Capability Representations for Brokering: A Survey

Gerhard Wickler
Dept. of AI, University of Edinburgh
80 South Bridge, Edinburgh, Scotland
gw@dai.ed.ac.uk

Austin Tate
AIAI, University of Edinburgh
80 South Bridge, Edinburgh, Scotland
a.tate@ed.ac.uk

Abstract

In this article we review knowledge representation formalisms that lend themselves to the representation of capabilities of intelligent agents. The aim of representing capabilities is, of course, that we want to reason about them. The reasoning task we are most interested in is capability brokering, i.e. the task of finding an agent which has a capability that can be used to address a given problem. Thus, the first area we review here is agent cooperation and communication from which the problem originates. However, the capability representations found here often lack sophistication and are rather ill-defined. Thus, we have turned to logics next, which include some of the best understood representations in AI and beyond. These representations are rather generic though. Capabilities are essentially the actions an agent can perform and thus, we review action representations next. While these representations are very promising, they lack the expressiveness required to represent certain more complex capabilities. Models of problem solving address this issue for complex reasoning capabilities, and this is the final area we review in this paper.

1 The Problem of Capability Brokering

One approach to achieving artificial intelligence is the rational agent approach [Russell and Norvig, 1995, page 7]. In this approach, AI is viewed as the study and construction of rational agents. An agent is described as an entity that perceives and acts. Rationality means that it acts as to achieve its goals, given its beliefs. More precisely, [Wooldridge and Jennings, 1995, page 116] identify four properties an agent should have: autonomy, social ability, reactivity, and pro-activeness. Pro-activeness means that an agent should be able to exhibit goal-directed behaviour by taking the initiative. Pro-activeness is directly related to rationality. Social ability, the property we will be most concerned with, means that an agent interacts with other agents (possibly humans) via some kind of agent communication language. Together, pro-activeness and social ability imply that an agent should communicate not with just any other agent but with those agents that can help it achieve its goals. But first, it has to find these agents.

This problem is very similar to what [Davis and Smith, 1983, page 76] call the connection problem in distributed problem solving. One assumption they are making is that the agents that exist are fixed. We will assume here that an agent exists in a dynamic environment with other agents. As the environment changes new agents might come into existence or existing agents might disappear. Agent autonomy means that an agent has to operate without the direct intervention of humans, i.e. that it has to find out by itself about existing other agents, specifically, agents that can help it achieve its goals. [Genesereth and Ketchpel, 1994, page 51] distinguish two basic approaches to the connection problem: direct communication, in which agents handle their own coordination and assisted coordination, in which agents rely on special system programs to achieve coordination.

The best known work in AI on agent communication is probably the Knowledge-Sharing Effort [Fikes *et al.*, 1991, Neches *et al.*, 1991]. Part of this effort is the development of the Knowledge Query and Manipulation Language (KQML), a high-level agent communication language [Finin *et al.*, 1993, Finin *et al.*, 1997, Labrou and Finin, 1997]. KQML, like most approaches to the connection problem, advocates assisted coordination through facilitators and mediators. However, the support offered by KQML for this task is still an active research issue, especially for more complex agents [Kuokka and Harada, 1995b].

[Decker *et al.*, 1997] have recently described a solution space to the connection problem that identifies nine different types of middle-agents depending on which agents know about capabilities and preferences of agents initially. By a preference they mean meta-knowledge about what types of information have utility for a requester. In a solution to the connection problem in which capabilities are initially known to the provider and the middle-agent only, and in which preferences are initially known to the requester and the middle-agent only, the middle-agent is what they call a broker. Capability brokering and, more specifically, representations for capabilities that facilitate brokering are what we shall review in this article.

2 Software Agents

In this section we will look at work in the wider area of intelligent software agents and, more specifically, at approaches to capability brokering found there.

An overview of this section, which provides a conceptualisation of the relationships between the different sub-fields and approaches/systems described in this section, is given in figure 1. The figure also contains cross-links to other areas and highlights the work which was considered to be most important for representing and reasoning about capabilities.

2.1 Distributed AI

Intelligent software agents are often seen as part of a wider area of *Distributed Artificial Intelligence* (DAI) [Bond and Gasser, 1988, Chaib-Draa *et al.*, 1992,

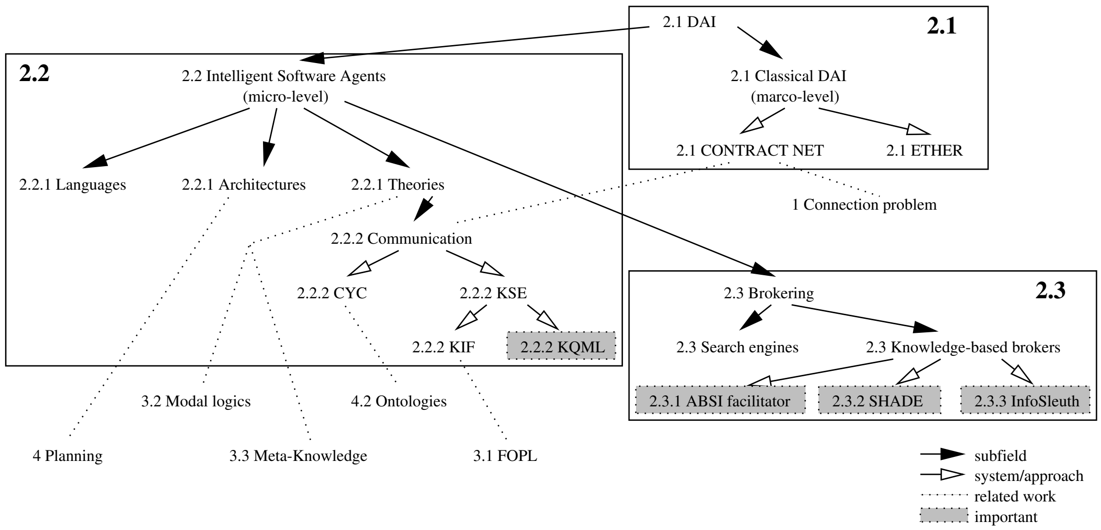

Figure 1: Overview of this section

Jennings, 1996] which motivates us to briefly consider this area first. DAI is the subfield of AI that is interested in concurrency in AI computations. Its main concerns have been *distributed problem solving*, i.e. how the task of solving a particular problem can be divided amongst a number of available problem solvers, and *multi-agent systems*, i.e. how a collection of autonomous intelligent agents can coordinate their knowledge, goals, skills, and plans jointly to take action or to solve problems.

DAI has not been very concerned with the problem of capability brokering. As pointed out in [Wooldridge and Jennings, 1995, page 142], the classical emphasis in DAI has mostly been on the *macro-level*, i.e. on social phenomena and the emergent behaviour of a group of problem solvers. Research in intelligent software agents emphasises the *micro-level*, i.e. the architecture and theories of individual agents. The latter is where the problem of capability brokering has been addressed previously and at which we will look in section 2.2.

Two architectures that grew out of DAI are worth mentioning here. Firstly, there is the CONTRACT NET [Smith, 1977, Davis and Smith, 1983]. In the CONTRACT NET architecture a given problem is first decomposed into sub-problems. These sub-problems are treated as contracts and a process consisting of contract announcement, bidding, and contract awarding is used to distribute problem solving. This process of negotiation, i.e. the extensive and explicit use of communication to distribute the problem (cf. section 2.2.2), was an important contribution of this work. Another contribution was the definition of the connection problem [Davis and Smith, 1983, page 76] which is essentially the problem of capability brokering.

Secondly, there is ETHER [Kornfeld, 1979, Kornfeld, 1981], a pattern-directed invocation formalism for parallel problem solving. ETHER provides a PLANNER-like language where procedure invocation is driven by pattern matching. Unlike in previous approaches, control over the distribution is not in the hands of the user. Instead, the patterns are used to distribute the problem-solving process. The basic mechanism for the distribution is by broadcasting of patterns. Nowadays virtually all DAI systems use patterns to distribute the problem-solving process, but the specifics of ETHER are not used anymore which is why we will not look at it any further.

2.2 Intelligent Software Agents

Intelligent software agents have recently received a lot of attention within AI [Russell and Norvig, 1995, Bradshaw, 1997, Huhns and Singh, 1998]. However, the definition of *agent* is elusive, i.e. there is still considerable lack of consensus on what exactly an agent is or what the research questions are that need to be addressed. An overview of possible definitions of agency and a comprehensive structuring of the field has been presented in [Wooldridge and Jennings, 1995] and we shall mostly adopt their approach. They distinguish agent theories, agent architectures, and agent languages as the three major subfields of agent research.

2.2.1 Languages Architectures and Theories

Firstly, the subfield of *agent languages* is mainly concerned with tools that allow one to program hardware or software computer systems using the concepts developed in agent theories as outlined below. Such tools include, for example, the Agent Behaviour Language [Wavish, 1992], the agent-oriented programming paradigm [Shoham, 1993], Concurrent MetateM [Fisher, 1994], or the Java Agent Template.[1] As this area is not concerned with the representation of and reasoning about capabilities, we shall not dwell on it here.

Secondly, the subfield of *agent architectures* is concerned with issues surrounding the construction of computer systems that satisfy the properties specified by agent theories (below). The classical approach in AI is the deliberative architecture based on the physical symbol system hypothesis, i.e. an architecture that contains an explicitly represented, symbolic model of the world [Newell and Simon, 1976, Russell and Norvig, 1995]. The main alternative to the deliberative approach is the reactive approach based on the so-called subsumption architecture [Brooks, 1986, Brooks, 1991]. Finally, a number of hybrid approaches to agent architectures have also been attempted. However, none of these architectures explicitly supports capability brokering. Since deliberative agents will need to take well-planned actions it is often assumed that such an agent should be based on AI planning technology [Wooldridge and Jennings, 1995, page 131]. We will look at AI planning more closely in section 4 and at existing agents using a planner specifically in section 4.4.

Finally, formal *agent theories* are essentially specifications for agents where an agent is described as an intentional system that has beliefs, desires, etc. [Seel, 1989]. Agent theories can be seen as representational frameworks for such attitudes. The dominant approaches are based on modal logics (cf. section 3.2) and meta-languages (cf. section 3.3). The former lead to the adoption of the possible worlds semantics which has been used to define what it means for an agent to know something and to reason about knowledge and belief [Hintikka, 1962, Kripke, 1963]. Various alternatives were also developed to avoid the problem of logical omniscience [Levesque, 1984, Konolige, 1986].

[1] JAT is available on the WWW at URL: `http://cdr.stanford.edu/ABE/JavaAgent.html`

Similarly, but to a lesser extent, there have been logics of goals or desires [Cohen and Levesque, 1990, Wooldridge, 1994]. Although these approaches have addressed many attitudes of agents, there remains the problem of integrating them into one framework for an all-embracing agent theory. The issue in agent theories we shall be most concerned with here is that of *agent communication* which also addresses the connection problem.

2.2.2 Agent Communication

At least two major efforts are currently under way which both assume *knowledge sharing* to be the key to successful agent communication and cooperation.

The first effort addressing the agent communication problem is the CYC project [Guha and Lenat, 1990, Guha and Lenat, 1994, Lenat, 1995]. The basic idea here is that agents need to have a large amount of commonsense knowledge before they can intelligently work together. Since the CYC researchers believe that commonsense knowledge can not be learned automatically without having a large body of it in the first place, most of the work in CYC has been on hand coding such knowledge and on developing large ontologies using micro-theories. We shall return to the issue of ontologies in section 4.2.

The second major effort addressing the agent communication problem is the *ARPA Knowledge Sharing Effort* [Fikes *et al.*, 1991, Neches *et al.*, 1991, Genesereth and Ketchpel, 1994]. They envisage a generic agent communication language as consisting of three parts: the vocabulary, the inner language which carries the content that is being communicated, and the outer language which represents mainly the speech act that this message represents. The vocabulary is to be defined within one or more ontologies that will be shared by the communicating agents [Gruber, 1993b, Gruber, 1993a, Farquahar *et al.*, 1996]. Again, we shall return to the issue of ontologies in section 4.2. A generic knowledge representation language called KIF [Genesereth, 1991, Genesereth *et al.*, 1992] to and from which all other content languages should be translatable has been suggested for the content to be communicated, including the content of messages about capabilities (cf. section 2.3).

2.2.3 The Knowledge Query and Manipulation Language

Research on the outer language mentioned above has resulted in the definition of the *Knowledge Query and Manipulation Language* (KQML) [Finin *et al.*, 1992, Finin *et al.*, 1997, Labrou and Finin, 1997]. We shall describe KQML briefly here.

The syntax of KQML is simply based on a balanced parenthesis list. The first element in this list represents the *performative* of this message[2]. The performative indicates the type of speech act this message is. For example, the performative `ask` indicates a question being asked and the performative `tell` indicates a statement being made. For each performative in KQML there is also a protocol that defines with which type of messages other agents should reply to this message if any. For example, there should always be a reply to an `ask`-message and the performative of this reply message should be `tell`. Although there is a set of predefined performatives in KQML it is not meant to be binding. Agents may choose to use this set or invent their own performatives. They may also choose not to implement certain predefined performatives. However, if a predefined performative is used it should be used with the protocol for this performative defined in the KQML specification.

The performative is followed by a number of *keyword-value pairs*. Again, there is a number of predefined keywords like `:sender` or `:content` that all have a fairly obvious meaning. For example, the value following the keyword `:sender` should be the name of the agent sending this message and the value following the keyword `:content` should be the actual content of the message. The content of a KQML message is meant to be opaque to the message. However, in interpreting a KQML message it is necessary to decide where the content ends and thus, it is necessary to look at the content at least for this. There are also a number of

[2] In the literature on KQML and speech acts the term *performative* is sometimes also used to refer to the whole message.

fairly obvious constraints between the different parts of a KQML message. For example, if the language field names a specific content language then the content should be in this language.

KQML, like most approaches to the connection problem, advocates assisted coordination through agents called *facilitators* and *mediators*. A facilitator in KQML is an agent that performs various useful communication services. One of the main services offered by a facilitator is to help other agents find appropriate clients and servers. How client and server agents can find the facilitators is a problem to which KQML does not prescribe a solution. Neither is the mechanism to be used by the facilitators to find appropriate servers for clients specified in KQML. However, there are a number of related performatives and protocols for these performatives that can be seen as the definition of an interface to the facilitators. This interface definition is one of the most important contributions of this work as far as capability brokering is concerned. Some of the most important performatives for brokering in KQML are described in table 1.

One issue worth noting at this point is that KQML requires the content of an advertisement to be identical to the content of the capability-seeking message. This is very restrictive and, as we shall see, most existing brokers ignore this part of the KQML specification and provide a more sophisticated matching service.

2.3 Brokering Agents

Returning to the connection problem, which is the main problem of capability brokering, [Genesereth and Ketchpel, 1994] distinguish two basic approaches to this problem: *direct communication*, in which agents handle their own coordination and *assisted coordination*, in which agents rely on special system programs to achieve coordination. ETHER and the CONTRACT NET, both described in section 2.1, fall into the first category. The facilitation approach defined in KQML falls into the second category and this is currently the dominating approach in intelligent agent research.

- **advertise**: With this performative the sender informs the receiver (which should be the facilitator) that the sender is willing and able to process certain messages. KQML specifies that the processable message being advertised is given as the content of this message, i.e. the content is a KQML message again. Furthermore, the performative of the content message should be one of a limited set and there are certain basic constraints on the sender and receiver of the advertisement and embedded message. No reply message is required.

- **subscribe**: With this performative the sender informs the receiver (which should be the facilitator) that it wants to be updated every time that the would-be response to the content message is different from the last response to the sender of this message. Thus, like for `advertise` the content must be a KQML message and similar constraints apply. In response, the facilitator should send one reply to the embedded message immediately and further messages as they occur.

- **recommend-one**: With this performative the sender informs the receiver (which should be the facilitator) that it wants to know about one agent that has advertised that it will process the message given as the content of this message. The expected reply to this message is a message with the performative `forward`, the content of which should be an advertising message. The content of this `recommend-one` message and the `advertise` message should be identical.

- **recommend-all**: This performative is like `recommend-one`, only that the reply should name all the agents that have advertised to process the given content message.

- **broker-one**: Again, this performative is like `recommend-one` in its form. The difference is that with this performative the sender asks the facilitator to find an agent that can process the given message and then send it the given message. If there will be a reply to this message, this reply should be forwarded to the sender of the `broker-one` message.

- **recruit-one**: Again, this performative is like `recommend-one` in its form. The difference is that with this performative the sender asks the facilitator to find an agent that can process the given message and then send it the given message. The difference to `broker-one` is that any reply should go directly to the sender of the `recruit-one` message rather than through the facilitator.

