This is an introduction to the foundations of knowledge representation in Cyc. Our first topic is: Why use logic?
Knowledge representation requires a representation language. Candidate representation languages range from natural languages (such as English or Turkish) to logic-based languages to object-oriented programming languages and others. CycL, the language used for knowledge representation in Cyc, is a (high-level) logic-based language. This section explores the reasons for that choice, and the advantages of logic-based knowledge representation.

One issue in the choice of representations is expressiveness. Since we want a great deal of expressiveness for the kind of knowledge Cyc is going to contain, it is sometimes suggested that we use natural language. The expressiveness of natural language, though, goes beyond what we need. It also gives rise to special problems if one wants not only to store, but also to reason with, the represented knowledge. Logic-based representation, in contrast, gives us enough expressiveness, and facilitates the reasoning as well.

Natural language is obviously very expressive. But this can lead to problems. Consider the first three sentences on the slide. Each of these means roughly the same thing and each of them has the implication that Jim’s falling occurred before his injury. If we want to represent that implication, do we write a rule for every natural language expression that could possibly express this point?

Logic-based languages offer a simplified, more efficient approach. First, we identify the common concepts – for example, the relation “x caused y” – at the heart of the English sentences. Then, we use logical relationships to formulate rules about those common concepts. For example, “if x caused y, then x temporally precedes y”.

NL vs. Logic: Expressiveness

NL:
Jim’s injury resulted from his falling.
Jim’s falling caused his injury.
Jim’s injury was a consequence of his falling.
Jim’s falling occurred before his injury.

NL: Write the rule for every expression?

Logic: identify the common concepts, e.g.
the relation: x caused y
Write rules about the common concepts, e.g.
x caused y → x temporally precedes y
NL vs. Logic:
Ambiguity and Precision

**NL:**
Ambiguous

- x is at the bank.
  - river bank?
  - financial institution?

**Logic:**
Precise

- x is running-InMotion → x is changing location
- x is running-DeviceOperating → x is operating
- x is running-AsCandidate → x is a candidate

*Reasoning:* Figuring out what must be true, given what is known. Requires precision of meaning.

Another issue in the choice of a knowledge representation language is ambiguity. Natural language is highly ambiguous. For example, if we say, “x is at the bank,” we don’t know whether what is meant is the riverbank or a financial institution. If we say that x is running, we don’t know whether x is changing location, operating (like a piece of machinery), or running as a candidate for office. On the other hand, with a logical representation we can precisely define the concepts we use. We can, for example, define a distinct concept corresponding to each of these three senses of “running.” This allows us to place the appropriate rules on their respective concepts, whereas they could not all be placed on the one ambiguous word.

This matters greatly for representation of knowledge in the Cyc Knowledge Base. After all, we are representing the knowledge for a purpose: we want to use the represented knowledge in reasoning. Reasoning means, at least in part, figuring out what must be true, given what is known. In order to reliably figure out what follows from what you know, you must be able to specify the starting point. Reasoning requires a clear understanding of exactly what knowledge you have. In other words, reasoning requires precision of meaning.
Logic: Well-understood operators enable reasoning:

Logical constants: not, and, or, all, some

Not (All men are taller than all women).
All men are taller than 12”.
Some women are taller than 12”.

Not (All A are F than all B).
All A are F than x.
Some B are F than x.

Logic also has the advantage of offering us a calculus of meaning. Logic features several well understood logical operators, such as those listed on the slide. They are well understood in the sense that they have been studied for years and their operation is well-documented.

For example, consider the sentences: “It is not the case that all men are taller than all women.” And also: “It is the case that all men are taller than twelve inches.” It follows from these two sentences that some women are taller than twelve inches.

You can express the format of these sentences and find the conclusion based on the logical constants alone, without knowing what the particular non-logical words (such as men, taller, and women) mean. This is very helpful for reasoning.
When choosing a formal symbolic language rather than a natural language, why choose logic rather than something like frames and slots? With frames and slots (and some object-oriented languages) reasoning depends on the mode of the representation, so there’s less reuse of the knowledge. Because of that, you either get less coverage or more bulk as you try to represent the knowledge from every direction in which you could possibly want to use it.

Also, the knowledge representation must be designed around the indexing, and the implicit knowledge is lost if the code is separated from the knowledge base. When using logic, the knowledge representation (KR) is mode-independent and the indexing is independent of the KR. And the implicit knowledge is independent of the KB.

Let me explain what this means . . .
Here’s an example. On the left side of the slide, we have a frame-and-slot type representation (or an object-and-attribute type representation). So we have an object, carl, and carl has the animal type elephant. Carl also has the mother claire. We have another object claire. claire has the animal type elephant, and claire’s mother is elaine. Now, this particular representation will allow us to look up carl and discover that his mother is claire.

However, if we look up claire, we’ll discover that her mother is elaine, but we won’t discover that she’s the mother of carl. In order to be able to do that kind of lookup with this type of representation, we need to create a separate index going from the mother attribute values (or slot values) to the objects of which they are mothers.

To the right of the frame-and-slot type representation, we have a logical representation. The first sentence says, Carl is an instance of Elephant; the second says, Carl’s mother is Claire. What’s worth noting about this representation is that the meaning and implications of the second sentence, Carl’s mother is Claire, can be accessed based on any of its argument values. So, we could look up mother and get all of the animals bearing this relationship. We could look up Carl and find out who his mother is. We could look up Claire and find out whose mother she is or who her mother is. So this single representation does the work of many different representations in the object-oriented or frame-and-slot representation. Cyc takes advantage of this independence, by indexing all argument places -- such as are filled by mother, Carl, Claire -- of each assertion. This comprehensive indexing enables efficient knowledge reuse.
Implicit knowledge

Also, in a logical representation, implicit knowledge is kept in the KB. On the left side of the slide, we have a frame-and-slot representation. We have object, elephant, and we have a number of attributes for it. Notice if you wanted to mark somewhere that you could figure out the weight of the elephant from the other attributes, there’s nowhere in the knowledge representation itself to put that information. You’d have to create a piece of code that attaches to the weight slot with instructions. But that means if you separate the knowledge base from the code base, you lose that information.

In the box on the right half of the slide, we have our logical representation. Here we have both that Elephant is a type of mammal, and we have a rule. The logical representation of that rule declaratively states that if something is an elephant and it’s male and it’s about Y meters tall, then it’s about 2Y tons heavy. That knowledge stays in the knowledge base even when you separate it from the code.
To summarize, the advantages of logic-based knowledge representation include: expressiveness (we get enough expressiveness using logic-based knowledge representation without the extra problems that natural language would bring us), precision (so we know exactly what the represented knowledge means), a calculus of meaning (so that we can reason with the knowledge based on logical constants), and use-neutral representation (which makes the represented knowledge more reusable).

Indexing is separated from the knowledge representation, so we get more reuse and the ability to access knowledge in unanticipated ways. And the implicit knowledge is maintained in the KB itself and retained even if we separate it from the inference engine.