Capability Assessment in UCPOP: Some Experiments in the Blocks World

Gerhard Wickler
Department of Artificial Intelligence, University of Edinburgh
80 South Bridge, Edinburgh EH1 1HN, Scotland
gw@aisb.ed.ac.uk

Abstract

This paper defines and addresses the competence and capability assessment problem. A simple method based on the meta-knowledge available in a reasoning system is given that can be used to assess competence. This idea is applied to the planner UCPOP to assess capability in the Blocks World domain. Several theoretical and practical problems lead to some experiments that involve generating partial search spaces and analyzing those to assess capability. A number of features is given that can be used to decide the capability assessment problem and the experiments show how these can be used to derive thresholds for a domain that indicate capability in this domain.

1 Competence and Capability Assessment

1.1 A Unified Method for Assessment

> "The intended role of knowledge representation in artificial intelligence is [to] reduce problems of intelligent action to search problems" [Ginsberg, 1993; page 38].

Knowledge representation is the process of encoding the knowledge needed to solve a range of problems in some formalism. A problem can then be solved by searching for a solution to a given problem instance. The **competence assessment problem** is the problem of assessing whether the search process will be successful.[1] In a knowledge-based system this search is called reasoning.

[1]This would mean that we are always competent in finite search spaces. However, there are many search spaces that are finite in theory but infinite for all practical purposes today, e.g. the potential configurations of a chess board. Thus, what we really want to assess is whether the search will be successful within reasonable resources.

One of the problems with search is that the most general techniques are also generally the least efficient. Different techniques for reasoning have therefore been developed in AI to exploit different features of the representation, features of the task, or features of the domain at hand. For example, a resolution-based theorem prover might employ knowledge about the length of the generated clauses, a planner might assume that the only changes to the world are the ones mentioned in the definition of the operators, or a game-playing system might use a sophisticated evaluation function or exploit symmetries. A number of well-known techniques for representation and reasoning over these representations is described in [Brachman and Levesque, 1985].

One specific kind of reasoning is *planning*. Planning is of particular interest because a planner is often assumed to be a central component of an artificial agent [Wooldridge and Jennings, 1995; Weld, 1996]. There are a number of techniques for AI planning [Allen *et al.*, 1990]. Often planning is not the only task an intelligent agent has to perform. The agent might also be required to execute the generated plan [Pryor, 1994] and perform other types of reasoning or non-planned actions. We will refer to the question whether an agent *can* achieve some state of the world as the **capability assessment problem**.

Since attempting to achieve a state of the world will often involve planning, capability assessment will involve competence assessment for the according planning problem. On the other hand, reasoning is a kind of action and knowledge states to be achieved can be seen as goals for a planner and thus, competence assessment can be seen as a specific kind of capability assessment. We will use the term competence assessment for general reasoning problems and use capability assessment for reasoning about actions involving planning.

Two questions that have to be answered to address the competence and capability assessment problem respectively are the following:

1. Can I derive $< proposition >$?[2]

   e.g. Mont Blanc is x metres high, etc.

2. Can I achieve $< state >$?

   e.g. being on top of Mont Blanc, etc.

The **basic method** of how competence assessment can be done could, if not must, look something like this:

> We have to abstract from the given problem and from the knowledge base that will be used to try to solve this problem and test whether the two match in some way.[3]

[2]This question can be seen as asking about the agent's *knowledge* if the *knowledge* refers to the deductive closure of the knowledge base.

[3]For a more elaborate discussion of this rule cf. [Wickler and Pryor, 1996].

1.2 A Fundamental Problem

One problem concerning competence assessment is that derivation of a proposition can be hard. Only in a simple formalism like propositional logic is it possible to decide whether a given proposition follows from the agent's knowledge base or not. This can be done simply by trying to derive the given proposition. However, this is not addressing the competence question directly; it is trying to solve the problem and obviously gives no efficiency gain. Furthermore, if the underlying formalism is more complex, for example, first order logic, then the problem is known to be undecidable. Notice that the difficulty comes from the problem itself.

