Sample Representations of Cell Structure and Processes and Supporting Relation Vocabulary

Vinay K. Chaudhri and Bill Jarrold
Artificial Intelligence Center
SRI International, Menlo Park, CA, 94025, USA

Introduction

We describe example representations of cell structure and processes designed to answer some of the questions that may be found at the back of a college-level Biology textbook. While we are using the task of answering questions as a context for the development of the representation proposed here, the resulting knowledge is useful in many other ways. For example, the vocabulary in the knowledge base can serve as the reusable framework for interoperability across various Biological systems. The knowledge base, once fully developed, could be potentially used as a platform for scientific discovery.

The representations shown here have been constructed by Biologists using a knowledge acquisition and question answering system – Automated User-Centered Reasoning and Acquisition (AURA) [1,2]. AURA embeds an electronic version of a college-level textbook as a reference source against which the Biologists perform knowledge encoding [3]. To construct the domain-specific representations, the users begin from a library of generic components available in AURA, called the Component Library (CLib), and a controlled vocabulary of relations, called the Slot Dictionary [4]. CLib is generic in the sense that the same generic classes and relations are useful in modeling different biological processes. For example, both the biological concept of active transport across a membrane and mRNA leaving the nucleus involve the concept of an object changing its location (i.e., moving) and can be created as elaborations of the reusable generic concept from CLib called *Move*.

Eukaryotic Cell

In Figure 1, we show a screen shot of the representation of a Eukaryotic cell from the AURA system. The graph represents an example instance of a Eukaryotic cell, its parts, and how they are related to each other. The node labeled *Eukaryotic-cell* is the root of this graph. It has a rectangular shape denoting that it is an *Entity*, and white background denoting that it is the root of the graph. An arrow from *Eukaryotic-cell* to *Mitochondria* has the label *has-part.* Labels also have a white background, but no border. This edge means that every instance of a *Eukaryotic-cell* has as its part an instance of *Mitochondrion.* The *has-part* edge also has two annotations, *10+ Ribosomes* and *2+ Plasma-Membrane,* which denote that every *Eukaryotic-cell* has ten or more instances of *Ribosomes* and two or more *Plasma-Membrane*s. The other parts of the *Eukaryotic-cell* (e.g., *Nucleus*, *Cytosol*, *Chromosome*, and *Ribosome*) can be interpreted analogously.

In the graph of Figure 1, we also show the spatial arrangement between parts. For example, an edge labeled *is-inside* between *Cytosol* and *Cytoplasm* denotes that *Cytosol*

is inside the *Cytoplasm.* The graph asserts the *is-inside* relation for a specific instances of *Cytosol* and *Cytoplasm* that are both parts of the same instance of *Eukaryotic-cell.*

