Reasoning About Action II: The Qualification Problem

Matthew L. Ginsberg
David E. Smith

Computer Science Department
Stanford University
Stanford, California 94305

Reasoning About Action II: The Qualification Problem

Abstract

We present a computationally effective approach to representing and reasoning about actions with many qualifications. The approach involves treating actions as qualified not by specific facts that may or may not hold when the action is executed, but instead as potentially qualified by general constraints describing the domain being investigated. Specifically, we suggest that the result of the action be computed without considering these qualifying domain constraints, and take the action to be qualified if and only if any of the constraints is violated after the computation is complete.

Our approach is presented using the framework developed in [6], where we discussed a solution to the frame and ramification problems based on the notion of possible worlds, and compared the computational requirements of that solution to the needs of more conventional ones. In the present paper, we show that the domain constraint approach to qualification, coupled with the possible worlds approach described earlier, has the remarkable property that essentially *no* computational resources are required to confirm that an action is unqualified. As before, we also make a quantitative comparison between the resources needed by our approach and those required by other formulations.

1 Introduction

1.1 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 [13]. 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 [3]); 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.

McCarthy first identified the qualification problem in 1977 [10] in the context of the missionaries and cannibals puzzle. He noted that in order to be able to use a boat to cross a river one would need

> *... a qualification that the vertical exhaust stack of a diesel boat must not be struck square by a cow turd dropped by a passing hawk or some other event that no-one has previously thought of. [10]*

A more familiar example of the qualification problem (also 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 $\mathtt{holds}(p, s)$ 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 set of consequences of the action a given that the preconditions hold.[1] An action can now be characterized by an axiom or axioms of the following form:

$$c \in C(a) \wedge \mathtt{holds}(p(a), s) \rightarrow \mathtt{holds}(c, \mathtt{do}(a, s)),$$

where $\mathtt{do}(a, s)$ refers to the situation after the action a has been performed. The qualification problem is that there are a great many clauses 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.

1.2 The default approach

There has been a recent resurgence of interest in problems of commonsense reasoning about actions and their consequences. Several authors [9, 12, 11, 15] 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 consequences $C(a)$ with $c \in C(a)$ and additional qualifications $q(a)$, we could write

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

[1] This description of an action is the one we used in [6].

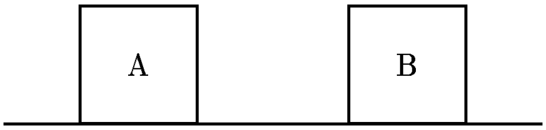


Figure 1: Move A to B's location

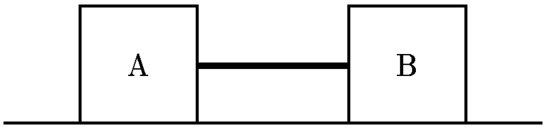


Figure 2: The dumbbell problem

together with the default rule

$$\frac{:\neg\mathtt{disabled}(a,s)}{\neg\mathtt{disabled}(a,s)}$$

In other words, if the action's preconditions hold in state s, and the action is not 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 from being executed. They can be assumed to be false, unless the contrary has been shown by some form of forward inference.

Unfortunately, there are still 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 $\mathtt{move}(b,l)$ 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:

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

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:

$$\mathtt{on}(x,l) \wedge \neg\mathtt{connected}(x,z) \rightarrow \mathtt{disabled}(\mathtt{move}(z,l)), \tag{2}$$

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.)

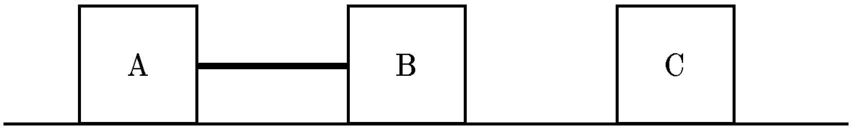


Figure 3: The blocked dumbbell problem

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:

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

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 these epistemological problems, this complexity also 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 and record which of the subsequent actions may be blocked.

As an example, suppose that we are working in a blocks world domain containing n blocks. If none of the blocks is connected to any other, then for some specific block b, there will be $n-1$ locations to which b cannot be moved because these locations are occupied by other blocks. Since there are $n-1$ disablers for the action of moving each block, there will $o(n^2)$ disablers that must be computed and stored for the entire domain. (This result continues to hold if some of the blocks are connected together.)

In Section 5, we will discuss a backward chaining approach to this problem, and show that it, too, suffers from severe computational difficulties.

Implemented systems for reasoning about action have taken a similarly "exhaustive" approach to dealing with qualification. In STRIPS [2] or QA-3 [7], for example, all qualifications need to be listed explicitly.[2] The computational properties of approaches such as these are comparable to those of the default approach we have described in this section.

1.3 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

[2]Although these approaches do not use a `disabled` predicate, they could easily be modified to do so. No nonmonotonic reasoning would be needed, because they maintain complete descriptions of their domains.

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:

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

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 those 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 squashed or 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 explicitly stated results, we need to distinguish legitimate qualifications for an action from possible ramifications of the action.[3]

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 of an action grows exponentially with the complexity of the domain [6], 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.

[3]The situation is not quite this simple, since in many cases it may be *desired* for the ramifications and qualifications of an action to interact. The self-fulfilling dumbbell problem is one example; in Section 3.1 we introduce a set of examples involving *self-defeating* actions.

We will describe this approach in terms of an extension to our earlier work on the frame and ramification problems [6]. In that work we showed how the result of an action could be taken to be the nearest possible world in which the explicit consequences of the action held. The possible worlds here are those defined by Lewis [8] and explicated in [4]. For example, imagine a robot considering moving a bookcase from one location to another. The expected result would be the nearest world to the current one in which the bookcase was at its new location. In this world, the bookcase would no longer be at its old location, and everything on or in it would also be at the new location. Furthermore, heating ducts and pictures might be covered in this new world as a consequence of the new position of the bookcase.

In [6], we also developed a method for efficiently computing these possible worlds. It involves examining proofs of the negation of the explicit consequences of an action, and removing one premise from each such proof. We review the definition of possible worlds and the mechanism for computing them in Section 2, and will rely on this information in subsequent sections.

Before proceeding, however, there are some general issues that should be discussed. First, we should emphasize that there is no intrinsic connection between describing qualification in terms of domain constraints and the possible worlds construction of [4, 6]. The approach we will present can be incorporated into any approach to reasoning about action that is based on domain constraints. The reason we discuss our solution in terms of the possible worlds construction is that the computational properties of this approach combine conveniently with domain constraint descriptions.

Second, there are significant shortcomings involved in thinking of qualification in terms of domain constraints. The reason is that a domain constraint such as that appearing in (4) is *instantaneous*, describing a restriction on the state of the world at a moment in time. It conveys no information at all about the underlying reason for any particular qualification, and gives no indication of what the result will be if we attempt a qualified action, such as that of Figure 1.

If we want to predict the result of an action such as this, we will in general need a more detailed description of our domain. In Figure 1, for example, we might describe the move action in terms of forces and accelerations, allowing us to determine the results of the qualified action.[4] Perhaps A bumps into B and stops, perhaps B is displaced after all, or perhaps one of the blocks is damaged as a result of the impact.

