Possible Worlds and the Qualification Problem

Matthew L. Ginsberg and David E. Smith[1]
The Logic Group
Computer Science Department
Stanford University
Stanford, California 94305

Abstract

In this paper, we propose a solution to McCarthy's *qualification problem* [10] based on the notion of *possible worlds* [3, 6]. We begin by noting that existing formal solutions to qualification seem to us to suffer from serious epistemological and computational difficulties. We present a formalization of action based on the notion of possible worlds, and show that our solution to the qualification problem avoids the difficulties encountered by earlier ones by associating to each action a set of domain constraints that can potentially block it. We also compare the computational resources needed by our approach with those required by other formulations.

I. Introduction

A. The problem

An important requirement for many intelligent systems is the ability to reason about actions and their effects on the world. There are several difficult problems involved in automating reasoning about actions. The first is the *frame problem*, first recognized by McCarthy [11]. The difficulty is that of indicating all those things that do not change as actions are performed and time passes. The second is the *ramification problem* (so named by Finger [2]); the difficulty here is that it is unreasonable to explicitly record all those things that *do* change as actions are performed and time passes. The third problem is called the *qualification problem*. The difficulty is that the number of preconditions for each action is immense. In a previous paper [4], we presented a computationally effective means for solving the frame and ramification problems. In this paper we extend this method to deal with the qualification problem.

A familiar example of the qualification problem, due to McCarthy, is the "potato in the tailpipe" problem. One precondition to being able to start a car involves having the key turned in the ignition, but there are many others. For example, there must be gas in the tank, the battery must be connected, the wiring must be intact, and there can't be a potato in the tailpipe. It would hardly be practical to check all of these unlikely qualifications each time we were interested in using the car.

To describe the qualification problem more formally, we will use a simple situation calculus to talk about the world. Let the predicate $\texttt{holds}(p, s)$ indicate that the proposition p holds in the state s. We also denote by $p(a)$ the preconditions of an action a, and by $c(a)$ the consequences of the action a given that the preconditions hold. An action can now be characterized by an axiom or axioms of the following form:

$$\texttt{holds}(p(a), s) \rightarrow \texttt{holds}(c(a), \texttt{do}(a, s)),$$

where $\texttt{do}(a, s)$ refers to the new situation that arises after the action a has been performed. The qualification problem is that there are a great many preconditions and qualifications appearing in the complete precondition $p(a)$. It is difficult to enumerate them all, and computationally intractable to check them all explicitly.

This overall problem consists of three distinct difficulties:

1. The language or ontology may not be adequate for expressing all possible qualifications on the action a,
2. It may be infeasible to write down all of the qualifications for a even if the ontology is adequate, and
3. It may be computationally intractable to check all of the qualifications for every action that is considered.

In this paper we will be concerned only with the second and third of these issues – how to conveniently express qualifications and how to reason with them in a computationally tractable way. We will not consider the problem of recognizing or recovering from qualifications that cannot be described within the existing ontology or language of a system.

B. The default approach

There has been a recent resurgence of interest in problems of commonsense reasoning about actions and their consequences. Several authors [7, 8, 9, 13] have suggested that the qualification problem can be effectively addressed by grouping together all of the qualifications for an action under a `disabled` predicate. This predicate is then assumed false by default in any particular situation. For example, given an action a with explicit preconditions $p(a)$, explicit

[1] Both authors supported by DARPA under grant number N00039-86-C-0033 and by ONR under grant number N00014-81-K-0004.

Figure 1: Move A to B's location

Figure 2: The dumbbell problem

consequences $c(a)$ and additional qualifications $q(a)$, we could write

$$\begin{aligned}\texttt{holds}(p(a), s) \wedge \neg\texttt{disabled}(a, s) \\ \rightarrow \texttt{holds}(c(a), \texttt{do}(a, s))\end{aligned}$$

$$\texttt{holds}(q(a), s) \rightarrow \texttt{disabled}(a, s),$$

together with the default rule

$$\frac{\texttt{M}\neg\texttt{disabled}(a, s)}{\neg\texttt{disabled}(a, s)}.$$

In other words, if the action's preconditions hold in state s, and the action is not provably disabled, then the consequences will hold in the state resulting from the execution of the action. The advantage of this approach is that a system does not need to reason about all of the obscure qualifications that might prevent each action. They can be assumed to be false, unless the contrary has been shown by some form of forward inference.

