Ontology Representation : design patterns and ontologies that make sense

Hoekstra, R.J.

Publication date
2009

Chapter 2

Knowledge Representation

> "Once the characteristic numbers of most notions are determined, the human race will have a new kind of tool, a tool that will increase the power of the mind much more than optical lenses helped our eyes, a tool that will be as far superior to microscopes or telescopes as reason is to vision."
>
> *Gottfried Wilhelm Leibniz, Philosophical Essays*

2.1 Introduction

The goal of AI research is the simulation or approximation of human intelligence by computers. To a large extent this comes down to the development of computational reasoning services that allow machines to solve problems. Robots are the stereotypical example: imagine what a robot needs to know before it is able to interact with the world the way we do? It needs to have a highly accurate internal representation of reality. It needs to turn perception into action, *know* how to reach its goals, what objects it can use to its advantage, what kinds of objects exist, etc. Because this problem solving takes place in a different environment (inside a computer) than human problem solving, it is subject to different restrictions, such as memory capacity, processing power and symbol manipulation. Where human reasoning can resort to a wide array of highly redundant patterns, machines will inevitably resort to parsimonious and incomplete representation, suitable only for solving a particular set of problems.

The field of knowledge representation (KR) tries to deal with the problems surrounding the incorporation of some body of knowledge (in whatever form) in a computer system, for the purpose of automated, intelligent reasoning. In this sense, knowledge representation is *the* basic research topic in AI. Any artificial intelligence is dependent on knowledge, and thus on a representation of that knowledge in a specific form.

The history of knowledge representation has been nothing less than turbulent. The roller coaster of promise of the 50'ies and 60'ies, the heated debates of the 70's, the decline and realism of the 80's and the ontology and knowledge management hype of the 90's each left a clear mark on contemporary knowledge representation technology and its application. In particular, the idea of a Semantic Web (discussed in Chapter 3) led to the integration of insights from two distinct fields in symbolic AI: knowledge *representation*, and knowledge *acquisition* (expert systems). Two areas that had showed little or no interaction for three decades, at least not significantly, since the divide between *epistemic* and *heuristic* aspects of an intelligent system (McCarthy and Hayes, 1969).

In this chapter I give an overview of the historical origins and rationale of knowledge representation, and the family of languages known as *description logics* in particular. These languages turned out to play an important role in the development of semantics on the web, discussed in Chapter 3, and could not have reached their current prominence without the wide adoption of the word 'ontology' in the AI literature of the nineties (see Chapter 4).

2.2 Two Schools

In the early second half of the 20th century, AI housed two very different schools of thought. The first was very much based on the idea that knowledge is best captured using a general purpose, clean and uniform language: *logic*. With roots in philosophy, it was oriented towards the adequate representation of our theoretical understanding of the *structure* of the world, and assumed that a small set of elegant first principles can account for intelligence. The second school's main interest was the approximation of *human* intelligence, and human *behaviour* in particular. Its main proponents had a background in psychology and linguistics, rather than philosophy or mathematics, and were less concerned with rigourous formal semantics. They built systems that 'just work', based on the assumption that human intelligence is a hodgepodge of many different ad hoc conceptions and strategies. Roger Schank coined the two groups the *neats* and the *scruffies*, respectively.

In short, research in KR can be roughly categorised as having either a *philosophical* or *psychological* nature. In an influential paper McCarthy and Hayes (1969) discussed a number of fundamental issues for AI (amongst which the famous frame problem). They operationalise (artificial) intelligence as follows:

> "...an entity is intelligent if it has an adequate model of the world (including the intellectual world of mathematics, understanding of its own goals and other mental processes), if it is clever enough to answer a wide variety of questions on the basis of this model, if it can get additional information from the external world when required, and can perform such tasks in the external world as its goals demand and its physical abilities permit."
>
> (McCarthy and Hayes, 1969, p.4)

This definition introduces the distinction between a *representation* of the world, and a *mechanism* that uses problems and information expressed in that representation to perform problem solving and decision making. Artificial

	Semantics (Epistemological Adequacy)	**Reasoning** (Heuristic Adequacy)
Philosophy	First Order Logic	Theorem Provers (exhaustive, combinatorial)
Psychology	Semantic Nets, Frames, (Rules)	Problem Solvers (goal directed, rational)
Practice	Rules	Expert Systems

Table 2.1: Schools and systems in Knowledge Representation

intelligence systems should attain a balance between both *epistemological* adequacy and *heuristic* adequacy.

The distinction between these approaches was very much evident in AI research in the seventies. Mylopoulos (1981) organised the schools in a taxonomy; KR languages can first of all be divided into procedural and declarative ones. The first is aimed at attaining heuristic adequacy, the second has an epistemological perspective. The declarative languages can be further subdivided into logic-based and semantic network ones, see Table 2.1. It wasn't until the end of the seventies that so-called *hybrid* systems attempted to combine declarative and procedural views (see Figure 2.1).

2.2.1 Intelligence by First Principles

The idea of automated intelligent reasoning has been around for a long time, and can be traced back to ancient Greece. Aristotle's *syllogisms* are often seen as the first example of a formalisation of valid reasoning. With the separation of mind and body in the 17th century by Descartes, and in common sense, the road was opened up to apply newly found mathematical insights to model and imitate parts of human thought as mechanistic processes. Perhaps the most appealing examples are Pascal's arithmetic machine for addition and subtraction, and Leibniz' improvements to it to support multiplication, division and computing the square root.

In the mean time other great minds, such as John Wilkins (the first secretary of the Royal Society) were busy working on a systematic account of all of human knowledge using his *Real Character*. The real character encoded words in such a way that each had a unique non-arbitrary name. For this, Wilkins used a three-layered tree structure. All concepts are distributed over forty Genuses; these in turn are divided into Differences which are separated as Species. Each of these layers adds one or more letters, such that any path through the tree can be represented as a unique four-letter word.

More influential, however was Leibniz' invention of the binary system of numbers that lies at the heart of his calculator. He entertained the thought of encoding 'notions' as unique binary encoded numbers. Using a simple method

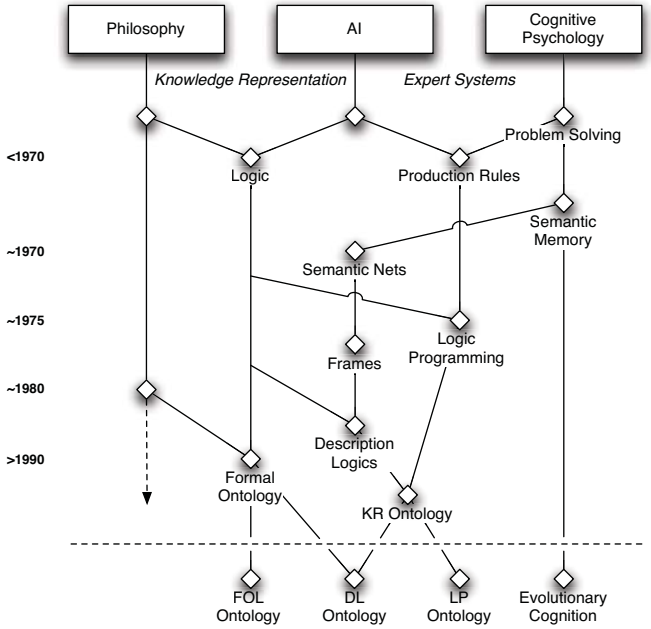

Figure 2.1: History of knowledge representation

of combining these numbers by applying mathematical operators, a machine – the *calculus ratiocinator* – could be built that would 'think'. Unfortunately, gathering all notions proved to be too formidable a task, and for various reasons – his work on the differential calculus beckoning – Leibniz was unable to continue his work in this direction. A few decades later Linnaues was more successful when he built his *Systema Naturae* of a more limited domain.

In Leibniz' view, intelligent reasoning is a form of *calculation*: the manipulation of symbols according to a set of logical axioms. Still (or even more) prevalent once computers became available, logic based knowledge representation was very popular during the 1960's after the development of automatic theorem proving using *resolution*. The idea of a general-purpose logical engine fuelled the idea that logics could be used as the basis of all intelligent action. Despite the fact that logic based knowledge representation has the advantage of a well-understood formal semantics, standard inference rules, and relatively simple notation, it has several drawbacks as well (Mylopoulos, 1981).

Logic based languages did not provide a way to organise knowledge in separately understandable modules. And as time progressed it became clear that this engine was not "powerful enough to prove theorems that are hard on a human scale", which lead to the "great theorem-proving controversy of the late sixties and early seventies" (Newell, 1982, p.90-91). Instead of being universally applicable, research in theorem proving increasingly focussed on smaller, theoretically hard topics. The result was a rather allergic reaction to anything smelling of *uniform procedures*, and at the start of the seventies it seemed that logic as knowledge representation language was very much done for in AI Hayes (1977); Newell (1982).