This redescription, however, is *also* in terms of instantaneous domain constraints (Newton's third law, in this case). If we want to determine the result of attempting a qualified action, this sort of an ontological shift will be needed. The reason for this is that we need to work with a level of description sufficiently detailed that the action being considered is *not* qualified. We see from this that, short of working at a level of detail sufficient to guarantee that no action is *ever* qualified, the problem of identifying qualified actions will persist. The approach we will describe in this paper can identify qualified actions using a domain description at any level of detail. In addition, the ability to identify qualified actions at a

[4] Yoav Shoham has also suggested using this reformulation to determine whether or not the action was qualified in the first place.

high level of abstraction can be used to determine under what circumstances a more detailed formulation must be considered.

1.4 Organization

Section 2 provides an overview of the possible worlds construction developed in [6]. In Section 3, we show how qualifications can be succinctly described by associating a set of qualifying domain constraints with each action, and show how this approach can be used to deal with examples like the dumbbell and blocked dumbbell problems.

Section 4 discusses the implementation of our ideas, and presents several examples of the approach in action. A remarkable feature of the approach we are proposing is that it takes essentially *no longer* to check an unqualified action to see if it is qualified and to then compute the result than it does to merely compute the result itself.

In Section 5, we present a comparison between the computational requirements of our inferential approach and those of the "exhaustive" approach presented in Section 1.2 and used in existing systems. We also compare our approach to the "partially exhaustive" approach introduced in Section 3.2.

Finally, in Section 6 we discuss some epistemological issues arising out of our approach, describe some difficulties with it, and suggest some alternatives for consideration.

2 Possible worlds

Our solution to the qualification problem can be best understood within the framework provided by an existing method for reasoning about action. The framework that we will use for this purpose is the one based on possible worlds and discussed in [6]. That work is reviewed 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.

This possible world cannot be constructed merely by inserting the consequences of the action into the database, since the database may become inconsistent if we do so. In the blocks world, for example, the consequence of moving a block b to a location l is that the block is located at its new location: $\texttt{on}(b,l)$. This is inconsistent with the fact $\texttt{on}(b,l_0)$ giving the block's *original* location.

The essential idea is that if our world description S is inconsistent with the consequences C of some action, we work with a maximal subset of S that *is* consistent with C.

2.1 Formalization

Suppose, then, that we are given an initial world description in the form of some set S of facts, and that C is some collection of facts that we wish to add to S, even though the set $S \cup C$ may be inconsistent. We will define a *potential world* for C in S to be any consistent subset of $S \cup C$ that contains C, and a *nearest* potential world, or *possible* world to be a *maximal* such consistent subset.[5]

As discussed in [6], 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 consider only 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.

This leads us to define possible worlds as follows:[6]

Definition 2.1 *Assume given a set S of logical formulae, a set P of the protected sentences in our language, and an additional set C.* A possible world for C in S *is any subset $T \subseteq S \cup C$ such that:*

1. *$C \subseteq T$,*
2. *$P \cap S \subseteq T$,*
3. *T is consistent, and*
4. *T is maximal subject to these constraints.*

We will denote the set of all possible worlds for C in S by $W(C, S)$.

We discuss in Section 2.3 the problem of generating possible worlds automatically. The basic idea is to detect potential contradictions by examining proofs of the negations of facts in C, and to manipulate the database in such a way that all of these proofs fail.

To use these ideas to reason about action, we define the *result* of an action a in a situation s, which we will denote by $r(a, s)$, to be the intersection of the possible worlds associated with its set of consequences $C(a)$. Formally,

$$r(a, S) = \begin{cases} S, & \text{if } W = \emptyset; \\ \bigcap\{w \in W\}, & \text{otherwise.} \end{cases} \tag{5}$$

The set W is given by

$$W = W(C(a), S), \tag{6}$$

and is the collection of possible worlds in which the consequences of the action hold.

2.2 An example

As an example of this construction, consider the scenario in Figure 4, repeated from [6]. The domain contains a table, a chest, a plant, a portrait and a bookcase (which itself

[5]Note that both potential worlds and possible worlds need not be closed under logical deduction, and also that they will in general be uncommitted on sentences that are logically independent of S. The more accurate phrase "possible partial world" seems rather unwieldy, however.

[6]This definition is related to one appearing in [1]. It is shown in [4] to be equivalent to ideas appearing earlier in Reiter's default logic [14].

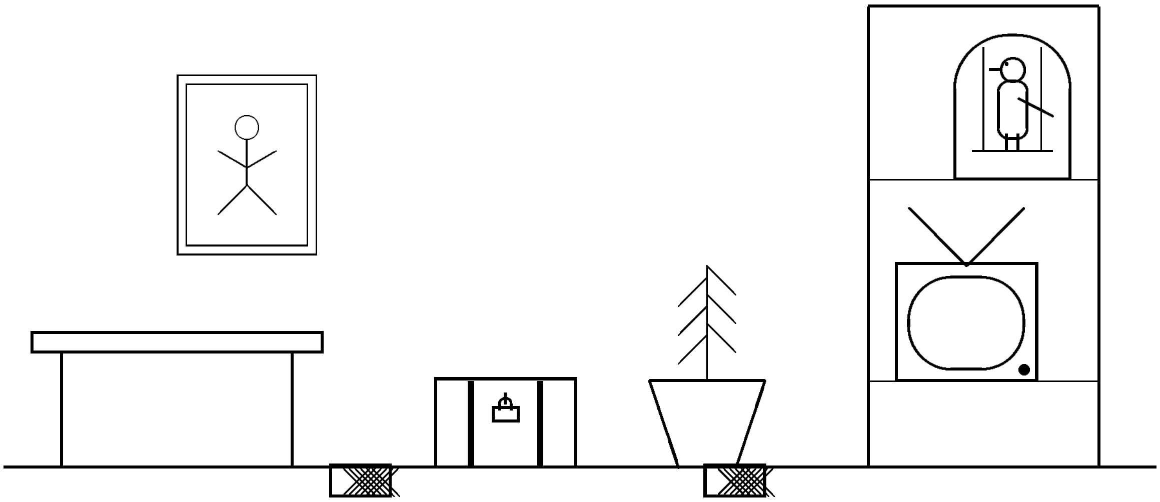


Figure 4: A household domain

contains a bird and a television). Ventilation for the room is provided by a pair of ducts under the floor; if both of these are blocked, the room becomes stuffy. Putting an object on the table will obscure the picture.

