Mylopoulos: First of all, a few things about the basic terms I will use. A knowledge representation scheme is a formalism for representing knowledge. The term knowledge representation will be used to refer to knowledge representation schemes, knowledge representation languages, and so on. The corresponding term in database modelling is data model or semantic data model. We're talking about natural ways of capturing some sort of reality using a specification language. The things produced when using a knowledge representation scheme are referred to as knowledge bases which correspond in database modelling and programming languages with conceptual schema and program specification respectively.

What is a knowledge base? What is modelled by a knowledge base? I am happy to report that there is no one answer from people working in knowledge representation. Some people will argue that a knowledge base is a model of some world. Others will say that it is linguistic information about some world, and so on. Depending on the viewpoint, there are differences in knowledge representation schemes.

Secondly, what is included in the knowledge base? Traditionally, knowledge bases include both abstract knowledge (like information that you have about persons in general) and concrete knowledge (like the address and telephone number of a particular person, John Smith). In database modelling there is a distinction between the kind of information that is stored in the conceptual schema and that stored in the database. A knowledge base includes both. This has a couple of consequences. First, the knowledge representation scheme would be such that you can talk about both the abstract and concrete knowledge in the knowledge base. You can ask queries of and perform reasoning on both types of knowledge. Second, in the same way that new objects, e.g., new persons, are added to the knowledge base during its lifetime, new concepts can be added, e.g., the concept vehicle.

There are about three basic types of knowledge representation schemes: logical, network or semantic networks, and procedural. Logical representation schemes contain true assertions about what is being modelled. In the semantic network scheme, you find the entities being modelled and the relationships between them, then you create corresponding objects and relations in the knowledge base. For procedural representations, we're interested in state transformations. The model has different states and allowable transformations between them; you want to model how these transformations can come about.

Logical Representation Schemes

Let's consider logical knowledge representation schemes in more detail. A knowledge base is a collection of assertions. The basic modelling notions are those provided by logic, i.e., constants, variables, functions or predicates, logical connectives, quantifiers, etc. A basic feature of logical representations is a well defined semantics that comes with first order logics. A second feature is the availability of proof theory. Using well defined procedures, you can deduce new facts which are not explicit in your knowledge base. This means that logical representations are particularly well suited in situations where reasoning is of primary importance. The third feature (and perhaps, disadvantage) is that there is no natural way of representing heuristic information, e.g., information that can tell you that to establish some goal you should use certain facts, start with certain facts, not try other facts at all, etc. An example of a logical representation scheme is PROLOG, a programming language in which a user program is a collection of assertions. In order to get around the drawback of not being able to represent heuristic information, PROLOG allows one to provide a dual interpretation to any one assertion. One interpretation is proof theoretic, the second is procedural. A procedural interpretation tells you to establish some subgoals in order to establish a particular goal. Subgoals for a particular goal are determined by the assertions in your knowledge base.

Heuristic or Control Information

Hayes: John, I object to your saying that it's difficult to represent heuristic information in the logical approach. There is indeed a profound difficulty in representing heuristic information, period. Nobody knows how to do it properly. But there's nothing special to logic about that difficulty. That is not a question of representation.

Balzer: You take a homogeneous structure for knowledge. However, most work done in representing control heuristics is done by procedural embedding. Therefore to include heuristics, you wind up with a mixed knowledge base rather than a homogeneous structure.

Hayes: To me procedural embedding is a counterexample. If someone comes up with some good heuristic knowledge, I bet you can represent it in predicate calculus as well as anything else.

Mylopoulos: I'm not sure one can. The question is, could it be represented more naturally, whatever that means, by using a procedural representation scheme. The claim has been made that

procedural representation schemes represent heuristic information more directly.

Hayes: That's a very controversial statement.

McLeod: John, could you please define heuristic information or heuristic knowledge?

Hayes: I can give you a definition. I'm not going to define the word heuristic, that's in the dictionary. In this context you have knowledge representation and inference rules; however, in a given situation it is not determined what to do next. If you look at it as defining a computation, what you have is a non-deterministic computation. Heuristics tell you what to do next.