Table 1: KQML facilitation performatives [Labrou and Finin, 1997]

Before we turn to a survey of existing brokers that facilitate assisted coordination, it is also worth noting that a kind of brokering has been performed on the Internet for some time now by *search engines* [Witten *et al.*, 1994, Howe and Dreilinger, 1997]. However, most search engines match requests, usually only consisting of a few keywords, to text pages on the Internet. The matching is essentially based on a reverse word frequency count algorithm[3] and can hardly be called knowledge-based. This is not the kind of brokering we are interested in here.

Various terms have been used for the special system programs on which assisted coordination relies, some of which we have used in this review, e.g. facilitator or broker. [Decker *et al.*, 1997] have recently suggested a categorisation of what they call *middle-agents*. They use the term middle-agent to mean any special system program used in assisted coordination. They distinguish different kinds of middle-agents according to where preference and capability knowledge resides. Preferences are meta-knowledge about what types of information have utility for a requester and capabilities are meta-knowledge about what types of requests can be serviced by a provider. The table summarising their categorisation is repeated here in table 2. According to this categorisation, in a scenario in which the capabilities of problem-solving agents are initially only known to the provider and the middle-agent and the problem of the problem-holding agent are initially only known to the requester and the middle-agent, the middle-agent is called a *broker*.

Brokers are the kind of middle-agent we are most interested in looking at in this article. We shall now briefly review some brokers that use explicit representations as the basis for brokering.

[3] Actually, there are also other techniques which are being used in search engines, but none of them is based on what can be considered an understanding of the retrieved document.

preferences initially known by	*capabilities initially known by*		
	provider only	provider + middle agent	provider + middle + requester
requester only	(broadcaster)	"front-agent"	matchmaker/ yellow-pages
requester + middle agent	anonymizer	broker	recommender
requester + middle + provider	blackboard	introducer/ bodyguard	arbitrator

Table 2: Middle-agent roles; from [Decker *et al.*, 1997, page 579]

2.3.1 The ABSI Facilitator

One of the earliest middle-agents that can be considered to be a broker in the above sense is the Agent-Based Software Interaction (ABSI) *facilitator* [Singh, 1993a, Singh, 1993b]. This broker is meant to be used in a system of agents operating in the ABSI architecture [Genesereth and Singh, 1993] and is based on a variant of an early specification of KQML [Finin *et al.*, 1993]. For the facilitator to perform its brokering service, agents must first notify the facilitator of the KQML messages they can process, i.e. they must advertise their capabilities. One important restriction imposed by the ABSI facilitator is that agents must not advertise that they can handle a message which they might subsequently fail to process.

The ABSI facilitator provides performatives for package forwarding, information monitoring, and content-based routing. Content-based routing is basically what we have called capability brokering. Of the KQML brokering performatives described in table 1, the ABSI facilitator essentially supports `advertise` and `broker-one`. Capabilities are represented by KQML messages as defined in the KQML specification. The content of these capability-representing KQML messages must be in KIF. For capability retrieval, the content of a capability-seeking message and the capability advertisement need not be identical for them to match, as the KQML specification would require. Instead a kind of unification defined by meta-descriptions in the KIF manual [Genesereth *et al.*, 1992] is used to match capabilities and preferences. Additionally, a Prolog-based inference engine can be used to evaluate additional conditions on the matched meta-variables.

2.3.2 SHADE and COINS

Two other brokers based on the KQML protocol are the SHADE and COINS matchmakers [Kuokka and Harada, 1995a, Kuokka and Harada, 1995b]. These brokers are implemented entirely as a declarative rule-based program within the MAX forward-chaining agent architecture [Kuokka, 1990]. As opposed to the ABSI facilitator, it is assumed that SHADE and COINS will make false positive and false negative matches. Thus, part of the work on these brokers was on addressing the problem of recovery after such a false match.

Both, SHADE and COINS, support the full range of KQML performatives described in table 1 and more. The difference between SHADE and COINS lies in the capability representations they can handle. In both cases, capabilities are represented as KQML messages, but SHADE works over a formal, logic-based content language and COINS operates over free-text information. Thus, COINS is effectively what we have called a search engine above. SHADE expects either KIF [Genesereth *et al.*, 1992] or MAX [Kuokka, 1990] augmented to support string patterns as terms for its content language. MAX is more appropriate for representing highly structured data such as objects or frames. The actual matching of capabilities and preferences is performed by a Prolog-like unification algorithm. Advertisements and requests must match based solely on their content; there is no knowledge base against which inference is performed. Limited inference for future versions is envisaged though.

2.3.3 InfoSleuth

The aim of the *InfoSleuth* project [Bayardo *et al.*, 1997, Nodine and Unruh, 1997, Nodine *et al.*, 1998] is to develop technologies that operate on heterogeneous information sources in an open, dynamic environment. To achieve this flexibility and openness, InfoSleuth integrates agent technology, ontologies, information brokerage, and Internet computing. InfoSleuth's architecture is comprised of a network of cooperating agents communicating in KQML. One of these agents is the broker agent which receives and stores capability advertisements from all other InfoSleuth agents. The task of the broker agent is to provide a semantic match-making service that pairs agents seeking a particular service with agents that can perform that service.

Minimally, every agent must advertise to the broker its location, name, and the language it speaks. Queries must be in KQML using KIF as the content language and "InfoSleuth" as the ontology. Matching is performed as an intersection function between the user query and the data resource constraints in the capability advertisement. The way this works is that KIF sentences, that are the content of capability advertisements and user queries, are translated into the deductive database language LDL++ [Zaniolo, 1991] and maintained in such a database.

Most Important Issues Here

- The work on the CONTRACT NET described in section 2.1 gave us the connection problem which is basically the problem addressed in this thesis.
- The inter-agent communication language KQML described in section 2.2.3 is probably the most advanced language for this purpose.
- The KQML-based brokers described in section 2.3 perform essentially the same task we are most interested in.

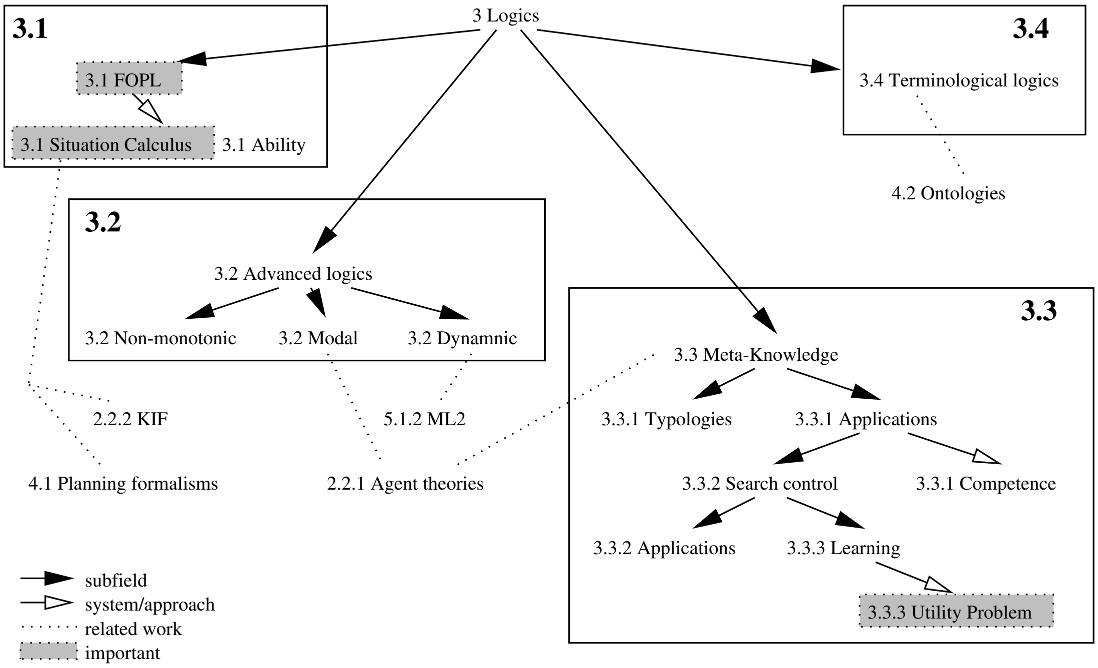

Figure 2: Overview of this section

3 Modelling Capabilities with Logics

In this section we will look at how some logics have been or could be used to represent the capabilities of intelligent agents.

An overview of this section, which provides a conceptualisation of the relationships between the different sub-fields and approaches/systems described in this section, is given in figure 2.

3.1 First-Order Predicate Logic

A generic knowledge representation formalism such as *first-order predicate logic* (FOPL) [Chang and Lee, 1973, Loveland, 1978, Gallier, 1986] might well have turned out to be sufficient for representing and reasoning about capabilities. Advantages of FOPL include its well-defined semantics and the fact that it is probably the best-researched knowledge representation formalism in AI and beyond. This is certainly good enough a reason to begin our review of logics as capability representations using FOPL.

Representations of capabilities in FOPL have been attempted in early approaches to reasoning about actions, e.g. [Green, 1969]. One of these early approaches is the *situation calculus* [McCarthy and Hayes, 1969, Shanahan, 1997] which has actually been an active topic of AI research for more than three decades now. However, its main concern has not been with reasoning about capabilities but with reasoning about actions in general, which can be seen as a much broader area than reasoning about capabilities for brokering.

Very briefly, the ontology of the situation calculus is made up of *situations* which can be thought of as snapshots of some world; *fluents*, which take on different values in different situations and can be thought of as time-varying properties; and *actions*, which can be executed to change the value of fluents. The atomic formula $Holds(f, s)$ is used to denote that the fluent f is true in situation s. Note that the fluent f, although it might look like an atomic formula, is a term, i.e. it represents an object in the domain represented. The function term $Result(a, s)$ is used to denote the situation obtained by executing the action a in the situation s. Sentences in first-order logic called effect axioms can now be written to represent the effects and preconditions of actions.

Unfortunately the effect axioms alone turn out to be epistemologically inadequate and further so-called frame axioms are needed in the representation, leading to the *frame problem* in AI [Hayes, 1974, page 69], [Shanahan, 1997]. Furthermore, the representation of fluents as objects in the domain seems counter-intuitive. In summary, the strong point of the situation calculus has traditionally been the theoretical framework it provides for the representation of actions based on a well-defined semantics.[4] A number of more direct action representations which also address the frame problem have been proposed in AI and we shall review some of them in section 4.1. Still, the situation calculus remains a highly expressive action representation with probably the clearest semantics of any such representation.

[4] Recent work described in [Gruninger and Fox, 1994, Gruninger *et al.*, 1997] addresses some practical aspects of reasoning with a formal situation calculus.

McCarthy and Hayes' original work was not limited to the representation of and reasoning about actions and their effects, but also included the general concept of *ability* [McCarthy and Hayes, 1969, pages 470–477]. In this work, they have attempted to formalise what it means for an agent to be able to do something by defining a predicate $can(p, \pi, s)$ meaning "agent[5] p can bring about the condition π in situation s." The interesting result here is, as they point out, that it is not at all clear what this proposition means. However, although highly relevant for the epistemological underpinnings of our work, we shall not go into the philosophical problems entailed here.

3.2 Advanced Logics

Since we have mentioned the situation calculus and the frame problem, it is also worth noting that there is a group of logics that have been mainly developed to address this problem. These logics are referred to as *nonmonotonic logics* [Ginsberg, 1987, Brewka, 1991], [Davis, 1990, section 3.1]. The classic approaches here are Default Logic [Reiter, 1980] and Circumscription [McCarthy, 1980b, McCarthy, 1980a]. However, as these approaches address the problem by changing the inference mechanism rather than fundamentally changing the representation, they are of little interest to us and we shall not look at them further here.

Many approaches to agent theories (cf. section 2.2.1) are based on *modal logics* [Chellas, 1980, Chagrov and Zakharyaschev, 1997], [Davis, 1990, section 2.7] and the possible worlds semantics [Hintikka, 1962, Kripke, 1963] and thus, we shall have a look at these logics next. Agent theories are specifications of agents. These specifications can be used by agents to reason about other agents. Our aim is to reason about the capabilities of other agents.

A modal logic augments a calculus, e.g. predicate calculus, with a number of operators, called *modal operators*, that take sentential arguments. Modal oper-

[5] They are looking at a world of interacting discrete finite automata for which we will use the term agent here.

ators are usually non-extensional, i.e. they do not commute with quantifiers, or are referentially opaque. The semantics of a modal language is based on Kripke structures which consist of a collection of possible worlds, connected by an accessibility relation. We say a possible world $\mathcal{W}_1$ is accessible from a possible world $\mathcal{W}_2$ in a Kripke structure if they are connected by the accessibility relation. In each possible world, a sentence in modal logic can be either true or false, i.e. a sentence may have different truth values in different possible worlds.

For example, in propositional modal logic the truth values of propositions can vary across different possible worlds. Propositions can be connected with the usual connectives (e.g. negation, conjunction, disjunction) to form more complex sentences. The only syntactical extension is the introduction of usually two new, dual types of sentences: $\Box A$ (*necessarily* A) and $\Diamond A$ (*possibly* A), where A is again a sentence in modal logic. The informal semantics is that $\Box A$ is true in a possible world $\mathcal{W}$ if and only if A is true in every possible world accessible from $\mathcal{W}$ and that $\Diamond A$ is true in a possible world $\mathcal{W}$ if and only if A is true in at least one possible world accessible from $\mathcal{W}$. Other modal operators may also be defined.

Modal logics have been used in agent theories mostly to reason about the knowledge of other agents [Wooldridge and Jennings, 1995, section 2]. This is done by interpreting $\Box A$ as a modal knowledge operator, i.e. an agent *knows* A if in every world that is consistent with its knowledge, A is true. Reasoning about knowledge using modal logics was probably first comprehensively integrated into a framework for reasoning about actions by [Moore, 1985].

A further extension of modal logics are *dynamic logics* [Harel *et al.*, 1982, Harel, 1984]. Dynamic logics were developed to reason about programs and their executions. Syntactically, the only change from normal modal logic is that $\Box A$ is replaced by $[\alpha]A$ and $\Diamond A$ is replaced by $\langle\alpha\rangle A$, where α is a program. A program implicitly defines an accessibility relation, i.e. only those worlds are accessible that are possible states after the execution of the program. $[\alpha]A$ is then defined as true in $\mathcal{W}$ if and only if A is true in every possible world accessible from $\mathcal{W}$ with the accessibility relation defined by α, i.e. if A is necessarily true after the execution of α.

Note that dynamic logics are the first logics introduced here that explicitly include capabilities in the form of programs in their ontology. However, representing knowledge in and automated reasoning over dynamic logics has proven not very practical and thus, we shall not pursue this path any further.

3.3 Meta-Level Knowledge

Experiments in [Larkin *et al.*, 1980, Chi *et al.*, 1981], and other work described in [Barr, 1979, Andrews, 1981], have shown that experts in a field often do not have more domain knowledge than novices, but instead they use this knowledge more efficiently; they have more *meta-knowledge*. Being an expert in a domain means to be more competent in this domain or, to be more capable of solving problems in this domain. Thus, there is a strong correlation between the availability of meta-knowledge and capability or competence in a domain. Similar arguments can be found in [Laske, 1986, Lecoeuche *et al.*, 1996, VanLehn and Jones, 1991]. We have argued in [Wickler and Pryor, 1996] that available meta-knowledge can be re-used for competence assessment. The emphasis here is on the re-use aspect which would make this approach very attractive to our aims as it could save us a lot of work. Thus, we shall now look at meta-knowledge and its representations to see whether this knowledge can be re-used for capability brokering.

3.3.1 Types of Meta-Level Knowledge

A number of *classifications of meta-level knowledge* have been attempted. For example, an early classification by [Davis and Buchanan, 1977] distinguishes schemata for reasoning about objects, function templates for reasoning about functions, rule models for reasoning about inference rules, and meta-rules for reasoning about strategies. In [Lenat *et al.*, 1983] there is not so much a categorisation of meta-knowledge, but instead they give a number of examples where such knowledge is being used. These examples include meta-knowledge: for rule selection; to record needed facts about knowledge; for rule justifications; to detect bugs; etc. The last example they give concerns meta-knowledge to describe a program's abilities. Unfortunately they do not describe a representation for this type of meta-knowledge. Similarly, [Maes, 1986] argues that meta-level knowledge is needed for introspection and classifies it by the tasks it is needed for, e.g. in assumption-based reasoning, in learning, in handling inconsistent, incomplete, and uncertain knowledge etc. This shows that there is a need for explicit meta-knowledge in knowledge-based systems.

