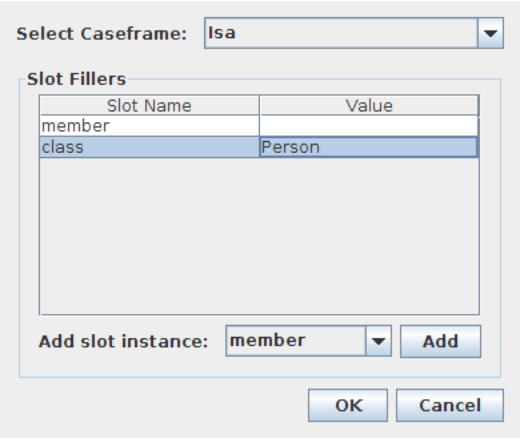

Figure 8: The QBE dialog box indicating a query for instances of the `Isa` caseframe with the class `Person`.

In addition, the user may themselves use variables in their query by prepending a question mark, "`?`", to the variable symbol, a feature useful for querying metaknowledge. For example, a user may view the graph with all instances of the `SourceOf` caseframe where the `Source` slot contains `Ahmed` and the `Theme` slot contains `(LocationOf Ziyad ?x)` to display all relations representing locations

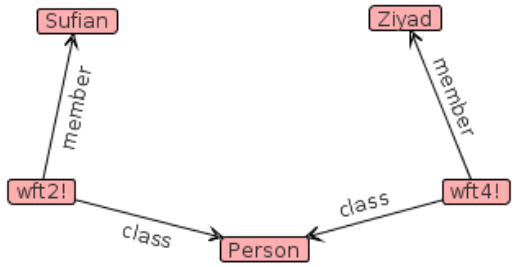

Figure 9: The result of finding instances of the `Isa` caseframe with the class `Person` in the calling example discussed earlier.

Ahmed has said Ziyad has been. At this time the user must enter `(LocationOf Ziyad ?x)` into the filler of the `Theme` slot manually, though it would be advantageous to have an interface allowing for embedded queries.

The graph the user is interacting with may contain relations not of interest, or perhaps not all the relations of interest. In this case the user may show or hide the relations attached to a node by right clicking a node and selecting the appropriate option. They may choose to show or hide the remainder of the relations with arrows either in to or out of the selected node. This shows or hides entire frames from the graph view. That is, it is never the case that a `wft` node with only some of the arrows out of it will be visible. One place this has particular use is once a QBE filter has been applied. Consider the example shown in Figure 9. We can choose to show all of the relations attached to the node `Sufian` if we are interested in knowledge associated with him, resulting in Figure 10. In this way the knowledge engineer can begin with a QBE-based query and only expand nodes of interest.

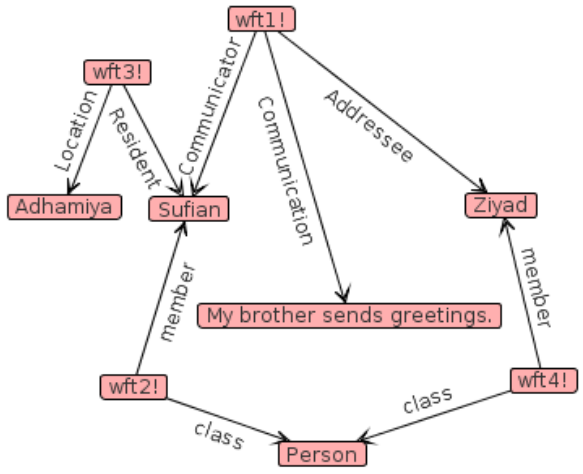

Figure 10: An expansion of the graph shown in Figure 9 wherein relations attached to the node `Sufian` are shown.

This example is of course quite trivial given the size of the graph being represented. In a much larger KB this becomes extremely useful as a method to visualize specific portions of a KB. The SNePS 3 user interface is currently being used

to visualize large propositional graphs for soft information fusion [Prentice *et al.*, 2010]. The graphs being visualized are many thousands of nodes.

In order to see the usefulness of QBE on a larger scale, we consider a knowledge base from this project containing 2,075 terms representing place names and various information about the places from the NGA GEOnet Names Server [National Geospatial-Intelligence Agency, 2011]. When these terms are added to the knowledge base and visualized, the graph shown in Figure 11 is produced. Large KBs are very difficult to work with conceptually as there is a significant burden on the knowledge engineer to understand the complex structure. The purpose of visualizing a KB though, should be to give the knowledge engineer an idea of the contents of the KB while reducing this burden as much as possible. As you can see, just visualizing the graphical version of the KB is unhelpful - it's impossible to pick out even a single relation.