McLeod: What would be an example of a heuristic?

Mylopoulos: A situation where you must search and you want to provide some information as to what strategies to try first or at all. That information would be heuristic.

Goldstein: That's not quite right. Polya's definition of heuristic is a tactic for solving a problem that is not guaranteed to succeed. If you have a set of tactics for analyzing a problem, that were, in fact, guaranteed to succeed, that would not be a "heuristic."

Shaw: They are related. Heuristic search gives you an order intended to make a search proceed faster, but is not guaranteed to succeed.

Hayes: Here is the difference. Looked at in the context of heuristic search, one particular search strategy or heuristic is to search the search space depth first. However, you can regard the resulting complicated thing as an algorithm, a backtracking algorithm. So whether you call it a heuristic or an algorithm, is a delicate point. Whether you regard your total system as a programming language of sorts or as heuristic search plus a control is purely a matter of terminology.

Rich: The important thing about a heuristic is it could be wrong.

Sibley: So heuristic knowledge has un-knowledge as well as knowledge?

Wedekind: If you know it, it can't be a heuristic.

Goldstein: I would like to distinguish the logical and procedural approaches. Consider the well-formed formula A→B which in traditional predicate calculus does not carry the additional meaning of whether you want to assert A and prove B, or assert not(B) and prove not(A). The proof procedure is independent of the well-formed formula. In Hewitt's thesis, around 1970, he waved the flag to say there's knowledge that says I want to go from A to B, rather than from not(B) to not(A). I want a language that allows me to say that. That was the historical origin of the "procedural" school of knowledge, in contradistinction to the logical approach.

Hayes: Let's use the term "control knowledge" since heuristic leads to all these fights.

Network Representation Schemes

Mylopoulos: Now, let's consider network representation schemes, or semantic network representation schemes. Here, a knowledge base is a collection of objects and, usually, binary relations between them. You will be hard pressed to find a collection of notions that underlies the various semantic network schemes proposed. Perhaps the best collection of primitive notions is given by Brachman. He proposed five levels:

1) directed labelled graphs plus an interpreter which allows graph manipulations,
2) a representation for logical assertions,
3) the epistemological level,
4) the semantic level, and
5) the linguistic level.

At the first level a semantic network scheme is viewed as a directed labelled graph and an interpreter is provided that allows you to manipulate the graph.

Katz: Nets themselves don't offer interpreters, they are an additional mechanism just as proof procedures and reasoning with predicates are additional to logical schemas. If you're going to allow mechanisms which operate on nets to be a property of the nets, then mechanisms that operate on logical schemas in predicate calculus notation have to be considered part of logical schemas.

Reiter: The interpreter and the network structure are separate. Graph manipulation is clearly part of the interpreter. Indeed, some kind of heuristic knowledge is embedded in the interpreter.

Mylopoulos: Yes. At this level a semantic network is just a directed labelled graph. Any additional properties or knowledge to be modelled in the knowledge base is in the interpreter. For example, the interpreter will have to know that a particular arc labelled "LT" has the usual properties of the less-than relationship.

Logical Assertion Representation

There is a second level of semantic network representation schemes, which views a semantic network as a representation for logical assertions. According to this view, a semantic network is a useful data structure for representing knowledge using notions from the logical representation schemes.

Hardgrave: Could you give a rough definition of a semantic network and how it differs from other kinds of network?

Mylopoulos: There is no one set of features that one can say underlies all semantic network representation schemes. Instead, we talk about semantic network representations which simply offer us a service for manipulating a directed labelled graph. That was the first level.

Hardgrave: Well if I draw a few nodes with a line, I can call them a semantic network? (Mylopoulos) Yes. (Hayes) But you have to say that the nodes are supposed to be objects and the edges are supposed to be relations? (Mylopoulos) Yes. (Hendrix) And that they support some kind of mapping with the real world.

Mylopoulos: The place where you put the relation with the real world is in the interpreter. In other words, the procedures for manipulating and searching the network worry about the various properties of the relationships and identities that are being modelled by relations and nodes of your semantic network.