There are also a number of *examples of systems* that have used explicit meta-knowledge for a number of purposes. For example, [Filman *et al.*, 1983] describe several experiments using meta-language and meta-reasoning to solve problems involving belief, heuristics, and points of view; [Attardi and Simi, 1984] describe a meta-language for reasoning about logical consequence; [Ginsberg, 1986] describes a meta-level framework for the construction of knowledge base refinement systems; [Haggith, 1995] describes a framework for reasoning about conflicts in knowledge bases. Many other systems using explicit meta-knowledge do exist (cf. [Maes and Nardi, 1988]). This illustrates the availability of meta-knowledge in knowledge-based systems.

Of particular interest to us is work on using meta-knowledge for *competence assessment* [Voß *et al.*, 1990] as this is directly related to capability retrieval. One of their aims was to develop a system that knows when it cannot solve a given problem without having to fail in an attempt to solve it. For this task they extended their problem solver with a number of reflective modules that performed some simple tests. The representation of knowledge in the reflective modules is procedural rather than declarative though, and the modules work for configuration tasks only. Furthermore, competence assessment was internal to the developed system and no external broker-like agent could perform the competence assessment.

Up to this point, there have been few approaches which use meta-knowledge to represent capabilities and certainly no solution that could be used for capability brokering, as we had hoped.

3.3.2 Search Control Knowledge

Although many different uses for meta-level knowledge have been suggested there has been one area where the use of meta-knowledge has had the largest impact: *search control* [Davis, 1980, Bundy and Welham, 1981, Georgeff, 1982]. The idea here is that spending some time on where one is going to search in a large search space is more efficient than just searching. As mentioned above, experts in a domain often distinguish themselves from novices not by having more relevant domain knowledge, but by using it more efficiently. This suggests that the kind of meta-knowledge that is strongly correlated to capability knowledge as we need to represent it is, in fact, search control knowledge. Thus, we shall now look at search control knowledge to see whether this knowledge can be re-used for capability brokering.

There are a number of *domains* for which search control knowledge has been found and employed. For example, [Bundy *et al.*, 1979] describe a system that uses meta-level inference to solve mechanics problems; [Wilkins, 1982] uses meta-knowledge to control search in chess; [Minton *et al.*, 1985] have used explicit search control knowledge in parsing; [Murray and Porter, 1989] have used knowledge to control search for consequences of new information during knowledge integration. Planning is of particular interest to us (cf. section 4.1) and there are a number of planners that use sophisticated search control techniques[6]. For example, [Tate, 1975] describes in his Ph.D. thesis how the structure of a given goal and its sub-goals can be exploited to control search; [Croft, 1985] examines in his work what exactly the choice points are during planning and develops heuristics

[6] To avoid confusion here, the term meta-planning has been introduced by [Wilensky, 1981] but does not refer to the use of explicit meta-knowledge to control search in planning.

to control search at these points; [Fox *et al.*, 1989] view planning as a constraint satisfaction problem and develop the concept of problem texture that is meant to aid in controlling search.

Thus, there exists a large body of search control knowledge that might be re-usable as capability knowledge. However, a closer inspection of the approaches described above reveals that the search control knowledge is often built into the system to maximise efficiency, i.e. it is represented only implicitly. In [Wickler and Pryor, 1996] we have attempted to re-use this implicitly represented search control knowledge to assess competence. However, our aim here is an explicit capability representation and the implicitness of the above search control knowledge is unlikely to provide us with insights as to how to represent capabilities.

3.3.3 Learning Search Control Knowledge

What we are looking for at this point are systems that contain explicitly represented search control knowledge that can be re-used for capability brokering. Most systems that *learn search control knowledge* belong to this group. This is because techniques from symbolic machine learning are often aimed at constructing an explicit representation of what they are trying to learn. If this learned search control knowledge could be re-used for capability brokering it would have the added advantage that we would not even have to find the knowledge ourselves. Thus, we shall now look at systems that learn search control knowledge.

Two general problem-solving architectures have been used to investigate this possibility: SOAR [Laird *et al.*, 1987, Rosenblum *et al.*, 1993] and PRODIGY [Minton *et al.*, 1989, Veloso *et al.*, 1995]. The basic learning algorithm in SOAR is chunking and learning from outside guidance [Golding *et al.*, 1987]. In PRODIGY explanation-based learning has been applied to learn explicit search control rules [Minton and Carbonell, 1987]. Explanation-based learning is a technique that has also recently been applied to learning search control rules for a SNLP-like planner [Kambhampati *et al.*, 1996]. The results described there are rather promising as far as the speed-up over SNLP ([McAllester and Rosenblitt, 1991], cf. section 4.1.2) is concerned. Similarly, [Ihrig and Kambhampati, 1997] describe the successful application of explanation-based learning to a case-based planner. Inductive learning of search control rules has been described in [Leckie and Zukerman, 1991], and [Eskey and Zweben, 1990] describe their work on leaning search control knowledge for the closely related scheduling problem. This shows that there is sufficient work in this area to provide a large body of explicit search control knowledge that might be re-usable as capability representations.

However, the fact that explicit search control knowledge can slow down problem-solving has not gone unnoticed [Etzioni and Minton, 1992]. The more search control rules have been learned, the more time it takes to evaluate all of them. Early approaches to this *utility problem* have just counted how often a specific search control rule was fired and deleted it if the success-rate went below a certain threshold [Minton *et al.*, 1987]. Later approaches attempted to approximate the learned search control knowledge to save time [Chase *et al.*, 1989]. [Wefald and Russell, 1989] have even tried to theoretically define when a search control rule has no benefit. [Kambhampati *et al.*, 1996] have avoided the utility problem by only learning provably correct rules, which are not very many.

As far as capability descriptions are concerned, forgetting or approximating search control knowledge means having a less accurate capability description. Considering the advantages of this approach, i.e. no need for a new representation or the manual development of new knowledge, this inaccuracy seems acceptable. However, there are more worrying results that question the usefulness and thus, the availability of explicit search control knowledge in the long term. Specifically, [Ginsberg, 1996a] has looked at a number of problems to which AI systems have been applied and found that, consistently, the most efficient approaches use relatively uninformed search. Why this is the case is not of much interest to us here, but this problem, which is ultimately rooted in the utility problem, has lead us to abandon the re-use approach argued for in this section.

3.4 Terminological KR Languages

By a *terminological knowledge representation language* we mean any formalism for defining and reasoning about concepts in the mould of [Brachman, 1979] and KL-ONE [Brachman and Schmolze, 1985]. Such systems are of little direct relevance here as there has not been a comprehensive attempt to represent capability knowledge in such a formalism. That is not to say that it is not possible though. The reason why we want to mention these languages here is that these formalisms provide the framework for the definition of ontologies to which we will return in section 4.2. For example, Ontolingua [Gruber, 1992] can be seen as rooted in terminological KR languages.

Most Important Issues Here

- First-order predicate logic is probably the best understood knowledge representation in AI and beyond and the situation calculus is an important action representation that could be used in KIF-based brokers.

- Although meta-knowledge initially looked like a very promising approach to capability representations because it potentially allows the re-use of a large existing body of knowledge, recent results related to the utility problem discussed in section 3.3.3 indicate that this approach is not desirable.

4 Actions in AI Planning

In this section we will review approaches to action representations in AI planning, the area we see most closely related to capability modelling.

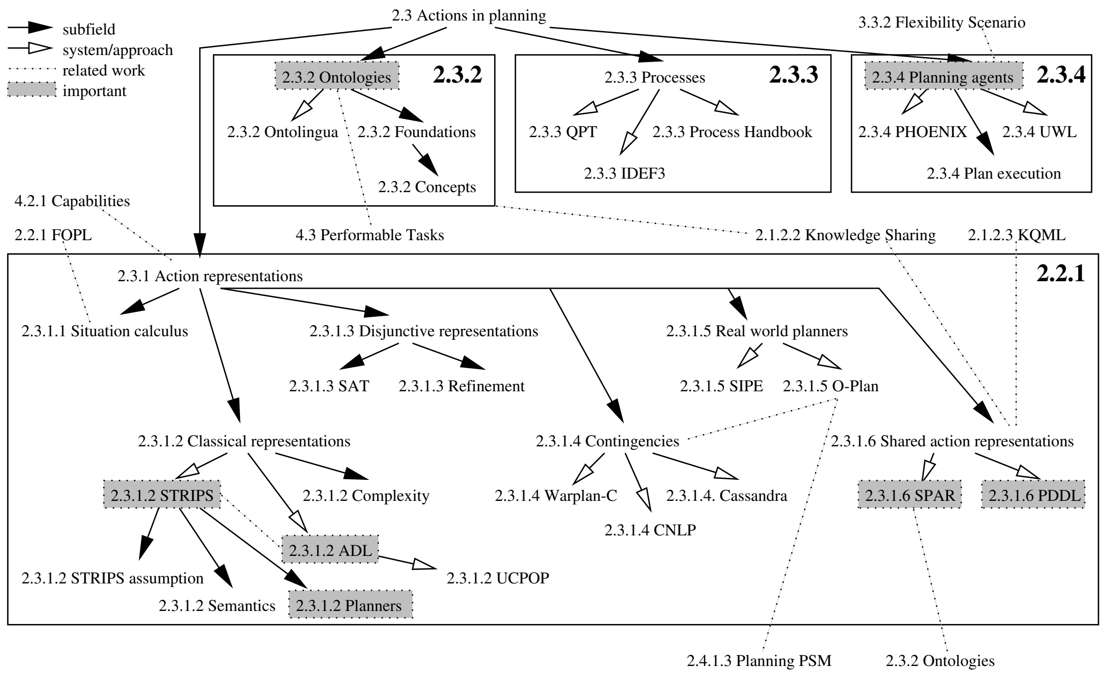

Figure 3: Overview of this section

An overview of this section, which provides a conceptualisation of the relationships between the different sub-fields and approaches/systems described in this section, is given in figure 3.

4.1 Action Representation Formalisms

There are *two reasons* why action representations as used by AI planners are of particular importance for our work. Firstly, a primitive action, one of the inputs to the classical planning problem [Tate *et al.*, 1990, page 28], can be interpreted as the representation of a capability. The second reason for our interest in action representations and AI planning is more complex. As pointed out before, intelligent agents are often assumed to use a planner to determine a course of action that achieves a given objective [Wooldridge and Jennings, 1995, page 131]. Thus, it is plausible to assume that, for a given objective, there exists a planning problem that has a solution if the agent is capable of achieving this objective. Under this assumption, the problem of capability assessment may be reduced to the plan existence problem.

4.1.1 First-Order Logic and the Situation Calculus

The planning problem was first addressed in AI e.g. in [Green, 1969] and in the situation calculus [McCarthy and Hayes, 1969, Shanahan, 1997].[7] Both of these approaches did not devise a new representation for actions but used *first-order predicate logic* to represent world states, actions, and their effects. Using first-order logic lead to a number of problems, most notably, the frame problem. Although there has been considerable progress towards representations of actions in first-order logic that avoid the frame problem, it can by no means be considered solved. Since we have already discussed first-order logic as a capability representation in section 3.1, we shall not go into more detail at this point.

4.1.2 Classical Non-Hierarchical Representations

One of the earliest systems in AI to address the planning problem using a task specific representation was the STRIPS system [Fikes and Nilsson, 1971, Fikes *et al.*, 1972]. The STRIPS *representation of actions* basically consists of:

- **an action pattern**: the identifier of the action and some variables describing the parameters;
- **a precondition formula**: a formula that must be true before this action can be applied;
- **an add list**: a list of formulae that will be true as a result of this action; and
- **a delete list**: a list of formulae that will no longer be true as a result of this action.

In the original definition of the STRIPS representation, the different formulae in the representation were allowed to be full first-order logic and a resolution

[7] The earliest AI system addressing this problem was probably GPS [Newell and Simon, 1963].

theorem prover was used to reason about world states. However, in a later description [Nilsson, 1980, chapter 7] only conjunctions of literals are permitted, which greatly simplifies the planning process. This later version is what is now generally referred to as the STRIPS representation. The significant advance of this representation over the situation calculus is the STRIPS *assumption*: only what is mentioned in the representation changes when an action is performed, i.e. anything that is not listed in the add or delete list will not change.

One interesting aspect of the STRIPS representation is that there was no *formal semantics* defined for STRIPS for a rather long time. In a classic paper, [Hayes, 1974] pointed out that many representations in AI suffered from this problem, and that formalisms that have no semantics should not be considered representations. Still, it was not until [Lifschitz, 1986] that a semantics for STRIPS was formally defined. Lifschitz also illustrates how the first intuitive approaches are not always quite the right definitions. It has to be said, though, that the STRIPS representation proved to be an extraordinarily successful action representation. There are still planners being developed today that use exactly this representation.

The STRIPS *planner* on the other hand suffered from many problems that were addressed in a number of subsequent systems, mostly following the STRIPS approach [Georgeff, 1987, Allen *et al.*, 1990, Tate *et al.*, 1990]. The final incarnation of a planner in the mould of STRIPS is probably the partial-order causal-link planner SNLP [McAllester and Rosenblitt, 1991]. However, there has been no significant advance in the representation of actions used by these systems, and this is the aspect we are most interested in here.

One of the more serious limitations of the STRIPS representation is its expressiveness. For example, the situation calculus is considered a much more expressive representation. It was not until [Pednault, 1989] that a serious attempt at exploring the middle ground between these two approaches was made. The result of this work was the new *action description language* (ADL) that combined the expressiveness of the situation calculus with the STRIPS assumption to retain the best of both worlds. The underlying idea in ADL was to exploit the fact that effect axioms in the situation calculus all more or less have the same syntactical format. Pednault used this pattern to define ADL and how ADL expressions should be expanded into situation calculus formulae. In this way, the semantics of ADL was grounded in the situation calculus but the syntax looked much more like STRIPS with precondition, add, and delete formulae.

What Pednault did not do was design a planner for ADL. Although one could translate the representation into first-order logic and reason about it with a theorem prover, this was clearly not the intension. The first planner that was based on a restricted version of ADL was UCPOP [Penberthy and Weld, 1992, Barrett *et al.*, 1995]. The basic extension of UCPOP*'s version of* ADL over the STRIPS representation was the introduction of conditional effects. Effects are the union of add and delete lists and conditional effects are effects that only occur if certain secondary preconditions hold before the action is executed. Also, conditional effects can occur any number of times with different instantiations for a given action instance. By restricting the domains of all variables to known, finite domains it was possible to extend the basic SNLP algorithm to handle conditional effects.

Complexity of STRIPS Planning As we have mentioned above, one of the reasons why we are interested in AI planning is because the capability assessment problem may be reduced to the plan existence problem. [Bylander, 1994] has shown that the problem of determining whether a given instance of the planning problem has any solution is, even for propositional STRIPS, a PSPACE-complete problem. Thus, assessing capability via plan existence is not a promising route as far as capability retrieval is concerned.

An investigation into whether and why different types of planning algorithms are more efficient than others can be found in [Barrett and Weld, 1994]. In the course of this work, they defined the complexity of a planning problem. We could potentially use such a complexity measure to estimate whether a plan will be found within given resources, i.e. to address the plan existence problem. However, the complexity of the benchmark problems they used was given in terms of the length of the shortest plan solving them, i.e. in general, the complexity was only known once the problem was solved.

4.1.3 Disjunctive Representations

In [Kautz and Selman, 1992] a new approach to planning has been suggested. Instead of refining a partial plan through search they have reformulated the planning problem as a *satisfiability problem* to which they applied their stochastic hill climbing algorithm GSAT [Selman *et al.*, 1992]. The difficult task here is the reformulation. [Blum and Furst, 1995] independently found such a reformulation that led to a significant increase in performance over conventional planners as demonstrated by their planner, Graphplan. This new formulation was later improved and combined with Walksat, an evolution of GSAT, to give even better results [Kautz and Selman, 1996].

Now, why is it that these satisfiability planners could so drastically outperform all deductive partial-order causal-link approaches? This question has been addressed in [Selman, 1994, Kambhampati, 1997] and they suggest that the essential difference lies in the fact that the representations used by satisfiability planners are capable of representing a new kind of *disjunction in plans*, i.e. sets of plans that contain disjunctions of actions to be included in the final plan. As a response to this finding there are now some deductive planners that also use disjunctive representations, e.g. COPS [Ginsberg, 1996b] or Descartes [Joslin and Pollack, 1996]. However, they do not seem to have the performance of satisfiability planners yet.