Unfortunately, there are some serious difficulties with this approach. Consider a simple blocks world consisting of a floor with two blocks on it, as shown in Figure 1, and a single operation $\texttt{move}(b, l)$ that moves the block b to location l. One qualification on this action is that the intended destination for a move operation must be vacant. We might express this as:

$$\texttt{holds}(\texttt{on}(x, l), s) \rightarrow \texttt{disabled}(\texttt{move}(y, l), s). \quad (1)$$

If x is in some location l, the action of moving y to that location is disabled.

Now suppose that we complicate matters by allowing blocks to be connected together as shown in Figure 2. (We will henceforth refer to this as the dumbbell problem.) If we try to move the block A to the location occupied by B, B moves also, and will therefore not be in the way when A arrives. In this case, the fact that B is in the way is *not* a qualification on the action. So we need to modify (1) to become:

$$\texttt{on}(x, l) \wedge \neg\texttt{connected}(x, y) \rightarrow \texttt{disabled}(\texttt{move}(y, l)), \quad (2)$$

indicating that an object at the destination of an intended motion disables that action unless it is connected to the object being moved. (We have dropped the situation variable in (2) in the interests of simplicity.)

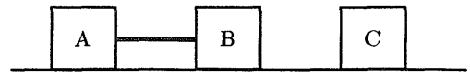


Figure 3: The blocked dumbbell problem

The "blocked dumbbell" problem shown in Figure 3 requires that we introduce still more qualifications on the move operator. Now the presence of C blocks the action, since B is unable to move to *its* new location. We have to modify (2) to produce something like:

$$\begin{aligned}\texttt{on}(z, l') \wedge \texttt{connected}(x, y) \wedge \neg\texttt{connected}(y, z) \\ \wedge\texttt{induced-position}(y, l', \texttt{move}(x, l)) \rightarrow \\ \texttt{disabled}(\texttt{move}(x, l)).\end{aligned} \quad (3)$$

This axiom states that a **move** action will be disabled if an object connected to the object being moved is prevented from reaching its new location.

The increased complexity is a consequence of the fact that the disabling rules (2) and (3) need to anticipate the *ramifications* of the **move** action, but the possible ramifications become increasingly numerous and complicated as the complexity of the domain increases.

In addition to epistemological problems, this complexity leads to computational difficulties. As the number of ramifications grows, it becomes impractical to forward chain on the direct results of an action in order to determine all of the subsequent actions that may be disabled. We will see in Section IV-A that a backward chaining approach to this problem is also intractable.

C. Approach

In the examples above, the move operation always failed because there was something in the way. It would therefore seem that we should be able to derive the above qualifications from more general constraints on the world. In the blocks world, one of the domain constraints is that an object cannot be in two places at once; another domain constraint is that no two objects can ever be in the same place at the same time. We could state these formally as:

$$\begin{aligned}\texttt{on}(x, l) \wedge l \neq l' &\rightarrow \neg\texttt{on}(x, l') \\ \texttt{on}(x, l) \wedge z \neq x &\rightarrow \neg\texttt{on}(z, l),\end{aligned} \quad (4)$$

If we try to move a block to a location that is already occupied, the resulting world will be in contradiction with the domain constraint (4). We conclude that the action cannot be performed.

A similar argument can be made for the potato in the tailpipe problem. In this case, it is inconsistent for an engine to be running with a blocked exhaust. It follows that a car with a blocked exhaust cannot be started.

Unfortunately, there is a serious flaw in these arguments. The trouble is that we have not distinguished between things that an action can change (ramifications) and things that prevent it from being carried out (qualifications). In our blocks world example, it may very well be that a block in the way will defeat a move operation. On the other hand, it might be the case that the robot arm is sufficiently powerful that any block in its way simply gets knocked aside. Given only the domain constraint, we have no way of knowing which is the case.