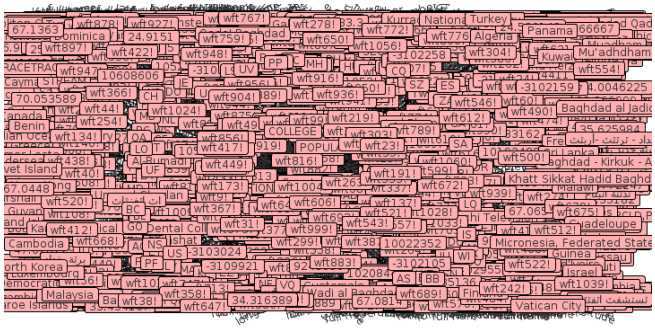

Figure 11: A large network of places and knowledge about them from the NGA GEOnet Names Server.

Very rarely is a knowledge engineer really interested in the entire KB at one time though, and constraining the display would be useful. Assume we are really only concerned with places that are countries, and we'd like to see the nodes for all of those. We can do this with a simple QBE-based query (by filling `Country` in to the `class` slot of the `Isa` caseframe in the user interface, and leaving the filler of the `member` slot empty). The graph is then restricted to showing only the relevant knowledge, as shown in Figure 12.

As you can see the resulting graph is much easier to understand; an underlying structure is visible and a node of interest is easier to pick out. The knowledge engineer could easily find and expand a country node if she were interested in the relations attached to it. You will notice though that the propositional graph contains many `wft` nodes which may not be helpful to the user of the graph. Under some situations these can be removed by redrawing, or collapsing the graph.

4 Collapsing the Graph

It is important to display the `wft` nodes in the graph either if the function has more than two arguments, as is the case for `wft1` in our calling example, or if the term is, itself, an argument of another function, as is the case for `wft5`. However, in the case of functional terms with just two slots that are not fillers of other slots, such as `wft2`, `wft3`, `wft4` and

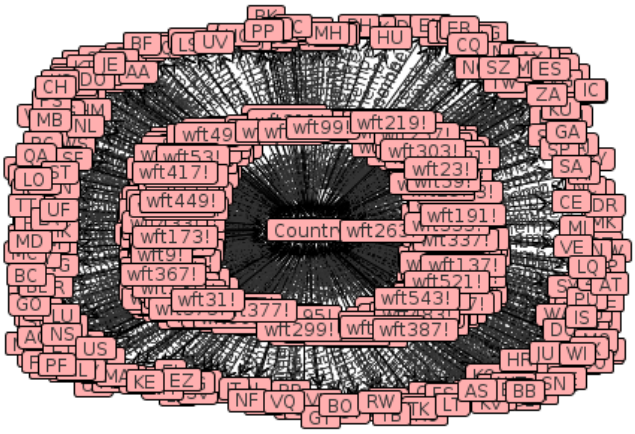

Figure 12: The KB shown in Figure 11 filtered by QBE to show only places which are countries.

`wft6`, the `wft` nodes just clutter the display. In this case, the `wft` node with its two outgoing arcs can be converted into a single arc labeled by the function symbol. This is referred to as "collapsing" the graph. The resulting collapsed graph is semantically equivalent to the uncollapsed version, it is only an alternate representation of the knowledge in the KB. Figure 13 is a collapsed version the graph shown in Figure 6.

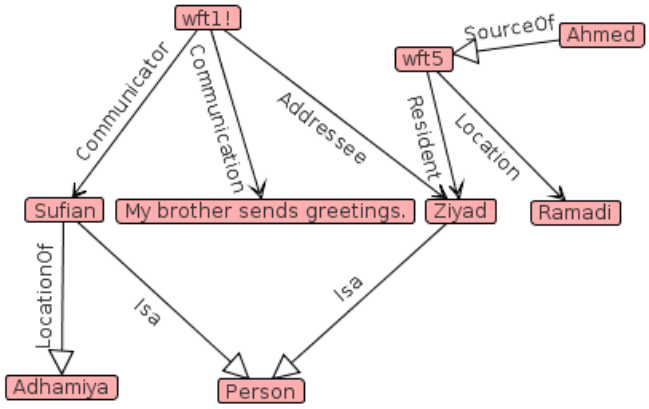

Figure 13: A collapsed version of the graph shown in Figure 6 in which binary relations are redrawn without their `wft` nodes.

Because the graph representation is backed by frames with ordered slots, the GUI can automatically determine which parts of the graph can be collapsed, the direction of the new arcs, and their labels. Any frame with only two slots can be collapsed. The arc is drawn from the filler of the slot containing the first argument of the function, to the filler of the slot containing the second argument of the function, and the label of the new arc is the function symbol associated with the frame. The arrow head used on the new arcs is a different style from that used on uncollapsed arcs as a visual reminder to the user that it is a collapsed arc.

Notice that, in contrast to some other systems, for exam-

ple those assumed by [Bodenreider, 2002], not all SNePS 3 `wft` nodes can be collapsed in this way. This is due to the increased expressibility of SNePS 3: the ability to represent n-ary relations for $n > 2$, and the ability to express propositions about propositions.

