On Strongest Necessary and Weakest Sufficient Conditions

Area: Reasoning Techniques – Deduction and Abduction

Fangzhen Lin (flin@cs.ust.hk)
Department of Computer Science
The Hong Kong University of Science and Technology
Clear Water Bay, Kowloon, Hong Kong

Abstract

Given a propositional theory T and a proposition q, a sufficient condition of q is one that will make q true under T, and a necessary condition of q is one that has to be true for q to be true under T. In this paper, we propose a notion of strongest necessary and weakest sufficient conditions. Intuitively, the strongest necessary condition of a proposition is the most general consequence that we can deduce from the proposition under the given theory, and the weakest sufficient condition is the most general abduction that we can make from the proposition under the given theory. We show that these two conditions are dual ones, and can be naturally extended to arbitrary formulas. We investigate some computational properties of these two conditions and discuss some of their potential applications.

1 Introduction

Given a propositional theory T and a proposition q, a sufficient condition α of q is one that will make q true under T: $T \models \alpha \supset q$. Symmetrically, a necessary condition α of q is one that has to be true for q to be true under T: $T \models q \supset \alpha$.

For instance, consider the following theory T:

$$rain \supset grassWet,$$
$$sprinklerOn \supset grassWet.$$

A sufficient condition for $grassWet$ to be true under T is for $rain$ to be true. Another (trivial) one is $grassWet$ itself. Yet another one is $sprinklerOn$. And yet another one is $rain \vee sprinklerOn$.

In this paper we shall propose a notion of weakest sufficient and strongest necessary conditions, study their properties, and consider ways of computing them. There are many potential applications. The following are two examples:

- **Abduction** As we can see from the above example, given an observation, there many be more than one abductive conclusions that we can draw. It should be useful to find the weakest of such conclusions, i.e. the weakest sufficient condition of the observation.

- **Definability** It is often necessary to determine whether a given theory yields a definition of a proposition in terms of a set of other propositions. For example, such computation is essential in both Simon's [12] and Pearl's [10] approaches to causation. This is also what is needed in order to compute successor state axioms [11] from causal theories. We believe definability is best handled using our proposed two conditions. While a proposition may or may not be definable in terms of a set of base propositions, our strongest necessary and weakest sufficient conditions, as we shall show, always exist and are unique up to logical equivalence. Furthermore, these two conditions are useful to have even when a proposition cannot be defined in terms of others. For instance, in the context of reasoning about actions, this situation arises when an action has an indeterminate effect on a proposition. In this case, the strongest necessary and weakest sufficient conditions of this proposition can be used to completely capture the effect of this action on the proposition.

This paper is organized as follows. We include logical preliminaries in section 2. In section 3, we define the notion of strongest necessary and weakest sufficient conditions, and prove some results that characterize these conditions. In section 4, we extend these two conditions to that of arbitrary formulas. In Section 5 we outline some algorithms for computing these two conditions and discuss some experimental results. Finally, we make some concluding remarks in section 6.

2 Logical preliminaries

We assume a propositional language. As usual, a truth assignment is a function from the set of propositions in the language to $\{true, false\}$, and the satisfiability relation $M \models \varphi$ is defined as usual between a truth assignment M and a formula φ.

Given a set P of propositions, a formula that mentions only propositions in P is called a *formula of* P. It is clear that the truth value of such a formula is determined by the truth values of the propositions in P only.

Given a formula φ, and a proposition p, we denote by $\varphi(p/true)$ the result of replacing every p in φ by $true$. We extend this notation to a set of formulas as well: if T is a set of formulas, then by $T(p/true)$ we mean the formula

$$\bigwedge_{\varphi \in T} \varphi(p/true).$$

We define $\varphi(p/false)$ and $T(p/false)$ similarly. It is clear that for any truth assignment M, $M \models \varphi(p/true)$ iff $M' \models \varphi$, where M' is the truth assignment that is exactly like M but $M'(p) = true$. A similar result holds for $\varphi(p/false)$.

A notion of forgetting, or eliminations of "middle terms" as Boole ([2], page 99) called it, will be essential here. Given a formula φ, and a proposition p, the result of forgetting p in φ, written $forget(\varphi; p)$, is the following formula:

$$\varphi(p/true) \vee \varphi(p/false).$$

Now given a finite set of propositions P, the result of forgetting P in φ, written $forget(\varphi; P)$, is defined inductively as:

$$forget(\varphi; \emptyset) = \varphi,$$
$$forget(\varphi; P \cup \{q\}) = forget(forget(\varphi; q); P).$$

It can be shown that for any formula φ and any propositions p_1 and p_2, $forget(forget(\varphi; p_1); p_2)$ and $forget(forget(\varphi; p_2); p_1)$ are equivalent. So $forget(\varphi; P)$ is well defined.