The same is true for the potato in the tailpipe problem. Given a car with a potato in its tailpipe, how are we to know whether turning the key in the ignition will have no effect, or will blow the potato out of the tailpipe? Surely a potato in an exhaust nozzle of the space shuttle would not prevent it from taking off, but nowhere have we provided any information distinguishing the two cases.

The problem is essentially this: given that the results of an action may include arbitrary inferential consequences of the stated results, we need to distinguish legitimate qualifications for an action from possible ramifications of the action. One solution to this problem is to explicitly identify, for each potential ramification of an action, whether or not it can act to qualify the action in question. Unfortunately, the number of potential ramifications to an actions grows exponentially with the complexity of the domain [4], so that any approach to the formalization of action that requires the exhaustive enumeration of all of an action's ramifications will become computationally intractable when dealing with complex domains.

The approach we will take to this problem is to indicate, for each possible action, which subset of the domain constraints can potentially block the action. In our blocks world example, the domain constraint that no two things can be in the same place at the same time qualified the failing move operations. In the car example, the constraint about exhaust blockages leads to the qualification.

D. Organization

In the next two sections, we present a formalization of the informal approach discussed in Section I-C, and go on to show that this approach does indeed give intuitively correct answers when dealing with a variety of qualified and unqualified actions.

In Section IV we briefly compare the computational requirements of the possible worlds approach with those of existing descriptions, such a QA-3 type planner using monotonic situation calculus [5] or a system using a default approach such as that described in Section I-B.

II. Possible worlds

The approach to qualification that we are proposing builds upon our earlier work on the frame and ramification problems [4]. We will review that work very briefly here; the essential idea is to take the result of an action to be the nearest possible world in which the explicit consequences of the action hold.

To formalize this, suppose that we have some set S of facts describing the condition of our world before taking an action a with consequences $C(a)$. Now after the action a is taken, we need to add the facts in $C(a)$ to our world description S; the difficulty is that the simple union $S \cup C(a)$ may be inconsistent.

In order to avoid this difficulty, we consider consistent subsets of $S \cup C$. In other words, we define a possible world for C in S to be any consistent subset of $S \cup C$. Now note that the nearness of a particular subset to the original situation described by S is reflected by how large the subset is: if $C \subseteq T_1 \subseteq T_2 \subseteq S \cup C$, T_2 is at least as close to S as T_1 is. This leads us to define the *nearest* possible world for C in S to be a *maximal* consistent subset of $S \cup C$. Note that maximal here refers to set inclusion, as opposed to cardinality, so that a subset of $S \cup C$ is maximal if it has no consistent superset in $S \cup C$.[2]

There is one additional subtlety that we need to consider. Specifically, there will often be facts that will *always* hold, so that we want to only consider subsets of $S \cup C$ that contain them. Domain constraints such as (4) often have this property; we can expect (4) to hold independent of the modifications we might make to our world description. We cater to this formally by supposing that we have identified some set P containing these protected facts.

Definition 1 *Assume given a set S of logical formulae, a set P of the protected sentences in our language, and an additional set C. A* nearest possible world for C in S *is defined to be any subset $T \subseteq S \cup C$ such that $C \subseteq T$, $P \cap S \subseteq T$, T is consistent, and such that T is maximal under set inclusion subject to these constraints.*

In general, we will have no use for possible worlds other than the nearest ones, and will therefore refer to the nearest possible worlds for C in S simply as *possible worlds* for C in S.

III. Qualification and possible worlds

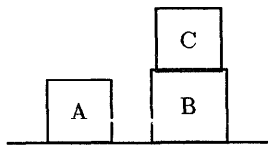
A. Manipulating domain constraints

To describe qualification in this framework, we will describe an action a using a precondition $p(a)$, a consequence set $C(a)$, and a *qualification set* $Q(a)$. The qualification set contains those domain constraints that can qualify the success of the action.