As far as action representations are concerned, these new planners can be considered a step backwards rather than forward. All the actions the satisfiability planners can reason about are strictly propositional, a limitation that stems from the satisfiability algorithm used. Thus, this work is of little interest to us. The above planners do however reason about disjunction in plans and, as far as plan representations are concerned, this presents a significant advance. This is not an issue here though.

4.1.4 Contingencies

An interesting extension of the STRIPS-based action representations presented this far has been introduced in *contingency planning*. Essentially, the idea here is that certain actions may have several alternative outcomes. The first planner to address this problem was Warplan-C [Warren, 1976]. The representation used by Warplan-C was again based on the STRIPS representation. The major difference was that several alternative sets of effects can be specified for an action, each of which is given a contingency label. Each set of effects was represented as an add and a delete list, as it would be for a normal STRIPS action. The number of contingencies was assumed to be small and not all actions were expected to lead to contingencies. There has also been some work on extending O-Plan (see below) to deal with contingencies [Secker, 1988], but the current version does not contain any such extension.

A more recent contingency planner is CNLP [Peot and Smith, 1992], which is basically a non-linear version of Warplan-C. The underlying action representation did not change from Warplan-C though. A variant of CNLP's algorithm has been used in the Cassandra planner [Pryor and Collins, 1996] which, like UCPOP, also handles conditional effects. In Cassandra's action representation the different contingencies were represented as conditional effects, where the contingency label can be seen as a secondary precondition of the different effects in the different contingencies.

Effectively, contingencies can be seen as introducing disjunctions into effects, thus significantly extending the expressiveness of the representation. Thus, these representations are of great interest to us.

4.1.5 Real World Planners

Most of the planners mentioned this far are research vehicles and have not been applied to realistic domains. However, there are at least two planners that have been used outside a research environment: *O-Plan* [Currie and Tate, 1991, Tate *et al.*, 1994, Tate, 1995] and SIPE [Wilkins, 1988]. Both these systems are quite similar in that they support a very rich representation to support planning in realistic domains.

The O-Plan planner essentially consists of: a set of knowledge sources which can address different types of flaws or issues in a plan; a set of constraint managers to evaluate different types of constraints in a plan; and a controller for these modules. The *openness* of O-Plan means new modules can be added to the planner without too much effort. The representation used by O-Plan is based on the <I-N-OVA> constraint model of activity [Tate, 1996a] which views a plan as a set of constraints on possible behaviour. The actual action representation language used in O-Plan is called O-Plan Task Formalism (TF) [O-Plan TF, 1997, Tate *et al.*, 1998]. O-Plan TF is primarily based on a hierarchical model of action expansion. The representation of primitive actions, the aspect we are most interested in, has a great degree of richness, allowing for a number of constraint types in the representation, e.g. complex temporal constraints, resource constraints, etc. The ability to add new constraint managers as required gives O-Plan the high flexibility needed for realistic domains. Another interesting aspect of the O-Plan planner is that is has been modelled as a CommonKADS problem-solving method [Kingston *et al.*, 1996] (cf. section 5.1.3).

When it comes to modelling realistic domains, the richness offered by the representations of these real world planners presents a significant advance over the earlier STRIPS-based representations. However, our aim for now is not to develop a broker for a realistic domain which might require such richness in its capability representations. Furthermore, whether more richness necessarily means more expressiveness is an open question. The most interesting aspect of these planners for our work is the openness of O-Plan which gives it its flexibility.

4.1.6 Shared Action Representations

Part of the current movement towards knowledge sharing and shared representations (cf. section 2.2.2) involves the development of *shared action representations.* In section 2.2.3, we have already looked at KQML which can be considered one such language, as a KQML expression represents an action. At least two other efforts with the aim of standardising a common action representation that facilitates knowledge sharing are currently under way. We will look at these next.

One of the latest proposals is the *Shared Planning and Activity Representation* (SPAR) [SPAR, 1997, Tate, 1998]. The principal scope of SPAR is to represent past, present, and possible future activity and the command, planning, and control processes that create and execute plans meant to guide or constrain future activity. It can be used descriptively for past and present activity and prescriptively for possible future activity. The way SPAR aims to facilitate knowledge sharing is not only through a language with an open syntax, but also by providing an ontology of fundamental concepts for representing and reasoning about actions. A brief look at the SPAR ontology will follow in section 4.2.

Another shared action representation is the *Planning Domain Definition Language* (PDDL) that was developed as a common format for all competitors in the AIPS'98 planning competition [Ghallab *et al.*, 1998]. The scope of PDDL is far more limited than SPAR: PDDL was only aiming for a representation that covers the representations used by competing planning algorithms. One of the interesting features of this language is that it contains explicit flags for different extensions to the basic language that have to be set in a problem description if the according extension is used. A planner not supporting these extensions can then simply check these flags to see whether it can generate plans for the described problems.

Both representations are only meant as an interlingua and not as representations which are reasoned over directly. Still, both these languages, and KQML, as well, offer very interesting features that a capability representation must also have, such as SPAR's openness and flexibility.

4.2 Ontologies of Actions

A logic only defines the syntax and semantics for a representation, but it is the ontology that defines the vocabulary. Approaches to knowledge sharing therefore agree on the need for *shared ontologies* (cf. section 2.2.2). Thus, we too will need a shared ontology of actions to represent and reason about capabilities. One of the best known languages for defining sharable ontologies is Ontolingua [Gruber, 1992], which has been defined as part of the knowledge sharing effort. Methodological issues for developing ontologies are discussed in [Gruber, 1993a, Fernández *et al.*, 1997, Gómez-Pérez, 1998].

Foundations for sharable ontologies of actions are described in [Tate, 1996b]. According to Tate, an ontology can be composed of four major parts. Firstly, there is the meta-ontology which contains fundamental ontological elements used to describe the ontology itself. Secondly, there is the top level ontology which is the minimal ontology used as a framework by all detailed ontologies. Thirdly, there is a library of shared ontological elements which may be shared across a number of detailed ontologies but need not be included. Finally, there are the detailed ontologies which build on the top level ontology and may include ontologies from the library.

A fundamental question is *which concepts* the different parts of a shared ontology of actions should contain. There are a number of such ontologies that have suggested different concepts, mostly for the meta-ontology and the top level ontology. For example, ontologies of actions were defined in the Process Interchange Format (PIF) [Lee *et al.*, 1996], the Enterprise ontology [Uschold *et al.*, 1996], the Core Plan Representation (CPR) [Pease and Carrico, 1997], the Shared Planning and Activity Representation (SPAR) [SPAR, 1997], and recently in work on models of problem solving [Gennari *et al.*, 1998] (cf. section 5.2). The SPAR ontology, for example, defines concepts for entities, environments, activities, actions, events, time points, objects, agents, locations, calendars, relationships, activity constraints, world models, plans, processes, objectives, issues, etc. Concepts are related to each other in a semantic network style representation and each concept is defined by a semi-formal description.

We believe an ontology of actions to be a considerable aid for the representation of capabilities.

4.3 Process Modelling

One of the drawbacks of STRIPS-based action representations, as described above, is that they are insufficient for reasoning about processes. This is because they only refer to two states, the one just before the described action and the one just after the action has been completed. Processes cannot be described adequately in this way. A first attempt to reason about simultaneous, interactive processes, characterised by a continuum of gradual change, that may be activated involuntarily, and that take up time, was proposed in [Hendrix, 1973]. This line of work ultimately lead to the *Qualitative Process Theory* (QPT) [Forbus, 1984]. Not only does QPT handle all the above difficulties, it also can be used to come up with useful conclusions even when not all the quantities for a given process are given.

The IDEF3 process capture method has been used to model processes of a different kind [Mayer *et al.*, 1992, Lydiard, 1996]. IDEF3 is part of the IDEF family of methods funded by the US Air Force to provide modelling support for systems engineering and enterprise integration. The IDEF3 method allows different user views of temporal precedence and causality relationships associated with enterprise processes to be captured. The information is presented in a series of diagrams and text, providing both a process-centred view of a system, via the Process Flow Network , and an object-centred view of a system via the Object State Transition Network. This method can tolerate incomplete and inconsistent descriptions and is flexible enough to deal with the incremental nature of the information acquisition process.

[Malone *et al.*, 1997] describe a novel theoretical and empirical approach to tasks such as business process redesign, enterprise modelling, and software development. The project involves collecting examples of how different organisations perform similar processes, and organising these examples in an on-line *process handbook*. The handbook is intended to help people redesign existing organisational processes, invent new organisational processes, learn about organisations, and automatically generate software to support organisational processes. A key element of the work is an approach to analysing processes at various levels of abstraction, thus capturing both the details of specific processes as well as the "deep structure" of their similarities.

4.4 Agents Planning with Capabilities

Although it has been argued that deliberative agents should be based on AI *planning* technology [Wooldridge and Jennings, 1995, page 131], most existing agents are not. The earliest agents based on a planner are probably found in [Cohen *et al.*, 1989]'s PHOENIX system which includes planning agents that operate in the domain of situated forest fire management.

We have argued at the beginning of this section that the capability assessment problem may be reduced to the plan existence problem under certain assumptions. One of these assumptions was that there will be no problems during the execution of a plan, but we know that this assumption is overly optimistic. Early work that can be seen as the foundation for a planning agent's architecture is presented in [Ambros-Ingerson and Steel, 1988], which describes IPEM, a clear and well-defined framework for the integration of planning, plan *execution*, and execution monitoring. More recent work in the area of plan execution and opportunity recognition with reference features is described in [Pryor, 1996].

Probably the most noteworthy agents that do use a planner are the intelligent softbots developed at the University of Washington [Etzioni *et al.*, 1993, Weld, 1996, Etzioni, 1997]. One of the most interesting aspects of this work for us is the action representation used by the softbots. They found that STRIPS-based representations lack certain essential features that they needed for their Internet softbots. The action representation language they have developed to model operating system commands, UWL [Etzioni *et al.*, 1992], has two major extensions over conventional languages. Firstly, it allows the modelling of information gathering through goals of the type `(find-out` *literal*`)`. Secondly, one can specify for certain conditions to remain unchanged by an action with a `(hands-off` *condition*`)` goal expression. Arguably, the former is subsumed by reasoning about knowledge as discussed in section 2.2 and the latter is a side effect of having to refer to objects by their properties.

Our main concern is with the capability reasoning performed by the broker, prior to the assignment of tasks to problem-solving agents. Hence, problems occurring during plan execution are not addressed in this thesis.

Most Important Issues Here

- Primitive actions in classical non-hierarchical action representations (section 4.1.2) like STRIPS and ADL could well form the basis for a capability description language.

- Furthermore, capability descriptions should be open like the O-Plan representation giving it flexibility, they should allow for the representations of ontologies of actions like the SPAR ontology, and they should allow for the flagging of language properties similar to PDDL.

- A planner should be used by the problem-solving agents to determine a course of action. A planner could also be used by a broker to combine capabilities of various problem solvers to solve a given problem.

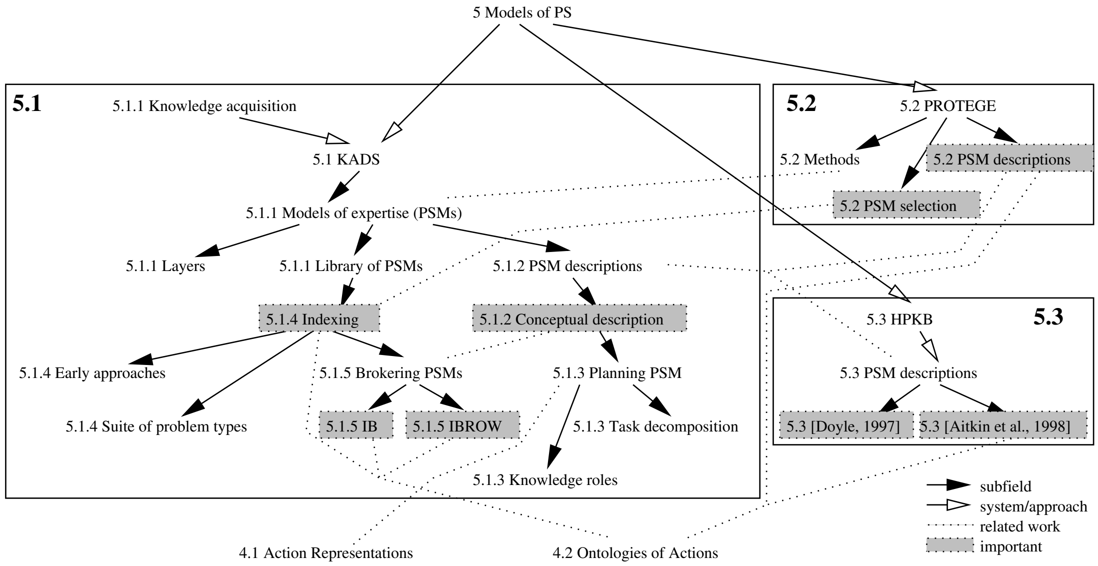

Figure 4: Overview of this section

5 Models of Problem Solving

In this section we will review approaches to modelling problem-solving methods.

An overview of this section, which provides a conceptualisation of the relationships between the different sub-fields and approaches/systems described in this section, is given in figure 4.

5.1 KADS-Based Approaches

We were interested in models of problem solving to investigate whether these models can be seen as capability models and thus, can be used for capability brokering. One of the largest and longest-running projects that deals with the modelling of problem-solving methods is the KADS *project* [Wielinga and Breuker, 1986, Breuker and Wielinga, 1989, Wielinga *et al.*, 1992]. Therefore, we shall now look at the KADS representation of models of problem-solving.

5.1.1 The KADS Methodology

The KADS methodology is a tool for *knowledge acquisition* and the building of knowledge-based systems (KBSs). Knowledge acquisition is a constructive process in which the knowledge engineer uses data about the behaviour of an expert to make design decisions regarding a KBS to be built. In this view, a KBS is an operational model that is the result of knowledge acquisition. The process of knowledge acquisition consists of knowledge elicitation, knowledge interpretation, and formalisation.

The KADS methodology's first principle is that the knowledge acquisition process should result in a number of *intermediate models*. These are: the organisational model, the application model, the task model, the model of cooperation, the model of expertise, the conceptual model, and the design model. The organisational model and the application model are models of the environment the KBS is meant to be used in and the problem it is meant to address. The task model specifies how the function of the system is to be achieved and contains the task decomposition. The model of cooperation assigns tasks and sub-tasks to agents. The model of expertise specifies the problem-solving expertise required to perform the problem-solving tasks assigned to the system at the knowledge-level [Newell, 1982]. The conceptual model is an abstract description of the objects and operations the KBS should know about. Finally, the design model is a high-level specification of the KBS, the operationalisation of which should be the KBS itself.

The model that we are most interested in here, as it appears to be the closest to a capability model, and that has received the most attention within the KADS community is the *model of expertise*. KADS suggests a decomposition of this model according to the types of knowledge it contains into several layers:

- **The domain layer**: The domain knowledge embodies the conceptualisation of the domain for a particular application in the form of the domain theory. It contains concepts, properties, and relations between concepts and their properties.

- **The inference layer**: The inference knowledge embodies primitive inference actions over the domain knowledge, also referred to as the knowledge sources. Domain knowledge is mapped into the meta-classes or knowledge roles that represent the generic input and output of the inference actions by the domain view. The inference structure describes the flow of knowledge between the inference actions similar to a data flow diagram.

- **The task layer**: The task knowledge embodies the control knowledge needed to perform reasoning at the inference layer. This includes knowledge of how the overall task is to be decomposed into subtasks.

- **The strategic layer**: The strategic knowledge determines what goals are relevant to solve a particular problem. However, this layer was dropped in KADS-II/CommonKADS.

One of the key issues in the KADS methodology is that it strongly advocates the re-use of knowledge, which is to be achieved through a *library* of such knowledge. The library is divided into two parts: the domain division, which is concerned with generic and re-usable domain knowledge, and the task division, which contains the description of the *interpretation models* or *models of problem-solving*. An interpretation model is a model of expertise with an empty domain layer, i.e. it is a domain-independent description of a problem-solving method (PSM) [Benjamins *et al.*, 1997]. These are exactly the models we are interested in.

5.1.2 Descriptions of PSMs

The KADS library contains a number of generic PSMs that represent the experience gained in many years of knowledge engineering [Breuker *et al.*, 1987, Breuker and Van de Velde, 1994]. The *description* of each PSM in the library consists of three parts: a verbal description, a conceptual description, and a formal description. The verbal description is a description in natural language. The conceptual description uses a frame-like language derived from the Conceptual Modelling Language CML [Wielinga (ed) *et al.*, 1994, chapter 3]. The formal description which exists only for a few PSMs is given in ML^2 (see below).