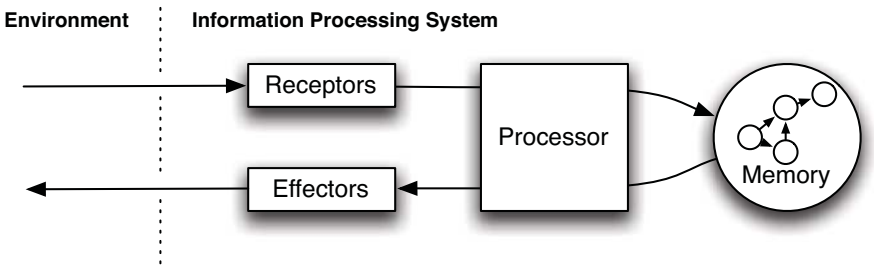

Figure 2.2: Structure of an Information Processing System (adapted from Newell and Simon (1972))

Another problem was that logics are not well suited to represent procedural knowledge. The PROLOG programming language (Kowalski, 1974; Bratko, 1986) was designed to alleviate this problem by the interpretation of implications as procedure declarations (see Figure 2.1).

2.2.2 Production Systems

At the start of the seventies, much of AI research was targeted at building psychologically sound computer models. Such models were used in cognitive psychology research for the study of both language semantics and human reasoning. It soon became evident that people within the two areas of research entertained differing views on what 'knowledge' is. While one group maintained that knowledge is all about 'how' (the heuristic view), the other group advocated the 'what' (the epistemological view). The latter is discussed in Section 2.2.3.

During the 1950s, Newell and Simon (1972) developed the *Logic Theorist* and the *General Problem Solver* (GPS) programs which were able to perform various tasks using a combination of theorem proving and heuristics. Although they started out in a similar vain as the purist logicians, they soon developed a quite different approach. In their view, human thinking and problem solving is by information processing: "the human operates as an information processing machine"(Newell and Simon, 1972, p.21). Even though this *information processing system* (IPS) perspective soon turned out to be overly simplistic as a correct model of the human mind; it is a useful abstraction, and has developed into a major knowledge representation paradigm. The general structure of an IPS is that of a processor that interacts with its environment – using a receptor and effector – and stores information about the environment in its memory (see Figure 2.2).

The processor consist of a set of elementary information processes (eip) and an interpreter that determines the sequence of processes to be executed as a function of the symbols stored in memory. This view was very much oriented towards the way in which computers can be used to do *thinking*. Of primary concern, here, were the questions as to 1) how elementary information processes should be expressed, and 2) what strategies should be implemented as part of the interpreter.

Systems that follow the IPS architecture of Newell and Simon are generally a type of *production* system (Buchanan and Shortliffe, 1984). These systems were first conceived of as a general computational mechanism by Post (1943), used to describe the manipulation of logical symbols. In its simplest form, a production rule consists of a left hand side (condition) and a right hand side (action), usually in the form of an `if ...then ...` statement. A production rule is essentially the operationalisation of a reasoning step (i.e. an eip) in an IPS: given some input structure of symbols, the rule produces a new (modified) structure of symbols. An interpreter iteratively evaluates all productions in the system until it finds one that matches one or more symbols stored in memory. This evaluation of the condition of rules is a passive operation that has no impact on those symbols. When the interpreter evaluates some input to the conditions of a rule, it is said to 'fire', and performs the operations specified on the right hand side on relevant symbols in memory. Because of its dependency on the order in which the interpreter carries out evaluation, a production is not applied in the same way as the full combinatorics of logical implication. Where the consequent of an implication necessarily holds at all times – all information implicit in the knowledge base holds at all times – the consequent of a production rule only comes into effect after the rule has fired.

Production rules were (and still are) used for a wide variety of applications. They can be categorised according to two views: as a means for psychological modelling on the one hand (as in IPS), and for *expert systems* on the other. In cognitive psychology, production rule systems were part of an effort to create programs that capture human performance of simple tasks. This performance includes typical human treats such as forgetting, mistakes etc. and rules were a promising paradigm for capturing heuristics in human problem solving. For these scruffies, human intelligence was rather a "particular variety of human behaviour" (Davis et al., 1993, p.10); the 'intelligence' of reasoning can only be assessed by virtue of its correspondence to human intelligence, and not necessarily by whether it is clean and logical. To them, production systems provided a clear formal way to represent the basic symbol processing acts of information processing psychology (Davis and King, 1984). The production rule semantics allowed an escape from the nothing-or-all inferences of theorem proving, and could be used to capture the local, rational control of problem solving.

During the eighties, rule-based knowledge representation was applied in a number of large scale projects, and found its way into many enterprise industrial and government applications. Because of their rather practical, application oriented perspective, focus shifted from a cognitive perspective to building large knowledge-based systems, and creating and maintaining elaborate models that capture expert knowledge. Knowledge-based expert systems, emphasise problem-solving *performance* at the cost of psychological plausibility. Production rules are used to capture expert knowledge about a particular task or domain, and enable the system to support, augment or even surpass human problem solving. Production rules can be modified and extended relatively easily, which makes them a convenient paradigm for incremental system development. A well known example of such a system is is the MYCIN expert system for medical diagnosis (Buchanan and Shortliffe, 1984). Rather than attempting to simulate diagnosis by human experts, it captures and formalises the (often implicit) knowledge, i.e. the 'rules of thumb' used by those experts, into the form of production rules. Because of the emphasis on performance,

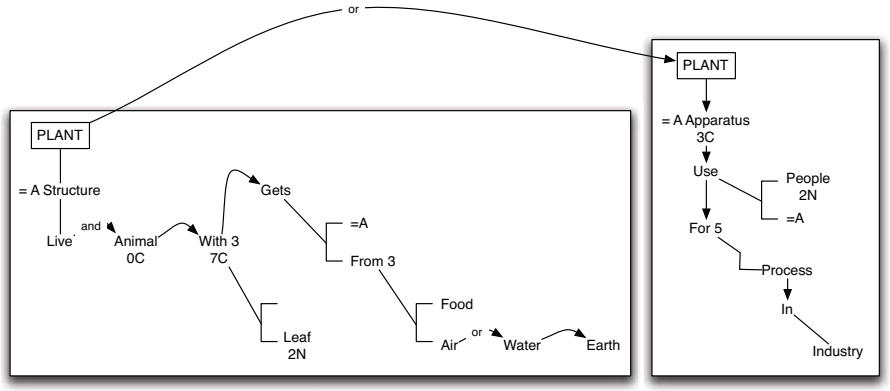

Figure 2.3: A semantic network, using the notation of Quillian (1966)

interest soon grew towards the improvement of the interpreter with more efficient strategies.

2.2.3 Semantic Networks

While production systems were rather good at mimicking the heuristics of human problem solving, they were severely limited when it comes to another major area in AI: natural language understanding. Looking at it in terms of an IPS, a natural language processing system is all about mapping terms and structures in natural language to their cognitive interpretation in memory. In other words, lexical terms are to be *grounded* in a model that represents their semantics, where the semantics should mimic the human understanding of these terms in memory. This application area brought forth a number of very influential knowledge representation technologies that count as the direct predecessors of the languages currently used for representing knowledge on the Semantic Web.

The theory of *semantic memory* by Quillian (1966), is based on the idea that memory consists of associations between mental entities, i.e. semantic memory is *associative* memory (Anderson and Bower, 1973). Quillian's semantic memory can be depicted using *semantic networks*, directed graphs where nodes are terms and the relationships between those terms are represented as arcs (see Figure 2.3). In semantic networks, different senses of a word concept (a node) can be organised into *planes*, and can be related through *pointers* (edges). Pointers within a plane form the structure of a definition. Pointers leading outside a plane indicate other planes in which the referenced words themselves are defined. The use of planes and pointers allowed the 'import' of a word definition by reference. Quillian (1966) distinguished five kinds of pointers: subclass, modification, disjunction, conjunction, subject/object.

Though graph-based representations had been used extensively in the past for a wide variety of representations, Quillian was the first to use semantic networks for representing human knowledge:

> "His intent was to capture in a formal representation the 'objective' part of the meanings of words so that 'humanlike use of those meanings' would become possible"
>
> (Brachman, 1979, p. 5)

Underpinning the cognitive perspective of this approach, was his later research using reaction time in assessing the factual truth of statements (Collins and Quillian, 1969, semantic verification) to test the psychological plausibility of semantic network models. Of particular interest was the question as to whether the retrieval of inferred property values (over the subclass relation) would take more time than directly represented property values. The fact that indeed this was the case provided backing for *property inheritance* over a superclass-subclass taxonomic hierarchy in semantic network representations of human semantic memory. Furthermore, he intended his model to be suitable for automatic *inference*, allowing for querying information implicit in the model. Effectively turning the semantic network paradigm into not only a representation but a *simulation* of human memory.

The expressive power of the original semantic networks soon became too restricted, and a number of innovations and extensions followed. Quillian's original set of five link types turned out to be insufficient, and was superseded by the ability to type pointers using *named* attributes, i.e. a means to use a token to point to a type. Furthermore, a distinction was introduced between concepts and examples (later *instances*), in Carbonell (1970). These innovations led to a plethora of widely variant, rather unprincipled, semantic network 'languages'. Many of which applied the technique to domains other than psychology.

A true show-stopper, however, was that new link types, and even concept types, were not explained and the interpretation of semantic networks was left to the 'intuition' of the reader (Brachman, 1979): semantic networks did not really have semantics. For instance, both the concept–instance distinction and the type–token distinction were obfuscated by the use of the infamous 'IS-A' link (Woods, 1975; Brachman, 1983).