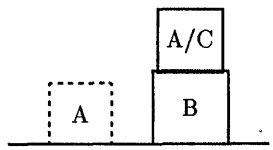
As an example, we can characterize the `move` operator as follows:

$$p(\texttt{move}(b, l)) = \texttt{clear}(b)$$

[2] This definition originally appears in [1]. It is shown in [3] to be equivalent to ideas appearing earlier in Reiter's default logic [12].



Move A onto B with C in the way



Domain constraint violated

Figure 4: A qualified action

$$C(\texttt{move}(b,l)) = \{\texttt{on}(b,l)\}$$
$$Q(\texttt{move}) = \{\texttt{on}(x,l) \wedge z \neq x \rightarrow \neg\texttt{on}(z,l)\}. \quad (5)$$

The precondition is simply that the block being moved be clear, and the consequence is to relocate the block at the destination of the move operation. The qualification set consists of the single domain constraint stating that only one block can be in any particular location at any given time. In other words, move actions will be qualified if something gets in their way.

To determine whether or not an action will succeed, we first remove the domain constraints in Q from our world description, and only then do we construct the nearest possible world in which the consequences of the action hold. If the domain constraints are violated in all of the worlds so constructed, the action is qualified; if there is some world in which none of the domain constraints is violated, the action succeeds (and the corresponding world is the result of the action).

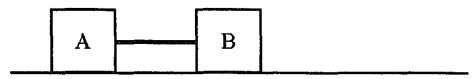
For `move` actions, this involves computing the consequences of the action assuming that any number of blocks can occupy the same location. If no two objects coincide in the resulting world, the action succeeds: nothing got in the way.

B. Examples

We now examine a series of potentially qualified actions, and show that this definition does indeed give us the desired result in all cases. Consider first the simple example shown in Figure 4a. The initial state is given by:

$$\begin{array}{ll} * & \texttt{on}(A,\texttt{floor}) \\ & \texttt{on}(B,\texttt{floor}) \\ & \texttt{on}(C,B). \end{array} \quad (6)$$

We now attempt to move A onto B.



Move A to B's location

Domain constraint intact

Figure 5: An unqualified action

To construct the possible world for $\texttt{on}(A,B)$, we must remove the fact that A is on the floor, since the domain constraint indicating that A can be in only one place at a time is not in the qualification set $Q(\texttt{move})$. But we do *not* need to remove the fact that B is also on top of C, since the domain constraint that A and B cannot coincide is not being considered. The resulting world is shown in Figure 4b, where the $*$ labelling (6) indicates that it has been removed from our world description. The domain constraint (4) is violated in this world, and the action is therefore qualified.

As a second example, consider the dumbbell problem, which is repeated in Figure 5a. The initial state is given as:

$$\begin{array}{ll} * & \texttt{on}(A,l_1) \\ * & \texttt{on}(B,l_2) \\ & \texttt{connected}(A,B). \end{array}$$

We also need axioms describing the `connected` predicate. We might have[3]:

$$\texttt{connected}(x,y) \wedge \texttt{on}(x,l_1) \rightarrow \texttt{on}(y,l_2) \quad (7)$$
$$\texttt{connected}(x,y) \wedge \texttt{on}(x,l_2) \rightarrow \texttt{on}(y,l_3). \quad (8)$$

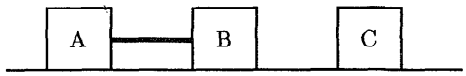
We assume that the axioms describing connection and the fact $\texttt{connected}(A,B)$ are all protected.

Even in the absence of the domain constraint saying that A and B cannot both be located at l_2, $\texttt{on}(B,l_2)$ is inconsistent with the consequence $\texttt{on}(A,l_2)$ because of the domain constraint (8) describing the effect of the connection between A and B. Thus (4) continues to hold, and the action is not qualified. The result is given by:

$$\begin{array}{l} \texttt{on}(A,l_2) \\ \texttt{connected}(A,B). \end{array}$$

Using (8), we can now derive $\texttt{on}(B,l_3)$ from these two facts, so that B's new location is a ramification of the move action. See Figure 5.