As useful knowledge representation formalisms will usually be undecidable, we can only hope for an *approximate* answer to a competence question about propositions. That is, we hope for an answer like "Yes, it is *likely* that I can solve this problem." In this way we could overcome the undecidability and efficiency problem mentioned above.

The second kind of question, the capability question, has the same problem associated with it. In general it is not possible to decide whether there is a plan that achieves a given set of goals. Hence, one can only get an approximate assessment here as well.

Looking at human problem solvers, we find the same behaviour: the initial assessment might be wrong. This is simply a result of the fact that the competence or capability decision is based upon an abstraction of the given problem, and some details might reveal the problem as more difficult than initially thought.

1.3 Exploiting Necessary Criteria

One way to assess problem-solving ability is to use necessary criteria that can be evaluated quickly. One such criterion is whether the agent knows about the concepts and instances involved in the problem at hand. If there is no knowledge available about those to the agent then it will most likely not be able to solve the problem.

Although there is a philosophical problem involved here, namely what it means to *know* something about an instance or concept [Russell, 1910; Hintikka, 1962], we can efficiently check whether a knowledge base contains knowledge about some instance or concept that is represented by a given symbol. If the knowledge base does not contain the symbol then there is no knowledge about the thing it represents available.

There are also domain dependent criteria that can be used to assess competence and capability. For example, [Voß *et al.*, 1990] describe a number of knowledge sources that can be used to assess competence in an assignment task.

2 The Competence Assessment Problem

2.1 Meta-Knowledge Indicates Competence

Competence knowledge is *knowledge about other knowledge*. Our claim here is that meta-rules [Davis and Buchanan, 1977] contain knowledge about competence. Given a problem instance with its goal we can use the following simple rule to assess competence:

> If **there is a meta-rule** applicable to the given problem instance that tells us which object-level knowledge to apply then it is likely that **we are competent** with respect to this problem instance.

The intuition behind this rule is quite simple: in trying to assess our competence for a given problem we seek ways to approach the problem. If we know of a promising way to tackle the problem then we assume that we are competent. The existence of meta-knowledge that tells the inference engine how to proceed in the search space represents exactly the condition of knowing how to tackle the problem. Of course, it is possible that after actually starting to solve the problem the system discovers that the promising approach did not lead to the solution. This simply accounts for the fact that the competence assessment problem is not decidable in general.

2.2 Some Possible Objections

There are a number of possible objections at this point. For example, if the meta-knowledge still leaves us with a considerable number of promising rules then we know of quite a few ways to approach to problem which might be interpreted as not really knowing how to tackle the problem. Another objection could be that meta-rules do not contain all the knowledge we need to assess competence and we will fully agree with this point. We might need to explicitly represent further knowledge to get to a better assessment. Yet another objection could be that a rule-based approach is not general enough. However, a production system framework covers many AI techniques [Nilsson, 1980]. Still, one could argue that, for example, a case-based reasoning approach or a constraint-based approach is more suitable for the competence assessment problem. This is missing the point though.

The problem we are given is to decide on the basis of the knowledge-base (including all levels of knowledge) and the problem instance at hand whether it is likely that we can solve the problem. There are two issues here: how to abstract from the problem instance and knowledge base; and how to evaluate whether the two abstractions match. We believe that the hard part is the abstraction process because one has to identify some general features in terms of which the problem instance can be described. This problem is similar to the one in earlier work on concept learning systems: if the right attribute is not given to the system any mechanism must fail to derive an appropriate concept description. Only once the potentially important features have been identified it does makes sense to think about the mechanism. This is also true for case-based reasoning: extracting the right features to classify and retrieve cases is a fundamental problem.