Formally, the initial situation shown in the figure can be described as follows (the * and # symbols should be ignored for the moment):

	on(bird,top-shelf)	rounded(bird)
*#	on(tv,bottom-shelf)	rounded(plant)
	on(chest,floor)	
*	on(plant,duct2)	duct(duct1)
	on(bookcase,floor)	duct(duct2) (7)
*	blocked(duct2)	in(bottom-shelf,bookcase)
	¬obscured(picture)	in(top-shelf,bookcase)
#	¬stuffy(room)	

There are also the associated domain constraints:

$$\texttt{on}(x,l) \wedge l \neq l' \rightarrow \neg\texttt{on}(x,l') \tag{8}$$

$$\texttt{on}(x,l) \wedge z \neq x \wedge l \neq \texttt{floor} \rightarrow \neg\texttt{on}(z,l) \tag{9}$$

$$\texttt{rounded}(l) \rightarrow \neg\texttt{on}(x,l) \tag{10}$$

$$\texttt{duct}(d) \wedge \exists x.\texttt{on}(x,d) \rightarrow \texttt{blocked}(d) \tag{11}$$

$$\exists x.\texttt{on}(x,\texttt{table}) \leftrightarrow \texttt{obscured(picture)} \tag{12}$$

$$\texttt{blocked(duct1)} \wedge \texttt{blocked(duct2)} \leftrightarrow \texttt{stuffy(room)} \tag{13}$$

The first domain constraint indicates that an object can be in only one place at any given time, and the second that two different objects cannot be in the same place (except for the floor, which can support many objects). The third indicates that no object can be on top of a rounded object (the bird cage and the plant both fit this description). Domain constraint (11) indicates that anything on a duct blocks it. The final two domain constraints define the conditions under which the portrait will be obscured or the room will be stuffy.

There is a single action in this domain, that of moving an object from one location to another. We will denote this action by $\mathtt{move}(x, l)$, where x is the object being moved and l is the intended destination. The preconditions given for the `move` action in [6] were that the object being moved be clear, that its destination either be clear or be the floor, and that no attempt be made to place an object on top of a rounded object:

$$p(\mathtt{move}(x, l)) \equiv \mathtt{clear}(x) \wedge [\mathtt{clear}(l) \vee l = \mathtt{floor}] \wedge \neg\mathtt{rounded}(l). \tag{14}$$

The consequence of the action is that the object is relocated at its destination:

$$C(\mathtt{move}(x, l)) = \{\mathtt{on}(x, l)\}.$$

Note that the given precondition is intended to be complete, in the sense that its satisfaction is sufficient to guarantee the success of the action.

Assuming that we are not prepared to drop the domain constraints appearing at the end of the above description, there is a unique possible world corresponding to the new fact $\mathtt{on}(\mathtt{tv}, \mathtt{table})$, the consequence of moving the television to the table. This possible world corresponds to the removal of the facts indicating that the television is on the bottom shelf of the bookcase and that the picture is not obscured. It is necessary to remove these facts since, in light of the domain constraints indicating that an object can be in only one place at any given time and that anything on the table obscures the picture, they are each inconsistent with the new location of the television.

There are two possible worlds corresponding to moving the television to duct 1. In one of these (marked with a # in the domain description), the room becomes stuffy; in the other (marked with a *), the ventilation system displaces the plant from duct 2. In the absence of additional information allowing us to select between these two possible worlds (how heavy is the plant?), we take the conservative approach of removing from the domain description facts marked with *either* a # or a *.

Note that we have defined an action whose consequences have *no* possible world as having no effect on the domain being investigated; in [6], this was justified by an argument that, "an action whose consequences have no possible world is effectively impossible." It is our intention in the current paper to show that this effective impossibility captures the essence of the qualification problem: that qualifications on actions correspond to an attempted violation of the constraints on the domain being investigated.

2.3 Automatic generation of possible worlds

In [4, 6], we discussed the automatic construction of the possible worlds for C in S. The basic idea is to remove from S just enough to invalidate any proof of $\neg c$ for each $c \in C$. To formalize the construction, we need the following definitions:

Definition 2.2 *Let C and S be sets. We define a* proof set *for C to be any subset T of $S - C$ such that:*

1. *$T \cap P = \emptyset$. All of the sentences in T are unprotected.*
2. *$T \cup P \cup C$ is inconsistent. The negation of some sentence in C follows from the facts in T and the protected sentences in our language.*
3. *T is minimal subject to these conditions. The proof set does not contain any extraneous or irrelevant sentences, so that removing any sentence from it will invalidate the proof.*

Definition 2.3 *Let $A = \{S_i\}$ be a collection of sets. A* hitting set *for A is any set H such that $H \cap S_i \neq \emptyset$ for every i.*

Theorem 2.4 *Given sets of sentences C and S, let $\{S_i\}$ be the set of all proof sets for C in S. Then the possible worlds for C in S are precisely those sets of the form*

$$S \cup C - H$$

for some minimal hitting set H for $\{S_i\}$.

A proof of this result can be found in [4] or [6].

As an example, consider the action of moving the television to the table in our household domain. In the initial situation, there are two proofs that the television is not on the table. One uses the fact that the television is in the bookcase, and can be in only one place at a time. The other uses the fact that the picture is not obscured, as it would be if something were on the table. The proof sets for $\texttt{on}(\texttt{tv}, \texttt{table})$ in our initial situation are therefore:

$$\begin{aligned} S_1 &= \{\texttt{on}(\texttt{tv}, \texttt{bottom-shelf})\} \\ S_2 &= \{\neg\texttt{obscured(picture)}\} \end{aligned}$$

The minimal hitting set for these two proof sets is

$$H = \{\texttt{on}(\texttt{tv}, \texttt{bottom-shelf}), \neg\texttt{obscured(picture)}\},$$

and it follows from this that the unique possible world for $\texttt{on}(\texttt{tv}, \texttt{table})$ is as described in Section 2.2.

In this example, the computational effort involved in constructing the possible world was incurred in the construction of the various proof sets, as opposed to the combinatoric manipulations needed to generate the minimal hitting sets appearing in Theorem 2.4. We assumed in [6], and will continue to assume in the current paper, that the hitting set construction does not involve substantial computation.

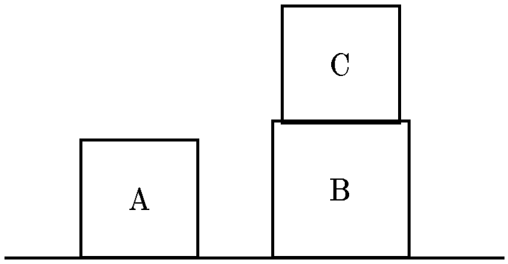


Figure 5: Move A onto B

3 Qualification

3.1 The basic problem

Instead of considering the complex household domain of Section 2.2, suppose that we consider the extremely simple example of a qualified action shown in Figure 5. The initial situation is given by:

$$\begin{array}{ll} * & \mathtt{on}(A, \mathtt{floor}) \\ * & \mathtt{on}(C, B) \\ & \mathtt{on}(B, \mathtt{floor}), \end{array} \tag{15}$$

and there are the two domain constraints

$$\mathtt{on}(x,l) \wedge l \neq l' \quad \rightarrow \quad \neg\mathtt{on}(x,l') \tag{16}$$
$$\mathtt{on}(x,l) \wedge z \neq x \wedge l \neq \mathtt{floor} \quad \rightarrow \quad \neg\mathtt{on}(z,l), \tag{17}$$

indicating, respectively, that blocks can be in only one place at a time, and that no two blocks can be in the same place at the same time (except for the floor).