[3] An alternative formulation would describe the `connected` predicate arithmetically, assigning a numeric position to objects in our domain. We are using the description given only for reasons of simplicity.



Move A to B's location

Domain constraint violated

Figure 6: The blocked dumbbell

In the blocked dumbbell problem (Figure 6a), the initial description is:

$$\begin{array}{ll} * & \texttt{on}(A, l_1) \\ * & \texttt{on}(B, l_2) \\ & \texttt{on}(C, l_3) \\ & \texttt{connected}(A, B). \end{array}$$

As above, B must move when A does, since the two blocks are connected. But C need not be dislodged if we ignore the domain constraint in $Q(a)$: the only reason it has to move is that it cannot remain at B's implied destination. Thus the domain constraint is violated in the resulting world and, as depicted in Figure 6, the action fails.

The pulley problem shown in Figure 7a is somewhat different. Here, moving A toward B causes B to move toward A, and a ramification of the action is to *introduce* a qualification. The action should fail.[4]

The initial state is given by:

$$\begin{array}{ll} * & \texttt{on}(A, l_1) \\ * & \texttt{on}(B, l_2) \\ & \texttt{pulley}(A, B). \end{array}$$

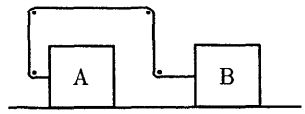
If we denote by l_4 the location halfway between l_1 and l_2, the axioms describing the pulley system are:

$$\texttt{pulley}(x, y) \wedge \texttt{on}(x, l_1) \rightarrow \texttt{on}(y, l_2) \quad (9)$$

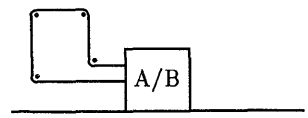
$$\texttt{pulley}(x, y) \wedge \texttt{on}(x, l_4) \rightarrow \texttt{on}(y, l_4). \quad (10)$$

Ignoring the domain constraint stating the blocks cannot coincide, the possible world relocating A halfway between l_1 and l_2 removes the facts marked with a * above; the domain constraint (4) is violated in this world, since the physics of the pulley system implies that both blocks must be located at l_4.

[4] As with the "self-fulfilling" dumbbell problem, this sort of "self-defeating" action poses severe problems for earlier descriptions of qualification.



Move A halfway to B



Domain constraint violated

Figure 7: The pulley

IV. Comparison with other approaches

Existing approaches to qualification proceed by explicitly indicating under what circumstances the action is qualified; if none of these circumstances can be proven to have arisen, the action is assumed to be unqualified. We will refer to this as the "exhaustive" approach to qualification because of the need to list all of the qualifications explicitly. If any of the listed qualifications is present, the action is blocked. The non-exhaustive, or *inferential* approach that we proposed in Section III takes a more relaxed view, enabling us to determine inferentially which domain facts potentially qualify the action in question.

In this section, we compare these two approaches. We are interested in the additional computational resources needed by the various methods in order to both describe the qualifications on an action, and to determine whether or not any particular action is in fact qualified. We begin by considering the exhaustive approach.

A. Exhaustive approaches

In a domain with a distinct action types, we showed in [4] that any of these actions can have up to $2^{\kappa a}$ distinct ramifications, where $\kappa \leq 1$ is a number that can be expected to be fairly small for large domains, although the product κa will increase as the domain becomes more complex. As we have seen, it is theoretically possible for any or all of these ramifications to qualify any particular action, although we would expect in general that only some fraction $\lambda 2^{\kappa a}$ of them will.