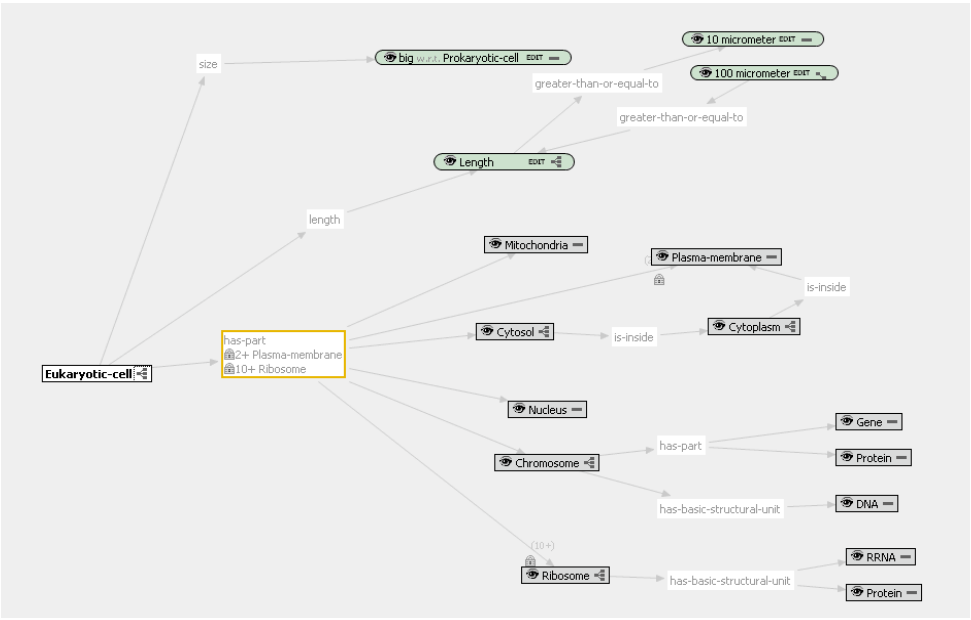

Figure 1: An AURA representation of a Eukaryotic cell

We also encode the *size* and *length* of a *Eukaryotic-cell.* The nodes with oval ends contain values. The CLib defines an ontology of values that includes *scalar* and *cardinal* values. The value of *size* is a scalar value *big Prokaryotic-Cell,* denoting that the size of the *Eukaryotic-cell* is big relative to a *Prokaryotic-cell.* The *length* of a *Eukaryotic-cell* is stated to be between the cardinal values of *10 micrometer* and *100 micrometer* using the relation *greater-than-or-equal-to*.

This representation is not complete and can be extended by a Biologist through the AURA user interface. The relations *has-part, is-inside, size, length, greater-than-or-equal-to* are based on the Slot Dictionary of the CLib, We will illustrate these and several other relations in subsequent concepts.

Various icons on the nodes in the graph (e.g., an eye, a tree symbol, a short rectangle) are user interface controls for manipulating the graph. Using these, nodes in the graph could be expanded, or contracted, or hidden. Some expansion operations are not purely visual and may lead to inference in the knowledge base.