At the logical level, semantic networks are simply convenient data structures for representing logical assertions. They offer indexing schemes of various sorts for getting all the assertions, for example, that talk about a particular person, or talk about persons in general.

Epistomological Level

There is a third level which Brachman calls epistemological level provides facilities for organizing concepts in terms of epistomological primitives: that is basic units of knowledge. KLONE is an example of this approach.

Semantic Level

At this level the constructs represent the meaning of sentences in a language. A good example of the work in this category is that of Roger Shank. He uses a representation that includes primitive actions and various rules for putting them together in the knowledge base to express more complex meaning.

Linguistic Level

There is a last level, the linguistic level, where sentences are organized into a knowledge base. The objects in the network represent words, or sentence fragments.

The Role of the Interpreter

I cannot really say that semantic networks have a really well defined semantics. To tell what objects and relations in the knowledge base mean, you have to look at the interpreter. If you are dealing with a fairly large number of object types and relation types, then you are in trouble because you have to keep expanding your interpreter, and it's very hard to see exactly what you are storing in it. It's supposed to be a big program of some sort. To get around this problem of large and unwieldy interpreter procedures or logical assertions can be associated with different objects and relations. Consequently, the interpreter is distributed throughout the knowledge base and it's easier to see exactly what is going on.

Balzer: John, I'm not happy with the different levels of semantic nets. The important thing for me is just this programmable interpreter. All levels are basically relational connections among objects. The designers of these different semantic networks chose to accentuate certain relations as being more important and best able to draw conclusions from them the way they program up the interpreter. Different people use semantic networks differently but basically the same thing is going on. You just program up the interpreter to do the right thing.

Mark: I disagree with that point slightly. What's being talked about here is ways to constrain the building of networks and this has some effect on the interpreter. In OWL, there is a definite intention to structure things in a "linguistic" way. Brachman's major point in KLONE is to have a sound epistemology in the way we structure the net, in order to simplify the interpreter.

Reiter: There's another important distinction: network hackers are generally concerned with representing prototypical objects and default reasoning of some kind. That's about the only feature of networks that I can discern that distinguishes them from the logical representation. I have yet to see a semantic network representation that couldn't be fairly directly mapped into some sort of predicate logic, and simply view the interpreter as a set of control (heuristic) restrictions. It seems to me the real distinction has to do with this issue of representing default assumptions.

On the Equivalence of Representations

Weber: Can I conclude from what you just said that you can prove there is some kind of equivalence between these two representations?

Hendrix: Everybody has their own network representation. There are all kinds of hassles about which ones are equivalent to which. I concluded that partitioned semantic networks are equivalent to predicate calculus, and vice versa, but, that doesn't mean anything.

Balzer: The state of the art in representations of the network is very, very similar to the state of the art in data abstraction in programming languages. Everybody's got their own thing, with a few little twists that cause them to be different.

Hardgrave: Essentially all of them come back to being equivalent to a Turing machine, somehow we've got to find a metric, something above Turing machines, to give us some classification.

Goldstein: I'll give you a metric. Again, I think you have to look at the history of it. What's going on today is this grey area. In the early history, the logical approach used notions of proof procedure from logic. The idea behind semantic nets in Quillan's work is that a particular kind of reasoning procedure that was not in the predicate calculus modes of that time came along with it. He spoke about it as a spreading activation. The kind of question being dealt with was "what President had something to do with cherry trees?" That is a difficult thing to reason out in the predicate calculus. The notion of a semantic network, in its early formulation, was a network with George Washington, cherry tree, President

and links between them. You do this spreading activation and look to where it hits; that would give your answer "Washington is that president." Now there are reasoning schemes associated with networks. In the years that followed, the networks have been expanded so that most of the things you could do with predicate calculus are doable, e.g., quantifiers. The early form of semantic net was very pure, very restricted, and had this activation algorithm.

Hardgrave: So what's the bottom line. Is it equivalent to a Turing machine?