The move action is described by

$$\begin{aligned} p(\mathtt{move}(x,l)) &\equiv \mathtt{clear}(x) \\ C(\mathtt{move}(x,l)) &= \{\mathtt{on}(x,l)\}, \end{aligned} \tag{18}$$

where we are treating the fact that the target block must be clear as a qualification rather than as an explicit precondition.

Suppose that we now attempt to move block A onto block B. Since the precondition to the action is satisfied (that A, but possibly not B, be clear), we proceed by constructing the nearest possible world in which the result of the action, $\mathtt{on}(A,B)$, holds. This possible world involves removing from our world description the two facts marked with a * in the domain description (15). We remove the fact that A is on the floor because it conflicts with the domain constraint (16) saying that a block can be in only one place at a time. The fact that C is on B is removed because of (17), which says that B can support only one other block.

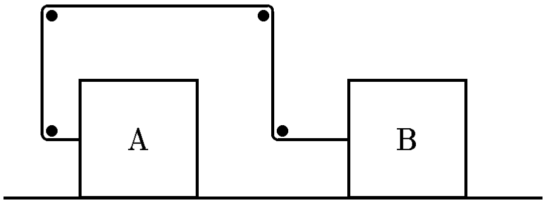


Figure 6: Move A halfway to B

The difficulty is that because of the domain constraint (17) indicating that a block can support only one other, the qualification to the action (that block B is already occupied) is defeated as a ramification of the result of the action. Requiring that *nothing* be removed from the original database in the possible world construction is also unacceptable, since the fact giving A's original location *should* be removed when A is moved.

Overturning the fact that A was originally on the floor is an intended ramification of the `move` operator; overturning the fact that B was originally occupied is not. Furthermore, the information supplied is completely symmetric with respect to these two facts, so that it will not be possible to resolve this problem without introducing additional information.

Before proceeding, however, we should note that the problem is not simply to determine which of a set of domain facts are "ramifications" and which others are "qualifications", since these two sets may overlap. In the dumbbell problem of Figure 2, for example, the ramification of moving A (that B moves as well) *should* overcome the qualification.

The "dual" to this situation is one where a ramification of the action *introduces* a qualification. In the pulley problem shown in Figure 6, for example, moving A toward B causes B to occupy A's intended destination. Although the action initially appears to be unqualified, the qualification arises and the action is blocked. We will refer to actions such as this as *self-defeating*.

3.2 Protecting domain facts

One solution to the problem of distinguishing qualifications from ramifications is to use the possible worlds construction to describe the results of an action, but to treat some of the domain facts as protected when the action is executed.

We might, for instance, require that $\mathtt{on}(x, l)$ be protected if we move a block z with $z \neq x$, so that only the location of the block being moved can change. In Figure 5, there will now be no possible world corresponding to the result $\mathtt{on}(A, B)$, since the fact giving C's original location is protected. Since the result of an action for which there are no possible worlds is defined by (5) to be the same as the situation in which the action was attempted, the action effectively fails in this case.

Formally, we introduce a new predicate $\mathtt{protected}(f, a)$. The intention is that

$$\mathtt{protected}(f, a)$$

indicates that the fact f is protected when we consider a potential action a. In the blocks world, we would have:

$$x \neq z \rightarrow \texttt{protected}(\texttt{on}(x, l), \texttt{move}(z, l')). \tag{19}$$

The location of a block x is protected when we attempt to move another block z. The incorporation of this rule into our system will result in behavior equivalent to that corresponding to (1).[7]

Once again, however, it may be extremely difficult to determine which domain facts are protected for any particular action. In the dumbbell problem of Figure 2, we have a domain constraint stating that, "B's location is two units to the right of A's." There is no formal way to distinguish this from our earlier constraint, "B's location is not the same as A's." The fact that we distinguished the two domain constraints (16) and (17) by isolating the domain fact giving B's location does not help us. Instead, we need to modify (19) to become:

$$x \neq z \wedge \neg\texttt{connected}(x, z) \rightarrow \texttt{protected}(\texttt{on}(x, l), \texttt{move}(z, l')). \tag{20}$$

Note, however, that the information in (20) above *is* sufficient to describe the blocked dumbbell problem of Figure 3. Although B's location is not protected when we attempt to move A, C's location is protected, and the action fails.

The approach represented by (19) and (20) is capable of dealing with the blocked dumbbell problem in a simple way because it allows us to identify *potential* qualifications on the action in question; whether or not these actually qualify any particular instantiation of the action is then determined in light of the rest of the domain facts.

We see that this "protection" approach provides us some relief from the need to explicitly list all of the qualifications to an action. The approach also has computational advantages over the approach described in Section 1.2, where one must examine all possible disablers in order to determine if a particular action fails. In the protection scheme, only those disablers that are relevant to the action in question will be tested.

Unfortunately, the protection approach also runs into trouble in complex situations. Suppose we return to the household domain pictured in Figure 4, and consider once again the action of moving the television to duct 1. Suppose also that we add the domain constraint that the ventilation system is sufficiently powerful to displace any light objects:

$$\texttt{stuffy(room)} \wedge \texttt{duct}(d) \wedge \texttt{on}(x, d) \rightarrow \neg\texttt{light}(x). \tag{21}$$

We also assume that the plant is light:

$$\texttt{light(plant)}. \tag{22}$$

The result of moving the television to duct 1 is now for the plant to be dislodged, so that the plant's location should not be protected for the action of moving the television to the duct.

[7]There is a difference between a "disabling" approach and the current one in that we need to provide *metalevel* information in the form of a rule such as (19), while disablers are generally base-level facts. The computational properties of the two approaches are identical, however.

The plant's location *is* protected, however, if we attempt to move the television to duct 2 instead of duct 1. We see from this example (and the preceding one) that it will not in general be possible to delimit the potential qualifications to an action simply by examining the action itself; once again, we also need to consider the ramifications of the action's success.

We will refer to this approach as *partially exhaustive*. We will describe as *exhaustive* formalisms such as that described in Section 1.2, which require that the user specify the qualifications precisely. In an exhaustive description, if an indicated qualification occurs, the action is blocked. The protection approach allows us merely to identify *potential* qualifications; whether or not any of these succeed in blocking the intended action is determined in light of the rest of the domain facts.

3.3 Manipulating domain constraints

We argue in [6] that any approach to the formalization of action that requires the enumeration of all of the ramifications of actions will suffer from severe computational problems when dealing with complex domains. Unfortunately, it does not appear to be possible to extend the formalism of the last section to deal in a thoroughly non-exhaustive fashion with the full range of examples that we have considered.

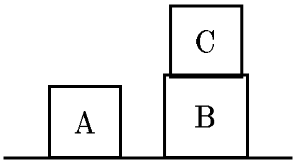
As evidenced by the final household example of the last section, the problem is that determining whether or not a specific fact such as `on(plant, duct2)` qualifies an action such as `move(tv, duct1)` depends not merely on the action and the fact, but on the fashion in which they interact. The attempt to move block A to block B's location in Figure 3 fails not simply because of C's location, but *because C gets in the way*. In the simple qualification problem in Figure 5, moving A once again fails because C gets in the way. Similarly for the self-defeating pulley problem in Figure 6. The action in the dumbbell problem in Figure 2 succeeds because B doesn't get in the way.