Prokaryotic Cell

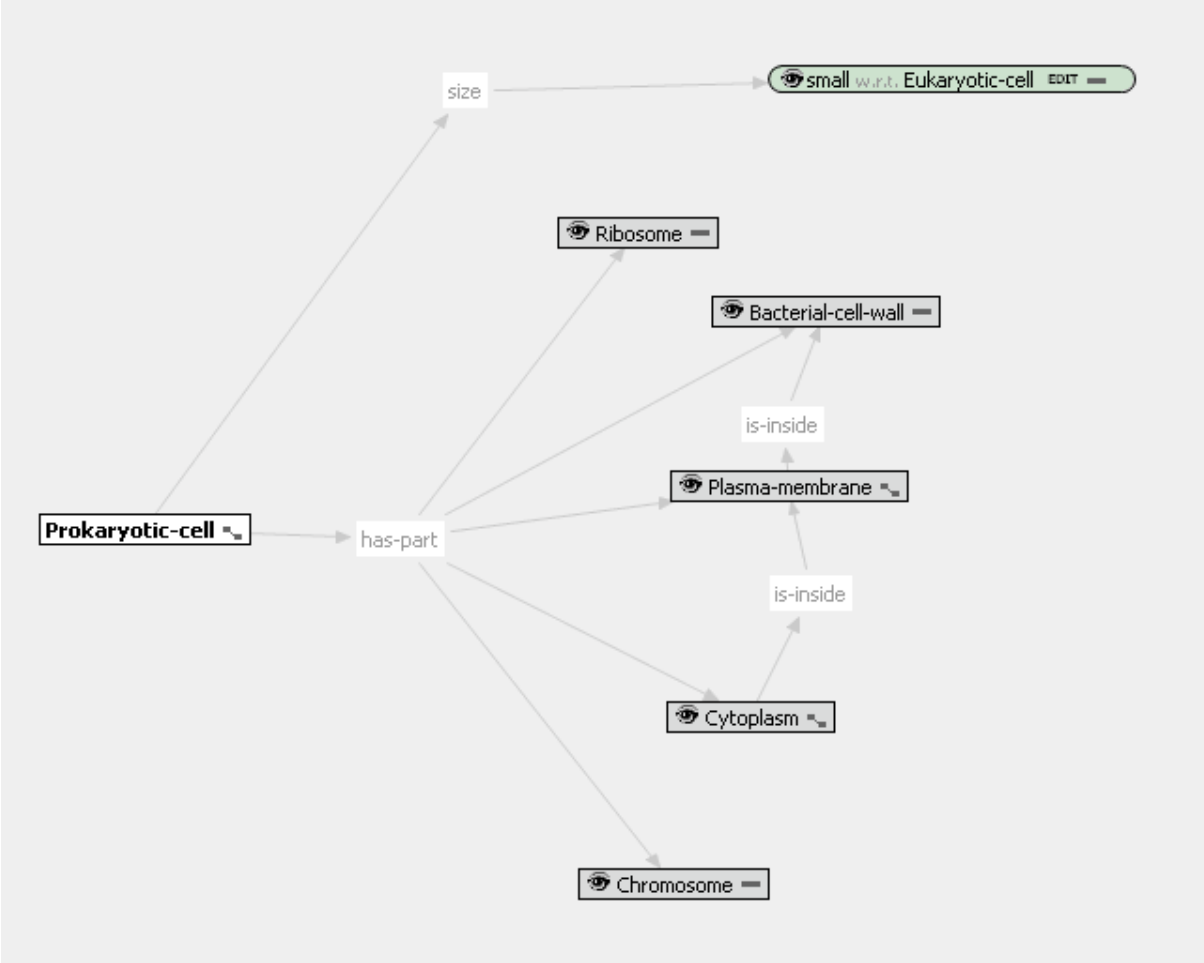

Figure 2: An AURA representation of a Prokaryotic cell

In Figure 2, we show a representation of a Prokaryotic cell that indicates its parts and their relative spatial arrangement. The size is stated using a scalar value indicating that a Prokaryotic cell is "small" relative to the scale of sizes of Eukaryotic cells. The representation of the Prokaryotic cell uses some of the very same relations (e.g., *has-part, is-inside*) that were used in representing a Eukaryotic cell.

Mitosis

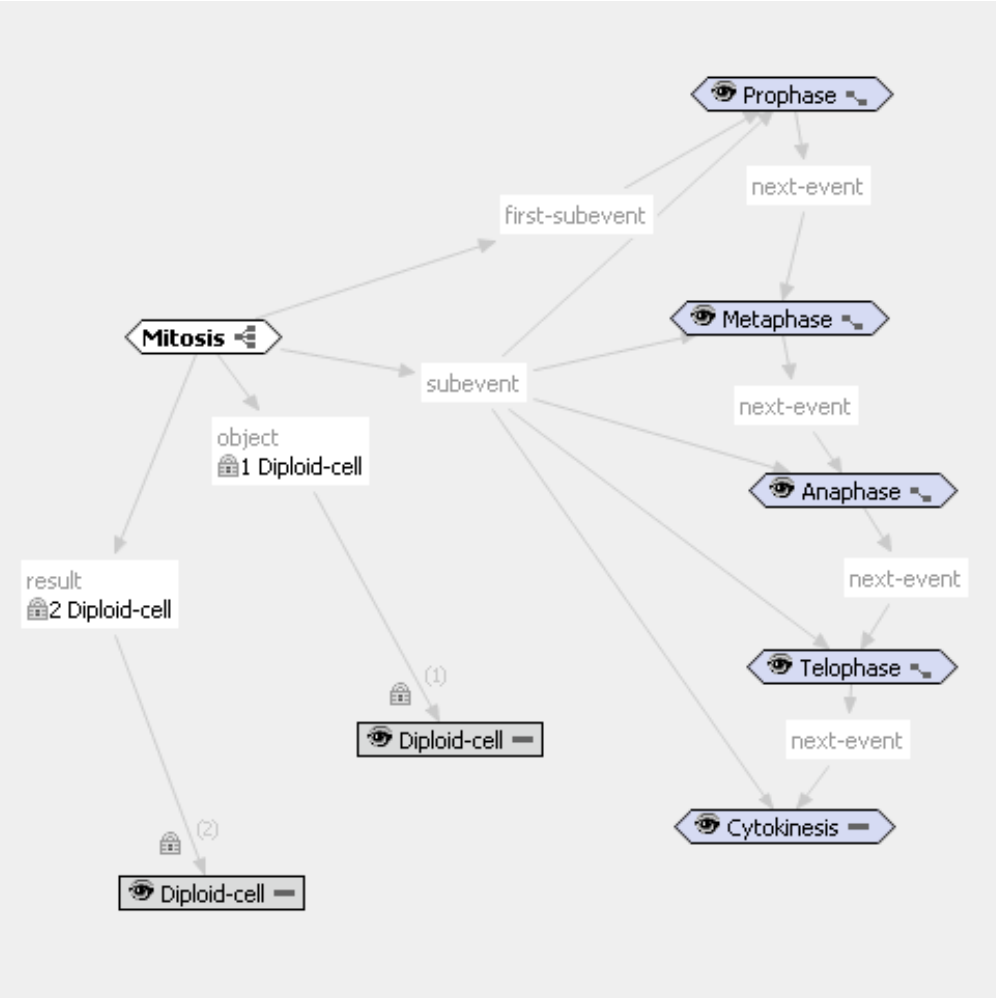

Figure 3: An overview of the process of mitosis

In Figure 3, we show the representation of the process of mitosis that is central in the context of cell processes. The nodes that represent events are shown in an angular shape.