Goldstein: It was an attempt to model human memory. Just as in computer science analysis of Turing machines has turned into complexity, where we talk about time and space constraints. That's the whole game within AI representations, namely, can you characterize the space or time needed to reach a particular node.

Hayes: There's a much more refined notion in the difference between Turing machine equivalences and equivalences for representation schemes. It has to do with the fact that a representation scheme has some kind of semantical meaning.

Hardgrave: But what are entities and relations? (Hayes) It doesn't matter. (Hardgrave) Computers simply manipulate symbols. When you try to tie them back to the real world you're in trouble. (Hayes) But the symbols have meaning. (Hardgrave) But the computer will never understand that meaning, no matter what you do to it.

Hayes: We understand it, and we can decide whether two schemes are equivalent by asking whether you can do a translation that preserves the meaning.

Hardgrave: If two people perceive the same symbols differently, that's really outside the realm of computers.

Hayes: It's not outside the realm of representation theory. We are here discussing systems for representing knowledge, not just systems for hacking. Everything's equivalent within memory.

Other Features of Semantic Networks

Mylopoulos: The important feature of semantic network representation schemes is the organizational principle that it provides. One can organize all the objects and relationships that constitute the knowledge base. Some of those principles are is_a (generalization), part_of (aggregation) and instance_of (classification), which are also used in database management.

I would like to mention the notion of context which is useful in trial and error situations. You follow the particular alternative until you reach a dead end or the goal. If you reach a dead end, you backtrack and try some other alternative. Some notion of context is useful here. Various semantic network schemes provide means for representing contexts.

The basic feature that semantic network representation schemes are good for is to access information because of the associationist viewpoint. Using an object and its relations with other objects, you can access those other objects from it. Examples of knowledge representation schemes are KLONE at BBN, OWL, which Bill Mark and his group has developed, and PSN which has been worked on at the University of Toronto.

Procedural Representations

Let's turn now to procedural representation schemes in which the knowledge base is basically a collection of active agents; let's just refer to them as procedures for the time being. The primitive notions in terms of which you are modelling are actions. There are primitive actions such as creating objects, destroying objects, changing the properties of objects, passing messages, receiving input, sending output, etc. There is some notion of an activation mechanism to express under what circumstances another instance of a procedure is activated. There is some notion of control transfer from one active agent to another active agent, which can take place under different circumstances.

Activation mechanisms start with the use of the standard programming language, procedure call mechanism, and move to what has been called pattern directed procedure invocation first provided in Planner. In Planner, procedures are demons which, like database triggers, are activated when a particular condition takes place.

The obvious starting point for control structures is the hierarchical control structures in standard programmed languages. You can move to more heterarchical regimes as far as a situation where everything in a knowledge base is treated as an active agent. Everything moves in terms of messages passed between active agents.

A basic feature of procedural representation is efficiency of execution. I don't mean the number of machine cycles or machine instructions. I mean instead that if you have to perform a fairly conceptual kind of task, e.g., cooking, using a procedural representation scheme, then you will follow the steps that are relevant to the successful execution of that task more directly than if you were using another sort of representation scheme. Another important feature, or disadvantage, of procedural representation is the difficulty in understanding exactly what kind of knowledge is included in your procedure. The reason for this is that any one fact, such as "every person is mortal" may be used by all sorts of procedures. So if you want to know where the fact is used or to modify that fact, you really have to look at your whole knowledge base. There is no direct association between the facts that are represented in your knowledge base and the places where those facts are used.

Weber: Isn't that the same in the case of semantic networks? Can you always trace back to the procedures in the interpreter where a particular fact was used?

Mylopoulos: No. In a semantic network representation, whatever you are representing should be represented in terms of objects and relations on the semantic network itself. What is in the interpreter

is what you hide under the rug in terms of procedural representation or something else.

Carbonell: It is sort of the same. You can encode a procedure declaratively in the semantic network, and have an interpreter that will convert that representation into something that will directly encode to the English phrase "First you walk around here, then you go out the door, and then you go down the steps, and that's how you get to New York." The determination of proceduralality depends on whether one has an interpreter that directly interprets that procedure represented or not.