We can apply this to the NGA GEOnet KB which we applied a QBE filter to earlier (and shown in Figure 12) to increase the usability of that graph further. In collapsing this graph, the binary relation `Isa` becomes only two nodes instead of three as shown in Figure 14 and the graph is now easier to view and manipulate.

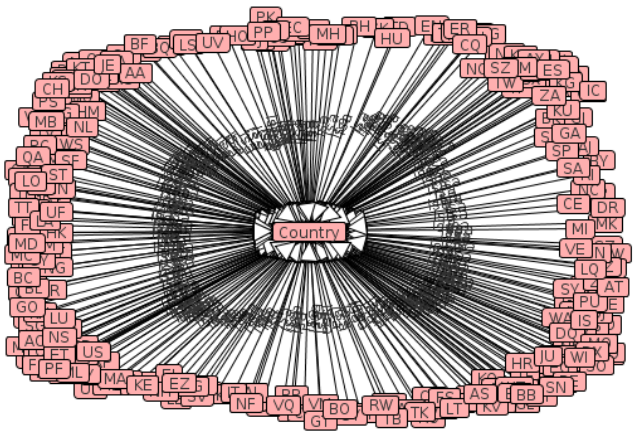

Figure 14: A collapsed version of the graph shown in Figure 12 to show places which are countries with binary relations collapsed.

As you can see, this increases the usability of the graph interface even beyond that of using QBE alone, allowing for very quick visualization of important data from complex hierarchies. These techniques together are more useful than automatic clustering of entire sections of a graph or using a zooming function. Automatic clustering can change the semantics of the graph and hide important knowledge and structural features. Scaling a graph on the other hand does not change the semantics, but it still hides relevant parts of the graph along with irrelevant parts, complicating the work of the knowledge engineer.

5 Evaluation

The user interface discussed here has been used for visualizing and interacting with graphs on the order of several thousand nodes. The techniques are known to scale to this level, and we are confident they will scale beyond it as work continues with increasingly larger knowledge bases. Any limitations in scale we believe will be due to the JUNG visualization system rather than the techniques discussed here. Unfortunately we were unable to find user interfaces for generalized KR systems suitable for comparison with the SNePS 3 user interface. Instead, we will discuss the concepts developed here in the context of ontology editors since they face many of the same challenges discussed here. We looked at several ontology editors [Stanford Center for Biomedical Informatics Research, 2011; Harris, 2011; Revelytix, Inc., 2011] and found that all of them provide the same basic functionality in mostly similar ways, so we will discuss this in the context of Protégé, which appears to be the most popular.

The increased expressivity of the SNePS 3 logic differentiates this work from ontology editing tools. Ontologies limit themselves to the use of binary relations to show the relationships between classes and individuals. In Protégé, to add a restriction between two classes, one needs only choose the initial class, the restricted property (such as `is-a` or `part-of`), and the restricted filter (the second argument in the binary relation). The interface is very simple for this - the user chooses from a tree of options. The frame-based interface for adding an instance of a caseframe we use allows for n-ary relations and adding instances of frames as the values for slots, requiring a more complex interface. For purely binary relations the work done in the Protégé interface works well, but it does not scale beyond that.

The methods of interaction with an ontology in Protégé are somewhat similar to what we have discussed here. The addition of classes and instances to an ontology occurs outside of any graph interface, and the graphs are used for visualization purposes only. The ontology editors surveyed also have basic reasoning abilities and the ability to present a query to the reasoner. Querying a graph by performing reasoning is something which we wish to incorporate into a future version of the SNePS 3 interface. Currently QBE only filters the graph itself (that is, the current contents of the KB). It would be useful for the system to, at least optionally, perform inference at query time. When doing this, we will want QBE to become more expressive as well - allowing, minimally, for conjunctive queries. As discussed earlier, we also would like a graphical interface for filling a slot with another frame.

The visualization requirements of a KR system and an ontology viewer are somewhat different. Ontologies always have a defined root and lend themselves to a hierarchical tree-like view, while this is not always the case in a propositional graph. This characteristic leads to ontology editors which show a definable number of levels of the graph in detail, along with some number of nodes along the path to the root (as in Protégé's OWLViz viewer). The graphs are also always shown with a view akin to what we call the "Collapsed View" in the SNePS 3 interface, as there are only binary relations to be shown. These features in combination allow ontologies which are very large (containing millions of concepts) to be viewed easily.

SNePS 3 and its user interface can be (and have been [Prentice and Shapiro, 2011]) used for interaction with ontology structures, providing for the most part a superset of the visualization and interaction features provided by ontology editors.

