Competence and Capability Assessment

Gerhard Wickler Louise Pryor
Dept. of AI, University of Edinburgh, 80 South Bridge, Edinburgh EH1 1HN, Scotland
{gw|louisep}@aisb.ed.ac.uk

Abstract

This paper discusses the competence and capability assessment problems, the differences between them, and a unified method to solve them. Competence can be assessed by exploiting meta-rules present in the knowledge base. Capability assessment focuses on planning; meta-knowledge can be used here as well. There are different stages in the planning process where meta-rules can be added and these represent more or less useful aspects of capability. We give an analysis of SNLP as an example of our approach. Finally, we shall discuss how competence and capability assessment are related to flexible computation.

1 Problem-Solving and Intelligent Agents

The problem we attempt to address is best illustrated by looking at an example. Consider the problem-solving activity of human problem solvers given the following simple physics problem [Larkin *et al.*, 1980]:

> A block of mass m starts from rest down a plane of length l inclined at an angle Θ with the horizontal. If the coefficient of friction between block and plane is μ, what is the block's speed as it reaches the bottom of the plane?

Given that the human problem solvers have some knowledge of physics in the form of equations that are appropriate to the problem, they will most likely answer the question whether they *can* solve this problem with "yes", i.e. they will state that they are competent to solve this particular problem instance. We will refer to this question as the **competence assessment problem**. However, then asked *what* the solution is they would probably reply that to work out the *exact* speed of the block at the end of the plane they needed some more time.

Now, this is interesting. Assessing whether one *can* solve a problem seems to be much easier than actually solving the problem, at least for a human problem solver. This claim is supported by research on the knowing not phenomenon [Barr, 1979]. They point out that "people often know rapidly and reliably that they *do not know* something."

To be able to solve the competence assessment problem efficiently would be a very useful and important ability for an intelligent computational agent trying to solve a problem as well.

Intelligent agents do not necessarily have all the knowledge they need to solve a problem but they might know of other agents they can communicate with that have the knowledge they lack. Thus, in order to solve its problem the first agent has to find another agent that can provide the desired knowledge, formulate and ask the relevant question, wait for the reply, and continue processing. The second agent receives the query and tries to find an answer. If it finds an answer it returns it and does something else.

A problem occurs when the agent is unable to solve the given problem and wastes time looking for the solution anyway. It may be the case that other queries cannot be worked on during this time. The simplest approach to trying to avoid this situation is for the question-answering agent to proceed in two steps:

1. Assess its own *ability* to solve the given problem;
2. Try to find the answer.

In an environment where multiple agents have to solve a number of problems, competence as-

sessment can be used to distribute the problems amongst the agents.

2 Competence and Capability

2.1 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].

One of the problems with search is that the most general techniques are often the least efficient. Different techniques for reasoning have therefore been developed in AI to exploit different features of the representation, task, or 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 in this context 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 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**.

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

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

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

2. Can I achieve $< state >$?

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

For example, looking at the sliding block problem again we need to assess competence for a proposition P, i.e. a question of the first type. From the problem "What is the block's speed as it reaches the bottom of the plane?" we take the proposition "The block's speed as it reaches the bottom of the plane is x" to insert into the competence question.

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.

2.2 A Fundamental Problem

The 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.

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

Hence, one can only get an approximate assessment here as well.

Looking at human problem solvers again, 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. If, in the example from the beginning of this paper, the plane turns out to be so long that the effects of aerodynamics become important, then human problem solvers may fail at this problem although they originally thought they were competent.

3 The Competence Assessment Problem

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.[2]

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.

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

3.1 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.

4 The Capability Assessment Problem

4.1 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 agent's knowledge base.

Unfortunately there seems to be very little meta-knowledge telling us how to tackle the problem in current planners. Now, why is this? As described above, the classic planning problem is very much domain independent. Meta-rules on the other hand do refer to the domain as can be seen in the example meta-rule for determination of an investment. Since classic planners almost never 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 has available 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.

It is very unfortunate that there seem to be no meta-rules in current planners because we pointed out reuse of meta-knowledge as one of the major advantages of our competence assessment approach. That there is no such meta-knowledge in classic planners is not strictly true though: in hierarchical planning, for example, abstract actions guide search. Once a plan is found on one level of abstraction the refinement process can be seen as one in which intermediate goal states are specified for the level below. The problem with this approach is that it allows only one specific kind of meta-knowledge to be exploited, namely how to refine abstract actions. Furthermore, it is not represented as knowledge for guiding search explicitly. Meta-rules are a more generic way to represent this and other knowledge.

So how can we integrate meta-rules into a

planning framework? We shall see how this can be done by looking at the effect of the competence assessment rule from a more technical point of view. Meta-rules 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.

4.2 Meta-Rules for Planning

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]. Instead we will concentrate on the planning algorithm that can be seen as the state of the art: SNLP [McAllester and Rosenblitt, 1991].

There are three non-deterministic steps in SNLP: goal selection, operator selection, and causal link-protection. We will now illustrate by means of examples what the decision is we have to take in each of these steps. We will give examples of meta-rules that represent knowledge to aid search control. Finally, we will explain which capabilities are indicated by the meta-rules.

4.2.1 Goal Selection

At any stage in the planning process let G contain the open conditions or goals that still need to be achieved in order to transform the current plan state into a solution plan state, i.e. one in which the preconditions of each step $s_i \in S$, the set of actions currently in the plan, are all necessarily true in the input situation of s_i. For SNLP this is true when G is empty. Otherwise we have to select a goal from G that we attempt to achieve and remove from G. The best goal to chose would be the one that maximizes the probability that the current plan state can be transformed into a solution plan state.