Mitosis has fives steps indicated using the *subevent* relationship, and *Prophase* is the first event indicated using the *first-subevent* relationship. The ordering among the events is indicated using the *next-event* relationship. The process of *Mitosis* operates on a *Diploid-cell* indicated using an *object* relation, and produces another *Diploid-cell.* The *object* and *result* relations have annotations on them indicating that the process has exactly one *Diploid-cell* as its object and exactly two *Diploid-cell*s as their result.

Citokinesis

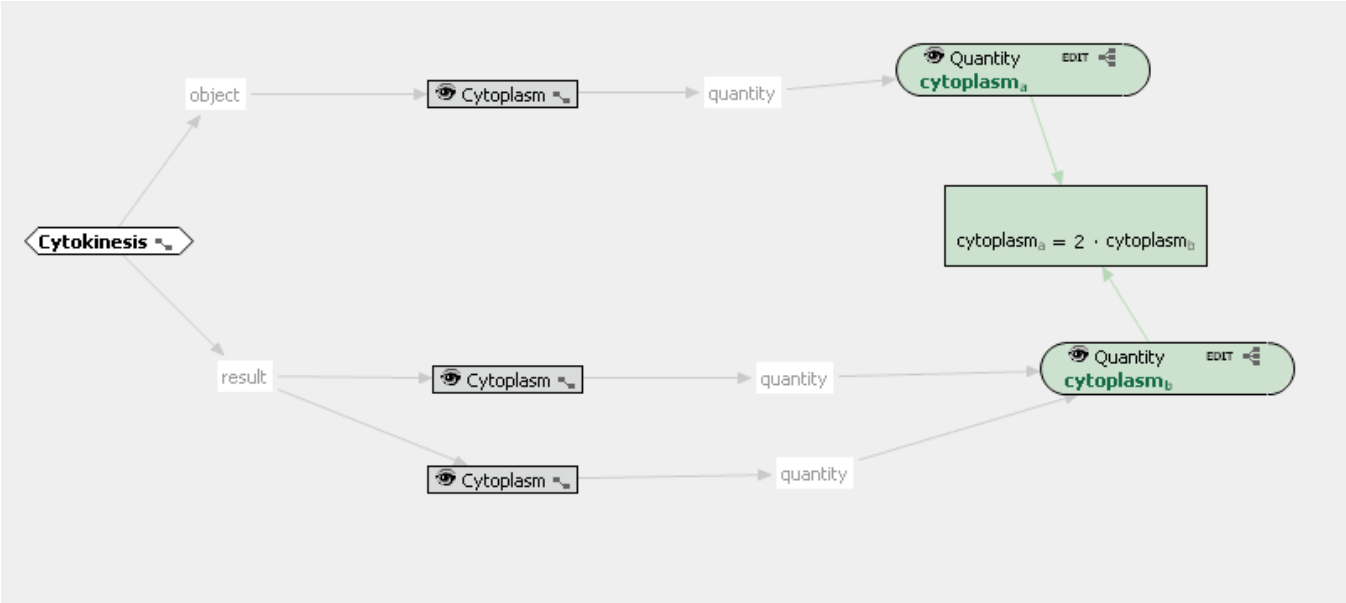

Figure 4: A representation of cytokinesis

In Figure 4, we show an illustration of the cytokinesis step of mitosis. During cytokinesis, the cytoplasm divides into two equal parts. The concept of *Cytokinesis* is a subclass of the class *Divide* in the CLib. *Divide* is an example of a domain-independent concept that is predefined in the library. Figure 4 also illustrates the use of a mathematical equation to indicate that the cytoplasm divides into two equal parts.