Since Boole [2] first defined it in 1854, this notion of forgetting, or eliminations of middle terms, has been re-discovered several times. Weber [14] re-introduced it and used it for updating propositional knowledge bases. It is also a special case of a more general notion of forgetting defined on first-order language by Lin and Reiter [8] for capturing database regression [7] and certain notion of relevance in problem solving. Lewis [5] (page 155) observed that "For purposes of application of the algebra[1] to ordinary reasoning, elimination is a process more important than solution, since most processes of reasoning take place through the elimination of 'middle' terms." As we shall see, this notion of "the elimination of middle terms" is certainly central for us here.

3 Strongest necessary and weakest sufficient conditions

We begin with strongest necessary conditions. As we shall see, weakest sufficient conditions are dual ones.

Definition 1 *Let T be a theory, P a set of propositions in T, and q a proposition in T but not in P. A formula φ of P is said to be a* necessary condition *of q on P under T if $T \models q \supset \varphi$. It is said to be a* strongest necessary condition *if it is a necessary condition, and for any other necessary condition φ', we have that $T \models \varphi \supset \varphi'$.*

The following proposition is straightforward:

Proposition 1 *If both φ and φ' are strongest necessary conditions of q on P under T, then $T \models \varphi \equiv \varphi'$.*

This means that if there is a strongest necessary condition, then it is unique up to logical equivalence under the background theory. So sometimes, we also call a strongest necessary condition *the* strongest one. As we shall see, strongest necessary conditions always exist.

[1] The author's note: "the algebra" = Boolean algebra as we know it today.

Example 1 The following examples provide some intuitions about our notion of strongest necessary conditions.

1. $T_1 = \{q \supset (p_1 \wedge p_2)\}$. The strongest necessary condition of q on $\{p_1, p_2\}$ under T_1 is $p_1 \wedge p_2$, and the strongest necessary condition of q on $\{p_1\}$ is p_1.

2. $T_2 = \{q \supset (p_1 \vee p_2)\}$. The strongest necessary condition of q on $\{p_1, p_2\}$ is $p_1 \vee p_2$, and the strongest necessary condition of q on $\{p_1\}$ is $true$ because neither p_1 nor $\neg p_1$ follow from q under T_2.

3. $T_3 = \{q \supset p_1, q\}$. A strongest necessary condition of q on $\{p_1\}$ is p_1. Another one is $true$. Of course, $T_3 \models p_1 \equiv true$.

The following theorem captures this notion of strongest necessary conditions in terms of truth assignments. It says that a necessary condition φ of q is a strongest one on P under T if and only if for any model of T, if it satisfies φ, then it can be modified into a model of q without changing the truth values of the propositions in P:

Theorem 1 *Let T, P, and q be as in Definition 1. Let φ be a necessary condition of q on P under T. Then φ is a strongest necessary condition of q on P under T iff for any model M of T and φ there is an assignment M' such that:*

1. *for any $p \in P$, $M'(p) = M(p)$.*

2. *$M' \models T \cup \{q\}$.*

We can use this theorem to verify that a certain necessary condition is in fact the strongest. For instance, consider $T_1 = \{q \supset p_1 \wedge p_2\}$ in Example 1 above. Clearly, p_1 is a necessary condition of q on $\{p_1\}$. It's a strongest one because given any model of T_1, if it satisfies p_1, then we can modify it into another model of T_1 and q while preserving the truth value of p_1. However, although it is also a necessary condition of q on $\{p_1, p_2\}$, it is not the strongest one this time, because if $M \models p_1 \wedge \neg p_2 \wedge \neg q$, then $M \models T_1$, but it cannot be modified into a model of q and T_1 without changing the truth value of p_2.