6 Conclusions

The nature of SNePS 3 KBs being conceived of as simultaneously graph, frame, and logic-based extends naturally into the interactions one may do graphically with such KBs. Users are able to add knowledge to the KB and visualize it in a

user friendly way. The combination of frame and graph based techniques in QBE and binary relation collapsing respectively has resulted in a highly usable method for interacting with KBs. It is now possible to view, modify, and understand a large KB without the need to sift through complex text outputs or graphs which are too large to be viewed easily. We believe the techniques given here are ideal for visualizing and interacting with the data associated with a KR system.

7 Acknowledgments

We would like to thank the SNePS Implementation Group for their consistent feedback on the user interface. This work has been supported in part by a Multidisciplinary University Research Initiative (MURI) grant (Number W911NF-09-1-0392) for "Unified Research on Network-based Hard/Soft Information Fusion", issued by the US Army Research Office (ARO) under the program management of Dr. John Lavery.

References

[Bodenreider, 2002] Olivier Bodenreider. Experiences in visualizing and navigating biomedical ontologies and knowledge bases. In *Proceedings of the ISMB'2002 SIG meeting: Bio-ontologies*, pages 29–32, 2002.

[Fillmore, 1968] Charles J. Fillmore. The case for case. In E. Bach and R. T. Harms, editors, *Universals in Linguistic Theory*, pages 1–88. Holt, Rinehart and Winston, New York, 1968.

[Fillmore, 1976] Charles J. Fillmore. Frame semantics and the nature of language. In *In Annals of the New York Academy of Sciences: Conference on the Origin and Development of Language and Speech*, volume 280, pages 20–32, 1976.

[Harris, 2011] Nomi Harris. OBO-Edit. http://oboedit.org/, 2011.

[ISO/IEC, 2007] ISO/IEC. *Information technology — Common Logic (CL): a framework for a family of logic-based languages, ISO/IEC 24707:2007(E)*. ISO/IEC, Switzerland, First edition, October 2007. available from `http://standards.iso/ittf/license.html`.

[National Geospatial-Intelligence Agency, 2011] National Geospatial-Intelligence Agency. NGA GEOnet names server (GNS). http://earth-info.nga.mil/gns/html/, 2011.

[Prentice and Shapiro, 2011] Michael Prentice and Stuart C. Shapiro. Using propositional graphs for soft information fusion. In *Proceedings of the 14th International Conference on Information Fusion*, 2011.

[Prentice *et al.*, 2010] Michael Prentice, Michael Kandefer, and Stuart C. Shapiro. Tractor: A framework for soft information fusion. In *Proceedings of the 13th International Conference on Information Fusion*, 2010.

[Revelytix, Inc., 2011] Revelytix, Inc. Knoodl. http://www.knoodl.com/, 2011.

[Ruppenhofer *et al.*, 2006] J. Ruppenhofer, M. Ellsworth, M. R. L. Petruck, C. R. Johnson, and J. Scheczyk. FrameNet II: Extended theory and practice. Unpublished manuscript, 2006.

[Shapiro *et al.*, 2007] Stuart C. Shapiro, William J. Rapaport, Michael Kandefer, Frances L. Johnson, and Albert Goldfain. Metacognition in SNePS. *AI Magazine*, 28:17–31, Spring 2007.

[Shapiro, 1993] Stuart C. Shapiro. Belief spaces as sets of propositions. *Journal of Experimental and Theoretical Artificial Intelligence (JETAI)*, 5(2&3):225–235, April–September 1993.

[Shapiro, 2000] Stuart C. Shapiro. An introduction to SNePS 3. In Bernhard Ganter and Guy W. Mineau, editors, *Conceptual Structures: Logical, Linguistic, and Computational Issues, Lecture Notes in Artificial Intelligence 1867*, pages 510–524. Springer-Verlag, Berlin, 2000.

[Shapiro, 2004] Stuart C. Shapiro. A logic of arbitrary and indefinite objects. In D. Dubois, C. Welty, and M. Williams, editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004)*, pages 565–575, Menlo Park, CA, 2004. AAAI Press.

[Shapiro, 2010] Stuart C. Shapiro. Set-oriented logical connectives: Syntax and semantics. In Fangzhen Lin, Ulrike Sattler, and Miroslaw Truszczynski, editors, *Proceedings of the Twelfth International Conference on the Principles of Knowledge Representation and Reasoning (KR2010)*, pages 593–595, Menlo Park, CA, 2010. AAAI Press.

[Stanford Center for Biomedical Informatics Research, 2011] Stanford Center for Biomedical Informatics Research. The Protégé ontology editor and knowledge acquisition system. http://protege.stanford.edu/, 2011.

[The JUNG Framework Development Team, 2010] The JUNG Framework Development Team. JUNG - Java universal network/graph framework. http://jung.sourceforge.net/, 2010.

[Zloof, 1975] Mosh M. Zloof. Query by example. *AFIPS*, 44, 1975.