For each PSM the *conceptual description* defines functions which are essentially the primitive inference actions, function structures which are more or less the inference structures, and a control structure which is the control regime applied in this PSM. The conceptual description of a function consists of a description of the dynamic input and output knowledge roles of this function, the static knowledge roles it accesses, its goal, a specification of the relation between input and output, and an operation type which is the type of primitive inference this function performs. The function structure is a collection of the functions this structure is composed of. The control structure is a specification of the control flow over these functions including how the overall task accomplished by this PSM is to be decomposed.

Although the conceptual description contains the right kind of knowledge to be considered a capability representation, it also allows for informal content in many places. Thus, it can not be used as is for automated brokering, but it provides us with insights for designing a capability representation.

Formal Specifications of Models of Expertise ML^2 is a formal specification language based on KADS models of expertise [van Harmelen and Balder, 1992, Aben, 1995]. It allows different levels of formalism for domain, inference, and task layer. The domain layer is to be specified essentially in typed first-order logic. The inference layer extends this by allowing the reification of expressions, i.e. a form of meta-expressions, and reflective reasoning about these named expressions. Finally, the specification of the task layer is to be defined in Quantified Dynamic Logic (cf. section 3.2). While this formalism is certainly powerful, it has been "claimed that highly trained mathematicians are needed to write, to understand and to verify a formal specification" [Aben, 1995, page 20].

5.1.3 Planning as a PSM

As we have pointed out in section 4.1, *planning* is one area of particular interest to us because the ability to generate a plan to solve a given problem can be interpreted as the capability of solving this problem. Thus, we will have a brief look at the PSM for planning described in the CommonKADS library of PSMs now [Valente, 1994, Valente, 1995, Barros *et al.*, 1996].

The first step in the description of a PSM is the identification of the *knowledge roles*. For the planning task, four dynamic and two static roles have been identified. The dynamic knowledge roles are the current state, the goal, the plan, and the conflicts. The current state is a description of the initial state of the world, and the goal is a set of conditions to be achieved in a future state of the world. The plan consists of a set of plan steps, ordering constraints, variable bindings, and causal links (cf. section 4.1). The conflicts represent the discrepancy between the conditions in the goal and what the plan achieves. The static knowledge roles are the world description and the plan description. The world description consists of the state description, e.g. fluents in the situation calculus, and the state changes, effectively the possible actions in the domain. The plan description comprehends the optional plan structure, a hierarchical decomposition of the actions, and the plan assessment knowledge used to evaluate plans.

The *task decomposition* for the planning PSM is summarised in figure 5. Tasks are represented by ellipses in this figure and PSMs are represented by boxes. For example, the planning task can be addressed with a propose-critique-modify PSM which leads to three sub-tasks: propose expansion, critique plan, and modify plan that have to be performed in this order. Each of these tasks can again be addressed by some PSM until no further decomposition is possible.

It has been shown that this PSM description does describe, at the knowledge

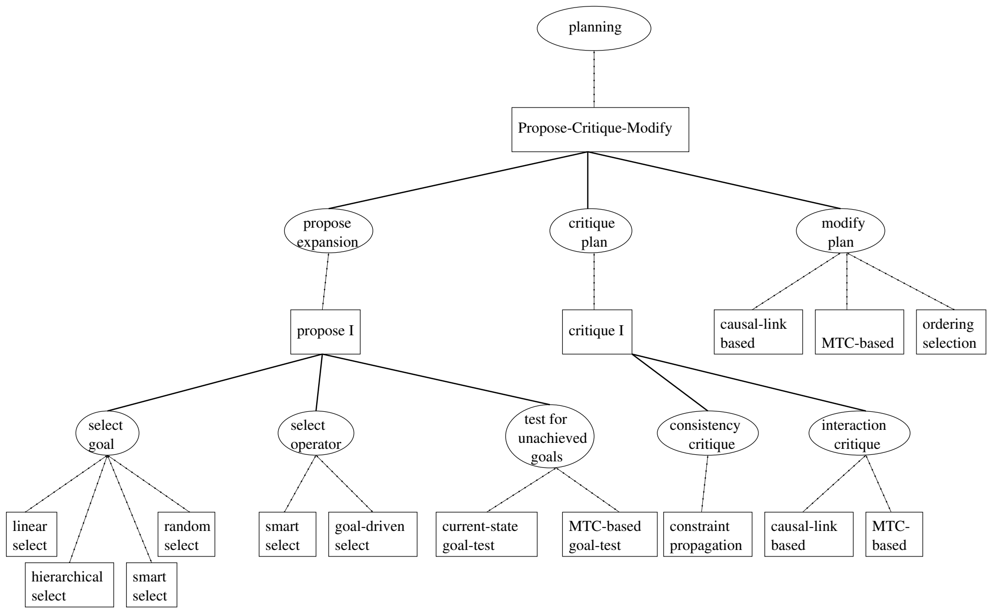

Figure 5: Task-Method Decomposition; from [Barros *et al.*, 1996, page 15]

level, the problem-solving behaviour of many modern planners. However, it is difficult to see how this description can be used to decide whether a planner will be able to find a solution for a given planning problem, i.e. how this description can be used to assess capability.

5.1.4 Indexing PSMs in the Library

During the knowledge acquisition process the knowledge engineer might identify a PSM from the library as appropriate for the task at hand and then use this model to focus the knowledge acquisition process, e.g. by attempting to elicit domain knowledge to fill relevant knowledge roles and by defining the domain view [Brazier *et al.*, 1995]. Thus, the retrieval of an appropriate model from the library, also referred to as the *indexing problem*, is an important step in the KADS methodology, just like it is for capability brokering. However, it is also expected that the model from the library will need further refinement before it can be transformed into the design model. This refinement is called knowledge differentiation in KADS. It is worth noting at this point that the resulting model, the design model, is not meant to be operational in KADS.

One approach to the indexing problem in KADS was the definition of a *taxonomy of generic tasks* which is supposed to help the knowledge engineer to identify an appropriate PSM in the library [Breuker and Van de Velde, 1994, page 59]. The idea was for the knowledge engineer to follow one path down a hierarchy that ends in the most specific PSM suitable for the task at hand. However, in practise this turned out not to be so simple. Another approach tried in the KADS project was the indexing of PSMs in the library with *task features* [Aamodt *et al.*, 1993] for which they have suggested a quite elaborate list of such features.

The latest insight however seems to be to take a more indirect approach. The basic argument in [Breuker, 1997] is that one is given a problem and different kinds of PSMs might be able to solve this problem. The PSM selection mechanism should reflect this by providing a *suite of problem types* and associating a number of PSMs with each problem type. The selection amongst these could then be by assumptions made by the PSM, by the domain, or by the depth with which the PSM has been modelled. Another, recent criticism of the original indexing mechanism is that it is based only on yes/no distinctions and does not allow gradual refinement [van Harmelen and ten Teije, 1998].

Since we cannot use the KADS representation for PSMs to represent capabilities directly, we also cannot adopt their indexing mechanisms.

5.1.5 Brokering for PSMs

There are currently at least two approaches in progress that attempt to address the indexing problem with a *broker*. Naturally, we are interested in this work as the problem addressed is very similar to our problem.

IB, the *Intelligent Broker* [Fensel, 1997, Decker *et al.*, 1998], currently under research at the University of Karlsruhe is one such broker. The aim of this broker is not to facilitate agent cooperation, as it is for the brokers described in section 2.3, but to find a PSM for a given task on the Internet. Unlike most other brokers, IB is not based on KQML. The approach assumes the availability of an ontology of PSMs which is used for the description of PSMs and which the broker can use for its search. The ontology of PSMs they envisage does not yet exist but might well be based on the taxonomy of PSMs described in the KADS library of expertise models [Breuker and Van de Velde, 1994, page 59]. The language in which they intend to describe PSMs and on which their ontology will be based is not finished yet. This language will be called the Unified Problem-solving Method description Language (UPML), but only a draft specification exists [Fensel *et al.*, 1998a, Fensel *et al.*, 1998b]. Another task envisaged for this broker is the adaption of the selected PSM to the actual task which requires mapping entities in the given problem to the roles of the PSM.

Another project that is closely related to the work on IB is the ESPRIT-funded project IBROW^3 that started in January 1998 [Benjamins *et al.*, 1998, Armengol *et al.*, 1998]. The aim here, too, is to develop a broker that can select, configure, and adapt knowledge components from large libraries on the Internet. For selecting a problem-solving method from a library, the broker will reason about characteristics of the PSM, in particular about their competence and requirements. For this purpose PSMs will have to be described in some language. Although no such language has been selected or proposed yet, it is envisaged that an ontology will be at the heart of the approach. Furthermore, as the people involved with IBROW^3 are largely the same as for IB it is quite possible that the two systems and PSM description languages will be similar.

5.2 Protégé

The Protégé system [Musen, 1989, Eriksson *et al.*, 1995] addresses a problem very similar to the problem addressed in KADS and thus, we are interested in Protégé for very similar reasons.

Protégé provides a knowledge engineering environment in which a developer can specify tasks and select PSMs from a library of re-usable *methods*. Developers must identify, at least partially, the task of the system they are designing before they can select and custom tailor preexisting methods. This task-analysis leads to a system-role description in terms of the domain for the system, which serves as the basis for the selection of PSMs that accomplish the task and for the configuration of the selected methods for the task instance. In PROTÉGÉ, methods are actions that accomplish tasks. Methods can delegate problems as subtasks to be solved by other methods. They use the term mechanism for primitive methods that cannot be decomposed. In addition to supporting the development of problem solvers for knowledge-based systems, PROTÉGÉ generates domain-specific knowledge acquisition tools that elicit the expertise required by the PSMs to perform the latter's task.

For PSM *selection*, they believe that it will be difficult to make a comprehensive list of factors to consider. However, they do identify a set of recurring factors that are applicable to most tasks. This list of common factors includes the input and output of the task, the domain knowledge available, the solution quality required, the computational time and space complexity, and the flexibility of the method. Once a method has been selected it needs to be configured. This is largely a matter of selecting mechanisms or methods for a method's subtasks and defining the mapping between method terms and domain terms.

An essential part of the *method description language* developed in PROTÉGÉ is the *method ontology* which includes definitions of all the objects required by the PSM. Ideally, developers of PSMs would share a framework for defining inputs and outputs. [Gennari *et al.*, 1998] have begun to develop a "foundation ontology" for developers of PSMs. In this ontology, a PSM must have a name and a textual description. Furthermore, it contains ontology frames for input and output, a list of subtasks, and a list of constraints across the inputs and output but not among inputs or outputs. The latter are located inside the ontology frame for inputs and outputs, together with key classes and functions in this frame, and the API used which contains information about the ways in which the PSM makes run-time queries for additional information. Subtasks again come with a textual description, inputs, outputs, constraints between those, and information as to whether this subtask is required and whether it has a default implementation. The lowest level of detail in their method description language is the choice of a formal language for expressing the axioms that represent the requirements of the method. The current suggestion is that this language will be based on KIF [Genesereth, 1991, Genesereth *et al.*, 1992].

To summarise, not only are the problems addressed by KADS and PROTÉGÉ very similar, but so are the approaches. Methods in PROTÉGÉ correspond to models of expertise in KADS. Both approaches are based on a library of PSMs and the description languages they use are essentially informal. Furthermore, both approaches address the indexing problem and suggest that an ontology will be the key to the solution. Thus, virtually all of our comments on KADS also apply to PROTÉGÉ.

5.3 The HPKB Program

The DARPA-funded *High Performance Knowledge Bases (*HPKB*) program* is a research programme to advance the technology of how computers acquire, represent and manipulate knowledge.[8] The approach taken in HPKB is quite similar to the approach in KADS again. One of its aims is to speed up the development of knowledge-based systems significantly. One way to achieve such a goal is through the enablement of knowledge reuse, including the reuse of PSMs. This might ultimately lead to the fully automated configuration of knowledge-based systems. For this purpose they are interested in developing a language for describing PSMs and a number of groups are currently working on a proposal for such a language. For example, the latest work on PROTÉGÉ is one of the inputs to the HPKB effort.

[Doyle, 1997]'s *proposal* for a PSM description language was one of the earliest contributions for the HPKB program. According to his proposal, a capability

[8] cf. `http://www.teknowledge.com/HPKB/`

description should include: the task addressed by the method; the method ontology; the contextual properties; the behavioural properties; the cognitive properties; relationships to other methods; relationships to implementations; and other annotations. However, as this proposal was still an early draft we shall not go into detail here.

Another interesting input to this part of the HPKB program is the *language proposal* described in [Aitken *et al.*, 1998]. They suggest that a PSM can be viewed as a process or action. In this case process or action representations from AI planning might also work for PSMs. We shall review process modelling techniques in section 4.3 and we have looked at action representations in section 4.1. As a result of this view, the language they propose characterises a PSM or capability in three parts. Firstly, there is the competence of the capability. This includes the goal or objective, the problem type the PSM addresses, a generic solution, the solution components (conclusion, argument structure, and case model), solution properties, and the rationale which can be a textual description of when and why the PSM might be used. Secondly, there is the configuration of the capability. This includes the method ontology, the domain theory consisting of field, ontology/mapping, and representation, and the sub-methods. The third and last part is the PSM process which includes the environment, the resource constraints, the actor constraints, various world constraints, and sub-activities.

Compared to KADS or PROTÉGÉ, HPKB is still in its infancy. The language proposals are all draft and indicate types of knowledge to be represented rather than defining actual languages. Thus, we can only take these initial ideas into account when designing a capability description language.

Most Important Issues Here

- Knowledge engineering with models of problem solving is often based on a library of PSMs. This library contains at least semi-formal descriptions of PSMs and it is the description languages suggested by the different approaches we are most interested in.

- Especially KADS and KADS-related work has been concerned with the indexing problem for their library. The indexing problem is closely related to the capability retrieval problem.

6 Summary and Evaluation

In this article we reviewed various approaches to *representing and reasoning about capabilities*. In section 2 we looked at software agents, the area in which most of the work on brokering has taken place to date. This area has been mostly concerned with the reasoning aspect of capability brokering. This work introduced us to the agent communication language KQML.

The remaining sections in this article described approaches which were mostly related to the representation aspect of capability brokering. In section 3 we looked at the way various logics could have been used to represent capability knowledge. In section 4 we looked at how representations of actions, which are very similar to capabilities, have been encoded in AI systems. Finally, in section 5 we looked at models of problem solving methods to see whether these models represent capability information, and if so, how it was represented.

6.1 Desirable Characteristics for Capability Representations

The first step towards a new capability description language must be a characterisation of the *properties* or *attributes* we want this language to have.

We believe that two properties a new capability description language should have are *expressiveness* and *flexibility* (cf. [Wickler, 1999]). By expressiveness we mean the ability to express more than is possible in other representations. By flexibility we mean the possibility to delay decisions regarding the compromises that have to be made to knowledge representation time.

Our aim is to use capability representations for brokering. When designing a knowledge representation language it is important to take into account what kind of reasoning one wants to perform over this language. Thus, another characteristic we would like a new capability description language to have is that it is similar to languages which have been *used for capability brokering* successfully, as this would indicate that this language, too, can be used for brokering. Likewise, since capabilities can be seen as actions one can perform, we would also expect such a language to be similar to representations that have been *used to represent and reason about actions*.

As we expect the broker to perform its services autonomously, it is important that the capability representations are in some *formal* language; a new capability description language must have this attribute. Finally, every representation must *have a semantics* to qualify as a representation in the first place [Hayes, 1974], so we shall pay attention to this property as well.

6.2 Preliminary Evaluation

Given this characterisation of desirable properties for a new capability description language, we can now evaluate the approaches described in this article to identify which of the approaches have the above properties. The results of this preliminary evaluation are summarised in table 3. For simplicity, we have only listed the four general areas described in sections 2 to 5. Each of these areas comprises a number of approaches and the table obviously over-generalises and thus, should be seen as a table of general trends rather than an exact evaluation.