We can similarly define the notion of weakest sufficient conditions:

Definition 2 *Let T be a theory, P a set of propositions in T, and q a proposition in T but not in P. A formula ψ of P is said to be a* sufficient condition *of q on P under T if $T \models \psi \supset q$. It is said to be a* weakest sufficient condition *if it is a sufficient condition, and for any other sufficient condition ψ', we have that $T \models \psi' \supset \psi$.*

Strongest necessary and weakest sufficient conditions are in fact dual conditions. The easiest way to make this dual relationship precise is to extend these conditions from propositions to arbitrary formulas and to show that for any formula A, a formula φ is a strongest necessary (weakest sufficient) condition of A iff $\neg\varphi$ is a weakest sufficient (strongest necessary) condition of $\neg A$. This will be done in section 4

We now relate these two conditions to the notion of definability. We say that a theory T *defines a proposition* q on a set P of propositions iff there is a formula φ of P such that $T \models q \equiv \varphi$. The following proposition is straightforward:

Proposition 2 *A theory T defines a proposition q on P iff $T \models \varphi \supset \phi$, where φ is any strongest necessary condition of q on P and ϕ any weakest sufficient condition of q on P, both under the theory T.*

This basically reduces the problem of computing definability to that of computing strongest necessary and weakest sufficient conditions. An advantage of working with the latter is that the two conditions always exist and are unique up to logical equivalence under any given theory. Furthermore, they are useful to have even when they do not yield a definition of a proposition.

4 Strongest necessary and weakest sufficient conditions of a formula

Our notion of strongest necessary and weakest sufficient conditions can be extended from a proposition to an arbitrary formula.

Definition 3 *Let T be a propositional theory, α a formula, and P a set of propositions in $T \cup \{\alpha\}$. A formula φ of P is said to be a necessary condition of α on P iff $T \models \alpha \supset \varphi$. It is said to be a strongest necessary condition if for any other necessary condition φ', we have that $T \models \varphi \supset \varphi'$. Sufficient conditions and weakest sufficient conditions are defined similarly.*

Computing the strongest necessary and the weakest sufficient conditions of a formula can be reduced to that of a proposition, as the following proposition shows.

Proposition 3 *Let T, P, and α be as in Definition 3. A formula φ of P is the strongest necessary (weakest sufficient) condition of α on P under the theory T iff it is the strongest necessary (weakest sufficient) condition of q on P under the theory $T' = T \cup \{q \equiv \alpha\}$, where q is a new proposition not in T and α.*

Although not necessary in principle, the notion of sufficient and necessary conditions of a formula is very useful in expressing many properties. The following are some interesting ones.

The first one says that, as expected, if two formulas are equivalent under T, then they have the same strongest necessary and weakest sufficient conditions:

Proposition 4 *If $T \models \alpha \equiv \beta$, then for any set of propositions P, a formula φ is a strongest necessary (weakest sufficient) condition of α on P iff it is a strongest necessary (weakest sufficient) condition of β on P.*

The following proposition makes precise the dual relation between strongest necessary and weakest sufficient conditions:

Proposition 5 *A formula φ is the strongest necessary (weakest sufficient) condition of q iff $\neg\varphi$ is the weakest sufficient (strongest necessary) condition of $\neg q$, where all the conditions are on a common set of propositions and under a common theory.*

Proposition 6 *If φ_1, ..., φ_k are the strongest necessary conditions of $\alpha_1, ..., \alpha_k$, respectively, then $\varphi = \varphi_1 \vee \cdots \vee \varphi_k$ is the strongest necessary condition of $\alpha_1 \vee \cdots \vee \alpha_k$, where all the conditions are on a common set of propositions and under a common theory.*

Symmetrically, for weakest sufficient conditions, we have:

Proposition 7 *If φ_1, ..., φ_k are the weakest sufficient conditions of $\alpha_1, ..., \alpha_k$, respectively, then $\varphi = \varphi_1 \wedge \cdots \wedge \varphi_k$ is the weakest sufficient condition of $\alpha_1 \wedge \cdots \wedge \alpha_k$, where all the conditions are on a common set of propositions and under a common theory.*

This proposition is particularly useful in planning when we do not have a successor state axiom for every fluent: given a conjunctive goal $g_1 \wedge \cdots \wedge g_n$, to achieve it using action A, we need to compute the weakest sufficient condition of this conjunction, and reduce the conjunctive goal to it. By this proposition, instead of having to compute the weakest sufficient condition for every possible conjunction, it is enough to compute the condition for each conjunct ahead of time.

5 Computing strongest necessary and weakest sufficient conditions

There are several ways of computing these two conditions. We begin with prime implicates [13]. As with most reasoning tasks, strongest necessary and weakest sufficient conditions can be easily computed using prime implicates.