Example Questions and Their Answers

Here is a sampling of questions that can be answered using the representations we have considered so far:

1. What are the parts of a Eukaryotic cell?
2. What are the similarities and differences between a Eukaryotic and a Prokaryotic cell?
3. How many ribosomes does a cell have?
4. What is the size of a Eukaryotic cell?
5. What are the steps of mitosis?
6. In mitosis, which step follows anaphase?
7. How many cells result from mitosis?

Possible Applications to Bioinformatics

The focus in our project so far has been on technology development. Even though we have used Biology teachers or graduate students to author some of the knowledge shown here, we have not yet made a clear connection to work in bioinformatics. There are several possible avenues that we could explore for this purpose.

1. We could consider the relations in the Slot Dictionary in the CLib as possible primitives for modeling biological data. The Slot dictionary is available at http://www.cs.utexas.edu/users/mfkb/RKF/tree/components/specs/slotdictionary.html. An OWL representation of the Slot Dictionary as well as the CLib is available.
2. We could import the Gene Ontology (http://www.geneontology.org) and provide it as a built-in vocabulary for authoring knowledge into AURA.

While there could be many applications of the AURA technology to bioinformatics, the current technical note focused primarily on the Slot Dictionary, as this addresses a currently perceived need of the community.

Comparison to OBO Relations

In this section, we compare the relations in the slot dictionary to the relations that are under consideration by OBO Foundry. These relations are described at: http://www.obofoundry.org/ro/. Our goal in doing this comparison is to understand the similarities and differences between the vocabulary used in AURA versus the one being considered for OBO.

isa:

The OBO defines this relation as: For continuants: C is_a C' if and only if: given any c that instantiates C at a time t, c instantiates C' at t. For processes: P is_a P' if and only if: that given any p that instantiates P, then p instantiates P'.

In AURA the *subclasses* relation has a meaning closest to OBO's *isa.* Whereas OBO's isa meaning depends on whether relates two continuants versus two processes, the relation *subclasses* in AURA has the same meaning for Entities versus Processes (these two CLib classes seem to be closest in meaning to continuants versus processes, respectively).

part_of

The OBO defines this relation as: For continuants: C part_of C' if and only if: given any c that instantiates C at a time t, there is some c' such that c' instantiates C' at time t, and c *part_of* c' at t. For processes: P part_of P' if and only if: given any p that instantiates P at a time t, there is some p' such that p' instantiates P' at time t, and p *part_of* p' at t. (Here *part_of* is the instance-level part-relation.

In AURA, the relation *has-part* is most relevant because it is the most generic partonomic relationship in CLib. However, because it applies at the instance level, it more closely matches the meaning of OBO's *part_of*. By contrast, *has-part* is not reflexive.

There are several more specific partonomic relations in CLib such as *has-structural-part* and *has-functional-part*. *has-region* indicates spatial partonomy. The *material* relation specifies partonomy between substances. The *has-part* relation only applies between instances and there is no class-to-class analogue of it.

integral_part_of

The OBO definition is: C integral_part_of C' if and only if: C part_of C' AND C' has_part C.

We had difficulty understanding this definition because we did not find the definition of has_part anywhere else.

proper_part_of

As for part_of, with the additional constraint that subject and object are distinct.

The *has-part* relation in CLib, that was already mentioned above, is the most similar to the instance level version of proper_part_of, i.e. *proper-part-of*.

located_in

C located_in C' if and only if: given any c that instantiates C at a time t, there is some c' such that: c' instantiates C' at time t and c *located_in* c'. (Here *located_in* is the instance-level location relation.)

The *location* relation in CLib is the closest semantic match to this relation. It relates an *Entity* to a *Place.*

contained_in

C contained_in C' if and only if: given any instance c that instantiates C at a time t, there is some c' such that: c' instantiates C' at time t and c located_in c' at t, and it is not the case that c *overlaps* c' at t. (c' is a conduit or cavity.)

The relation *content-of* in CLib is the nearest match to this relation as it relates an instance of *Entity* which contains another instance of *Entity*. One difference with contained_in is that it does not require a conduit or cavity.

adjacent_to

C adjacent to C' if and only if: given any instance c that instantiates C at a time t, there is some c' such that: c' instantiates C' at time t and c and c' are in spatial proximity

There are several relations in CLib that are similar to adjacent_to, each related to the notion of adjacency. These include *abuts, is-near, is-along or is-beside.*

transformation_of

Relation between two classes, in which instances retain their identity yet change their classification by virtue of some kind of transformation. Formally: C transformation_of C' if and only if given any c and any t, if c instantiates C at time t, then for some t', c instantiates C' at t' and t' earlier t, and there is no t2 such that c instantiates C at t2 and c instantiates C' at t2.