Goldstein: Again, you have to look at it historically. The earliest semantic nets did not encode procedures in the network. They had a very simple interpreter that simply did association in the network. It had a number of motivations, including a psychological one. Later on more was done, resulting in distinct types of semantic nets.

Mark: To say you encode a procedure in a semantic network is not the same as encoding an object. We know a lot more about relationships between objects than we do about relationships between procedures.

Mylopoulos: If you use procedural notions to represent knowledge (e.g., you worry about who activates whom, and under what circumstances), then you are using the procedural representation. It doesn't matter whether semantic networks come in later on as a convenient data structure for representing the procedures. The same applies to representing logical assertions in terms of semantic networks. In a certain sense you are not using a semantic network representation scheme, you are using the logical representation scheme. The basic notions that you use to model world knowledge, whatever that is, are logical notions.

Codd: The problem is, though, that there are no explicit rules that constrain what knowledge you can find in any of the semantic network approaches. If we're going to adopt the semantic network approach we want some specific rules or checks that prevent us from hiding knowledge, knowledge that shouldn't be there in some sense.

Mylopoulos: The generic examples of such schemes are various types of production systems that have been used successfully to build expert systems for particular tasks. Planner, which was mentioned earlier, and Actors, which were also developed by Hewitt, and used this idea of a knowledge base simply being a collection of active agents that can pass messages between each other.

Hayes: Why didn't you include PROLOG?

Mylopoulos: I see PROLOG starting with the logical viewpoint and going some distance toward procedural accounting for some aspects of generic representations. So you could include it.

Hayes: The trouble is that assertional-procedural distinction. It's lost under meaning equivalence. You can find two different systems, one will be called procedural and one will be called asertional, yet they have essentially the same meaning. I think the distinction is fairly useless in discussing representation because it's not preserved under change.

Meyer: Since there are n theories I'm confused. One thing which hasn't been mentioned and which comes from an entirely different domain, the operating systems domain, bears a striking resemblance to pattern directed control mechanisms. That is CSP, Hoare's cooperating sequential processes which stem from Dijkstra's guarded commands, and has found its way into at least one language. It's very similar to procedural kinds of knowledge.

Frame Oriented Representations

Mylopoulos: I would like to talk about a recent development in knowledge representation called frame theory, or framework representation. It is not quite a type representation scheme but takes ideas from more than one of the basic semantic network categories. Frame theory is based on a notion of a frame. You can think of a frame as a complex data structure for storing and representing typical situations. A frame has slots with which you can associate defaults, constraints, and procedures that tell you what you might or might not do. The framework spends quite a bit of time worrying about the organization of the frames that constitute the knowledge base. The organization may come in terms of the same kinds of organizational principles used in semantic networks, e.g., similarity links between frames. There are mechanisms that can be provided for handling exceptional situations. For example, if all elephants are supposed to have four legs, and a particular elephant has three legs you still want to include it somehow in your knowledge base even though it doesn't satisfy the description of what an elephant is. It doesn't satisfy the frame that represents the concept of an elephant, or the stereotypical elephant.

Some of the features of frame based representations are that they are object-oriented, just like semantic network representation schemes. There is considerable emphasis on principles to organize a knowledge base. In addition to including notions from semantic network schemes, they also include notions from procedural representation schemes. KRL, Knowledge Representation Language and FRL, Frame Representation Language, started with this notion of a frame and developed knowledge representation schemes based on it. Some of the other representation schemes that were mentioned earlier, like OWL, PSN, and KLONE are very similar to FRL and KRL, but their starting point was perhaps different. Their authors saw their work as starting with semantic networks and building on top of it, whereas FRL and KRL people saw their work as particular representation of the frame theory.

The Use of Descriptive Information

Finally, let me go briefly over some of the uses one expects to make of facts that constitute a knowledge base. There is a difference between the AI, programming language and database uses of knowledge bases. A programming language person views a specification as having only one or two

uses. A database person uses a database schema or a database only for storage and retrieval. In the case of a knowledge base, one has to be ready to account for many different types of uses, including things like using inference rules to generate facts that were not in the knowledge base to start with; using access information for question answering, for example; matching, incomplete information and self knowledge.