Perhaps most manifest to current readers was the critique that the networks made no distinction between domain level constructs – conceptual relations in the domain – and knowledge structuring principles such as the subclass relation. As semantic networks relied heavily on graphical notation, this is most apparent in the uniformity of presentations. Woods stressed the importance of considering the semantics of the representation *itself*.

To summarise, semantic networks were developed as part of an effort in psychology to represent human semantic memory. Although they have been successful in many ways, they suffered from lack of proper semantics: "The 'semanticness' of semantic nets lies in their being used in attempts to represent the semantics of English words." (Brachman, 1979, p. 26).

2.2.4 Frames

The semantic network model received criticism from cognitive science itself as well. Most notable in this respect is Minsky (1975),[1] who argued against

[1] Though, as is common his ideas had been brooding in the community, cf. Schank and Abelson (1975); Woods (1975)

the paradigm of associative networks for representing semantic memory. In his view the 'chunks' of thought should be larger and more structured, and their "factual and procedural content must be more intimately connected". He proposed *frames* as knowledge structures that represent stereotyped *situations*. This meant a move away from the focus on word concepts in semantic nets to more contextual representation of knowledge. Frames are thus presented as part of a theory on the contents of thought (semantic memory), similar to the notion of *script* in Schank and Abelson (1975), used in natural language understanding.

A frame can be regarded as a group of interwoven nodes and relations (somewhat akin to Quillian's planes), but with a fixed structure. The 'top' levels of a frame represent that which is always true for the situation, the lower levels consist of terminals or '*slots*'. These terminals can specify conditions that its assignments (through specific instances or data) must meet. Minsky distinguishes simple conditions – in the form of 'markers' that require straightforward assignments to e.g. (smaller) sub-frames – from complex conditions that specify relations. The terminals of frames are filled with *default* assignments, which may be overridden when a frame is filled by a suitable particular situation.

Frame *systems* are collections of closely related frames. For instance, the effects of possible actions are reflected as transformations between the frames of a system. Frames within a system share the same terminals; this to allow the integration of information from different viewpoints on the same situation. For example, two frames representing the same cube at different angles share some of the faces of the cube.

Where semantic networks could already support a limited form of automatic inference, an important addition of the frame paradigm is the requirement of an *information retrieval network* that supports a standard matching procedure for determining whether a candidate situation fits a frame. This procedure tries to assign values to the frame's markers. In the case of a mismatch, the network should be able to propose a new replacement frame as possible candidate for a match. Matching is not performed solely on the constraints on a frame, but also by the current goals, which are used to determine constraint relevance.

The main contribution of the frame-based view was that it fixed a knowledge representation *perspective*. Semantic nets could be used to represent anything – not just word concepts – and were in many ways equivalent to generic graph representations. The frame proposal fixes the perspective on descriptions of *situations* in general, and *objects* and *processes* in a situation in particular. Minsky (1975) discusses how frames can be used to represent a wide variety of domains – vision, language understanding, memory – without compromising this perspective. His proposal is often viewed in line with the development of *object oriented programming*.

Furthermore, the frame-based approach incorporates a view on the *manipulation* of knowledge, i.e. the transitions between frames in a frame system. This effectively introduced the *reuse* of information as a knowledge organising principle. Lastly, it envisioned a procedure for *matching* specific situations to candidate descriptions in frames, and introduced *defaults* for dealing with incomplete knowledge.

2.2.5 Frame Languages

Research in the late seventies produced a number of – what in retrospect could be called – frame based KR languages. Not because they were explicitly developed to define Minsky's original frames (they were not), but because they share its emphasis on interrelated, *internally* structured concepts as primary language primitive.

Knowledge Representation Language (KRL) The Knowledge Representation Language (KRL), developed by Bobrow and Winograd (1976), was built on the idea that knowledge is organised around conceptual entities with associated *descriptions* and *procedures*. In their view, a KR language should be independent from the processing strategies or representations of a particular domain. It must provide a flexible set of underlying tools.

KRL descriptions represent partial knowledge about an entity, and can consist of multiple *descriptors* that can be grouped to capture differing viewpoints on the entity. KRL's descriptions are by *comparison* to a known entity (the *prototype*), extended with a further specification of the described entity. The prototype provides a *perspective* from which to view the object being described. The description of a concept entity can combine different *modes* of description (Bobrow and Winograd, 1976, p. 6), such as category membership, stating a relationship, or role in a complex object or event etc.

Reasoning in KRL is done by way of a *process of recognition* where newly introduced objects are compared to stored sets of prototypes. Specialised reasoning strategies can be attached to these prototypes in the form of *procedural attachments*. These procedures could be called depending various triggers (on classes) or traps (on objects) such as goal directed procedure calls (*servant* procedures) and side-effects of system actions (*demon* procedures). Such procedural attachments are coined procedural properties, as opposed to declarative properties.

Bobrow and Winograd make a strong claim that "it is quite possible ... for an object to be represented in a knowledge system only through a set of such comparisons" between prototypes (Bobrow and Winograd, 1976, p.7). The definition of an object is wholly contained within the system, but also functionally complete with respect to that definition as the system can answer any relevant query about it. This represents a fundamental difference in spirit between the KRL notion of representation and standard logical representations. Because the definition of an object is in terms of other objects, and vice versa, and its position within that network of comparisons is determined by a standard inference mechanism, it is the inference mechanism that determines the *meaning* of an object.

Structured Inheritance Networks (SI-Nets) Another frame-like language, the Structured Inheritance Networks (SI-Nets) of Brachman (1979) were an attempt to define an epistemologically well-founded class of KR languages (see Section 2.3.2): granted that we distinguish concepts and relations, how can we account for the apparent meaning of concepts that determines their position within a network? The most prominent of these languages, KL-ONE (Brachman, 1979; Brachman and Schmolze, 1985), is organised around *concepts*. Concepts are *intensional*, and can represent objects, attributes and relations in a

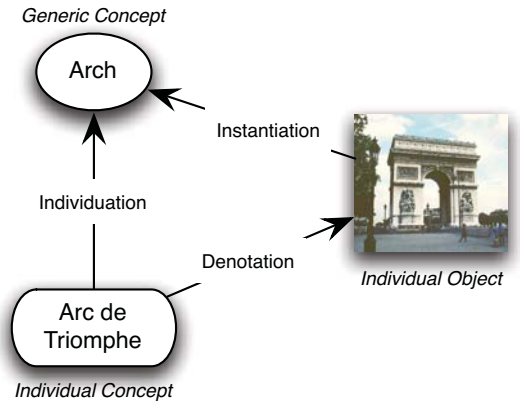

Figure 2.4: Instantiation, individuation and denotation, from Brachman (1979)

domain. Brachman furthermore distinguishes between *generic* and *individual* concepts:[2]

Generic Concept
represents a class of individuals by describing a prototypical member of the class.

Individual Concept
represents an individual object, relation or attribute by *individuating* more general concepts.

Individual Object
(or *instance*) is an object in the actual world that *instantiates* a Generic Concept, and is *denoted* by an Individual Concept, see Figure 2.4.

Similar to KRL entities, KL-ONE concepts are *structured* objects. They are described by *role/filler* descriptions, the 'slots' of the concept. These determine the type of entity that can fill the role, the number of fillers and the importance (modality) of the role. Where KRL uses five *modes* of description, KL-ONE is more abstract and distinguishes three role modality *types* – inherent, derivable and obligatory – that enable a modeller to distinguish between information needed for recognition and more neutral descriptions.

KL-ONE supports procedural attachments, but distinguishes *meta* descriptions – meta statements on concepts using KL-ONE's formalism – from *interpretive* attachments. The former, so-called *structural* descriptions (SD), can be used to prescribe the way in which role fillers interact for any individual; SD's relate two or more roles. The latter are similar to KRL's procedural attachments, and can be expressed using the interpreter language that implements KL-ONE itself (INTERLISP).

In summary, frame-based KR languages introduced a number of knowledge structuring principles that were absent in earlier semantic nets. They

[2]The distinction is similar to that between types (generic concepts), tokens (individual objects) and occurrences (individual concepts) in philosophy.

focus on structured concepts, which are defined by simple or very elaborate structural or procedural descriptions in terms of other concepts. Concepts are either generic or individual, and are clearly distinguished from their real world counterparts. The meaning of a concept is determined by its position within the network of other concepts, which is enforced by a standard inference mechanism based on the descriptions of the concept. Because of the combination of conceptual and procedural (production rule-like) primitives, languages such as KL-ONE and KRL are sometimes called *hybrid* systems.

2.3 Epistemology

"One man's ceiling is another man's floor"

Paul Simon, 1973

Frame based languages proved to be a significant improvement over other semantic networks.[3] They fixed a paradigm which was more rigourous and better suited for representing knowledge. The development of these languages was not only given in by the cognitive psychology argument of Minsky (1975), but also (and perhaps more importantly) by the growing awareness of the need to have a clear definition of what a knowledge representation *is*.

The interaction between psychological insights and knowledge representation practice led to two fundamental questions:

1. What is the relation between a machine representation and the thing (domain, body of knowledge) it denotes? and,

2. How does the representation language relate to the representation itself?

So, while at first developers of both semantic and procedural knowledge representations were primarily concerned with the *psychological plausibility* of their models, the proliferation of semantic nets and frame based languages sparkled growing concern about the *epistemological status* of knowledge, representation, and the KR languages themselves.

