Sound and Efficient Closed-World Reasoning for Planning

Technical Report 95-02-02

Oren Etzioni Keith Golden Daniel Weld*
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
{etzioni, kgolden, weld}@cs.washington.edu

February 21, 1995

Abstract

Closed-world reasoning is the process of inferring that a logical sentence is false based on its absence from a knowledge base, or the inability to derive it. Previous work on circumscription, autoepistemic logic, and database theory has explored logical axiomatizations of closed-world reasoning, and investigated computational tractability for propositional theories. Work in planning has traditionally made the closed world *assumption* but has avoided closed-world reasoning. We take a middle position, and describe a novel method for closed-world reasoning over the first-order theories of action used by planning algorithms such as NONLIN, TWEAK, and UCPOP. We show the method to be both sound and efficient. The method has been incorporated into the XII planner [22], which supports our Internet Softbot (software

*We thank Tom Dean, Denise Draper, Steve Hanks, Rao Kambhampati, Craig Knoblock, Neal Lesh, Martha Pollack, Rich Segal, Yoav Shoham, Mike Wellman, and Mike Williamson for helpful comments on earlier drafts of this paper. Omid Madani contributed to the formal aspects of the paper, and Rob Spiger contributed to the implementation. This research was funded in part by Office of Naval Research Grants 90-J-1904 and 92-J-1946, by a grant from the University of Washington Royalty Research Fund, and by National Science Foundation Grants IRI-8957302, IRI-9211045, and IRI-9357772. Golden is supported in part by a UniForum Research Award.

robot) [13]. In our experiments, closed-world inference consistently averaged about 2 milliseconds, while updates averaged approximately 1.2 milliseconds.

1 INTRODUCTION AND MOTIVATION

Classical planners such as NONLIN [46], TWEAK [5], or UCPOP [40, 47] presuppose correct and complete information about the world. Having complete information facilitates planning, since the planning agent need not obtain information from the external world — information absent from the agent's world model is assumed to be false (this is the infamous *closed world assumption* [43]). However, in many cases, an agent has incomplete information about its world. For instance, a robot may not know the size of a bolt or the location of an essential tool [35]. Similarly, a software agent, such as the Internet Softbot [17, 14, 13], cannot be familiar with the contents of *all* the bulletin boards, FTP sites, and files accessible through the Internet.[1]

What do we mean by incomplete information? In this paper, we focus on incomplete but correct information about the state of the external world (see Section 2.1 for a formal description). In contrast to work on relational database theory [23], we do not assume that all objects in the external world are known in advance; agents constantly encounter new objects. In addition, we do not assume that the world is static; agents constantly sense (or cause) changes to the world. In Section 5, we consider the implications of changes that the agent fails to sense.

Recent work has sketched a number of algorithms for planning with incomplete information (*e.g.*, [1, 35, 12, 28, 41]). These algorithms make the *open world assumption* — information not explicitly represented in the agent's world model is unknown. Because they make the open world assumption, none of the above algorithms handle universally quantified goals. The planners cannot satisfy even a simple goal such as "Print all of Smith's postscript files in the `/kr94` directory" because they have no way to guarantee that they are familiar with *all* the relevant files. In addition, these planners are vulnerable to *redundant information gathering* when they plan to "sense" information that is already known to the agent [16]. Since satis-

[1]Because our work is motivated by the softbot, most of our examples are drawn from the Internet and UNIX domains. However, we emphasize that our results are general and corresponding examples are easily found in physical domains as well.

fying the preconditions of an information-gathering action can involve arbitrary planning, the cost of redundant information gathering is unbounded in theory and large in practice [22].

We can still salvage a partial notion of complete information, even in the presence of unknown facts. Many sensing actions return exhaustive information which warrants *local closed world* information (`LCW`). For example, scanning with a TV camera shows *all* objects in view, and the UNIX `ls -a` command lists *all* files in a given directory. After executing `ls -a`, it is not enough for the agent to record that `paper.tex` and `proofs.tex` are in `/kr94` because, in addition, the agent knows that *no other* files are in that directory. Note that the agent is not making a closed world *assumption*. Rather, the agent has executed an action that yields closed world *information*.