The highest *expressiveness* can be found in logics and models of problem solving. Classical first-order predicate logic [Chang and Lee, 1973, Loveland, 1978, Gallier, 1986] has been used to represent many different kinds of knowledge and can thus be considered an expressive representation. However, many other logics offer still more expressiveness to allow the representation of highly complex circumstances (cf. section 3.2). Models of problem solving often allow natural

	2 brokers	3 logics	4 action reps.	5 models of PSMs
expressive	medium	high	medium	high
flexible	(yes)	no	some	no
brokered	yes	no	(yes)	(yes)
actions	no	no	yes	(yes)
formal	yes	yes	yes	(no)
semantics	(yes)	yes	(yes)	no

Table 3: Properties of different approaches

language as at least an aspect of their representation which accounts for their high expressiveness (cf. section 5.1.2). Most action representations (section 4.1) have only restricted expressiveness as they were designed to be used in formation of plans which in itself is a very complex process. There are, however, some action representations that offer more expressiveness, e.g. ADL [Pednault, 1989]. The representations used by the brokers we have described in section 2.3 are more difficult to classify as they are vague on what exactly the representation of capabilities they use will look like. Closer inspection reveals that, although they mostly allow KIF [Genesereth, 1991, Genesereth *et al.*, 1992] as the content language, the restrictions imposed are rather severe.

The highest *flexibility* of the representations we have looked at can be found in brokers and in some action representations. Most brokers are based on KQML (cf. section 2.2.3) which specifies that capabilities are to be described as KQML messages that can be processed. Thus, the capability description language is KQML, a language designed to have an opaque content which is expressed in a language specified at the wrapper level. In practise though, most brokers only allow a very limited range of languages that can be used as content in capability descriptions in KQML (cf. section 2.3). Most action representations (section 4.1) have very little flexibility, but there are a few noteworthy exceptions, e.g. SPAR [SPAR, 1997, Tate, 1998]. Like KQML, these languages allow the plugging in of different content languages which gives them their flexibility. Logics (section 3), although they provide a wide range of formalisms do not individually have this flexibility. Finally, models of problem solving (section 5) usually allow for some parts of their representations to be natural language descriptions and thus, cannot be considered flexible.

The next aspect we have looked at is whether the representation has been *used for brokering*. Obviously, the KQML-based representations described in section 2.3 pass on this criterion, but they are not the only ones. Action representations (section 4.1), in fact, can also be seen as having been used for brokering, as a planner that uses these representations at some point also needs to retrieve an action that can achieve a given effect. This is essentially the task performed during capability retrieval. A similar case could be made for the situation calculus (section 3.1) which is based on first-order logic, but it is really the ontology of the situation calculus that facilitates brokering, not the underlying representation. Thus, we are inclined to say that logics have not been used for brokering, allowing for exceptions. Models of problem solving (section 5) are again a borderline case as there are now several projects underway that are aimed at building brokers for problem-solving methods (PSMs) (cf. section 5.1.5). However, neither their representations nor their brokering mechanisms are defined yet.

The obvious representations that have been *used for representing and reasoning about actions* are, of course, the action representations (section 4.1). Models of problem solving (section 5) have also been used to represent and reason about actions, but the actions are usually restricted to the reasoning actions performed by some expert system. Still, reasoning about actions is what these representations were designed for. Brokering representations (section 2.3) and logics (section 3) both have also been used to represent and reason about actions, but this is not what they were specifically designed for.

As for the *formality* of the representation, the only area that does not qualify here are models of problem solving (section 5) because they usually allow for natural language as one aspect of their representation. As usual, there are exceptions, e.g. ML^2 (cf. section 5.1.2). However, ML^2 is so heavily logic-based that one could well count it into this area anyway. Closely related is the question of *semantics*. The area that has been most concerned with formal semantics is logics (section 3) in which almost every formalism has a well-defined formal semantics, otherwise it does not qualify as a logic. The semantics of action representations (section 4.1) and KQML (section 2.2.3) have also been defined to some degree, but there remain questions [Kuokka and Harada, 1995b] and descriptions are often informal. Finally, models of problem solving (section 5) being based on natural language fail here.

A new capability description language should retain the ideas behind these approaches that made them perform well in certain respects. In summary, a new capability representation:

- to preserve the structure found in action representations;
- to benefit from the expressiveness of highly powerful logics and the well-defined formal semantics that comes with them;
- to retain the flexibility of KQML by allowing for opaque content languages and to use the communication approach to brokering;
- to be formal to allow for autonomous brokering.

References

[Aamodt *et al.*, 1993] Agnar Aamodt, Bart Benus, Cuno Duursma, Christine Tomlinson, Ronald Schrooten, and Walter Van de Velde. Task features and their use in CommonKADS. Deliverable D 1.5, Free University of Brussels, Brussels, Belgium, January 1993.

[Aben, 1995] Manfred Aben. *Formal Methods in Knowledge Engineering.* PhD thesis, University of Amsterdam, Amsterdam, The Netherlands, February 1995.

[Aitken *et al.*, 1998] Stuart Aitken, Ian Filby, John Kingston, and Austin Tate. Capability descriptions for problem-solving methods. AIAI, University of Edinburgh, Edinburgh, Scotland, January 1998.

[Allen *et al.*, 1990] James Allen, James Hendler, and Austin Tate, editors. *Readings in Planning.* Morgan Kaufmann, San Mateo, CA, 1990.

[Ambros-Ingerson and Steel, 1988] José A. Ambros-Ingerson and Sam Steel. Integrating planning, execution and monitoring. In *Proc. 7th AAAI*, pages 83–88, Saint Paul, MN, August 1988. Morgan Kaufmann. Also in [Allen *et al.*, 1990, pages 135–740].

[Andrews, 1981] Peter B. Andrews. Theorem proving via general matings. *Journal of the ACM*, 28(2):193–214, April 1981.

[Armengol *et al.*, 1998] Eva Armengol, Richard Benjamins, Stefan Decker, Dieter Fensel, Enrico Motta, Rudi Studer, and Bob Wielinga. State of the art deliverable. Deliverable D1.4, University of Amsterdam, Amsterdam, The Netherlands, May 1998.

[Attardi and Simi, 1984] Giuseppe Attardi and Maria Simi. Metalanguage and reasoning across viewpoints. In Tim O'Shea, editor, *Proc. 6th ECAI*, pages 413–422, Pisa, Italy, September 1984. North-Holland.

[Barr, 1979] Avron Barr. Meta-knowledge and cognition. In *Proc. 6th IJCAI*, pages 31–33, Tokyo, Japan, August 1979. William Kaufmann.

[Barrett and Weld, 1994] Anthony Barrett and Daniel S. Weld. Partial-order planning: Evaluating possible efficiency gains. *Artificial Intelligence*, 67:71–112, 1994.

[Barrett *et al.*, 1995] Anthony Barrett, Dave Christianson, Marc Friedman, Chung Kwok, Keith Golden, Scott Penberthy, Ying Sun, and Daniel Weld. UCPOP user's manual (version 4.0). Technical Report 93-09-06d, University of Washington, Seattle, WA, November 1995.

[Barros *et al.*, 1996] Leliane Barros, André Valente, and Richard Benjamins. Modeling planning tasks. In Brian Drabble, editor, *Proc. 3rd International Conference on Artificial Intelligence Planning Sytems*, pages 11–18, Edinburgh, Scotland, May 1996. AAAI Press.

[Bayardo *et al.*, 1997] R. Bayardo, W. Bohrer, R. Brice, A. Cichocki, G. Fowler, A. Helal, V. Kashyap, T. Ksiezyk, G. Martin, M. Nodine, M. Rashid, M. Rusinkiewicz, R. Shea, C. Unnikrishnan, A. Unruh, and D. Woelk. Semantic integration of information in open and dynamic environments. In Joan M. Peckman, editor, *Proc. ACM SIGMOD International Conference on Management of Data*, Tucson, AZ, May 1997. ACM Press.

[Benjamins *et al.*, 1997] Richard Benjamins, Dieter Fensel, and B. Chandrasekaran. PSMs do IT! In *Proc. IJCAI Workshop on Problem-Solving Methods for Knowledge-Based Systems*, Nagoya, Japan, August 1997.

[Benjamins *et al.*, 1998] Richard Benjamins, Enric Plaza, Enrico Motta, Dieter Fensel, Rudi Studer, Bob Wielinga, Guus Schreiber, and Zdenek Zdrahal. IBROW^3 — an intelligent brokering service for knowledge-component reuse on the world-wide web. In *Proc. 11th Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Canada, April 1998.

[Blum and Furst, 1995] Avrim L. Blum and Merrick L. Furst. Fast planning through planning graph analysis. In *Proc. 14th IJCAI*, pages 1636–1642, Montréal, Canada, August 1995. Morgan Kaufmann.

[Bond and Gasser, 1988] Alan H. Bond and Les Gasser, editors. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann, San Mateo, CA, 1988.

[Brachman and Levesque, 1985] Ronald J. Brachman and Hector J. Levesque, editors. *Readings in Knowledge Representation.* Morgan Kaufmann, Los Altos, CA, 1985.

[Brachman and Schmolze, 1985] Ronald J. Brachman and James G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, April 1985.

[Brachman, 1979] Ronald J. Brachman. On the epistemological status of semantic networks. In Nicholas V. Findler, editor, *Associative Networks*, pages 3–50. Academic Press, New York, NY, 1979.

[Bradshaw, 1997] Jeffrey M. Bradshaw, editor. *Software Agents.* AAAI Press/ The MIT Press, Menlo Park, CA/Cambridge MA, 1997.

[Brazier *et al.*, 1995] Frances M. T. Brazier, Jan Treur, and Niek J. E. Wijngaards. Modelling interaction with experts: The role of a shared task model. Technical Report IR-382, Free University of Amsterdam, Amsterdam, The Netherlands, 1995.

[Breuker and Van de Velde, 1994] Joost A. Breuker and Walter Van de Velde, editors. *CommonKADS Library for Expertise Modelling.* IOS Press, Amsterdam, The Netherlands, 1994.

[Breuker and Wielinga, 1989] Joost Breuker and Bob Wielinga. Models of expertise in knowledge acquisition. In Giovanni Guida and Carlo Tasso, editors, *Topics in Expert System Design*, chapter 5, pages 265–295. Elsevier Science Publishers, Amsterdam, The Netherlands, 1989.

[Breuker *et al.*, 1987] Joost Breuker, Bob Wielinga, Maarten van Someren, Robert de Hoog, Guus Schreiber, Paul de Greef, Bert Bredeweg, Jan Wielemaker, Jean-Paul Billault, Massoud Davoodi, and Simon Hayward. Model driven knowledge acquisition: Interpretation models. KADS Deliverable Task A1, University of Amsterdam, Amsterdam, The Netherlands, 1987.

[Breuker, 1997] Joost Breuker. Problems in indexing problem-solving methods. In *Proc. IJCAI Workshop on Problem-Solving Methods for Knowledge-Based Systems*, Nagoya, Japan, August 1997.

[Brewka, 1991] Gerhard Brewka. *Nonmonotonic Reasoning: Logical Foundations of Commonsense*. Cambridge University Press, Cambridge, UK, 1991.

[Brooks, 1986] R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[Brooks, 1991] R. A. Brooks. Intelligence without representation. *Artificial Intelligence*, 47:139–159, 1991.

[Bundy and Welham, 1981] Alan Bundy and Bob Welham. Using meta-level inference for selective application of multiple rewrite rule sets in algebraic manipulation. *Artificial Intelligence*, 16(2):189–212, 1981.

[Bundy *et al.*, 1979] Alan Bundy, Lawrence Byrd, George Luger, Chris Mellish, and Martha Palmer. Solving mechanics problems using meta-level inference. In *Proc. 6th IJCAI*, pages 1017–1027, Tokyo, Japan, August 1979. William Kaufmann.

[Bylander, 1994] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1–2):165–204, September 1994.

[Chagrov and Zakharyaschev, 1997] Alexander Chagrov and Michael Zakharyaschev. *Modal Logic*. Clarendon Press, Oxford, UK, 1997.

[Chaib-Draa *et al.*, 1992] B. Chaib-Draa, B. Moulin, R. Mandiau, and P. Millot. Trends in distributed artificial intelligence. *Artificial Intelligence Review*, 6(1):35–66, 1992.

[Chang and Lee, 1973] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving*. Computer Science and Applied Mathematics Series. Academic Press, New York, NY, 1973.

[Chase *et al.*, 1989] Melissa P. Chase, Monte Zweben, Richard L. Piazza, John D. Burger, Paul P. Maglio, and Haym Hirsh. Approximating learned search control knowledge. In Alberto Maria Segre, editor, *Proc. 6th International Workshop on Machine Learning*, pages 218–220, Ithaca, NY, June 1989. Cornell University, Morgan Kaufmann.

[Chellas, 1980] Brian F. Chellas. *Modal Logic: An Introduction*. Cambridge University Press, Cambridge, UK, 1980.

[Chi *et al.*, 1981] Michelene T. H. Chi, Paul J. Feltovich, and Robert Glaser. Categorization and representation of physics problems by experts and novices. *Cognitive Science*, 5(2):121–152, April 1981.

[Cohen and Levesque, 1990] Paul R. Cohen and Hector J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42:213–261, 1990.

[Cohen *et al.*, 1989] Paul R. Cohen, M. L. Greenberg, D. M. Hart, and A. E. Howe. Trial by fire: Understanding the design requirements for agents in complex environments. *AI Magazine*, 10(3):32–48, Autumn 1989.

[Croft, 1985] D. Croft. Choice making in planning systems. In Martin Merry, editor, *Proc. 5th Expert Systems Conference*, pages 125–141, Warwick, UK, December 1985. University of Warwick, Cambridge University Press.

[Currie and Tate, 1991] Ken Currie and Austin Tate. O-Plan: The open planning architecture. *Artificial Intelligence*, 52(1):49–86, 1991.

[Davis and Buchanan, 1977] Randall Davis and Bruce G. Buchanan. Meta-level knowledge: Overview and applications. In *Proc. 5th IJCAI*, pages 920–927, Cambridge, MA, August 1977. MIT, William Kaufmann. Also in: [Brachman and Levesque, 1985, pages 389–396].

[Davis and Smith, 1983] Randall Davis and Reid G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, 1983. Also in: [Bond and Gasser, 1988, pages 333–356].

[Davis, 1980] Randall Davis. Meta-rules: Reasoning about control. *Artificial Intelligence*, 15(3):179–222, 1980.

[Davis, 1990] Ernest Davis. *Representations of Commonsense Knowledge*. Morgan Kaufmann, San Mateo, CA, 1990.

[Decker *et al.*, 1997] Keith Decker, Katia Sycara, and Mike Williamson. Middle-agents for the internet. In *Proc. 15th IJCAI*, pages 578–583, Nagoya, Japan, August 1997. Morgan Kaufmann.

[Decker *et al.*, 1998] Setfan Decker, Michael Erdmann, Dieter Fensel, and Rudi Studer. Reasoning with metadata: Ontobroker. University of Karlsruhe, Karlsruhe, Germany, 1998.

[Doyle, 1997] Jon Doyle. Problem-solving method language proposal. MIT, Cambridge, MA, October 1997.

[Eriksson *et al.*, 1995] Henrik Eriksson, Yuval Shahar, Samson W. Tu, Angel R. Puerta, and Mark A. Musen. Task modeling with reusable problem-solving methods. *Artificial Intelligence*, 79(2):293–326, December 1995.

[Eskey and Zweben, 1990] Megan Eskey and Monte Zweben. Learning search control for constraint-based scheduling. In *Proc. 8th AAAI*, pages 908–915, Boston, MA, August 1990. AAAI Press/The MIT Press.

[Etzioni and Minton, 1992] Oren Etzioni and Steven Minton. Why EBL produces overly-specific knowledge: A critique of the PRODIGY approaches. In Derek Sleeman and Peter Edwards, editors, *Proc. 9th International Workshop on Machine Learning*, pages 137–143, Aberdeen, Scotland, July 1992. Morgan Kaufmann.

[Etzioni *et al.*, 1992] Oren Etzioni, Steve Hanks, Daniel Weld, Denise Draper, Neal Lesh, and Mike Williamson. An approach to planning with incomplete information. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *Proc. 3rd KR*, pages 115–125, Cambridge, MA, October 1992. Morgan Kaufmann.

[Etzioni *et al.*, 1993] Oren Etzioni, Henry M. Levy, Richard B. Segal, and Chandramohan A. Thekkath. OS agents: Using AI techniques in the operating system environment. Technical Report 93-04-04, University of Washington, Seattle, WA, April 1993.

[Etzioni, 1997] Oren Etzioni. Moving up the information food chain. *AI Magazine*, 18(2):11–18, Summer 1997.

[Farquahar *et al.*, 1996] A. Farquahar, R. Fikes, and J. Rice. The Ontolingua server: A tool for collaborative ontology construction. Technical Report KSL 96-26, Stanford University, Stanford, CA, September 1996.