So why meta-rules? What we have shown above is how to use the knowledge in the meta-rules to get to an abstraction of the problem instance. Meta-rules perform a kind of abstraction from the problem instance and the object-level knowledge to decide what to do next. Thus, there are already a number of features contained in the meta-knowledge that are connected to features of the knowledge base. Instead of inventing new features, representing them in a different formalism, and doing the evaluation this way, it is more sensible to reuse the meta-knowledge in the system to decide the competence assessment problem.

2.3 Competence and Relevance

The above rule can be used to assess competence; it addresses the question *whether we have the right knowledge* to solve the given problem instance. Two points need to be mentioned here. Firstly, as pointed out above, the assessment is only approximate. Secondly, having the right knowledge is only a necessary criterion for competence. Other criteria like problem size and solvability need to be taken into account as well.

So how is competence related to relevance? We believe that having the right knowledge to solve a problem means having knowledge that is **relevant** to the problem at hand.

2.4 Looking Ahead

Norman and Bobrow address a problem related to assessing problem-solving ability [Norman and Bobrow, 1975]. The aim of their work was to evaluate whether it is worthwhile to pursue a certain line of reasoning given only limited (computational) resources. If the resources are unlikely to be sufficient then alternative lines of reasoning should be explored. The method they used works as follows:

1. A line of reasoning is followed for a number of steps;

2. Then the reasoning process itself is interrupted and the distance to the goal is estimated.

3. This together with the amount of resources used up up to now allows for an evaluation of the possibility to achieve the goal.

This also looks like a way to solve the competence assessment problem. However, the trouble with this method lies in estimating the distance to the goal. Depending on the underlying type of reasoning this can be a very difficult problem in itself. For example, deriving a proposition in first order logic can mean resolution theorem proving. The distance to the goal in this case is the number of clauses we still need to generate before we arrive at the empty clause. There does not seem to be a good estimate available. Similarly for planning, after achieving a number of goals it is not possible to decide whether the partial plan represents one that can be extended to a working plan that achieves all given goals.

What we want to take from Norman and Bobrow's work here is the idea of *looking ahead* to achieve a better competence assessment. Instead of using the rule to judge whether we know how to approach the given problem we might as well start the reasoning process and follow the most promising approach according to the meta-knowledge up to a certain point. This can be until a fixed number of steps have been taken, until a given branching factor is reached, or until no promising object-level knowledge seems available anymore. Then we could use the competence assessment rule to judge problem-solving ability at this point in the search. Since this assessment is based on more and better information we would expect it to be at least as good as an assessment based only on the given problem instance, if not better.

3 The Capability Assessment Problem

3.1 Meta-Knowledge and Capability

Planning is a specific kind of reasoning and therefore the rule for competence assessment given in the previous section should also be applicable to assess capability if we restrict the assessment to the planning phase. This requires the presence of meta-rules in the planning agent's knowledge base.

The classic planning problem [Wilkins, 1988] is very much domain independent. Meta-rules on the other hand do refer to the domain. Since classic planners almost by definition do not contain domain specific knowledge and neither do they allow for it to be specified as input, this knowledge is not available to them. If the planner knew about the primitive actions it can perform, i.e. had domain specific knowledge, then there is no reason why it should not also have knowledge *about* these actions and associated goals as well, i.e. the meta-knowledge. Since agents need this kind of knowledge about themselves there is no reason why they should not also have domain specific meta-knowledge. Hence, planning agents, unlike classical planners, should have the meta-knowledge needed for capability assessment.

Meta-rules can be used for competence assessment because they contain the knowledge how to approach the given problem. They help in deciding how to proceed at the non-deterministic steps in search, e.g. which applicable rule to fire next. The same idea can be used to assess capability. Details must obviously depend on the actual planning algorithm used.

3.2 Planning Algorithms: UCPOP

There are a number of different planning algorithms based on different ideas. It is beyond this paper to look at all of them but for a fairly recent and exhaustive overview see [Allen *et al.*, 1990]. In this paper we will concentrate on one planning algorithm, UCPOP [Penberthy and Weld, 1992; Barrett *et al.*, 1995].