Proposition 8 *Let T be a theory, q a proposition, and P a set of propositions. Let Π be the set of prime implicates of T that mention only propositions in $P \cup \{q\}$.*

1. *The strongest necessary condition of q on P under T is equivalent to the following conjunction:*

$$\bigwedge\{C \mid \neg q \vee C \in \Pi\}.$$

2. *The weakest sufficient condition of q on P under T is equivalent to the following disjunction:*

$$\bigvee\{\neg C \mid q \vee C \in \Pi\}.$$

However, computing the strongest necessary (similarly weakest sufficient) condition of a proposition using prime implicates is almost always a bad idea, unless one wants these conditions for all propositions on all possible sets of propositions and want them in the form of prime implicates. This is because, in general, there is no viable way of computing the set Π in Proposition 8 short of computing the set of all prime implicates of a theory, and once having the set of prime implicates, one can just "read" out these two conditions using Proposition 8.

A better way is in terms of the notion of forgetting that we defined in Section 2, by using the following theorem:

Theorem 2 *Let T be a theory, P a set of propositions, and q a proposition in T but not in P. Let P' be the set of propositions that are in T but not in $P \cup \{q\}$. Then we have:*

1. *The strongest necessary condition of q on P is $forget(T(q/true); P')$.*
2. *The weakest sufficient condition of q on P is $\neg forget(T(q/false); P')$.*

We illustrate the use of this theorem using the following theory

$$T = \{q \supset p_1 \vee p_2\}.$$

According to the theorem, the strongest necessary condition of q on $\{p_1, p_2\}$ is $forget(T(q/true); \emptyset)$, which is $T(q/true)$ and equivalent to $p_1 \vee p_2$, the same as given in Example 1. The strongest necessary condition of q on $\{p_1\}$ is $forget(T(q/true); \{p_2\})$, which is equivalent to $forget(p_1 \vee p_2; \{p_2\})$, which by definition is $(p_1 \vee true) \vee (p_1 \vee false)$, thus equivalent to $true$.

By its definition, for any formula φ and proposition p, $forget(\varphi; p)$ returns a formula that is twice the size of φ; and for a set P of propositions, $forget(\varphi; P)$ returns a formula whose size is 2^n times the size of φ, where n is the size of P. Indeed, computing forgetting in the worst case is expensive. For instance, it is easy to see that a formula is satisfiable iff forgetting all propositions in it returns $true$, and a formula is not satisfiable iff forgetting all propositions in it returns $false$. However, $forget(\varphi; p)$ can often be simplified. For instance, if φ is in disjunctive normal form, then $forget(\varphi; p)$ can be computed efficiently to yield a formula that is shorter than φ: it is the result of replacing both p and $\neg p$ in φ by $true$. More generally, as Darwiche [3] showed, if φ is what he called decomposable negation normal form, then $forget(\varphi; p)$ can be computed efficiently.

In this paper, we consider two ways of computing forgetting: directly using its definition or through (again) prime implicates. They are embodied below as Algorithms 1 and 2 for computing strongest necessary conditions. The ones for weakest sufficient conditions are similar.

Input A set T of axioms, a set P of propositions, and a proposition q not in P.

Output A formula of P that is the strongest necessary condition of q on P under T.

Algorithm 1

1. Let $T_1 = \{\varphi \mid \varphi \in T$ is a sentence of $P\}$, and $T_2 = T - T1$. Replace q in T_2 by $true$, and transform the resulting theory into a *minimal set of clauses* C (see below for a definition). Similarly, transform T_1 into a minimal set of clauses C_0.
2. Let P' be the set of propositions in C but not in P.
3. If P' is empty, then go to post-processing, otherwise select a proposition p in P', and delete it from P'.
4. Transform $C(p/true) \vee C(p/false)$ into a minimal set of clauses, assign it to C, and go back to step 3.
5. *Post-processing:* the resulting C is, although a strongest necessary condition of q, often an unwieldy set of clauses; it can be simplified as follows:
   - eliminate those clauses in C that are subsumed by one of the clauses in C_0;
   - for each clause α in C, transform $(C - \{\alpha\}) \cup C_0$ into a minimal set of clauses C_α, and eliminate α from C if it is subsumed by one of the clauses in C_α.

Where a minimal set of clauses is one such that:

- all unit clauses are resolved;
- none of the clauses is subsumed by any other clause in the set.