Rather than think of the action as being qualified by domain facts, suppose that we think of it as being qualified by the domain *constraints*. The approach we propose is this: Given an attempt to move a block to another location, imagine that the domain constraint (17) did not exist, so that many blocks could occupy identical locations. Now construct the result of the action, including all of its ramifications. If, in the resulting world, the domain constraint (17) is *still* not violated, the action was not qualified: Nothing got in the way. If the success of the action involves violating the constraint (17), something did get in the way, and the action should be qualified.

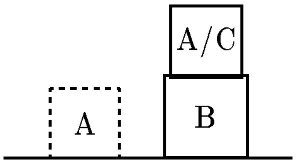
Here is the formal version of this solution. We describe an action a using a precondition $p(a)$, a consequence set $C(a)$, and a *qualification set* $Q(a)$. The qualification set contains those domain constraints that can qualify the success of the action.

Given an action a, we replace (6), defining W, the set of possible worlds for an action, to be the set of all elements of $W(C(a), S - Q(a))$ in which the elements of $Q(a)$ continue to hold:

$$W = \{V \in W(C(a), S - Q(a)) | V \cup Q(a) \text{ is consistent}\}. \tag{23}$$



Move A onto B with C in the way



Domain constraint violated

Figure 7: A qualified action

We now continue to take the result of the action to be

$$r(a,S) = \begin{cases} S, & \text{if } W = \emptyset; \\ \bigcap\{w \in W\}, & \text{otherwise,} \end{cases}$$

as in (5). Note that if the consequences of an action necessarily conflict with the domain constraints, so that $W(C(a),S) = \emptyset$, we get $r(a,S) = S$ using either the original definition (6) or the revision (23).

We now reexamine the examples we have considered thus far, showing that this new definition does indeed give us the desired result in all cases. The description of the `move` operator is given by:

$$\begin{aligned} p(\texttt{move}(b,l)) &\equiv \texttt{clear}(b) \\ C(\texttt{move}(b,l)) &= \{\texttt{on}(b,l)\} \\ Q(\texttt{move}(b,l)) &= \{\forall x,z,l'.\texttt{on}(x,l') \wedge z \neq x \wedge l' \neq \texttt{floor} \rightarrow \neg\texttt{on}(z,l')\}. \end{aligned} \tag{24}$$

We have made the quantifier in (24) explicit in order to make it clear that the qualification set consists of universally quantified domain constraints, as opposed to any particular instantiations thereof.

The precondition is simply that the block being moved be clear, and the consequence is that the block is 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, the action will be qualified if something gets in the way.

We begin with the simple example of Figure 7. The initial state is given by:

$$* \quad \texttt{on}(A,\texttt{floor}) \tag{25}$$

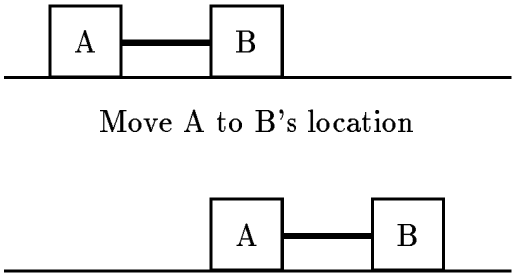


Figure 8: An unqualified action

$$\begin{array}{l} \texttt{on}(B, \texttt{floor}) \\ \texttt{on}(C, B). \end{array}$$

We now attempt to move A onto B.

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 C is also on top of B, since the domain constraint that two blocks cannot coincide is not being considered. The resulting world is shown in Figure 7, where the $*$ labelling (25) indicates, as before, that this sentence has been removed from our world description. The domain constraint (17) is violated in this world, and the action is therefore qualified.

As a second example, consider the dumbbell problem, which is repeated in Figure 8. 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 $\texttt{connected}$ predicate. We might have[8]:

$$\texttt{connected}(x, y) \wedge \texttt{on}(x, l_1) \rightarrow \texttt{on}(y, l_2) \quad (26)$$

$$\texttt{connected}(x, y) \wedge \texttt{on}(x, l_2) \rightarrow \texttt{on}(y, l_3). \quad (27)$$

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 two blocks 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

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

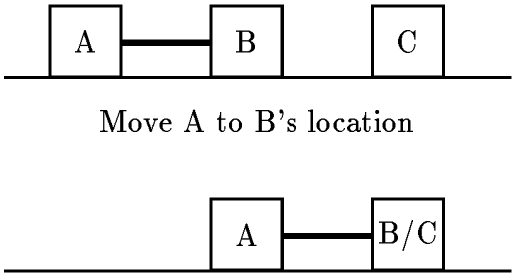


Figure 9: The blocked dumbbell

constraint (27) describing the effect of the connection between A and B. Thus (17) 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 (27), 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 8.

In the blocked dumbbell problem (Figure 9), 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 will 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 9, the action fails.

The pulley problem shown in Figure 10 is somewhat different. Here, the initial description is[9]:

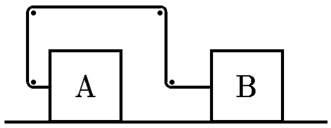
$$\begin{array}{ll} * & \texttt{on}(A, l_1) \\ * & \texttt{on}(B, l_2) \\ & \texttt{pulley}(A, B). \end{array}$$

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

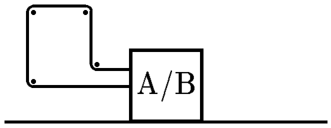
$$\texttt{pulley}(x, y) \wedge \texttt{on}(x, l_1) \rightarrow \texttt{on}(y, l_2) \tag{28}$$

$$\texttt{pulley}(x, y) \wedge \texttt{on}(x, l_4) \rightarrow \texttt{on}(y, l_4). \tag{29}$$

[9] Once again, an arithmetic description could be used instead.



Move A halfway to B



Domain constraint violated

Figure 10: The pulley

Ignoring the domain constraint stating that 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 (17) is violated in this world, since the physics of the pulley system implies that both blocks must be located at l_4.

Finally, consider the action of moving the television to duct 1 in the household scenario of Figure 4. Since the two worlds constructed in Section 2.2 did not involve a violation of the domain constraint (9), this action succeeds.[10] If, on the other hand, we were to attempt to move the television directly to duct 2 (i.e., on top of the plant), the domain constraint would be violated, since only one object can be on top of the duct. The action would therefore fail.

4 Implementation

4.1 Qualification determination

We have seen that in order to determine whether or not an action a is qualified, we need to see if the domain constraints in $Q(a)$ are violated in the possible worlds resulting from the execution of the action.

A naïve implementation of this idea would involve constructing these possible worlds while ignoring the facts in $Q(a)$, and then attempting to prove the negations of the domain constraints that had been previously ignored. Assuming that all of these domain constraints remained valid, the result of the action could be evaluated by recomputing the possible worlds while considering the qualifying domain constraints.

Fortunately, there is a much cheaper way of performing this computation. In computing and recomputing the possible worlds, we need to investigate the proof sets for the consequences of the action. Suppose that instead of ignoring the qualifying domain constraints in

[10] As before, we are unable to determine whether the plant moves or the room becomes stuffy.