UCPOP searches the space of plan states to find a (complete) plan that achieves the given goals from the given initial world state. A plan is a set of *steps* that are actions to be executed. Each step is included to satisfy a goal or a precondition for other steps. The information which goals or preconditions are satisfied by the steps is part of the plan and is stored in the *causal links*. Furthermore, a plan contains a set of *flaws*, i.e. the reasons why this plan is not yet complete. These are, for example, yet unsatisfied preconditions. Finally, a plan contains *ordering constraints* on the steps and *binding constraints* for the variables used.

The algorithm starts with the initial plan which contains two steps: one that results in the initial state and one the requires the goal state in this order. Basically what UCPOP does is to select a plan and refine it by removing one of its flaws, thereby creating a new plan, until a plan with no flaws is found. To refine a plan a flaw has to be selected and a way to remove this flaw has to be chosen. Thus there are three non-deterministic choices at which meta-knowledge can be used to guide search in UCPOP:

- Select a plan to be refined;
- select a flaw in the currently selected plan;
- select a way to remove this flaw from the plan.

Version 4 of UCPOP comes with a number of domains like, for example, the Blocks World, a traveling domain, the Towers of Hanoi domain, etc. For each of these domains there are a few problem instances defined, some of which can be solved by UCPOP in reasonable time.

UCPOP also contains a search controller that allows explicit meta-rules to be specified that can guide the non-deterministic choices as mentioned above. Several of the test domains have such search control knowledge associated with them like, for example, the Prodigy Blocks World, the domain we have chosen to use for the experiments described later.

3.3 Some Problems Related to Capability Assessment

As mentioned above, one of the major advantages we see in our approach of using meta-knowledge for capability assessment is that this knowledge will be re-used rather than duplicated in a different formalism. To show that this is possible we need to look at existing domains that already contain meta-knowledge. It is surely not difficult to come up with some meta-knowledge that can be used for capability assessment, but this would not substantiate our re-usability claim. Only if the existing knowledge can be used to get a good approximation for the capability assessment then the argument about re-use is valid. Since UCPOP provides us with a number of such domains it appears to be a good planner to evaluate our ideas.

However, there are some problems in using the UCPOP example domains for the evaluation of our capability assessment ideas. Firstly, the domains are too different. Giving a problem instance to the planner with a domain that is not the domain the problem instance was intended to be solved in will lead to immediate failure in the planning process. The reason for this is that the predicates used to describe the different domains do not overlap and thus, there is usually only one domain that contains operators that can achieve the goals of a problem instance. In realistic scenario we cannot expect this to be the case. The main problem for our work here is that a capability assessment phase preceding the actual problem-solving activity is not required for the domains provided with UCPOP.

A second problem for the evaluation of our ideas was discovered when looking at the implementation of the search control language that is used in UCPOP. They use a rule-based approach based on the ideas in Prodigy [Minton *et al.*, 1989]. However, some rules use predicates that are defined with some LISP code rather than having an explicit semantics and thus, cannot be considered an explicit representation of meta-knowledge that can be used directly to assess capability. Furthermore, the rules are used to compute an evaluation function that is used to decide where to continue the search. This imposes a total order on the open nodes where no such knowledge is available. The meta-rules themselves only impose a partial order on these nodes.

4 Experiments with UCPOP

4.1 Experiment Design and Expectations

The design of the experiments had to address the problems mentioned above. What we have done to evaluate our ideas in the UCPOP domains is to compare the planner using domain-dependent meta-knowledge with one that uses only domain-independent search control knowledge. This addresses the problem of the domains being too different. We make the assumption that by using domain-dependent meta-knowledge the planner becomes more competent in this domain, an assumption that can be supported by the fact that this planner will also perform much better in this domain. Thus, what we expect is that the domain-dependent meta-knowledge indicates capability in some way.