In his 1980 inaugural presidential address to the AAAI[4] Newell (1982) discussed exactly this topic: the nature of knowledge, and in particular the relation between knowledge and representation. In his view, the latter is used precisely and clearly in computer science while the former is often used informally. He identifies a problem with representation, in that it is attributed a 'somewhat magical' role.

> "What is indicative of underlying difficulties is our inclination to treat representation like a *homunculus* as the locus of *real* intelligence."
>
> (Newell, 1982, p.90)

[3] The above quote was taken from Brachman (1979).
[4] American Association of Artificial Intelligence

Table 2.2: Computer System Levels (Newell, 1982)

Level	Description
Knowledge Level	Knowledge and rationality
Symbol Level	Symbols and operations (also *program* level)
Logic Level	Boolean logic switches (e.g. AND/OR/XOR gates, consists of the *register-transfer sublevel* and *logic circuit sublevel*)
Circuit Level	Circuits, connections, currents
Device Level	Physical description

The most salient task in AI is identifying the proper representation of a problem. It can make the difference between combinatorial and directed problem solving: "... the crux for AI is that no one has been able to formulate in a reasonable way the problem of finding the good representation, so that it can be tackled by an AI system." (Newell, 1982, p.3). The capability to find the proper representation apparently requires some special kind of intelligence.

What epistemological adequacy is, turned out to differ widely, as we have seen in the previous section. McCarthy and Hayes propose to construct a practical philosophical system, based on our common sense understanding of the world. For semantic network-adepts, epistemological adequacy equates to psychological plausibility. But even the criterion of psychological plausibility is not suitably specific to distinguish between production systems and network representations. All three proposals, the logic-based, production-based and network approach, aim to answer the two fundamental questions raised at the start of this section. Although they formulate some description on what a knowledge representation should contain and how it relates to the world it represents, this description remains vague: none of them clearly defines a framework for this relation. And secondly, they do not give an answer to how knowledge relates to the language it is represented in: what are the primitives of knowledge representation?

2.3.1 Knowledge and Representation

In his discussion on the nature of knowledge, Newell (1982) presented the *knowledge* level as a *computer system level*. The execution of computer program code is made possible by its translation to physical operations on a circuit board. This translation passes through a number of steps, or 'levels' at which the program can be expressed (e.g. from java code at the symbol level, to java byte code to processor instructions etc.), see Table 2.2. Levels have a *medium*, the *system* it is used to express, primitive processing *components* and guidelines for their *composition*, and a definition of how system *behaviour* depends on the behaviour and structure of components.

Every computer system level can be defined autonomously – without reference to another level – and is reducible to a lower level. Because a description of a system at some level does not imply a description at a higher level, these

levels are not levels of abstraction: A level is rather a specialisation of the class of systems that can be described at the level directly below it. Computer system levels are concrete and really exist, they are "a reflection of the nature of the physical world" (p. 98). Newell postulated the existence of the knowledge level as a hypothesis:

> **The Knowledge Level Hypothesis**. There exists a distinct computer systems level, lying immediately above the symbol level, which is characterised by knowledge as the medium and the principle of rationality as the law of behaviour.
>
> (Newell, 1982, p.99)

The question is, how does the *knowledge* level fit into the rather technical framework of levels in a computer system? The idea is that when we perform knowledge representation – both procedural and declarative – we express our rather abstract, implicit notions of knowledge manipulating systems (people, or rather *agents*) in terms of a symbol level system. This includes the definition of the medium (knowledge), components (actions, goals), and laws of behaviour (rationality) prevalent at this level. Newell (1982) insisted that a knowledge level description of a system is not just a matter of *treating* a system as a rational agent as in the *intentional stance* of Dennett (1987). But rather that the level exists and behaves in the same way as any of the other computer system levels. The term 'knowledge level' is often used to describe representations of knowledge in terms of concepts, tasks, goals etc. The representation is said to be 'at' the knowledge level. For Newell, however, the knowledge level is the knowledge itself, a representation will always be 'at' the symbol level (Newell, 1993).

One implication of this perspective is that a knowledge representation is to be regarded as truly a representation of *knowledge* and not a representation of physical, philosophical or psychological *reality*. Though, by its very nature knowledge is itself some representation of *a* reality. The relations between concepts and individual objects of KL-ONE in Figure 2.4 are in fact reflections of our knowledge of that object.

Figure 2.5 is an extended version of Brachman's triangle and illustrates the relation between a knowledge representation and reality. At the far right are the individual objects that exist in reality, our knowledge of reality is cast both in terms of knowledge of these individuals as individual *concepts* and as generalisations over these individuals. A knowledge *representation* is constructed in a similar fashion. Individual concepts in representation are denoted by our knowledge of individual concepts; generic concept representations are individuated by these individual concepts. However, we can also choose to represent a generalisation over the generic concepts in human knowledge, these *meta concepts* individuate representations of generic concepts, and are instantiated by actual generic knowledge. Finally, a knowledge representation language provides constructs that allow us to formulate the concepts in our representation through instantiation.

Although it is clear that all knowledge representation occurs *by proxy* of our knowledge of reality, it is not always practical to take the separation between knowledge and reality explicitly into account. If a system is to represent reality

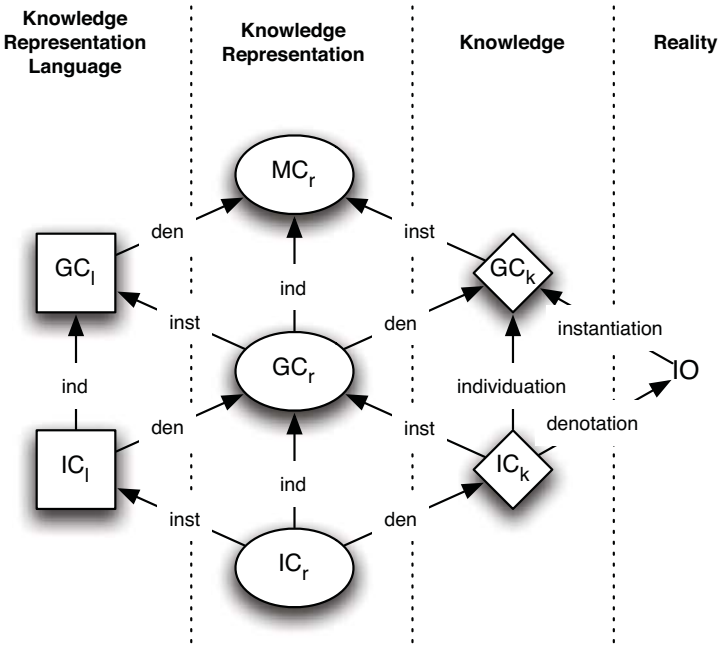

Figure 2.5: Relation between a representation and the world.

as we understand it, there is no more useful and well-tuned proxy than the mental models of reality we use and live by every day.

2.3.2 Representation and Language

The introduction of the knowledge level helps us to put the representation of knowledge into perspective, but does not address the suitability issue of the knowledge representation *languages* of Section 2.2.

Levels In his review of lessons learned in semantic nets, Brachman (1979) identified five distinct groups of primitive types used in these languages. He considered each of these groups to stand for a particular viewpoint, or conceptual 'level'. Any network, he argued, can be "analysed in terms of any of the levels" (p.27). In other words, a concept expressed in a language at one level, can be understood and expressed at all other levels as well. On the other hand, an interpreter usually commits to support only one of these sets.

At the *implementational* level, semantic nets are mere graphs, data structures where links are pointers and nodes are destinations for links. The *logical* level emerged in reaction to criticism that semantic nets did not have formal semantics. It perceives semantic nets as a convenient depiction of predicates or propositions (the nodes) and the logical relationships between them (the links). Originally, however, semantic nets were meant to capture the meaning of word concepts. At this *conceptual* level, links are case relations between nodes rep-

Table 2.3: Levels of Semantic Networks (Brachman, 1979)

Level	Primitives
Implementational	Atoms, pointers
Logical*	Propositions, predicates, logical operators
Epistemological	Concept types, conceptual sub pieces, inheritance and structuring relations
Conceptual	Semantic or conceptual relations (cases), primitive objects and actions
Linguistic	Arbitrary concepts, words, expressions

* Note that the logic*al* Level of Brachman is *not* the same as Newell's Logic Level

resenting word senses. Here, the primitives are less neutral, and encompass conceptual elements and relations, such as action types and cases (thematic roles) respectively. Not always are these primitives explicitly defined as part of the semantic net language, but on the whole the relations do have this flavour. One level higher, nodes and links are language dependent. *Linguistic* level networks are composed of arbitrary relations and nodes that exist in a domain. Each consecutive level adds a commitment to a particular interpretation of the structure of the world.