$Q(a)$ during the initial calculation, we treat them simply as unprotected.[11] Having done so, we can divide the proof sets for $C(a)$ into two groups: those that contain elements of $Q(a)$, and those that do not. We will call members of the first group *qualification* proof sets, and members of the second group *ramification* proof sets.

Now the initial computation of the possible worlds can proceed simply by considering the ramification proof sets, those that do not intersect $Q(a)$. In addition, examination of the complete collection of proof sets allows us to determine whether or not the qualifying domain constraints hold in the resulting possible worlds (avoiding the computational expense of a call to a theorem prover). The basic reason for this is that a qualifying domain constraint q will remain consistent with a possible world whenever that possible world corresponds to a hitting set that intersects *all* of the proof sets constructed for $C(a)$, as opposed to simply the ramification proof sets.

Formally, we have the following:

Definition 4.1 *Let C, S and Q be sets. We define a* ramification proof set *for C to be any subset T of $S - C$ such that:*

1. $T \cap P = \emptyset$,
2. $T \cup (P - Q) \cup C$ *is inconsistent, and*
3. T *is minimal subject to these conditions.*

Definition 4.2 *Let C, S and Q be sets. We define a* qualification proof set *for C to be any subset T of $S - C$ such that:*

1. $T \cap P \subseteq Q$. *All of the sentences in T are either unprotected or elements of Q.*
2. $T \cap P \neq \emptyset$,
3. $T \cup (P - Q) \cup C$ *is inconsistent, and*
4. T *is minimal subject to these conditions.*

Note that the collections of ramification and qualification proof sets depend on S, Q and T. These sets will be identified by context in the examples of interest to us.

With these definitions, we now have the following:

Theorem 4.3 *Let a be an action, and S a state. Now define $\{H_i\}$ to be the collection of minimal hittings sets for the ramification proof sets for $C(a)$. The possible worlds corresponding to the result of the action are now given by*

$$S \cup C(a) - H_i,$$

for those H_i that intersect all of the qualification proof sets for $C(a)$ in S.

[11]The time needed to generate the proof sets is independent of which sentences are protected, since this only affects whether or not a particular sentence is included in a particular proof set. The search space of possible proofs of negations of elements of C is unchanged.

Proof. We know that we can construct the possible worlds corresponding to the result of the action by first constructing the worlds while ignoring the domain constraints in $Q(a)$, and then removing those worlds in which these domain constraints are invalid.

If we ignore the domain constraints in $Q(a)$, then any proof of the negation of some element of $C(a)$ that depends on one of these domain constraints will fail. It follows from this that the possible worlds constructed while ignoring the domain constraint correspond exactly to the minimal hitting sets for the ramification proof sets for $C(a)$ in S.

Next, note that given such a minimal hitting set H, the possible world $S \cup C(a) - H - Q(a)$ will be consistent with $Q(a)$ if and only if $S \cup C(a) - H$ is consistent, and this will be true just in case H intersects all of the proof sets for $C(a)$ in S.

It is clear that H will intersect all of these proof sets if and only if it intersects all of the qualification proof sets and all of the ramification proof sets. Since H intersects all of the ramification proof sets by construction, the theorem follows. $\square$

Note the tremendous computational import of this result: The qualification check incurs essentially *no* additional computational cost if the action being investigated is unqualified. This is a result of the fact that the expensive operation in computing the result of an action is the construction of the proof sets for its consequences, and not in the resulting hitting set calculation. This is in stark contrast with the computational requirements of previous descriptions of qualification.

4.2 Examples

Once again, we work through the various examples we have considered thus far, showing our method at work.

The dumbbell We begin with the dumbbell problem of Figure 8. Recall that we are trying to move block A to l_2; there are three distinct proofs that it is not already there, leading to the proof sets:

$$\begin{aligned} S_1 &= \{\mathtt{on}(A, l_1)\} \\ S_2 &= \{\mathtt{on}(B, l_2), q\} \\ S_3 &= \{\mathtt{on}(B, l_2)\} \end{aligned}$$

The first proof argues that A cannot be at l_2 because it is located at l_1, and can be in only one place at any given time (this domain constraint is protected, and is not in the qualification set for the `move` action). The second proof is based on the fact that B is at l_2, and only one block can be at that location (we are denoting the domain constraint stating this by q). The final proof is somewhat more subtle, and is based on the fact that if A were at l_2, then B would have to be at l_3 because of the connection between them. But B is known to be at l_2, and can be in only one place at a time.

Of these three sets, S_1 and S_3 are ramification sets, and S_2 is a qualification set. The unique minimal hitting set for the two ramification sets is given by:

$$H = \{\mathtt{on}(A, l_1), \mathtt{on}(B, l_2)\}.$$

Since H intersects S_2, the action succeeds, and the result is as described in Section 4.2.

The blocked dumbbell Next, consider the blocked dumbbell of Figure 9. Now, there are *four* proofs that A is not located at l_2:

$$\begin{aligned} S_1 &= \{\texttt{on}(A, l_1)\} \\ S_2 &= \{\texttt{on}(B, l_2), q\} \\ S_3 &= \{\texttt{on}(B, l_2)\} \\ S_4 &= \{\texttt{on}(C, l_3), q\} \end{aligned}$$

The first three proofs are unchanged from the dumbbell problem. The fourth one argues that since C is at l_3, and only one block can be in any specific location, B must not be at l_3. From this point, the proof proceeds as for S_3.

Since the new proof set S_4 is a qualification proof set, the hitting set H is the same as in the previous example. Since H does not intersect S_4, the action fails.

The pulley Next, consider the pulley of Figure 10. There are three proofs that A is not located at l_4 (the point halfway between A and B's initial locations):

$$\begin{aligned} S_1 &= \{\texttt{on}(A, l_1)\} \\ S_2 &= \{\texttt{on}(B, l_2)\} \\ S_3 &= \{q\} \end{aligned}$$

The first proof, as usual, simply notes that A is elsewhere. The second is like S_3 in the dumbbell examples, noting that B is in the wrong place. The third is a consequence of the fact that, were A to be at l_4, the physics of the pulley system would force a violation of the qualifying domain constraint.

Here, S_1 and S_2 are the ramification proof sets. Since their minimal hitting set misses S_3, the action fails.

We have been assuming throughout this analysis that the connection represented by the domain fact $\texttt{pulley}(A, B)$ is protected. If, however, $\texttt{pulley}(A, B)$ is not protected, the proof sets become the following:

$$\begin{aligned} S_1 &= \{\texttt{on}(A, l_1)\} \\ S_2 &= \{\texttt{on}(B, l_2), \texttt{pulley}(A, B)\} \\ S_3 &= \{q, \texttt{pulley}(A, B)\} \end{aligned}$$

Now there are two hitting sets, given by

$$H_1 = \{\texttt{on}(A, l_1), \texttt{on}(B, l_2)\}$$

and

$$H_2 = \{\texttt{on}(A, l_1), \texttt{pulley}(A, B)\}.$$

Of these, only H_2 hits S_3, so we conclude that the action is not qualified, and the unique possible world resulting from its execution has the domain facts $\texttt{on}(A, l_1)$ and $\texttt{pulley}(A, B)$ removed. A moves and the pulley breaks, but B stays where it is.