Our experience with this algorithm has been that it spends the bulk of time on computing minimal sets of clauses in step 4. But if we do not require a set of clauses to be minimal, then the program quickly runs out of space.

Notice that according to our definition, for a set of clauses C to be a minimal one, only unit clauses in it need to be resolved. It is easy to see that given a set of clauses, transforming it into a minimal set can be done in polynomial time.

Alternatively, we can use prime implicates to compute forgetting. The following algorithm makes use of the fact that for any φ and P, $forget(\varphi; P)$ is equivalent to the conjunction of prime implicates of φ that do not mention any propositions in P.

Algorithm 3

1. Let $T_1 = \{\varphi \mid \varphi \in T$ is a sentence of $P\}$, and $T_2 = T - T1$. Transform T_1 into a minimal set of clauses C_0.

2. Generate the prime implicates of $T_2(q/true)$ using Tison's procedure [13], and let C be the set of those prime implicates that mention only propositions in P.

3. Go to post-processing as in Algorithm 1.

We have implemented the above two algorithms in SWI-Prolog[2], and run them on both random 3CNFs[3] and action theories used for computing successor state axioms in various action domains [6]. It turned out that Algorithm 2 is always slower than Algorithm 1, with on average a slow down of between 10 to 15 times. The reason is that computing prime implicates is expensive. Our notion of minimal sets of clauses seems to serve our purpose well here in cutting down the number and sizes of clauses without incurring too much cost in time.

Notice that in step 4 of Algorithm 1, after we have chosen a proposition to forget, we compute immediately the result of forgetting this proposition, $C(p/true) \vee C(p/false)$, by a set of minimal clauses. This is like breadth-first search, and as we have mentioned, is the bottle neck of this algorithm. Alternatively, we could work on each disjunct separately, and compute disjunctions only after all those propositions that need to be forgotten have been eliminated. As it turned out, this variant of Algorithm 1 performed far better than Algorithm 1 on random 3CNFs. For random 3CNFs with 50 variables and 215 clauses, this variant of Algorithm 1 spent on average 15 minutes[4] to return a strongest necessary condition of a proposition on a set of 10 other propositions, while Algorithm 1 did not return a solution after running overnight. However, this variant of Algorithm 1 is about 20% slower on action theories. The key is with the disjunction $C(p/true) \vee C(p/false)$: it can be simplified a lot for many benchmark action theories, but not much for random theories.

Regardless of which approach one uses, we have found that sometimes, it is a lot easier to compute the strongest necessary condition than the weakest

[2]SWI-Prolog is developed by Jan Wielemaker at University of Amsterdam

[3]Generated using a program by Kautz and Selman.

[4]On a SPARC Ultra2 machine running SWI-Prolog.

sufficient condition, and sometimes, it is the other way around. For instance, we have found that for may action theories, strongest necessary conditions are a lot easier to compute than weakest sufficient conditions. When one condition is much easier to compute, the following proposition will be very useful in computing the other one.

Proposition 9 *Let T be a theory, q a proposition, and P a set of propositions.*

1. *If φ is a necessary condition of q on P under T, and ψ a weakest sufficient condition of q on P under $T \cup \{\varphi\}$, then $\varphi \wedge \psi$ is a weakest sufficient condition of q on P under T.*

2. *If ψ is a sufficient condition of q on P under T, and φ a strongest necessary condition of q on P under $T \cup \{\neg\psi\}$, then $\varphi \vee \psi$ is a strongest necessary condition of q on P under T.*

Figure 1 shows this phenomenon for the problem of generating successor state axioms in a logistics domain using domain constraints such as "an object can be only at one location" and "if a package is inside a vehicle, and the vehicle is moved from one location to another, then the package is moved as well." In the figure, the x axle is the number of locations, and y axle the cpu run time in seconds on a SPARC Ultra 2 using Algorithm 1. The line labeled by *snc* corresponds to the strongest necessary condition of q on P, the one labeled by *wsc* the weakest sufficient condition of q on P, and *wsc*1 the weakest sufficient condition of q on P once the strongest sufficient condition has been added to the theory, where q stands for the proposition that the package is at location 2 after the action of moving the vehicle from location 1 to location 2 is successfully performed, and B is the set of propositions about the initial situation before the action is performed. Thus the weakest sufficient condition of q on B is the weakest condition about the initial situation that would ensure that the package is at location 2 after the action is performed, and the strongest condition is the strongest conclusion that one can infer about the initial situation once one knows that the package is at location 2 after the action is performed. The speedup of *wsc*1 over *wsc* is quite obvious, and the same is true for other action domains that we have experimented with, which include most of the benchmark planning domain in McDermott's PDDL library [9].