In line with the criticism of Woods (1975), who urged the consideration of the semantics of KR languages, and Minsky (1975), who argued for structured concepts, Brachman postulates that part of the promiscuity of semantic network languages lies in the absence of an intermediate level between the logical and conceptual levels. He proposed the introduction of an *epistemological* level which allows the definition of knowledge-*structuring* primitives as opposed to knowledge primitives:

> "The formal structure of conceptual units and their interrelationships as *conceptual units* (independent of any knowledge expressed therein) forms what could be called an *epistemology*."
>
> (Brachman, 1979, p.30)

To illustrate, even while we can argue about *which* properties exist, we can still agree that *properties* exist. See table 2.3 for an overview of Brachman's levels. Perhaps his levels are best understood as levels of detail or abstraction. When regarding a linguistic level representation, using plain English words etc., we can zoom in to see the case structure and concepts that underlie the language. If we then zoom in again, we can view the internal structure of these concepts and what makes that they can be related in certain ways. Not very surprisingly, his KL-ONE is designed for representing knowledge at the epistemological level.

We must be careful, however, not to misinterpret the analysis Brachman made as a deconstruction of layers in semantic-network based knowledge rep-

resentation only. In fact, it remains rather unclear *what* Brachman believes to be at a level. His description leaves room for the following alternative interpretations:

> **Language**
> A KR language can be designed to be adequate for the representation of knowledge using primitives at a particular level.
>
> **Knowledge**
> The knowledge represented using a KR language can be of a type corresponding to one of the levels. For instance, it is quite common to describe concepts using some form of logic, but we can just as readily represent logical primitives *as concepts*.

In the first sense, the knowledge *primitives* determine the level of a language; where in the second sense the level describes the *kind* of knowledge expressed in a model. The two interpretations of the levels are not wholly unrelated, and Brachman formulates a number of requirements for KR languages to adequately support the representation of knowledge at a particular level. Firstly, a language should be *neutral* with respect to knowledge at the level above it. Secondly, it should be *adequate* for supporting the level above it, i.e. it should be expressive enough to account for knowledge at that higher level. And thirdly, it should have well defined *semantics*: it should prescribe legal operations, and provide a formal definition of its primitives.

Types For a long time, the expert systems field seemed to steer clear of the epistemological crisis of declarative knowledge representation. And indeed, the levels presented in the previous section do not particularly fit the heuristic perspective of production systems. Although the PSI architecture includes a 'memory' in which knowledge of the world is stored declaratively, these symbol structures are usually very simple hierarchies with limited semantics, and were used chiefly as database-like place holders for instance data.

All of this changed when it became clear that production rule systems were not particularly usable in settings other than which they were designed for. This realisation emerged when Clancey (1983) attempted to use MYCIN rules in the GUIDON tutoring program, i.e. to "transfer back" expert knowledge from a rule base. The idea was to use the rules to explain each step in the diagnostic reasoning process. As the original design goal of MYCIN was for it to be built using a simple mechanism for representing heuristics that would support explanations and advice, it seemed at first that this educational use would be relatively straightforward. It turned out not to be. Merely using MYCIN's built in explanation facility did not work as expected. GUIDON was incapable of explaining many rules because of insufficient information about the way the rule base was structured. In order to support a tutoring setting, it is necessary to extract this "compiled knowledge"(Clancey, 1983).

Rules in MYCIN, but in other systems as well, implicitly encode the design rationale behind the way rules are fitted together. Clancey clarifies the different ways in which *design knowledge* is lost when building rules by distinguishing between *goals*, *hypotheses* and *rules*. These three categories are organised in a network of dependencies (see Figure 2.6). Goals are formulated as questions

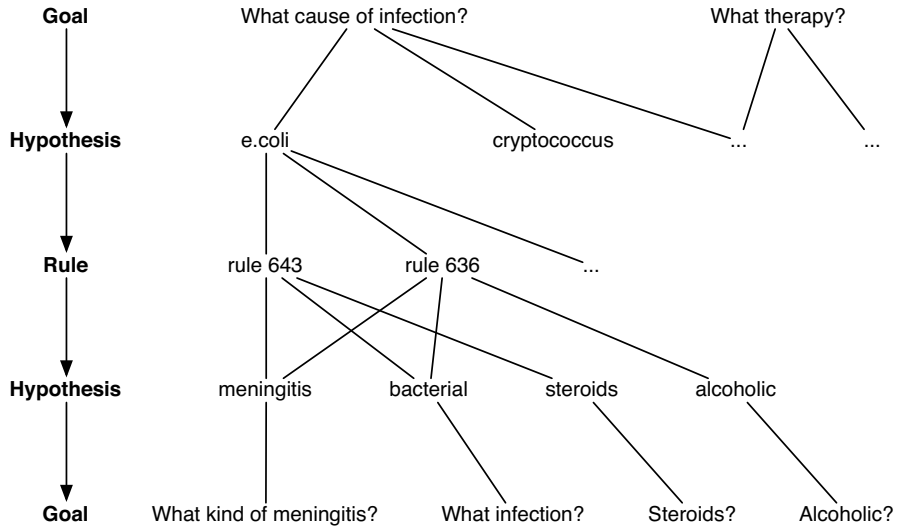

Figure 2.6: Network of goals, hypotheses and rules, adapted from Clancey (1983)

that need to be answered for the system to solve a problem (e.g. making a diagnostic decision). Hypotheses are potential answers to the questions, and are ascertained by rules that have the hypothesis as their consequent. Rules fire, i.e. make their consequent hold, when their antecedent is satisfied. This antecedent is composed of other hypotheses that answer some goal.

The decomposition gives insight in the way the different categories interact when MYCIN traverses the search space. For instance, when it tries to satisfy e.g. the "meningitis" hypothesis, MYCIN will in fact consider all related hypotheses that answer the more general goal "what infection?". The links between these categories are the 'points of flexibility' in a rule representation.

A problem solving *strategy* can be conveyed by making explicit the rationale behind the order of premises in a rule as this affects the order in which goals and hypotheses are pursued. A decision to determine the ordering of hypotheses in a particular way is a *strategic decision*, which can be stated in relatively domain-independent terms. The example given by Clancey is "consider differential broadening factors". Also some level of *structural* knowledge is necessary to "make contact with knowledge of the domain", it provides a "handle" by which a strategy can be applied. For instance, it seems reasonable to investigate common causes of a disease before considering unusual ones.

The *justification* for a rule is captured by the rationale for the connection between the conclusion in the consequent and hypotheses in the antecedent. Justification depends on knowledge of the domain. Clancey (1983) identifies four different types of justification, and consequently four different types of rules (the *domain theory*):

- *Identification* rules use properties of an object to identify it, e.g. "if it walks like a duck and talks like a duck, it is a duck".

- *Causal* rules convey a causal relation between two objects, e.g. "problem causes disease". In fact MYCIN distinguishes several different types of causal relation.
- *World fact* rules capture common sense knowledge about the world, e.g. "if the patient is male, he is not pregnant"
- *Domain fact* rules capture domain specific knowledge on the basis of domain definitions, e.g. "if a drug was administered orally and it is poorly absorbed in the GI tract, then the drug was not administered adequately"

This distinction between strategy and justification proved to be a very potent methodological tool. The structural domain theory can be used to make the strategy explicit that 'indexes' the domain model (the knowledge base). In other words, instead of a hodgepodge of entangled rules, the different components of a production rule system can now be considered separately and relatively independently. All this by a shift from the dominant heuristic perspective to a more epistemological analysis of the types of knowledge involved in production systems.

2.3.3 Adequacy

The distinction between epistemological and heuristic adequacy of McCarthy and Hayes (1969) turned out to have a deeper impact than originally envisioned. Although it was primarily presented as the two main goals of AI, it soon turned into a rift in artificial intelligence research between epistemological–declarative (semantic networks) and heuristic–procedural (production system) systems. Because they were treated as very distinct notions, their pursuit has produced very different representation perspectives. But as we have seen, these are really two sides of the same coin.

Firstly, heuristic and epistemic approaches tend to deal with the same *kinds* of knowledge. Albeit in quite divergent ways. Frame descriptions are not purely 'epistemological'; surely the assignment of default slot values based on a 'match' is a form of inference and presupposes some procedure for feeding new information to a knowledge base. Both KRL and KL-ONE are in fact *hybrid* systems and combine declarative concept definitions with procedural attachments. Vice versa, production systems encode identification knowledge and knowledge of facts, the primary issue in declarative approaches, and are thus just as susceptible to the epistemological crisis.

Secondly, the two approaches interact; how can a representation mean anything without some form of standard inference? McCarthy and Hayes's representation is "in such a form that the solution of problems follows from the facts expressed in the representation" (p.5), but how to check whether a solution indeed follows from the representation? In other words, the inference mechanism in frame languages is just a particular heuristic strategy of the kind production systems are intended to capture.

Newell's analysis showed that *both* aspects should be present for *any* knowledge level representation as roles of behaviour on the one hand, and the composition of components on the other. Both Brachman and Clancey hold that KR languages should provide the basic elements that shape our knowledge.

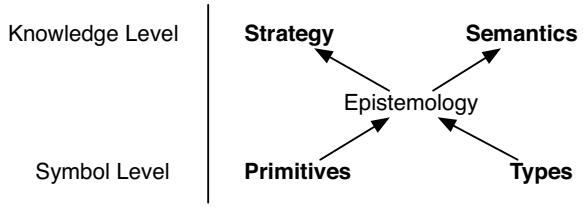

Figure 2.7: Levels of Knowledge Representation

Brachman emphasises the explicit definition of knowledge-structuring primitives. Clancey distinguishes *components* of a knowledge base, the interaction between different types of knowledge.