Since the meta-rules contain some predicates with a semantics that can only be inspected by evaluating the matching LISP code, we have decided to generate a small portion of the search space and then inspect the generated tree. Generating only a small part of the search space can be done relatively fast. Obviously, if we find the solution to the given problem within this limit we know that we are competent. Otherwise we want to analyze the generated search space in terms of features that indicate capability. The features we have used are the following:

- **size**: the number of nodes in the search tree
- **intern**: the number of internal nodes
- **leafs**: the number of leaf nodes (open nodes plus nodes without children)
- **av-bf**: the average branching factor ((size-1) / intern)
- **depth**: depth of the deepest node in the tree (root = 0)
- **h-open**: the depth of the highest node in the tree that is still to be expanded (excluding nodes without children)

The number of nodes generated, the number of internal nodes, and the number of leaf nodes are just standard features of trees and not intended to indicate competence directly. The average branching factor we would expect to be lower for a more capable planner since there might be children of a node that are not generated because we have meta-knowledge excluding these children from the search. Similarly, we would expect a better guided search to be more focussed. For a search tree of a fixed size this would mean that the depth achieved is higher for the more competent agent. Finally, if a node is unlikely to lead to the goal, we might have meta-knowledge that excludes this node from the search space rather than just giving it a low priority in the queue. Thus, we would expect a search tree generated by a more capable planner to have the open node with the lowest depth in the tree to be deeper, i.e. further away from the root, than such a node in a tree generated by a less capable planner.

The last problem mentioned above, the total order imposed by the evaluation function, could only be addressed by implementing a different search controller that uses the existing meta-knowledge. This has not been done for the experiments described here.

4.2 Experimental Results in the Blocks World

For our first experiment we have chosen the Sussman anomaly as the problem to be solved. It is one of the two problems for the Prodigy Blocks World that comes with UCPOP. Table 1 shows the features of partial search spaces generated with different size limits (columns) generated by UCPOP exploiting the domain specific

limit	5	10	20	30	50	70	100	1000
size	7	14	22	33	54	72	94	94
intern	5	9	15	22	34	43	54	54
leafs	2	5	7	11	20	29	40	40
av-bf	1.20	1.44	1.40	1.45	1.56	1.65	1.72	1.72
depth	5	7	10	15	15	21	28	28
h-open	4	6	8	8	12	12	12	12

Table 1: Features of search spaces generated by UCPOP for the Sussman anomaly in Prodigy's blocks world using meta-knowledge to control search.

limit	5	10	20	30	50	70	100	1000
size	7	12	22	34	52	72	102	1002
intern	3	6	10	16	27	39	54	578
leafs	4	6	12	18	25	33	48	424
av-bf	2.00	1.83	2.10	2.06	1.89	1.82	1.87	1.73
depth	3	4	7	12	20	20	22	37
h-open	2	3	3	3	3	4	4	4

Table 2: Features of search spaces generated by UCPOP for the Sussman anomaly in Prodigy's blocks world using best-first search.

search control knowledge. Table 2 gives the same features of search spaces when only best-first search is applied.

Several interesting results can be observed here. Most important, the features meant to indicate capability are consistently different. The average branching factor is between 1.2 and 1.72 for the more capable planner and between 1.73 and 2.1 for the planner not using the domain-specific meta-knowledge. Clearly, this feature seems useful to assess capability. Also interesting is the fact that the average branching factor seems to be even more different the smaller the search space is we are generating. This can be explained by looking at the meta-rules used. One rule, for example, states that a `stack` operator should only be used to achieve goals. This rule affects the search at the beginning but does not prune the search space when preconditions for other operators are tryed to be achieved.