[Fensel *et al.*, 1998a] Dieter Fensel, Richard Benjamins, Stefan Decker, Mauro Gaspari, Rix Groenboom, Enrico Motta, Enric Plaza, Guus Schreiber, Rudi Studer, and Bob Wielinga. UMPL: The very high idea. University of Karlsruhe, Karlsruhe, Germany, 1998.

[Fensel *et al.*, 1998b] Dieter Fensel, Richard Benjamins, Stefan Decker, Mauro Gaspari, Rix Groenboom, Enrico Motta, Enric Plaza, Guus Schreiber, Rudi Studer, and Bob Wielinga. The unified problem-solving method description language UMPL (version 1.0.7). University of Karlsruhe, Karlsruhe, Germany, July 1998.

[Fensel, 1997] Dieter Fensel. An ontology-based broker: Making problem-solving method reuse work. In *Proc. IJCAI Workshop on Problem-Solving Methods for Knowledge-Based Systems*, Nagoya, Japan, August 1997.

[Fernández *et al.*, 1997] M. Fernández, A. Gómez-Pérez, and N. Juristo. METHONTOLOGY: From ontological art towards ontological engineering. In *Working Notes of the AAAI Spring Symposium on Ontological Engineering*, Stanford, CA, March 1997. Stanford University, AAAI Press.

[Fikes and Nilsson, 1971] Richard E. Fikes and Nils J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3/4):189–208, 1971. Also in: [Allen *et al.*, 1990, pages 88–97].

[Fikes *et al.*, 1972] Richard E. Fikes, Peter E. Hart, and Nils J. Nilsson. Learning and executing generalized robot plans. *Artificial Intelligence*, 3(4):251–288, 1972.

[Fikes *et al.*, 1991] Richard Fikes, Mark Cutkosky, Tom Gruber, and Jeffrey Van Baalen. Knowledge sharing technology—project overview. Technical Report KSL 91-71, Stanford University, Stanford, CA, November 1991.

[Filman *et al.*, 1983] Robert E. Filman, John Lamping, and Fanya S. Montalvo. Meta-language and meta-reasoning. In *Proc. 8th IJCAI*, pages 365–369, Karlsruhe, Germany, August 1983. William Kaufmann.

[Finin *et al.*, 1992] Tim Finin, Don McKay, and Rich Fritzson. An overview of KQML: A knowledge query and manipulation language. Technical report, UMBC, Baltimore, MD, March 1992.

[Finin *et al.*, 1993] Tim Finin, Jay Weber, Gio Wiederhold, Michael Genesereth, Richard Fritzson, Donald McKay, James McGuire, Richard Pelavin, Stuart Shapiro, and Chris Beckauthor. Specification of the KQML agent-communication language. Technical report, The DARPA Knowledge Sharing Initiative External Interfaces Working Group, June 1993.

[Finin *et al.*, 1997] Tim Finin, Yannis Labrou, and James Mayfield. KQML as an agent communication language. In Jeffrey M. Bredshaw, editor, *Software Agents*, chapter 14, pages 291–316. AAAI Press/MIT Press, Menlo Park, CA/Cambridge, MA, 1997.

[Fisher, 1994] M. Fisher. A survey of concurrent METATEM—the language and its applications. In D. M. Gabbay and H. J. Ohlbach, editors, *Proc. 1st International Conference on Temporal Logic*, pages 480–505. Springer, 1994. LNAI 827.

[Forbus, 1984] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[Fox *et al.*, 1989] Mark S. Fox, Norman Sadeh, and Can Baykan. Constraied heuristic search. In *Proc. 11th IJCAI*, pages 309–315, Detroit, MI, August 1989. Morgan Kaufmann.

[Gallier, 1986] Jean H. Gallier. *Logic for Computer Science*. Harper and Row, New York, NY, 1986.

[Genesereth and Ketchpel, 1994] Michael R. Genesereth and Steven P. Ketchpel. Software agents. *Communications of the ACM*, 37(7):48–53, 147, July 1994.

[Genesereth and Singh, 1993] Michael R. Genesereth and Narinder Singh. A knowledge sharing approach to software interoperation. Report Logic-93-12, Stanford University, Stanford, CA, February 1993.

[Genesereth *et al.*, 1992] Michael R. Genesereth, Richard E. Fikes, Daniel Bobrow, Ronald Brachman, Thomas Gruber, Patrick Hayes, Reed Letsinger, Vladimir Lifschitz, Robert MacGregor, John McCarthy, Peter Norvig, Ramesh Patil, and Len Schubert. Knowledge interchange format version 3.0 reference manual. Report Logic-92-1, Stanford University, Stanford, CA, June 1992.

[Genesereth, 1991] Michael R. Genesereth. Knowledge interchange format. In *Proc. 2nd KR*, pages 599–600, Cambridge, MA, 1991. Morgan Kaufmann.

[Gennari *et al.*, 1998] John H. Gennari, William Grosso, and Mark Musen. A method-description language: An initial ontology with examples. In *Proc. 11th Workshop on Knowledge Acquisition, Modeling and Management*, Banff, Canada, April 1998.

[Georgeff, 1982] Michael P. Georgeff. Procedural control in production systems. *Artificial Intelligence*, 18(2):175–201, March 1982.

[Georgeff, 1987] Michael P. Georgeff. Planning. *Annual Reviews in Computing Science*, 2:359–400, 1987. Also in: [Allen *et al.*, 1990, pages 5–25].

[Ghallab *et al.*, 1998] Malik Ghallab, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. *PDDL—The Planning Domain Definition Language.* Yale University, New Haven, CT, March 1998. Draft 1.0.

[Ginsberg, 1986] Allen Ginsberg. A metalinguistic approach to the construction of knowledge base refinement systems. In *Proc. 5th AAAI*, pages 436–441, Philadelphia, PA, August 1986. Morgan Kaufmann.

[Ginsberg, 1987] Matthew L. Ginsberg, editor. *Readings in Nonmonotonic Reasoning.* Morgan Kaufmann, Los Altos, CA, 1987.

[Ginsberg, 1996a] Matthew L. Ginsberg. Do computers need common sense? In Luigia Carlucci Aiello, Jon Doyle, and Stuart Shapiro, editors, *Proc. 5th KR*, pages 620–626, Cambridge, MA, November 1996. Morgan Kaufmann.

[Ginsberg, 1996b] Matthew L. Ginsberg. A new algorithm for generative planning. In Luigia Carlucci Aiello, Jon Doyle, and Stuart Shapiro, editors, *Proc. 5th KR*, pages 186–197, Cambridge, MA, November 1996. Morgan Kaufmann.

[Golding *et al.*, 1987] Andrew Golding, Paul S. Rosenbloom, and John E. Laird. Learning general search control from outside guidance. In *Proc. 10th IJCAI*, pages 334–337, Milan, Italy, August 1987. Morgan Kaufmann.

[Gómez-Pérez, 1998] A. Gómez-Pérez. Knowledge sharing and reuse. In Liebowitz, editor, *Handbook of Applied Expert Systems.* CRC, 1998.

[Green, 1969] Cordell Green. Application of theorem proving to problem solving. In Donald E. Walker and Lewis M. Norton, editors, *Proc. 1st IJCAI*, pages 219–239, Washington, D.C., August 1969. Morgan Kaufmann. Also in: [Allen *et al.*, 1990, pages 67–87].

[Gruber, 1992] Thomas R. Gruber. Ontolingua: A mechanism to support portable onotologies. Technical Report KSL 91-66, Stanford University, Stanford, CA, June 1992.

[Gruber, 1993a] Thomas R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. Technical Report KSL 93-04, Stanford University, Stanford, CA, August 1993.

[Gruber, 1993b] Thomas R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.

[Gruninger and Fox, 1994] Michael Gruninger and Mark S. Fox. An activity ontology for enterprise modelling. In *Proc. Workshop on Enabling Technologies—Infrastructures for Collaborative Enterprises*. West Virginia University, 1994.

[Gruninger *et al.*, 1997] Michael Gruninger, C. Schlenoff, A. Knutilla, and S. Ray. Using process requirements as the basis for the creation and evaluation of process ontologies for enterprise modeling. *ACM SIGGROUP Bulletin Special Issue on Enterprise Modelling*, 18(3), 1997.

[Guha and Lenat, 1990] R. V. Guha and Douglas B. Lenat. Cyc: A midterm report. *AI Magazine*, 11(3):32–59, Fall 1990.

[Guha and Lenat, 1994] R. V. Guha and Douglas B. Lenat. Enabling agents to work together. *Communications of the ACM*, 37(7):127–142, July 1994.

[Haggith, 1995] Mandy Haggith. A meta-level framework for exploring conflicts in multiple knowledge bases. In John Hallam, editor, *Hybrid Problems, Hybrid Solutions*, pages 87–98. IOS Press, Amsterdam, The Netherlands, 1995.

[Harel *et al.*, 1982] David Harel, Dexter Kozen, and Rohit Parikh. Process logic: Expressiveness, decidability, completeness. *Journal of Computer and System Sciences*, 25:144–170, 1982.

[Harel, 1984] David Harel. Dynamic logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic Vol. II*, chapter 10, pages 497–604. D. Reidel Publishing Company, 1984.

[Hayes, 1974] Patrick J. Hayes. Some problems and non-problems in representation theory. In *Proc. AISB Summer Conference*, pages 63–79, University of Sussex, 1974. Also in: [Brachman and Levesque, 1985, pages 4–22].

[Hendrix, 1973] Gary G. Hendrix. Modeling simultaneous actions and continuous processes. *Artificial Intelligence*, 4:145–180, 1973. Also in: [Weld and de Kleer, 1990, pages 64–82].

[Hintikka, 1962] J. Hintikka. *Knowledge and Belief.* Cornell University Press, Ithaca, NY, 1962.

[Howe and Dreilinger, 1997] Adele E. Howe and Daniel Dreilinger. SAVVYSEARCH: A metasearch engine that learns which search engines to query. *AI Magazine*, 18(2):19–25, Summer 1997.

[Huhns and Singh, 1998] Michael N. Huhns and Munindar P. Singh, editors. *Readings in Agents.* Morgan Kaufmann, San Francisco, CA, 1998.

[Ihrig and Kambhampati, 1997] Laurie H. Ihrig and Subbarao Kambhampati. Storing and indexing plan derivations through explanation-based analysis of retrieval failures. *Journal of Artificial Intelligence Research*, 7:161–198, November 1997.

[Jennings, 1996] Nicholas R. Jennings, editor. *Foundations of Distributed Artificial Intelligence.* Wiley, New York, NY, 1996.

[Joslin and Pollack, 1996] David Joslin and Martha E. Pollack. Is "early commitment" in plan generation ever a good idea. In *Proc. 13th AAAI*, pages 1188–1193, Portland, OR, August 1996. AAAI Press/The MIT Press.

[Kambhampati *et al.*, 1996] Subbarao Kambhampati, Suresh Katukam, and Yong Qu. Failure-driven dynamic search control for partial order planners: An explanation-based approach. *Artificial Intelligence*, 88(1–2):253–315, December 1996.

[Kambhampati, 1997] Subbarrao Kambhampati. Challenges in bridging plan synthesis paradigms. In *Proc. 15th IJCAI*, pages 44–49, Nagoya, Japan, August 1997. Morgan Kaufmann.

[Kautz and Selman, 1992] Henry Kautz and Bart Selman. Planning as satisfiability. In Bernd Neumann, editor, *Proc. 10th ECAI*, pages 359–363, Vienna, Austria, August 1992. Wiley.

[Kautz and Selman, 1996] Henry Kautz and Bart Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proc. 13th AAAI*, pages 1194–1201, Portland, OR, August 1996. AAAI Press/The MIT Press.

[Kingston *et al.*, 1996] John Kingston, Nigel Shadbolt, and Austin Tate. CommonKADS models for knowledge-based planning. In *Proc. 13th AAAI*, pages 477–482, Portland, OR, August 1996. AAAI Press/The MIT Press.

[Konolige, 1986] Kurt Konolige. *A Deduction Model of Belief.* Morgan Kaufmann, San Mateo, CA, 1986.

[Kornfeld, 1979] William A. Kornfeld. ETHER–a parallel problem solving system. In *Proc. 6th IJCAI*, pages 490–492, Tokyo, Japan, August 1979. William Kaufmann.

[Kornfeld, 1981] William A. Kornfeld. The use of parallelism to implement a heuristic search. In *Proc. 7th IJCAI*, pages 575–580, Vancouver, Canada, August 1981. University of British Columbia, William Kaufmann.

[Kripke, 1963] S. Kripke. Semantical analysis of modal logic. *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik*, 9:67–96, 1963.

[Kuokka and Harada, 1995a] Daniel Kuokka and Larry Harada. Matchmaking for information agents. In *Proc. 14th IJCAI*, pages 672–678, Montréal, Canada, August 1995. Morgan Kaufmann.

[Kuokka and Harada, 1995b] Daniel Kuokka and Larry Harada. On using KQML for matchmaking. In *Proc. 1st International Conference on Multi-Agent Systems*, pages 239–245, San Francisco, CA, June 1995. AAAI Press/MIT Press.

[Kuokka, 1990] Daniel Kuokka. *The Deliberative Integration of Planning, Execution, and Learning.* PhD thesis, Carnegie Mellon University, Pittsburgh, PA, 1990.

[Labrou and Finin, 1997] Yannis Labrou and Tim Finin. A proposal for a new KQML specification. TR CS-97-03, University of Maryland Baltimore County, Baltimore, MD, February 1997.

[Laird *et al.*, 1987] John E. Laird, Allen Newell, and Paul S. Rosenblum. SOAR: An architecture for general intelligence. *Artificial Intelligence*, 33(1):1–64, 1987.

[Larkin *et al.*, 1980] Jill H. Larkin, John McDermott, Dorothea P. Simon, and Herbert A. Simon. Models of competence in solving physics problems. *Cognitive Science*, 4(4):317–345, October 1980.

[Laske, 1986] Otto E. Laske. On competence and performance notions in expert system design: A critique of rapid prototyping. In *Proc. 6th International Workshop Expert Systems and their Applications*, pages 257–297, Avignon, France, April 1986.

[Leckie and Zukerman, 1991] Christopher Leckie and Ingrid Zukerman. Learning search control rules for planning: An inductive approach. In Lawrence A. Birnbaum and Gregg C. Collins, editors, *Proc. 8th International Workshop on Machine Learning*, pages 422–426, Evanston, IL, June 1991. Northwestern University, Morgan Kaufmann.

[Lecoeuche *et al.*, 1996] Renaud Lecoeuche, Oliver Catinaud, and Catherine Gréboval-Barry. Competence in human beings and knowledge-based systems. In *Proc. 10th Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, November 1996.

[Lee *et al.*, 1996] Jintae Lee, Micheal Grunninger, Yan Jin, Thomas Malone, Austin Tate, Gregg Yost, and other members of the PIF Working Group. The PIF process interchange format and framework version 1.1. Working Paper #194, MIT Center for Coordination Science, Cambridge, MA, May 1996.

[Lenat *et al.*, 1983] Douglas B. Lenat, Randall Davis, Jon Doyle, Michael Genesereth, Ira Goldstein, and Howard Schrobe. Reasoning about reasoning. In Frederick Hayes-Roth, Donald A. Waterman, and Douglas B. Lenat, editors, *Building Expert Systems*, chapter 7, pages 219–239. Addison-Wesley, Reading, MA, 1983.

[Lenat, 1995] Douglas B. Lenat. CYC: A large-scale investment in knowledge infrastructure. *Communications of the ACM*, 38(11):33–38, November 1995.

[Levesque, 1984] Hector J. Levesque. A logic of implicit and explicit belief. In *Proc. 4th AAAI*, pages 198–202, Austin, TX, August 1984. University of Texas, William Kaufman.

[Lifschitz, 1986] Vladimir Lifschitz. On the semantics of STRIPS. In Michael P. Georgeff and Amy L. Lansky, editors, *Proc. Workshop on Reasoning about Actions and Plans*, pages 1–9, Timberline, Oregon, July 1986. Morgan Kaufmann. Also in: [Allen *et al.*, 1990, pages 523–530].

[Loveland, 1978] Donald W. Loveland. *Automated Theorem Proving: A Logical Basis.* Fundamental Studies in Computer Science Vol. 6. North-Holland, Amsterdam, The Netherlands, 1978.

[Lydiard, 1996] Terri Lydiard. Using IDEF3 to capture the air campaign planning process. University of Edinburgh, Scotland, March 1996.

[Maes and Nardi, 1988] Pattie Maes and Daniele Nardi, editors. *Meta-Level Architectures and Reflection.* North-Holland, Amsterdam, The Netherlands, 1988.