Furthermore, the separation of heuristics and epistemology is still important, but has been detached from the KR formalisms and has become rather an interaction between strategic and domain knowledge. A strategy 'indexes' domain facts, and the combination of the two constitutes a domain model.

Both Clancey and Brachman thus operate at the epistemological level, and argue that only knowledge representation at this level can account for knowledge level system behaviour. For Clancey the ultimate test is the *reuse* of a knowledge system for a different task (explanation vs. diagnosis: same domain, different strategy), for Brachman this was the built-in appropriateness of a language to represent a particular kind of (higher-level) knowledge.

More than a decade later, Davis et al. (1993) tried to settle the issue once and for all. A lot of the discussion in the field of knowledge engineering is, in their view, caused by a conflation of the different roles played by a knowledge representation. As we have seen, knowledge representation is first, and foremost, a *surrogate* for 'the thing itself' – a particular domain – used to enable reasoning as a simulation of the domain. As a representation cannot cover *all* of the domain, it needs to make a selection of relevant features of the domain and thus fixes a perspective on it. Every representation is therefore a set of *ontological commitments*, a commitment to the terms in which one should think about the domain (the domain and world facts of production systems). This commitment pertains not only to the contents of a particular model, but also to the KR language used. The ontological commitment accumulates in layers. For instance, a language for describing knowledge at the conceptual level commits to the existence of concepts, whereas a logical level language makes no such claims.

As we have seen in the preceding, a well designed representation language includes a standard inference mechanism. This makes a language a *fragmentary theory of intelligent reasoning*; it sanctions heuristic adequacy, and prescribes the way in which an AI system reasons on the basis of some adequately represented domain. Also, in its 'magical' role (Newell (1982), cf. Section 2.3.1) a KR language includes a commitment to a particular *way* of formulating problems that turns it into a *medium for pragmatically efficient computation*. The combination of language and representation is by itself a language which allows us to describe the world in a particular way; it is a *medium of human expression* (Stefik, 1986).

2.4 Components of a Knowledge Based System

As described in Section 2.3 the reusability of expert systems can be enhanced by separating the different types of knowledge in a model. In particular, this insight proved valuable in dealing with the problem first encountered by Wilkins and Leibniz in the 17th century: accumulating and representing all knowledge necessary for performing a particular task can be an arduous undertaking. Of the three schools discussed, the expert systems field was the one exposed to such large quantities knowledge, that this step of *knowledge acquisition* (KA) became regarded as worthy of study in its own right.

In early expert system development, models were built in an ad hoc, unprincipled and incremental manner. This lead to models such as that of MYCIN, in which expert knowledge was 'compiled' away, which made them very hard to understand for people other than the original designers. The models could not be reconstructed for other purposes than the one originally intended. Another problem was that this left no other means to check the correctness of a model, than by evaluating system behaviour as a whole. The knowledge acquisition field soon set out to develop a priori ways for ensuring expert system quality. Knowledge representation had become an engineering task, knowledge should be *modelled* rather than merely extracted from experts, and this process of knowledge engineering should be guided by a principled methodology. The methodology guarantees a level of quality by making design decisions explicit (see also Chapter 5).

2.4.1 Modelling Principles

In most approaches of the 1990's, knowledge engineering is regarded as a creative activity in which the construction of a knowledge base should be preceded by a modelling step. As in software engineering, the actual implementation of a KBS is guided by a *functional specification*. These approaches had a very strong methodological perspective, and covered every step in the construction of a KBS. From the identification of the purpose of a system and methods for knowledge elicitation and acquisition, to the specification and actual implementation of the system.

For instance, in the KADS methodology (Wielinga et al., 1992; van Heijst et al., 1997) knowledge acquisition is the construction of a knowledge level model (the *knowledge* or *conceptual* model).[5] Knowledge representation is then the implementation of this knowledge-level model in a knowledge base. This *design* model is a symbol level representation and takes into account additional considerations regarding e.g. computational efficiency of the system (See Figure 2.8). These considerations are recorded as *design decisions*. The PROTÉGÉ system of Puerta et al. (1992) adopted a similar approach where an initial knowledge level model was automatically translated into a CLIPS rule base.[6]

[5]Rather than a model consisting of 'concepts', the KADS conceptual model itself has concept status, it is a preliminary, abstract *version* (in another language) of the design model (a system component).

[6]CLIPS: C Language Integrated Production System, see `http://clipsrules.sourceforge.net`

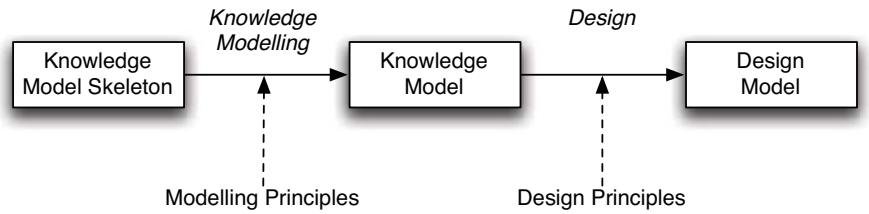

Figure 2.8: Steps in the knowledge engineering process (van Heijst et al., 1997)

Knowledge systems can be described at the knowledge level of Newell (1982), as long as the description is properly abstracted from the structural organisation of knowledge. In other words, the knowledge base (KB) of such a system should be described in terms of its *competencies*: "knowledge is to be characterised entirely *functionally*, in terms of what it does, not *structurally* in terms of physical objects with particular properties and relations" (p. 105). Levesque (1984) describes a functional perspective on KB characterisation. A KB can be treated as an *abstract data type*, where its behaviour is described in terms of a limited set of TELL and ASK methods. The capabilities of a KB are a function of the range of questions it can answer and assertions it can accept. How a KB implements this functionality should be hidden from the rest of the system. The advantage of this functional approach is that the knowledge engineer does not have to take into account considerations of implementation while constructing the knowledge model – at least to a large extent. As a result, the knowledge level model is less biased towards the implementation in a particular knowledge representation language.

Knowledge level models are more accessible to domain experts as they can be constructed and interpreted without in-depth understanding of technicalities. A knowledge level model can therefore help to reduce the *knowledge acquisition bottleneck* (Feigenbaum, 1980): the general difficulty to correctly extract relevant knowledge from an expert into a knowledge base.

Reuse of symbol level representations in the way discussed by Clancey (1983) turned out to be problematic because of the *interaction problem* (Bylander and Chandrasekaran, 1987). The problem that different types of knowledge in a knowledge base cannot be cleanly separated, because the purpose of the KBS, – translated into strategies – influences the way in which the domain theory is structured (recall Newell (1982) in Section 2.3). This is similar to the context dependency of the meaning of concepts. However, as knowledge models are formulated at a more abstract level – relatively untainted by issues of implementation and structure – it was thought that reuse of such models would be feasible. This "limited interaction hypothesis" (Wielinga et al., 1992) assumed that if domain knowledge was represented in a 'well structured and principled manner', general descriptions of methods should be possible. Unfortunately the structure of domain knowledge is often characteristic for a domain, and is not merely a matter of design (Valente et al., 1998).

The development of libraries of *skeletal* models, partially filled-in models of typical use, was thought to be the answer to the *hugeness* problem. Building a KBS requires a significant effort, not only because of the KA bottleneck, but

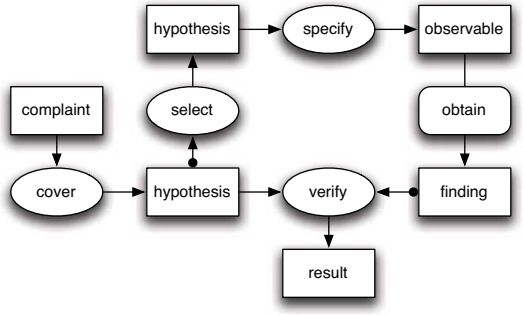

Figure 2.9: The standard *diagnosis* PSM, from Schreiber et al. (2000).

also because expertise domains in general are very large and complex. Skeletal models do not only help to reduce the amount of specification needed, but provide guidance with respect to the knowledge needed to build a KBS as well: they ensure coverage.

To enable knowledge level reuse, the roles played by elements in a model should be identified (Clancey, 1983; Bylander and Chandrasekaran, 1987). Approaches based on *role limiting* facilitate reuse by indexing symbol level representations – executable models – to types of knowledge at the knowledge level (Valente et al., 1998). These models are detailed blueprints for implementation in a system. Other approaches, such as KADS, rather took skeletal models to be sketches of models and provided no direct connection to implementation. The models in KADS supported reuse of 'understanding'.

Role limiting and knowledge typing are accomplished firstly by the separation between heuristics and epistemology.[7] For instance, KADS categorises elements as a type of *control* knowledge or as part of a *domain* theory (See also Figure 2.7). Control knowledge is the knowledge regarding how a system obtains it goals and solves problems. The *Common*KADS approach distinguished two types of control knowledge in expertise models (van Heijst et al., 1997; Valente et al., 1998; Schreiber et al., 2000):

Task Knowledge
is a abstract high level description of the decomposition of the goals within a KBS that must be achieved by problem solving.

Inference Knowledge
An inference is an primitive problem solving step which is solely defined in terms of its input/output signature. Its internal workings cannot be meaningfully expressed at the knowledge level even though they can perform very complex operations. Inferences can be combined in inference *structures*.