There is no reasonably close match to this relation in CLib. For processes that involve transformation, CLib provides relations such as *object, result, raw-material* to specify the inputs and outputs of the process.

derives_from

Derivation on the instance level (*derives_from*) holds between distinct material continuants when one succeeds the other across a temporal divide in such a way that at least a biologically significant portion of the matter of the earlier continuant is inherited by the later. We say that one class C derives_from class C' if instances of C are connected to instances of C' via some chain of instance-level derivation relations. Example: osteocyte derives_from osteoblast. Formally: C derives_immediately_from C' if and only if: given any c and any t, if c instantiates C at time t, then there is some c' and some t', such that c' instantiates C' at t' and t' earlier-than t and c *derives_from* c'. C derives_from C' if and only if: there is an chain of immediate derivation relations connecting C to C'.

There is no reasonably close match to this relation in CLib. For processes that involve transformation, CLib provides relations such as *object, result, raw-material* to specify the inputs and outputs of the process.

preceded_by

P preceded_by P' if and only if: given any process p that instantiates P at a time t, there is some process p' such that p' instantiates P' at time t', and t' is earlier.

The relation *next-event-of* in CLib is a close match to preceded_by. One difference is that the arguments are required to be instances of Event. Also there is an entailment (as per the English comment defining the term) that one Event "immediately" follows the other.

In addition, there are a number of termporal relationships that specify distinctions relevant to one time period preceeding another (such as does one object start after the ending of another or does is start after the start of another object). However, these relations apply only to instances of Time-Interval. To relate an Event or an Entity in time using one of these relationships, one would have to associate that Event or Entity with an instance of Time-Interval using a different relationship.

has_participant

P has_participant C if and only if: given any process p that instantiates P there is some continuant c, and some time t, such that: c instantiates C at t and c participates in p at t

There is no close match to this relation in AURA. Although there are relations such as *agent, object, instrument* and many more that relate participants of a process to the process, there is none that subsumes them all. In other words, there is none that captures the generic notion of participation in an instance of the CLib class *Event.*

has_agent

As for has_participant, but with the additional condition that the component instance is causally active in the relevant process

The closest match to this relationship in CLib is *agent*. An *agent* of an *Event* in CLib is the entity that initiates, performs or causes an event which is similar (but not identical to) the OBO definition for has_agent which involves the component instance being causally active.

instance_of

A relation between an instance and a class. For components: a primitive relation between a component instance and a class which it instantiates at a specific time. For processes: a primitive relation, between a process instance and a class which it instantiates, holding independently of time

The *instance-of* relation in CLib is a very close match to instance_of in OBO.

Having sketched out a mapping between the OBO relations and AURA relations, the following general remarks are in order.

- CLIB does not have an explicit distinction between class level and instance level relations. Most relations such as *is-part-of* do not have a class-level counterpart.
- Fluents and situation mechanism. The projection of a property through time is not a function of whether an object is a continuant or a process. Rather, projection is determined by a meta-relation known as fluent-status.

Acknowledgments

The AURA system has been developed under funding from Vulcan Inc.

References

[1] **Enabling Experts to Build Knowledge Bases from Science Textbooks** by Chaudhri, V., John, B.E., Mishra, S., Pacheco, J., Porter, B., and Spaulding, A. In *Proceedings of The Fourth International Conference on Knowledge Capture (K-CAP).* 2007.

[2] **Capturing and Answering Questions Posed to a Knowledge-Based System** by Clark, P., Chaw, S.-Y., Barker, K., Chaudhri, V., Harrison, P., Fan, J., John, B., Porter, B., Spaulding, A., Thompson, J., and Yeh, P.Z. In *Proceedings of The Fourth International Conference on Knowledge Capture (K-CAP),* October 2007.

[3] **Biology** by Campbell, N. A., and Reece, J. Sixth Edition (2001), Benjamin Cummings.

[4] **A Library of Generic Concepts for Composing Knowledge Bases** by Barker, K., Porter, B., and Clark, P. In *First International Conference on Knowledge Capture.* 2001.