The depth of the search space generated is also a feature that can be used to assess capability. The more capable planner usually generates deeper search trees. However, unlike for the average branching factor, the threshold varies with the size of the search space generated. On average, the trees generated by the more capable planner are about 25% deeper than the trees generated by the planner assumed less capable here. Even more distinguishing is the highest open node. For the less capable planner it never rises above four whereas it goes from

limit	5	10	20	30	50	70	100	1000
size	7	12	25	33	52	72	106	1002
intern	4	8	14	16	30	41	60	583
leafs	3	4	11	17	22	31	46	419
av-bf	1.50	1.38	1.71	2.00	1.70	1.73	1.75	1.72
depth	3	5	7	7	13	20	22	40
h-open	2	3	5	5	7	7	9	14

Table 3: Features of search spaces generated by UCPOP for the p22-problem in Prodigy's blocks world using meta-knowledge to control search.

limit	5	10	20	30	50	70	100	1000
size	8	13	22	32	53	72	102	1003
intern	3	5	10	16	27	37	56	542
leafs	5	8	12	16	26	35	46	461
av-bf	2.33	2.40	2.10	1.94	1.93	1.92	1.80	1.85
depth	3	5	10	12	20	20	20	39
h-open	2	2	2	2	3	3	3	4

Table 4: Features of search spaces generated by UCPOP for the p22-problem in Prodigy's blocks world using best-first search.

four to twelve for the more capable planner. Especially this feature seems to be very reliable measure for capability and shows how well the pruning of the search space works in this case.

Tables 3 and 4 summarize the results for basically the same experiments but with a different problem. This is the second of two problems defined in UCPOP for the Prodigy Blocks world. An interesting feature of this problem is that the goal state is not fully specified in the sense that some blocks mentioned in the initial state are not mentioned in the goal state.

Notice that even with the domain-specific meta-knowledge UCPOP is unable to find a solution within 1000 nodes. Although the more capable planner has the right knowledge to solve the problem, it still cannot solve it within this limit.

The results for this problem are fairly similar to the previous one. The average branching factor is higher for the less capable planner, the depth is higher for the more capable planner, and, again, the highest open node is the most distinctive feature. Notice also that the average branching factor is in similar ranges as for the Sussman anomaly, i.e. the threshold is not specific to the problem.

The above experiments show how the features vary if the size limit of the search space changes. We have also translated the tower inversion problem for three and four blocks into the Prodigy Blocks World and represented an additional

problem	control	size	intern	leafs	av-bf	depth	h-open
sussman	speed	22	15	7	1.40	10	8
	bf	22	10	12	2.10	7	3
p22	speed	25	14	11	1.71	7	5
	bf	22	10	12	2.10	10	2
tower3	speed	23	14	9	1.57	10	7
	bf	22	10	12	2.10	5	3
tower4	speed	23	13	10	1.69	11	6
	bf	24	11	13	2.09	5	2
circle	speed	23	15	8	1.47	9	4
	bf	24	11	13	2.09	5	2

Table 5: Features of search spaces generated by UCPOP for different problems in Prodigy's blocks world with a search limit of 20.

problem in which the goal contains a circle of blocks. The latter problem is, of course, not solvable. The search limit for these problems has been fixed to twenty. Table 5 summarizes the results in terms of features of the generated search spaces. In the second column (control), "speed" stands for the usage of domain-specific search control knowledge and "bf" for a best-first search strategy.

As with the previous problems the features listed can be used to assess capability. The average branching factor falls into ranges consistent with previous experiments and the depth generated by the more capable planner is in most cases considerably higher than for the less capable planner. For a search space limit of twenty, a depth of seven seems to be a good threshold. Similarly, a threshold of four seems to be a good value for the highest open node and, again, this also seems to be the most distinctive feature.

4.3 Breadth-First and Depth-First Search

One reason why the above features all allow the assessment of capability is that the basic search strategy in both cases is the same, best-first search. The difference is that the planner assumed more capable also uses domain-specific meta-knowledge. If the less capable planner used a different basic strategy like breadth-first or depth-first search the results would look quite different. Notice that the more capable planner always does a kind of best-first search by using its additional knowledge.