Tasks and inferences are combined in problem solving methods (PSM); these are descriptions of a particular way to perform some task: a PSM expresses a strategy, and reflects the "competence to decompose a task" (Valente et al.,

[7] Note that at the knowledge level, this does not correspond to a distinction between procedural vs. declarative: everything specified at the knowledge level is declarative.

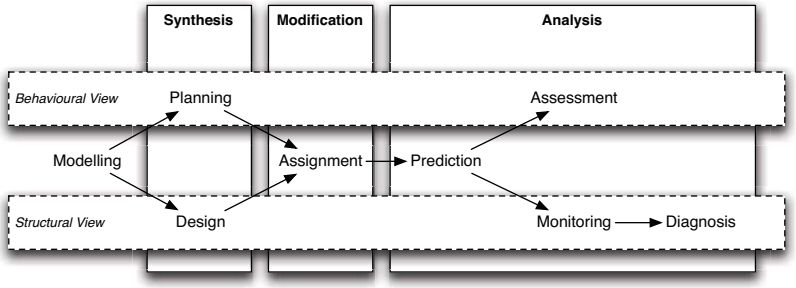

Figure 2.10: A suite of problem types, dependencies and views from Breuker (1994).

1998, p.400). See Figure 2.9 for an example problem solving method. Where role limiting approaches generally used a fixed task hierarchy, PSMs are orthogonal to such a hierarchy and can combine several tasks to perform a more complex one. In other words, if a task hierarchy divides a task into sub tasks, a PSM describes a particular path along those tasks in terms of an inference structure that implements them. Breuker and Van De Velde (1994); Breuker (1997) argue that a library of PSMs should be indexed by a suite of problem types, rather than by a taxonomy of tasks (Breuker, 1994).

Breuker's suite defines an intermediate level between task decompositions and problem solving methods. Central to this view are two standard sequences of problem types part of any task (Figure 2.10). The two sequences can be distinguished by a *behavioural view* and a *structural view*. The former incorporates the problems that deal with the interaction of a system with its environment: planning and assessment. The latter includes problems of design, monitoring and diagnosis and has a more internal perspective.

The interaction problem tells us that control knowledge and domain theory cannot be fully separated in the way assumed in the limited interaction thesis. On the other hand, a too close connection between the two kinds of knowledge severely limits reusability. A library of reusable knowledge components should therefore contain descriptions of domain knowledge: a PSM cannot be specified without at least some reference to domain knowledge. Firstly, a PSM expresses domain knowledge in terms of its role in a problem solving process. This captures the use of domain knowledge as an *epistemology* (see Section 5.5.2). Secondly, this epistemology is connected to its possible role fillers: the categories in the domain that are able to fulfil a role in problem solving. Reuse of PSMs is achieved by connecting categories in the epistemology to their counterparts in a generic domain theory. In short, the domain theory has two roles:

- To index problem solving methods for *reuse*, and in this way
- Communicates the *kinds* of things an expert system 'knows' about.

By the way it is constructed, an expert system cannot do otherwise than simply disregard anything it does not know about. In a way, the incorporation of a particular domain theory is therefore a significant *ontological commitment*,

in the sense of Davis et al. (1993). The domain theory expresses *that which persistently exists* for an expert system: it is an '*ontology*' (see Chapter 4).

2.5 Knowledge Representation

The views of KADS, *Common*KADS and the role limiting approaches did not just have impact on knowledge level specification (in the engineering sense) but also influenced the languages used for knowledge representation. While the psychological view and rationale of problem solving was abstracted to a meta-level of problem solving and reasoning strategies, knowledge based systems still require implementation at the symbol level. Even when this implementation is not intended to be *directly reusable*, as in role limiting, it needs to explicitly commit to a level of quality with respect to efficiency and computational properties.

In line with the functional paradigm of Levesque (1984), a knowledge base is in fact a *service* that can function as a black-box component of knowledge-based systems. As Levesque pointed out, to be reusable, such a component must be well described: it should guarantee answers to a specific set of tell/ask queries. This way, the different knowledge types of knowledge level specification can be instantiated as separate components of a knowledge based system *architecture*. This means that although *reasoning*, and problem solving in particular, is often a non-monotonic affair (McCarthy, 1980) where hypotheses are frequently asserted and retracted, *inference* does not have to be.[8]

Because KL-ONE like languages were already the most targeted to a specific type of knowledge, they gradually took on the singular role of expressing the domain theory of knowledge based systems.[9] This required the fine-tuning of standard inference over the *structural descriptions* that determine the position of a concept definition in a network: *classification*. At the same time, the logic approach first advocated by McCarthy and Hayes (1969) found its way back into mainstream knowledge engineering. Classification was specified as a form of *logical inference*. The functional specification advocated by Levesque was thereby underpinned by a logical formalism that specified the exact semantics of the representation language used by the knowledge component (Levesque and Brachman, 1985).

Levesque and Brachman (1987) pointed at a trade off between the expressive power and computational efficiency of representation formalisms. Inference on a highly expressive language will generally be inefficient, or even undecidable, while limitations on expressive power can ensure tractability. Relaxing either of these requirements would result in a system whose answers are not dependable. Its conclusions may not follow logically from the theory expressed by the knowledge base (soundness), or it does not give all possible answers (completeness). Levesque and Brachman (1987) therefore propose what

[8] For sure, in some applications a non-monotonic knowledge base can be beneficial for performance reasons as not the entire knowledge base needs to be re-evaluated whenever information is added or removed. However, non-monotonicity in general is no requirement as e.g. recent progress in decidable incremental reasoning over monotonic description logics shows (Parsia et al., 2006).

[9] Although logic programming languages such as Prolog were very popular for the representation of domain theories, their order dependence makes that a Prolog representation will always be slightly tainted by control knowledge.

Doyle and Patil (1991) call the *restricted language thesis*:

> "...general purpose knowledge representation systems should restrict their languages by omitting constructs which require non-polynomial (or otherwise unacceptably long) worst-case response times for correct classification of concepts"
>
> (Doyle and Patil, 1991, p.3)

However, as Doyle and Patil (1991) argue, this requirement cannot be posed in general for all representation languages used in knowledge based systems. In many applications, undecidable reasoning has no serious consequences and languages should therefore be evaluated on the basis of whether they "provide the rational or optimal conclusions rather than the logically sound conclusions" (Doyle and Patil, 1991, p.6). Although this is a fair point for knowledge based systems *as a whole*, it reintroduces the epistemological promiscuity of rationality at the implementation level, and thereby the interaction problem of Bylander and Chandrasekaran.

Successors of KL-ONE such as NIKL (Moser, 1983), KL-TWO (Vilain, 1984) and LOOM (MacGregor and Bates, 1987) and the Frame Language (Levesque and Brachman, 1987, FL), KANDOR (Patel-Schneider, 1984), KRYPTON (Brachman, 1983) and CLASSIC (Borgida et al., 1989; Brachman et al., 1991) were often still *hybrid* systems that combined terminological reasoning with procedural attachments. Once class membership or subsumption is established using *triggers* (Brachman et al., 1991), additional rules could fire that assigned default values to certain properties. Furthermore, Baader and Hollunder (1991) pointed out that these systems often implemented sound, but *incomplete* subsumption algorithms. The main reason was that sound *and* complete algorithms were only known for small and relatively inexpressive languages. Additionally, for many languages the subsumption problem was at least NP-hard, which meant that complete implementations would be intractable, while incomplete algorithms could be polynomial. Tractability is important for practical applications with often critical response time requirements.

2.5.1 Description Logics

The quest for more expressive but *decidable* combinations of language features became ever more prominent in the development of *description logics* (DL). These logics are often regarded as the direct successor of frame-based languages. However, where in frame-based systems descriptions (or descriptors) are secondary to the concepts they describe, in DL the descriptions themselves are first-class citizens. Current DLs have thus shifted away from concept-centered representation to description or *axiom* centred semantics. Concepts merely group together multiple descriptions to create some cognitively plausible aggregate. This paradigm shift is understandable, as frame-based systems, and early DL-systems, were not entirely 'clean' regarding the separation between concept and description. The $\mathcal{KRIS}$[10] system of Baader and Hollunder (1991) is one of the first sound and complete implementations of an expressive but decidable description logic.

[10] Knowledge Representation and Inference System

Description logics are fragments of first order logic (FOL) usually defined by means of a *model theoretic* semantics, i.e. its semantics is based on the interpretation of a language by means of set-theoretic structures. A sentence in such a language can only be made true or false given some *interpretation* of that sentence in terms of actual entities. For instance, the sentence "The person sits on the chair" can only be determined to be true given a mapping to a corresponding situation in reality, the *domain*, e.g. given a mapping from 'person' to an actual person, from 'chair' to an actual chair, and from 'sits' to an actual sitting relation between *that* person and *that* chair. An interpretation that makes the sentence true is said to be a *model* of that sentence. In terms of Brachman (1979) (cf. Figure 2.4), the interpretation of some sentence is the denotation relation between generic and individual concepts on the one hand, and individual objects and sets of objects in the domain on the other. See Figure 2.11 for a schematic depiction of the relation between concepts, individuals and the domain.