The household domain Finally, we consider the action of moving the television to duct 1 in the household scene of Figure 4. There are three proofs that the television is not located on duct 1:

$$\begin{aligned} S_1 &= \{\texttt{on}(\texttt{tv}, \texttt{bottom-shelf})\} \\ S_2 &= \{\neg\texttt{stuffy(room)}, \texttt{blocked}(\texttt{duct2})\} \\ S_3 &= \{\neg\texttt{stuffy(room)}, \texttt{on}(\texttt{plant}, \texttt{duct2})\} \end{aligned}$$

In the first, we note that the television is elsewhere. In the second, we use the facts that the room is not stuffy and the other duct is already blocked. The third uses the facts that the room is not stuffy and that the plant is on the other duct, from which it follows that the other duct is blocked.

All of these sets are ramification proof sets. It follows that the action is unqualified, and that the result is as described in Section 2.2.

4.3 Explaining failures

An attractive property of the earlier descriptions of qualification using domain facts was that they identified the specific fact that was blocking a qualified action. This property is shared by the construction we are proposing.

If a set H that hits all of the ramification proof sets misses some qualification proof set, there are two possibilities. If the qualification proof set contains facts other than the qualifying domain constraint, then these facts are the collective cause of the qualification. If no such facts exist, the proposed action is in direct conflict with the domain constraints.

We have seen examples of both of these possibilities. In the blocked dumbbell problem, the proof set that was missed by the hitting set was

$$S_4 = \{\texttt{on}(C, l_3), q\}.$$

This correctly identifies $\texttt{on}(C, l_3)$ as the source of the problem.

The pulley problem (with the `pulley` fact protected) is somewhat different. Here, the problematic proof set is given by

$$S_3 = \{q\}.$$

This indicates that the action is in direct conflict with the domain constraint, and there is therefore no possible world in which it succeeds.

5 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 have referred 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.

This is in contrast with the "partially exhaustive" approach of Section 3.2. There, we were able to merely indicate which domain facts would *potentially* qualify the action by conflicting with possible ramifications of it. In contrast with the exhaustive approach, however, a potential qualification will only block an action if the success of the action is inconsistent with the qualification because of a domain constraint. We refer to this approach as partially exhaustive because of the need to explicitly delimit the potential qualifications.

The fully non-exhaustive, or *inferential* approach that we finally proposed in Section 3.3 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 three approaches. As in [6], 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.

This comparison extends the comparison appearing in [6], which discussed the computational properties of various solutions to the frame and ramification problems. We will continue, therefore, to use the notation of the previous paper:

1. The number of distinct action types in the domain will be denoted by α.

2. The number of relation symbols in the domain will be denoted by r.

3. The number of explicit consequences of an action will be denoted by x. This is the size of an average consequence set; note that x has no correlation with the number of ramifications of a typical action.

4. The average time needed to investigate the truth or falsity of any particular domain fact (presumably using a backward-chaining theorem prover) will be denoted by t.

5. Finally, the ratio of the number of domain constraints needed to describe a domain to the number of monotonic persistence rules needed will be denoted by κ. Since we will generally need αr monotonic persistence conditions, it follows that there will be $\kappa\alpha r$ domain constraints describing the domain under consideration. It was shown in [6] that $\kappa \leq 1$ in general, and argued that κ will be small for large domains.

5.1 Space requirements

We first consider the amount of space needed to represent qualification information using the three approaches. For the exhaustive approach, for example, we showed in [6] that an action might have up to $2^{\kappa\alpha}$ distinct ramifications. Since any of these might qualify it, there may be up to $2^{\kappa\alpha}$ distinct qualification conditions ("disablers") on any particular action. In general, of course, things will not be this bad, and we will therefore assume that there are only $\lambda 2^{\kappa\alpha}$ distinct qualification conditions on each action, where λ is some number less than 1.

The partially exhaustive approaches will, in the worst case, also need to explicitly list all of the qualifications to any particular action. In general, however, we expect there to be some simplification because the qualifications are being interpreted using domain constraints. The example in Section 3.2 is typical: The blocked dumbbell problem can be described using the single qualification (20) in the partially exhaustive approach, but will require both (2) and (3) in the exhaustive approach.

In [6], the factor κ was introduced to represent the expected savings obtained by using an approach to the *ramification* problem that selected ramifications inferentially from a potentially larger set using domain constraints. Selecting among potential qualifications using domain constraints seems similar, and we can therefore expect the space requirements of the partially exhaustive approach to be only κ those of its exhaustive counterpart.

The inferential approach is rather different. Here, we simplify the computation by taking advantage of the fact that many apparently distinct disabling conditions are consequences of a single domain constraint. We will see in Section 6.3 that there are situations in which it is not possible to derive all of the disabling conditions in this fashion; for the time being, we will define as *uniform* any domain in which this can in fact be done.

In a uniform domain, we address the qualification problem by identifying, for each action type a, which of the $\kappa a r$ domain constraints 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 $x\kappa a r$ domain constraints in order to describe $Q(a)$ for each action, where x is the number of consequences in a typical $C(a)$.

Theorem 5.1 *In a uniform domain, the space requirements of the various approaches to the qualification problem are given by:*

space	*exhaustive*	*partial*	*inferential*
worst case	$\alpha 2^{\kappa\alpha}$	$\alpha 2^{\kappa\alpha}$	$\kappa\alpha^2 r$
typical case	$\lambda\alpha 2^{\kappa\alpha}$	$\lambda\kappa\alpha 2^{\kappa\alpha}$	$x\kappa\alpha r$

It is only the inferential approach that does not suffer from an exponential deterioration in performance as the domain becomes increasingly complex.

5.2 Time

As we have already shown in Section 4, the inferential approach requires *no* additional time to determine that an action is unqualified. The exhaustive approach is not so effective, since each of the disabling conditions needs to be checked. This will cost time $t2^{\kappa\alpha}$ in the worst case, and $\lambda t 2^{\kappa\alpha}$ on average.

The partially exhaustive approach also needs no additional time to determine if an action is qualified. If the action is qualified, the approach will be somewhat faster than the original possible worlds formalization described in [6], since the possible worlds construction can be

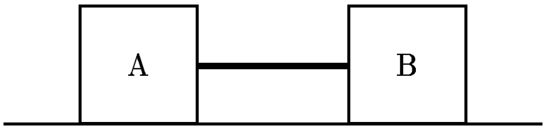


Figure 11: The dumbbell problem

curtailed early. If the action is unqualified, there will be no interaction between the possible worlds construction and the list of potential qualifications.

Theorem 5.2 *In a uniform domain, the time needed to confirm that an action is unqualified (above and beyond the time needed to compute its result) is given by:*

time	*exhaustive*	*partial*	*inferential*
worst case	$t2^{\kappa\alpha}$	0	0
typical case	$\lambda t2^{\kappa\alpha}$	0	0

6 Epistemological issues

6.1 Alternative approaches

Describing qualification in the fashion we have suggested is a fairly nonintuitive approach; it may seem more attractive to salvage the exhaustive approach in some way.