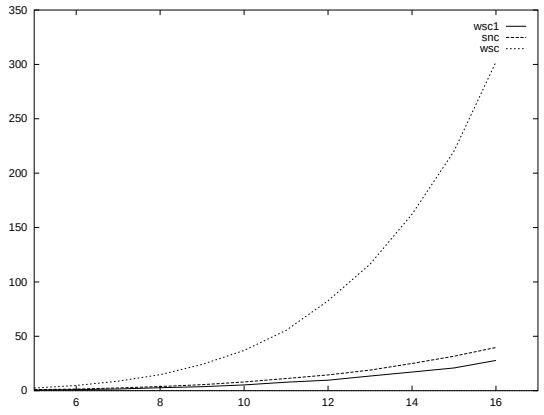

Figure 1: Logistics domain

6 Concluding remarks and future work

We have proposed a notion of strongest necessary and weakest sufficient conditions of a proposition, and considered ways of computing them. We believe these conditions have many potential applications in various areas including abduction and reasoning about actions.

There are several directions for future work. One of them is to extend the results here to the first-order case. This can be a difficult task. For instance, a result in [7] shows that forgetting in first-order case cannot in general be expressible in first-order logic. As a consequence, we expect that strongest necessary conditions of a proposition under a first-order theory cannot in general be expressible in first-order logic either. It seems that the best hope for dealing with the first-order case is to reduce it to the propositional one.

Finally, we have mentioned a few times that it is useful to compute strongest necessary and weakest sufficient conditions even when they do not yield a definition of a proposition. An example is in handling indeterminate actions, as discussed in [1, 4]. Another example is in handling incomplete knowledge bases. We believe the results here will be useful there.

References

[1] M. Bjäreland and L. Karlsson. Reasoning by regression: pre- and postdiction procedures for logics of action and change with nondeterminism. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*

(IJCAI–97), IJCAI Inc. Distributed by Morgan Kaufmann, San Mateo, CA., pages 1420–1425, 1997.

[2] G. Boole. *An Investigation of the Laws of Thought.* Walton, London, 1854. (Reprinted by Dover Books, New York, 1954).

[3] A. Darwiche. Compiling knowledge into decomposable negation normal form. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI–99), IJCAI Inc. Distributed by Morgan Kaufmann, San Mateo, CA.*, 1999.

[4] E. W. Dijkstra and C. S. Scholten. *Predicate Calculus and Program Semantics.* Springer-Verlag, New York, 1990.

[5] C. I. Lewis. *A Survey of Symbolic Logic.* University of California Press, Berkeley, 1918. (Reprinted by Dover Pub's., Inc., New York, 1960. Chapter II in The Classic or Boole-Schroder Algebra of Logic).

[6] F. Lin. From causal theories to successor state axioms: bridging the gap between nonmonotonic action theories and STRIPS-like formalisms. In *http://www.cs.ust.hk/faculty/flin*, 1999. Submitted.

[7] F. Lin and R. Reiter. How to progress a database. *Artificial Intelligence*, (92)1-2:131–167, 1997.

[8] F. Lin and R. Reiter. Forget it! In R. Greiner and D. Subramanian, editors, *Working Notes of AAAI Fall Symposium on Relevance*, pages 154–159. The American Association for Artificial Intelligence, Menlo Park, CA, November 1994.

[9] D. McDermott et al. (AIPS-98 Planning Competition Committee). PDDL – the planning domain definition language. Technical Report Tech Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

[10] J. Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–709, 1995.

[11] R. Reiter. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 418–420. Academic Press, San Diego, CA, 1991.

[12] H. Simon. Causal ordering and identifiability. In W. Hood and T. Koopmans, editors, *Studies in Econometric Method*. John Wiley and Sons, New York, 1953.

[13] P. Tison. Generalized consensus theory and application to the minimization of boolean functions. *IEEE Trans. on Electronic Computers*, EC-16/4:446–456, 1967.

[14] A. Weber. Updating propositional formulas. In *Proceedings First Conference on Expert Database Systems*, pages 487–500, 1986.