More formally, in DL a model consists of a domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$ that maps individual names to elements in the domain (individual objects), class names to subsets of the domain, and property names to binary relations on the domain. The vocabulary N of a DL knowledge base $\mathcal{K}$ correspondingly is a tuple $\{N_I, N_C, N_R\}$ consisting of a set of individual names N_I, a set of class names N_C, and a set of role names N_R. To give an example, applying the interpretation function to an individual $o \in N_I$ must result in a member of the domain: $o^{\mathcal{I}} \in \Delta^{\mathcal{I}}$. Similarly, the interpretation of a class $\mathsf{C} \in N_C$ is a subset of the domain ($\mathsf{C}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$), and the interpretation of a property $\mathsf{R} \in N_R$ is a subset of the set of all possible object pairs in the domain: $\mathsf{R}^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The meaning of axioms in a DL knowledge base is given by the relations between individuals, classes and properties in that knowledge base and corresponding constraints on models. For instance, if $\mathsf{A} \sqsubseteq \mathsf{B}$ (A is a subclass of B), then $\mathsf{A}^{\mathcal{I}} \subseteq \mathsf{B}^{\mathcal{I}}$ for all models of the knowledge base. In fact, this meaning does not depend on any meaning of objects in the domain, or on which objects make up the domain. Rather, the meaning of an axiom arises in its relation to the other axioms in a knowledge base: every DL knowledge base consists at least of the two classes top ($\top$) and bottom ($\bot$) which are defined as $\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$ and $\bot^{\mathcal{I}} = \emptyset$, respectively. The meaning of a DL knowledge base "derives from features and relations that are common to all possible models" (Horrocks et al., 2003).

The way in which the semantics of DL axioms is defined has several important implications. First, and foremost, every assertion in the knowledge base should be seen as a restriction on the possible models of that knowledge base. This means, conversely, that even though something might not have been explicitly stated, it may still be assumed to hold (i.e. be a model of the knowledge base). Description logics thus adopt the *open world assumption* which puts a rather counter-intuitive spin on negation: an axiom is only 'false' iff it has no model, i.e. that model is not a model of the knowledge base. Negation is therefore quite hard to enforce in DL.

A second implication is that there exists no direct link between a knowledge base and a domain. Although a knowledge base is defined in terms of the interpretation to the domain, determining adequacy of the knowledge base can only be determined internally, i.e. by 1) inferring that some axiom *can not* have a model (there cannot exist a (set of) objects that can make the axiom true)

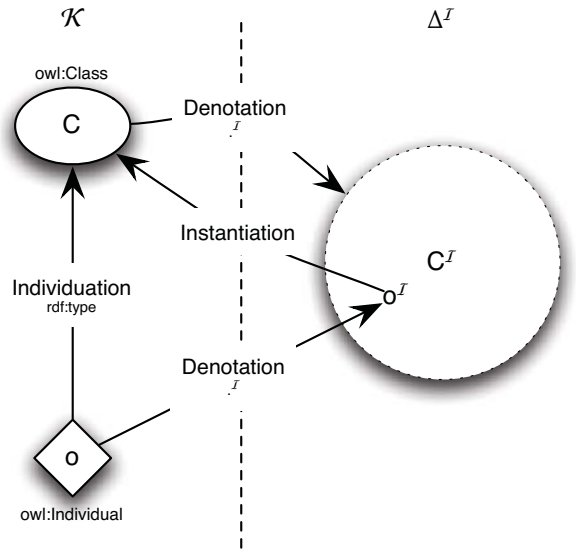

Figure 2.11: Instantiation, individuation and denotation in DL

or by 2) explicit assertions about individuals that relate directly to individual objects in the domain. Whether the latter relation holds or not is up to an observer external to the knowledge base. Related to this, is the fact that DLs do not necessarily adopt the *unique name assumption*, the assumption that an individual object is denoted by exactly one individual in the knowledge base: any number of individuals may share the same model.

As discussed earlier, the balance between computational complexity and decidability on the one hand, and expressiveness on the other plays a prominent role in description logics research (Baader et al., 2003). As a result, various description logics can be constructed out of several basic building blocks, for which the effect on combinations is well known.[11] These building blocks each have an associated letter, the combination of which gives the name of a particular description logic (see Table 2.4). For example, the DL $\mathcal{SHIQ}(d)$ is the attributive language with complex concept negation, transitive properties, inverse properties, qualified cardinality restrictions and datatype properties.

The emphasis on computational efficiency has furthermore lead to a distinction between axioms in the terminological, role and assertional boxes:

Terminological Box (TBox)
Contains class axioms, the generic concepts of Brachman (1979).

Role Box (RBox)
Contains descriptions of the characteristics of properties and the relations between them.

[11] See e.g. the Description Logics Complexity Navigator at `http://www.cs.man.ac.uk/~ezolin/dl/`.

Assertional Box (ABox)
Contains the assertions about individuals, the individual concepts of Brachman (1979).

Although the distinction between these boxes is not formally relevant, separating the various types of reasoning – classification for the TBox and realisation for the ABox – significantly improves reasoner performance (Levesque and Brachman, 1987). For instance, the use of class axioms such as *nominals*, that define a class by enumerating its individual members and thus cross the boundary between TBox and ABox, can have a remarkable effect on reasoning time (Klarman et al., 2008).

2.6 Discussion

As the issues raised in this chapter show, knowledge representation is not a trivial task and from its rise in the late sixties, the field underwent numerous changes. We discussed the computational crisis of uncontrolled logical inference in theorem proving, the debates on heuristic and epistemological adequacy, the status and psychological plausibility of knowledge representation languages, the epistemological promiscuity of expert system implementations and the rise of an *engineering* perspective on knowledge based systems development.

Although the three schools of knowledge representation (Section 2.2) embarked on distinct journeys, they each have a distinct role in this knowledge engineering view. The procedural perspective of production systems allows to express rational control in knowledge based reasoning, the semantic network paradigm grew into robust languages for expressing domain theories, and logic returned in full swing as means to define the semantics and guarantee correct inference of implicit knowledge expressed using these languages.

The most important lesson learned is certainly that though human experts may be very successful in applying highly heterogeneous knowledge, directly mimicking this human expertise has a serious detrimental effect on knowledge based systems development, maintenance and performance. Knowledge based systems should be *specified* at the knowledge level. This specification contains a description of the different types of knowledge involved in the system, and how they interact. The most prominent distinction can be made between the *domain theory* of a system – what it knows *about* – and the *control knowledge* that expresses the rationale of reasoning over the domain. Control knowledge consists of a decomposition of tasks and inference types, the performance of which can be specified by generic *problem solving methods*. Inference types are characterised by an input/output signature and can be implemented by separate knowledge *components* that have a matching *functional specification*. The different types of knowledge in an implemented knowledge based system operate at different levels: control knowledge is applied at a meta level with respect to the inferences performed by knowledge components.

The functional specification of knowledge system components is most vigourously adopted by *description logics* systems, the successors of KL-ONE like frame-based languages. Description logics are a family of decidable formal languages designed to express concepts, relations and their instances and are

thus naturally suited for expressing the domain theory of knowledge based systems. DL algorithms are optimised for calculating the inferred subsumption hierarchy, class satisfiability and class membership of instances (realisation).

Description logics research continued throughout the nineties, but played a relatively modest role until it entered a global stage when DL became the formalism of choice for knowledge representation on the *Semantic Web* (Berners-Lee, 1999). Arguably, the development of a web-based knowledge representation language is not trivial. Chapter 3 discusses the trade-offs and restrictions imposed on languages for the semantic web, and gives an overview of the resulting Web Ontology Language (Bechhofer et al., 2004, OWL).

Davis et al. (1993) characterised the domain theory of knowledge based systems as its *ontology*, a term that increased in popularity throughout the nineties. Although the exact meaning of this term is not directly relevant for the discussion of OWL as representation language in Chapter 3, ontologies do have a specific role to play in knowledge engineering. Chapter 4 discusses the various, often confusing, interpretations of what ontologies are, and gives a characterisation of an ontology as knowledge representation artefact.

Table 2.4: DL abbreviations

Abbreviation	Description
$\mathcal{AL}$	Attributive Language: • Atomic negation (negation of concepts that do not appear on the left hand side of axioms) • Concept intersection • Universal restrictions • Limited existential quantification (restrictions that only have fillers of owl:Thing)
$\mathcal{FL}^-$	A sub-language of $\mathcal{AL}$, without atomic negation
$\mathcal{FL}_o$	A sub-language of $\mathcal{FL}^-$, without limited existential quantification
$\mathcal{C}$	Complex concept negation
$\mathcal{S}$	$\mathcal{AL}$ and $\mathcal{C}$ with transitive properties. In $\mathcal{SROIQ}$: some additional features related to the RBox
$\mathcal{H}$	Role hierarchy (subproperties)
$\mathcal{R}$	Limited complex role inclusion axioms; reflexivity and irreflexivity; role disjointness.
$\mathcal{O}$	Nominals, i.e. enumerated classes and object value restrictions
$\mathcal{I}$	Inverse properties
$\mathcal{N}$	Cardinality restrictions
$\mathcal{Q}$	Qualified cardinality restrctions
$\mathcal{F}$	Functional properties
$\mathcal{E}$	Full existential quantification
$\mathcal{U}$	Concept union
(D)	Datatype properties, data values and datatypes

Source: http://en.wikipedia.org/wiki/Description_logic