Matching is used to analyze input data, e.g., to recognize concepts in sentences and pictorial data. Matching can be a very important operation in some knowledge representation schemes. In KRL, matching is a basic operation.

What is meant by incompleteness of knowledge representation is that in certain situations where you are dealing with micro-worlds or an artificial kind of world, you can expect the knowledge base to be incomplete in the sense that it doesn't give a complete description of whatever it is you are modelling. This has to be taken into account by inference mechanisms. Incompleteness is also important in another sense. If a knowledge base is always an incomplete description of what is being modelled, then you will always want to add knowledge, new facts, to it. The task of generating a knowledge base is never complete and may go on throughout the lifetime of the knowledge base. So knowledge acquisition, or the problem of bringing new facts into the knowledge base in such a way that you satisfy all the organizational principles and constraints that go with your knowledge representation scheme, is an important problem.

The last point worth mentioning concerns self knowledge. Let me give some examples. A particlar type of self knowledge, called the closed world assumption, assumes that the collection of objects in a knowledge base is complete, e.g., there are no other objects in the world being modelled that are not being represented at some time. This assumption is knowledge about your knowledge base, not about the world you are modelling; so it is self knowledge. A default mechanism is a sort of self knowledge. You default in a situation where you say, I know otherwise I can assume the value of that property, e.g., a person has two legs. Again, this information is not knowledge about the world being modelled, rather it is knowledge about the knowledge base itself.

Codd: I wouldn't want this set of features to be interpreted as distinguishing AI activities from database activities. These activities are in the database world. The main distinction is that much of the work in the database world assumes that we are dealing with regularly structured data. There are regularities that it pays to exploit and for which you introduce special, macro operations that would be hardly useable if you were dealing with a mass of heterogeneous structures or worlds.

Mylopoulos: We could include some macro operators in the representation itself.

Codd: The point you're trying to make is that these are the distinguishing features of knowledge representation schemes. Fine. I agree with that, but I don't want people to interpret it as a list that distinguishes AI activities from database activities.

Balzer: In my battling with database conceptual schemas I've seen a similar blurring of the different database representations, e.g., relational and network, because good features in one are being incorporated in others. Therefore they are coming closer together. I see very little difference between those structures and the kind described here for AI systems. The difference is mainly in the intended use of the system. Ted Codd said that database people attempt to capture important regularities because of large amounts of data to be processed. In the AI community we don't know ahead of time what those regularities are. Therefore we need some kind of runtime problem solving to figure out what it is that we can capture to make the connection between things. The main distinction is a delay of binding.

Codd: Oh no. I disagree with that. There exist relational systems which have knowledge acquisition in the sense that it exists in AI, namely, you can dynamically declare relations, as well as create, destroy, expand, and contract them dynamically and not bring the system down.

Balzer: That's a different kind of delay of binding than what I'm talking about. Let me say that's expansion of structure. It's user extension. In the delayed binding I was talking about, the system goes through some problem solving activity to figure out how to connect things.

Codd: What you are talking about is non-determinism, and I agree, that is a distinguishing feature. Many AI problem areas are concerned with non-determinism and that is a difference. That's not binding.

Hendrix: Perhaps equally important is that most conventional database systems don't deal at all with quantification. You may have quantification in the query language, but the representation itself doesn't deal with, in particular, existential variables, and, unless you are talking about a closed world, universal quantification, an important distinction. Modalities are another kind of thing.

Mylopoulos: Certain semantic data models do deal with particular kinds of quantification. With generalization you can represent quantification of the type, for all x, if p(x) then q(x). With aggregation there is some sort of universal statement where that for all x such that p(x) then there is a y which is related to x.

Hendrix: But not in the database itself. My point of view, coming from the AI community, is that we ought to use the best AI techniques available to encode information in the schemas.

Sowa: It is more a continuum. AI work has very irregular data with more description than there is data. Databases tend to have regularly structured data with less description than data. It's largely a question of the amount of descriptive information needed to describe the database.