[Maes, 1986] Pattie Maes. Introspection in knowledge representation. In *Proc. 7th ECAI, Vol. I*, pages 256–269, Brighton, UK, July 1986.

[Malone *et al.*, 1997] Thomas W. Malone, Kevin Crowston, Jintae Lee, Brian Pentland, Chrysanthos Dellarocas, George Wyner, John Quimby, Charley Osborn, and Abraham Bernstein. Tools for inventing organizations: Toward a handbook of organizational processes. MIT, Cambridge, MA, 1997.

[Mayer *et al.*, 1992] R. J. Mayer, T. P. Cullinane, P. S. deWitte, W. B. Knappenberger, B. Perakath, and M. S. Wells. Information integration for concurrent engineering (IICE) IDEF3 process description capture method report. Report AL-TR-1992-0057, Armstrong Laboratory, Logistics Research Division, 1992.

[McAllester and Rosenblitt, 1991] D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *Proc. 9th AAAI*, pages 634–639, Anaheim, CA, August 1991. AAAI Press/The MIT Press.

[McCarthy and Hayes, 1969] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In Bernhard Meltzer and Donald Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, Edinburgh, Scotland, 1969. Also in: [Allen *et al.*, 1990, pages 393–435].

[McCarthy, 1980a] John McCarthy. Applications of circumscription to formalizing commonsense knowledge. *Artificial Intelligence*, 28:89–116, 1980.

[McCarthy, 1980b] John McCarthy. Circumscription—a form of nonmonotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[Minton and Carbonell, 1987] Steven Minton and Jaime G. Carbonell. Strategies for learning search control rules: An explanation-based approach. In *Proc. 10th IJCAI*, pages 228–235, Milan, Italy, August 1987. Morgan Kaufmann.

[Minton *et al.*, 1985] Steven Minton, Philip J. Hayes, and Jill Fain. Controlling search in flexible parsing. In *Proc. 9th IJCAI*, pages 785–787, Los Angeles, CA, August 1985. Morgan Kaufmann.

[Minton *et al.*, 1987] Steven Minton, Jaime G. Carbonell, Oren Etzioni, Craig A. Knoblock, and Daniel R. Kuokka. Acquiring effective search control rules: Explanation-based learning in the PRODIGY system. In Pat Langley, editor, *Proc. 4th International Workshop on Machine Learning*, pages 122–133, Irvine, CA, June 1987. University of California, Morgan Kaufmann.

[Minton *et al.*, 1989] Steven Minton, Craig A. Knoblock, Daniel R. Kuokka, Yolanda Gil, Robert L. Joseph, and Jaime G. Carbonell. PRODIGY 2.0: The manual and tutorial. Technical Report CMU-CS-89-146, Carnegie Mellon University, Pittsburgh, PA, May 1989.

[Moore, 1985] Robert C. Moore. A formal theory of knowledge and action. In Jerry R. Hobbs and Robert C. Moore, editors, *Formal Theories of the Commonsense World*, chapter 9, pages 319–358. Ablex, Norwood, NJ, 1985. Also in: [Allen *et al.*, 1990, pages 480–519].

[Murray and Porter, 1989] Kenneth S. Murray and Bruce W. Porter. Controlling search for the consequences of new information during knowledge integration.

In Alberto Maria Segre, editor, *Proc. 6th International Workshop on Machine Learning*, pages 290–295, Ithaca, NY, June 1989. Cornell University, Morgan Kaufmann.

[Musen, 1989] Mark A. Musen. Automated support for building and extending expert models. *Machine Learning*, 4:349–377, 1989.

[Neches *et al.*, 1991] Robert Neches, Richard Fikes, Tim Finin, Thomas Gruber, Ramesh Patil, Ted Senator, and William R. Swartout. Enabling technologies for knowledge sharing. *AI Magazine*, 12(3):36–56, Fall 1991.

[Newell and Simon, 1963] Allen Newell and Herbert A. Simon. GPS, a program that simulates human thought. In E. A. Feigenbaum and J. Feldman, editors, *Computers and Thought*, pages 279–293. R. Oldenbourgh KG, 1963. Also in: [Allen *et al.*, 1990, pages 59–66].

[Newell and Simon, 1976] Allen Newell and Herbert Simon. Computer science as empirical enquiry. *Communications of the ACM*, 19:113–126, 1976.

[Newell, 1982] Allen Newell. The knowledge level. *Artificial Intelligence*, 18(1):87–127, January 1982.

[Nilsson, 1980] Nils J. Nilsson. *Principles of Artificial Intelligence*. Tioga, Palo Alto, CA, 1980.

[Nodine and Unruh, 1997] Marian Nodine and Amy Unruh. Facilitating open communication in agent systems: The InfoSleuth infrastructure. In N. Singh, A. Rao, and m. Wooldridge, editors, *Proc. 4th International Workshop on Agent Theories, Architectures, and Languages*, pages 281–295, Providence, RI, July 1997.

[Nodine *et al.*, 1998] Marian Nodine, Brad Perry, and Amy Unruh. Experience with the InfoSleuth agent architecture. In Brian Logan and Jeremy Baxter, editors, *Proc. AAAI Workshop on Software Tools for Developing Agents*, Madison, WI, January 1998. AAAI Press.

[O-Plan TF, 1997] AIAI, University of Edinburgh, Edinburgh, Scotland. *Task Formalism Manual*, January 1997. Version 3.1.

[Pease and Carrico, 1997] R. Adam Pease and Todd M. Carrico. Core plan representation. Armstrong Lab Report AL/HR-TP-96-9631, Armstrong Laboratory, US Air Force, January 1997. Object Modeling Working Group.

[Pednault, 1989] Edwin P. D. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter, editors, *Proc. 1st KR*, pages 324–332, Toronto, Canada, 1989. Morgan Kaufmann.

[Penberthy and Weld, 1992] J. Scott Penberthy and Daniel S. Weld. UCPOP: A sound, complete, partial order planner for ADL. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *Proc. 3rd KR*, pages 103–114, Cambridge, MA, October 1992. Morgan Kaufmann.

[Peot and Smith, 1992] Mark A. Peot and David E. Smith. Conditional nonlinear planning. In James Hendler, editor, *Proc. 1st International Conference on Artificial Intelligence Planning Sytems*, pages 189–197, College Park, MD, June 1992. Morgan Kaufmann.

[Pryor and Collins, 1996] Louise Pryor and Gregg Collins. Planning for contingencies: A decision-based approach. *Journal of Artificial Intelligence Research*, 4:287–339, May 1996.

[Pryor, 1996] Louise Pryor. Opportunity recognition in complex environments. In *Proc. 13th AAAI*, pages 1147–1152, Portland, OR, August 1996. AAAI Press/The MIT Press.

[Reiter, 1980] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Rosenblum *et al.*, 1993] Paul S. Rosenblum, John E. Laird, and Allen Newell, editors. *The* SOAR *Papaers: Readings on Integrated Intelligence*, volume I & II. MIT Press, Cambridge, MA, 1993.

[Russell and Norvig, 1995] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, Upper Saddle River, NJ, 1995.

[Secker, 1988] Judith A. Secker. Use of O-Plan for oil platform construction project planning. AIAI-PR 22, AIAI, University of Edinburgh, Edinburgh, Scotland, June 1988.

[Seel, 1989] N. Seel. *Agent Theories and Architectures.* PhD thesis, Surrey University, Guildford, UK, 1989.

[Selman *et al.*, 1992] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proc. 10th AAAI*, pages 440–446, San Jose, CA, July 1992. AAAI Press/The MIT Press.

[Selman, 1994] Bart Selman. Near-optimal plans, tractability, and reactivity. In Jon Doyle, Erik Sandewall, and Pietro Torasso, editors, *Proc. 4th KR*, pages 521–529, Bonn, Germany, May 1994. Morgan Kaufmann.

[Shanahan, 1997] Murray Shanahan. *Solving the Frame Problem.* MIT Press, Cambridge, MA, 1997.

[Shoham, 1993] Y. Shoham. Agent-oriented programming. *Artificial Intelligence*, 60(1):51–92, 1993.

[Singh, 1993a] Narinder Singh. A Common Lisp API and facilitator for ABSI. Report Logic-93-4, Stanford University, Stanford, CA, January 1993.

[Singh, 1993b] Narinder P. Singh. Implementation details for the new ABSI facilitator. Stanford University, Stanford, CA, April 1993.

[Smith, 1977] Reid G. Smith. The CONTRACT NET: A formalism for the control of distributed problem solving. In *Proc. 5th IJCAI*, page 472, Cambridge, MA, August 1977. MIT, William Kaufmann.

[SPAR, 1997] DARPA/Rome Laboratory. *Planning Initiative Shared Planning and Activity Representation*–SPAR, October 1997. Version 0.1.

[Tate *et al.*, 1990] Austin Tate, James Hendler, and Mark Drummond. A review of AI planning techniques. In James Allen, James Hendler, and Austin Tate, editors, *Readings in Planning*, pages 26–49. Morgan Kaufmann, San Mateo, CA, 1990.

[Tate *et al.*, 1994] Austin Tate, Brian Drabble, and Richard Kirby. O-Plan2: An open architecture for command, planning and control. In Monte Zweben and Mark S. Fox, editors, *Intelligent Scheduling*, chapter 7, pages 213–239. Morgan Kaufmann, San Francisco, 1994.

[Tate *et al.*, 1998] Austin Tate, Stephen T. Polyak, and Peter Jarvis. TF method: An initial framework for modelling and analysing planning domains. In *Proc. Knowledge Engineering and Acquisition for Planning: Bridging Theory and Practice*, Pittsburgh, PA, June 1998. Carnegie-Mellon University, AAAI Press.

[Tate, 1975] Austin Tate. *Using Goal Structure to Direct Search in a Problem Solver.* PhD thesis, University of Edinburgh, Edinburgh, Scotland, 1975.

[Tate, 1995] Austin Tate. Integrating constraint management into an AI planner. *Artificial Intelligence in Engineering*, 9:221–228, 1995.

[Tate, 1996a] Austin Tate. Representing plans as a set of constraints — the <I-N-OVA> model. In Brian Drabble, editor, *Proc. 3rd International Conference on Artificial Intelligence Planning Sytems*, pages 221–228, Edinburgh, Scotland, May 1996. AAAI Press.

[Tate, 1996b] Austin Tate. Towards a plan ontology. *AI*IA Notizige (Quarterly Publication of the Associazione Italiana per l'Intelligenza Artificiale)*, 9(1):19–26, March 1996.

[Tate, 1998] Austin Tate. Roots of SPAR—shared planning and activity representation. *The Knowledge Engineering Review*, 13(1):121–128, March 1998.

[Uschold *et al.*, 1996] Mike Uschold, Martin King, Stuart Moralee, and Yannis Zorgios. The enterprise ontology. Technical Report AIAI-TR-195, AIAI, University of Edinburgh, Edinburgh, Scotland, August 1996.

[Valente, 1994] André Valente. Planning. In Joost Breuker and Walter Van de Velde, editors, *CommonKADS Library for Expertise Modelling*, chapter 10, pages 213–229. IOS Press, Amsterdam, 1994.

[Valente, 1995] André Valente. Knowledge-level analysis of planning systems. *SIGART Bulletin*, 6(1):33–41, January 1995.

[van Harmelen and Balder, 1992] Frank van Harmelen and J. R. Balder. $(\mathrm{ML})^2$: A formal language for KADS models of expertise. *Knowledge Acquisition*, 4(1), March 1992.

[van Harmelen and ten Teije, 1998] Frank van Harmelen and Annette ten Teije. Characterising problem-solving methods by gradual requirements: Overcoming the yes/no distinction. In *Proc. 8th Knowledge Engineering: Methods and Languages*, Karlsruhe, Germany, January 1998. University of Karlsruhe.

[VanLehn and Jones, 1991] Kurt VanLehn and Randolph M. Jones. Learning physics via explanation-based learning of correctness and analogical search control. In Lawrence A. Birnbaum and Gregg C. Collins, editors, *Proc. 8th International Workshop on Machine Learning*, pages 110–114, Evanston, IL, June 1991. Northwestern University, Morgan Kaufmann.

[Veloso *et al.*, 1995] Manuela Veloso, Jaime Carbonell, Alicia Perez, Daniel Borrajo, Eugene Fink, and Jim Blythe. Integrating planning and learning: The PRODIGY architecture. *Journal of Theoretical and Experimental Artificial Intelligence*, 7(1), 1995.

[Voß *et al.*, 1990] Angi Voß, Werner Karbach, Uwe Drouven, and Darius Lorek. Competence assessment in configuration tasks. In *Proc. 9th ECAI*, pages 676–681, Stokholm, Sweden, August 1990. Pitman.

[Warren, 1976] David H. D. Warren. Generating conditional plans and programs. In *Proc. AISB*, pages 344–354, Edinburgh, Scotland, July 1976. University of Edinburgh.

[Wavish, 1992] P. Wavish. Exploiting emergent behaviour in multi-agent systems. In E. Werner and Y. Demazeau, editors, *Proc. 3rd European Workshop on Modelling Autonomous Agents and Multi-Agent Worlds*, pages 297–310. Elsevier Science Publishers, 1992.

[Wefald and Russell, 1989] Eric H. Wefald and Stuart J. Russell. Adaptive learning of decision-theoretic search control knowledge. In Alberto Maria Segre, editor, *Proc. 6th International Workshop on Machine Learning*, pages 408–411, Ithaca, NY, June 1989. Cornell University, Morgan Kaufmann.

[Weld and de Kleer, 1990] Daniel S. Weld and Johan de Kleer, editors. *Readings in Qualitative Reasoning about Physical Systems.* Morgan Kaufmann, San Mateo, CA, 1990.

[Weld, 1996] Daniel S. Weld. Planning-based control of software agents. In Brian Drabble, editor, *Proc. 3rd International Conference on Artificial Intelligence Planning Sytems*, pages 268–274, Edinburgh, Scotland, May 1996. AAAI Press.

[Wickler and Pryor, 1996] Gerhard Wickler and Louise Pryor. On competence and meta-knowledge. In Milind Tambe and Piotr Gmytrasiewicz, editors, *Proc. AAAI Workshop on Agent Modeling*, pages 98–104, Portland, OR, August 1996. AAAI Press, Menlo Park, CA.

[Wickler, 1999] Gerhard Wickler. *Using Expressive and Flexible Action Representations to Broker Capabilities for Intelligent Agent Cooperation.* PhD thesis, University of Edinburgh, Edinburgh, Scotland, 1999. (To appear).

[Wielinga and Breuker, 1986] Bob J. Wielinga and Joost A. Breuker. Models of expertise. In *Proc. 7th ECAI, Vol. I*, pages 306–318, Brighton, UK, July 1986.

[Wielinga *et al.*, 1992] B. J. Wielinga, A. Th. Schreiber, and J. A. Breuker. KADS: A modelling approach to knowledge engineering. *Knowledge Acquisition*, 4(1):5–53, March 1992.

[Wielinga (ed) *et al.*, 1994] Bob Wielinga (ed), Hans Akkermans, Heshem Hassan, Olle Olsson, Klas Orsvärn, Guus Schreiber, Peter Terpstra, Walter Van de Velde, and Steve Wells. Expertise model definition document. Report KADS-II/M2/UvA/026/5.0, University of Amsterdam, Amsterdam, The Netherlands, June 1994.

[Wilensky, 1981] Robert Wilensky. Meta-planning: Representing and using knowledge about planning in problem solving and natural language understanding. *Cognitive Science*, 5(3):197–233, July 1981.

[Wilkins, 1982] David E. Wilkins. Using knowledge to control tree searching. *Artificial Intelligence*, 18(1):1–51, January 1982.

[Wilkins, 1988] David E. Wilkins. *Practical Planning.* Representation and Reasoning Series. Morgan Kaufmann, San Mateo, CA, 1988.

[Witten *et al.*, 1994] Ian H. Witten, Alistair Moffat, and Timothy C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images.* Van Nostrand Reinhold, New York, NY, April 1994.

[Wooldridge and Jennings, 1995] Michael Wooldridge and Nicholas R. Jennings. Intelligent agents: Theories and practice. *The Knowledge Engineering Review*, 10(2):115–152, June 1995.

[Wooldridge, 1994] Michael Wooldridge. Coherent social action. In A. G. Cohn, editor, *Proc. 11th ECAI*, pages 279–283, Amsterdam, The Netherlands, August 1994. Wiley.

[Zaniolo, 1991] C. Zaniolo. The logical data language (LDL): An integrated approach to logic and databases. Technical Report STP-LD-328-91, MCC, Austin, TX, 1991.