Suppose we return to the dumbbell problem, repeated in Figure 11. One possible way to treat this would be to view the motion as a continuous instead of as a discrete process, considering the action of moving A first halfway to B, and noting that A's final destination is clear as a result.

We have two objections to this. The first is that it is not clear that this solves the problem: What if A and B were extremely close when the action began? What if they were touching? Indeed, why doesn't the action remain qualified because of the presence of the rod joining A and B together?

Our second objection is that there are physical systems and implementations where motion really *is* quantized. It may not be meaningful to discuss the possibility of moving A "halfway" to B.

By assuming that the motion is continuous, we are in effect changing the language we are using to describe the situation and action in question. Another such ontological shift, mentioned in the introduction, involves solving the dumbbell problem by describing the situation in terms of forces and accelerations. Although the approach of describing a physical system in a more detailed way (i.e, with forces instead of `move` operators) will generally be sound, this redescription can increase the complexity of the axiomatization enormously. For problems as simple as the dumbbell problem, it should be unnecessary to describe the situation at this level of detail. In addition, as mentioned in Section 1.3, it is not clear that the ontological shift does not reintroduce similar problems at the more detailed level.

The construction we have proposed, although counterintuitive, provides a satisfactory description of qualification *without* requiring an ontological shift.

6.2 Preconditions as qualifications

An issue we have left principally unaddressed is that of precisely when it is possible to view a given precondition as a qualification. There are in fact two issues here: when it is *possible* to drop a precondition, rederiving it inferentially, and when it is *advisable* to do so.

The preconditions that cannot be viewed as qualifications are those that reflect information not found elsewhere in the database. In our description of our household domain, for example, the precondition to the `move` action given in (18) was that the object being moved be clear. We can easily imagine a robot capable of moving stacks of objects; if our robot lacks this ability, we must say so explicitly. We cannot drop the precondition that the object being moved be clear unless we are prepared to describe the physics of the robot's arm in a way that would entail this precondition.

The question of when it is *useful* to interpret a precondition as a qualification is a much more delicate one. In the computational discussion of Section 5, for example, we assumed throughout that the action being executed was *not* qualified, so that the total time needed to evaluate the action was approximately

$$2t\kappa\alpha,$$

which is the expression given in [6] as the time needed to compute the result of the action.

In the inferential approach we have described, the time needed to evaluate a qualified action will *also* be $2t\kappa\alpha$, since the result must still be constructed in order to determine that the action is qualified. Since simply checking for the qualification as one of p explicit preconditions will take time at most tp, it may often be advisable to retain as preconditions those qualifications that are reasonably likely to hold.

6.3 Difficulties

The approach we have presented draws its computational power from the fact that many apparently distinct preconditions to an action may in actuality be manifestations of a small number of domain constraints. We have represented this information by associating a set of qualifying domain constraints with each action.

In some cases, the simple approach we have described may be unable to characterize the domain effectively. For example, the qualification set may itself depend on information appearing elsewhere in the database. In our household domain, we might want to have

$$Q(\texttt{move}(b,l)) = \{\forall x, z, l'.\texttt{on}(x,l') \wedge z \neq x \wedge l' \neq \texttt{floor} \rightarrow \neg\texttt{on}(z,l')\}$$

if the object b being moved is too heavy to be displaced by the ventilation system, but

$$Q(\texttt{move}(b,l)) = \{\forall x,z,l'.\texttt{on}(x,l') \wedge z \neq x \wedge l' \neq \texttt{floor} \rightarrow \neg\texttt{on}(z,l'),$$
$$\texttt{blocked}(\texttt{duct1}) \wedge \texttt{blocked}(\texttt{duct2}) \leftrightarrow \texttt{stuffy(room)}\}$$

if it is not. (This example is similar to one suggested by Vladimir Lifschitz.)

This can be formalized by introducing a metalevel (or modal) relation $Q(a,q)$ which holds if q is a qualifying domain constraint for the action a. Now we can handle the above example by writing

$$Q(\texttt{move}(b,l), \forall x,z,l'.\texttt{on}(x,l') \wedge z \neq x \wedge l' \neq \texttt{floor} \rightarrow \neg\texttt{on}(z,l')),$$

together with

$$\texttt{light}(b) \rightarrow Q(\texttt{move}(b,l), \texttt{blocked}(\texttt{duct1}) \wedge \texttt{blocked}(\texttt{duct2}) \leftrightarrow \texttt{stuffy(room)}).$$

A potential difficulty with this approach is that it requires that we prove sentences of the form $Q(a,x)$ in order to determine whether or not an action a is qualified. This additional invocation of a theorem prover potentially invalidates the discussion of computational complexity underlying Theorem 4.3.

A more serious difficulty stems from the fact that, as mentioned in Section 6.2, there are many qualifications which simply do not arise as consequences of domain constraints. In our household domain, the bookcase may simply be too heavy to move, and there may be no domain constraint corresponding to this. For a fact such as this, there is essentially no choice but to explicitly label it as a precondition to the action being considered. The example in Section 6.2 is similar; there, we remarked that the robot's inability to move a stack of blocks did not correspond to any domain constraint.

It seems, however, that preconditions that are truly independent of domain constraints are rare. Consider the example of the preceding paragraph: The immobility of the bookcase is a consequences of the fact that heavy objects cannot be moved. Given such a domain constraint, the construction we have described could be applied to the potential action of moving the bookcase, and would identify its weight as the source of the problem.

7 Conclusion

In this paper, we have presented a computationally viable method for dealing with the qualification problem. Our approach to qualification allows us to determine inferentially under what circumstances a particular action will be qualified, avoiding the need to list explicitly all of the obstacles that any particular action might encounter. Since actions in complex domains will have a tremendous number of potential ramifications, and since any of these ramifications will potentially qualify the action itself, deriving the qualifications inferentially is of comparable importance to avoiding the need to explicitly list all of the action's ramifications.

We went on to describe an implementation of our ideas incorporating the possible worlds construction presented in [6]. Remarkably, the approach we presented incurs essentially no computational cost for actions that are not in fact qualified.

The solution we have proposed to the qualification problem is included in the possible worlds planning system discussed in [5]. There, the possible worlds approach is also used to focus planning search; for any goal or subgoal, we construct the closest possible world to the current state of the planner for which the goal holds, using this to select between competing subgoals or planning actions. This planning system and the control procedure based on nearness of possible worlds are described in detail in [5].

Acknowledgement

This work has been supported by DARPA under grant number N00039-86-C-0033 and by ONR under grant number N00014-81-K-0004. 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 Yoav Shoham for comments on some of our examples, and would especially like to thank Vladimir Lifschitz 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] R. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

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

[4] M. L. Ginsberg. Counterfactuals. *Artificial Intelligence*, 30:35–79, 1986.

[5] M. L. Ginsberg and D. E. Smith. Possible worlds planning. In preparation.

[6] M. L. Ginsberg and D. E. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35:165–195, 1988.

[7] 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.

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

[9] V. Lifschitz. Formal theories of action. In *Proceedings of the 1987 Workshop on the Frame Problem in Artificial Intelligence*, Lawrence, Kansas, 1987.

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

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

[13] 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.

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

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