Suppose now that the investigation of any particular domain fact is done by backward inference and takes an amount of time t. Assuming that most actions are not qualified, so that the examination of each of the $\lambda 2^{\kappa a}$ qualifications is a necessary overhead to the investigation of the successful action, it follows that:

Theorem 2 *The computational overhead required by an exhaustive approach to qualification is given by:*

	worst case	*typical case*
space	$2^{\kappa a}$	$\lambda 2^{\kappa a}$
time	$t2^{\kappa a}$	$\lambda t 2^{\kappa a}$

B. The inferential approach

The approach to qualification presented in Section III describes qualifications not in terms of disabling conditions, but by using a "qualification set". In principle, it might be as difficult to describe the qualification set as to list all of the disabling conditions; in practice, however, it appears that a simple qualification set (such as that in (5)) will often correspond to *all* of the disabling conditions. The power of our approach to qualification is that it enables us to take advantage of this simplicity of description.[5] We will say that a domain is *uniform* if a single qualification set for each action generates all of the disabling conditions for it.

In a uniform domain, we address the qualification problem by identifying, for each action a, which of the κar domain constraints[6] are in $Q(a)$. This will require us to list as many as $\kappa a^2 r$ domain constraints in the various $Q(a)$'s, although it is likely that a domain constraint will only be in $Q(a)$ if it involves a relation symbol appearing in a's consequence set $C(a)$. In general, therefore, we can expect to need to list at most $x\kappa ar$ domain constraints in order to describe $Q(a)$ for each action, where x is the number of consequences in a typical $C(a)$ (i.e., the number of "direct" consequences of the action).

The additional time needed to investigate the action is that needed to check whether or not the domain constraints in $Q(a)$ are violated in the possible world constructed. We demonstrated in [4] that this is given by $n\kappa a(t+t_c)$, where $t_c \approx t$ and n is the number of constraints in $Q(a)$. This gives us:

Theorem 3 *In a uniform domain, the computational requirements of the inferential approach to qualification are given by:*

	worst case	*typical case*
space	$\kappa a^2 r$	$\kappa x a r$
time	$2\kappa a^2 t$	$2\kappa x a t$

Comparing this with theorem 2, we see that it is the inferential approach that does not suffer from an exponential deterioration in performance as the domain becomes increasingly complex.

[5] Shoham has argued that our method also takes its power from a partial order on possible worlds, but this is not the case. As can be seen from the examples in Section III-B, the method remains computationally effective if the partial order used is simply that given by set inclusion.

[6] We are using r here to represent the number of relation symbols in our domain. See [4].

Acknowledgement

We would like to thank the Logic Group for providing, as ever, a cooperative and stimulating — and demanding — environment in which to work. We would like to specifically thank Vladimir Lifschitz, John McCarthy, Drew McDermott and Yoav Shoham for many useful discussions; it was Vladimir who introduced the dumbbell problem during a discussion at the Timberline planning workshop.

References

[1] R. Fagin, J. Ullman, and M. Vardi. On the semantics of updates in databases. In *Proceedings Second ACM Symposium on Principles of Database Systems*, pages 352–365, Atlanta, Georgia, 1983.

[2] J. J. Finger. *Exploiting Constraints in Design Synthesis.* PhD thesis, Stanford University, Stanford, CA, 1987.

[3] M. L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30:35–80, 1986.

[4] M. L. Ginsberg and D. E. Smith. Reasoning about action I: A possible worlds approach. In *Proceedings of the 1987 Workshop on Logical Solutions to the Frame Problem*, Lawrence, Kansas, 1987.

[5] C. C. Green. Theorem proving by resolution as a basis for question-answering systems. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence 4*, pages 183–205, American Elsevier, New York, 1969.

[6] D. Lewis. *Counterfactuals.* Harvard University Press, Cambridge, 1973.

[7] V. Lifschitz. Formal theories of action. In *Proceedings of the 1987 Workshop on Logical Solutions to the Frame Problem*, Lawrence, Kansas, 1987.

[8] J. McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[9] J. McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[10] J. McCarthy. Epistemological problems of artificial intelligence. In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 1038–1044, Cambridge, MA, 1977.

[11] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Mitchie, editors, *Machine Intelligence 4*, pages 463–502, American Elsevier, New York, 1969.

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

[13] Y. Shoham. Chronological ignorance. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 389–393, 1986.