The reason why the average branching factor is lower for the more capable planner we believe to be that there are some rules that prevent certain children of nodes to be generated. This does not depend on the search strategy and it is quite plausible that this feature will be similarly useful compared to breadth-first or depth-first search. However, as tables 2 and 4 show, the average branching factor can vary the deeper we get into the search space. This means that it is possible that in a depth-first search where a high depth can be expected this feature will not be reliable anymore.

For the depth we have argued that a more capable planner should have a more focussed approach and thus explore the limited search space more in depth. Obviously this still works in comparison with a breadth-first search since this will generate trees of minimal depth. Depth-first search on the other hand will generate trees that will often be deeper than the trees generated by best-first search. This suggests a very focussed search but this is of course not focussed by knowledge, i.e. by being competent, but by random choices made.

The highest open node shows a similar behaviour. In a breadth-first search the highest open node is mostly at the depth of the tree minus one. For a small search space it is possible that this feature will indicate capability but since the depth of the tree will grow only very slowly in breadth-first search, this feature will become meaningful for larger search spaces again. In depth-first search the highest open node will usually be at a very low depth and hence this feature remains a capability indicator. Notice that the conjunction of the depth and the highest open node feature can be used as a strategy independent capability indicator.

5 Conclusions and Future Work

In this paper we have defined and addressed the competence and capability assessment problem. We have argued that meta-knowledge can be used to assess competence: if we know how to approach the problem we are likely to be competent. The main advantage here is that the knowledge represented in meta-rules can re-used to perform the assessment. The meta-knowledge tells us whether we have the right knowledge to address a given problem, i.e. whether we have knowledge relevant to the problem at hand. However, having the relevant knowledge is not sufficient to be able to solve a problem. The problem might be too large to be solvable with the given resources or there might not be a solution. Having the right knowledge is only a necessary criterion for being competent.

Furthermore, we have applied the ideas about competence assessment to UCPOP to show that they work in practise. However, this added some practical problems to the theoretical ones that had to be addressed in the experiments conducted to show that re-using the meta-knowledge indicates capability in the UCPOP domains. As a result we came up with several features that can be extracted from small portions of the search spaces that can be generated very fast. These features then can be used to assess capability. Whereas the experiments show that the features indicate capability for the problems used, the small number of problems available is a problem. However, to show that the knowledge was re-used it seemed necessary to use existing examples. We have plans to do similar experiments for different domains in future.

Another problem with the features is that some of them vary with the size of the search space explored. Thus, the threshold to distinguish capable from incapable planner is a function of the search limit. Not knowing this function an assessment of capability was only possible comparatively.

Finally, one of the problems with the search control in UCPOP is that it computes an evaluation function and thereby imposes a total order on the open nodes. It would be interesting to see how capability assessment changes if we maintained a partially ordered list of open nodes to guide search.

Acknowledgements

The work reported is sponsored as part of the O-Plan project by the Advanced Research Projects Agency (ARPA) and Rome Laboratory, Air Force Materiel Command, USAF, under grant number F30602-95-1-0022. The O-Plan project is monitored by Dr. Northrup Fowler III at the USAF Rome Laboratory. The U.S. Government is authorised to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either express or implied, of ARPA, Rome Laboratory or the U.S. Government.

References

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

[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.

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

[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].

[Ginsberg, 1993] Matt Ginsberg. *Essentials of Artificial Intelligence.* Morgan Kaufmann, San Francisco, CA, 1993.

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

[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.

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

[Norman and Bobrow, 1975] D. A. Norman and D. G. Bobrow. On data-limited and resource-limited processes. *Cognitive Psychology*, 7:44–64, 1975.

[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.

[Pryor, 1994] Louise Pryor. *Opportunities and Planning in an Unpredictable World.* PhD thesis, Northwestern University, Evanston, IL, June 1994.

[Russell, 1910] Bertrand Russell. Knowledge by acquaintance and knowledge by description. *Aristotelian Society Proceedings*, 11:108–128, 1910.

[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.

[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. University of Edinburgh, Edinburgh, Scotland, March 1996.

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

[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.