Let us look at the blocks world [Nilsson, 1980] to illustrate how a meta-rule aiding goal selection could work. The problems in this domain are typically stacking problems and it is obvious that it is best to start at the bottom. The following meta-rule expresses this knowledge:

> If there are goals `ON(x, y)` and `ON(y, z)` then work on the latter first.

Similarly one could specify that goals `ONTABLE(x)` should be preferred over all others. Other domains like the artificial domains in [Barrett and Weld, 1994] would benefit from this kind of goal ordering as well since this is the only decision in some of their domains, i.e. the ones where there is only one operator to achieve each goal.

So what capability does the above meta-rule indicate? A planner with this meta-knowledge would be capable to solve stacking problems, i.e. problems where the goal state is the conjunction of conditions with predicate `ON` and with a shared object as the appropriate arguments.

4.2.2 Operator Selection

Now, let c be the goal condition we have chosen to attempt to achieve next. Then let S_c be the set of operators that fulfill c, where $S_{add} \in S_c$. As for the goal selection, the best operator to chose from S_c would be the one that maximizes the probability that the current plan state can be transformed into a solution plan state.

Looking at the blocks world again, one could assume that it is better to put a block on the table if one needs an empty hand and the block is not needed on another block. Note that the latter expresses a goal preference not included in the following meta-rule:

> If the goal is `HANDEMPTY` and `putdown(x)` $\in S_c$ then prefer this operator.

Notice that meta-rules for operator selection can be attached to goals like the meta-rules in TEIRESIAS.

Capability indication for rules of this type is limited; they only tell us whether the agent knows how to approach some part of the problem. Of course, the real difficulty often stems from the emerging goal conditions.

4.2.3 Causal-Link Protection

The last non-deterministic step we consider is the causal-link protection. Causal links indicate which operators generate the preconditions for which other operators. If a new operator is

introduced into the plan state then this might delete preconditions for some of the operators in the plan. There are several ways to prevent this: promotion, demotion, and separation. The former two add ordering constraints on the actions; the latter adds constraints to make sure variables do not unify. As before we want to maximize the possibility of a solution plan state.

Now suppose that we have an extended blocks world where we can plan using multiple hands but we know that hands are in general a tight resource. Then protecting links by separation of variables of the type `Hand` is not a good idea.

> If s_k threatens $s_i \xrightarrow{p} s_j$ and protection by separation involves noncodesignation constraints over variables of type `Hand` then avoid protecting the link with these constraints.

What this means in terms of capability assessment is that we are capable of dealing with resources of the type mentioned in the rule. Note, however, that this requires variables to be typed.

5 Competence Assessment and Flexible Computation

Flexible computation is about methods and algorithms that allow trading solution quality against available resources. In other words, flexible computation is about how to get a maximum benefit with only limited resources. Taking, for example, a planning problem where the task is to find the shortest plan that achieves some goals, we could apply an algorithm that first searches for any complete plan that achieves the goals and then searches for a shorter plan. The latter need not be mearly a continuation of the search but could be a completely different technique as, for example, branch merging in contingency planning [Pryor and Collins, 1996; page 323]. The search for the shorter plan could be interrupted at any time with a complete plan being available as the result. Given more time, it is reasonable to hope that the algorithm will find a shorter plan. This is exactly the kind of tradeoff flexible computation tries to deal with. In the planning example time is the limited resource that has to be traded for solution quality. Other resources that may be limited could be memory or information requests on a network.

Although the method for competence assessment described in this paper can be extended to be a flexible method in itself, i.e. a method that results in a better assessment given more time (see section 3.1), we see its main role as a facilitator for flexibility in an environment where a number of intelligent agents work together on a number of problems. Competence assessment is about guessing whether an agent will be able to find a solution with the given resources. In an environment where multiple agents work together to solve a number of problems it is not necessarily the case that all the given problems can be solved with the given resources. Our aim in this case must be to maximize the number of solved problems using only the given resources. Furthermore, if the agents have different knowledge and/or problem-solving methods available to attempt to solve the given problems then the distribution of the problems can have significant impact on the number of problems solved within the given resources. Our claim is that performing a competence assessment before attempting to solve a problem will ultimately save resources that can be used to solve more problems. In this way competence assessment facilitates the maximization of results given limited resources.

Note that a special case of the above scenario would be to have multiple agents addressing the same problem type and a number of problems that are a decomposition of one larger problem. In this case we are using competence assessment to decide which aspect of the larger problem we will address and how this will be done. Again, competence assessment is not itself the flexible method but a facilitator.

The issue that competence assessment addresses is the representation of performance profiles. As discussed in section 3, the advantage of our method over more static representations is that it allows the reuse of meta-knowledge and thereby avoids the problem of having to define features of the problem that can be used for the performance profiles.

6 Future Work

One aim for the future is to investigate domain specific heuristics for planning. We want to implement these heuristics as meta-rules and show that a planner using meta-knowledge in this

form could be more efficient then approaches to planning not using this knowledge. This efficiency gain is not our main goal though.

We envisage a problem-solver consisting of a number of agents equipped with different techniques to tackle a given problem. Different agents are more or less efficient at solving different types of sub-problems. The problem then is to distribute effort amongst the different agents in order to solve the overall problem. We see competence and capability assessment as one approach to dynamically minimizing the overall effort spent; an agent only attempts to contribute to the overall solution if it thinks it is competent to solve a specific sub-problem. This work is to be integrated into the O-Plan framework [Tate *et al.*, 1994]. Some experiments with UCPOP [Barrett *et al.*, 1995] have already been conducted and the results indicate that efficient capability assessment is feasible.

The current assessment method tells us whether we can do something not how good we expect to be at doing it. This is another issue that needs further work and might be beneficial to the agent problem-solving scenario.

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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[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.