The agent stores the limited information it has about the external world in a database $\mathcal{M}$, which we refer to as its "incomplete world model." To represent `LCW`, we utilize an explicit database of meta-level sentences such as "I know all the files in `/kr94`." The sentences describe the limited instances over which the information in $\mathcal{M}$ is in fact a complete model of the external world. The information in the `LCW` database is equivalent to the "closed roles" found in knowledge-representation systems such as CLASSIC [2] and LOOM [3], to predicate completion axioms [6, 27], and to circumscription axioms [33, 32]. Our contributions include:

- A sound and efficient calculus for answering queries based on the `LCW` database. The calculus answers queries such as: if the agent is familiar with all the files in the directory `/kr94`, and with all group-readable files on the file system, does it follow that the agent is familiar with all the group-readable files in `/kr94`?

- A sound and efficient calculus for updating the `LCW` database as the state of the world changes. The update calculus answers questions such as: if the agent knows the lengths of all the files in `/kr94`, and a file is added to `/kr94`, is the agent still familiar with the lengths of all files in that directory? What if a file is deleted from `/kr94`?

- Efficient algorithms, based on the calculus, for querying a locally complete database of domain propositions (Appendix A). We experimentally evaluate the performance of our query and update algorithms in the UNIX domain (Section 4).

1.1 Previous Work

Below, we briefly review the large body of related work on circumscription, autoepistemic logic, and database theory. At the end of this section, we summarize the key differences between this body of work and ours.

The bulk of previous work has investigated the *logic* of closed world reasoning (*e.g.*, [26, 10, 44, 34, 29]), and the semantics of theory updates (*e.g.*, [21, 24, 8]). Results include logical axiomatizations of the closed world assumption (CWA), exploring the relationship between CWA and circumscription, distinguishing between knowledge base revision and knowledge base update, and more. Although decidable computational procedures have been proposed in some cases (*e.g.*, [20], and the Minimality Maintenance System [42]), they remain intractable. Update procedures have been described that involve enumerating the possible logical models corresponding to a database (e.g., [48, 7]), or computing the disjunction of all possible results of an update [25]. In contrast, we adopt the WIDTIO (When In Doubt Throw It Out [49]) policy. As [9] points out, this method is easy to implement efficiently but has the disadvantage that in the worst case, all knowledge in the database has to be retracted. We demonstrate experimentally that our update scheme is effective in practice.

Levy [30] has pointed out a close relationship between closed-world reasoning and the problem of detecting the independence of queries from updates. However, the computational model in the database literature (Datalog programs) is different from our own. Furthermore, tight complexity bounds for this problem have not been reported in the database literature (*e.g.*, [31] merely reports decidable algorithms).

Recently, some excellent analyses of computational complexity have emerged [4, 9], which show that the different approaches described in the literature are highly intractable in the general case. Stringent assumptions are required to make closed-world reasoning tractable. For example, Eiter and Gottlob [9, page 264] show that propositional Horn theories with updates and queries of bounded size yield polynomial-time algorithms. However, all positive computational tractability results reported in [4, 9] are restricted to *propositional* theories. Motivated by the need for closed-world reasoning in modern planning algorithms, we have formulated a rather different special case where the knowledge bases record first-order information, queries are first-order conjunctions, and updates are atomic.

In short, there are three fundamental differences between the results in this paper and previous work. First, the bulk of previous work has focused on the *logic* of closed-world reasoning, not on its computational tractability. The thorough analyses of computational complexity in [4, 9] have found closed world inference and update to be inherently intractable in the general case. In fact, previous work has only found tractable algorithms for restricted classes of propositional theories. Second, we have formulated efficient closed-world reasoning algorithms for first-order theories of the sort used by modern planners.[2] Finally, we not only show that our algorithms run in polynomial time (for queries and updates of bounded size), but also carry out a host of experiments demonstrating them to be efficient in practice (Section 4).

1.2 Organization

The paper is organized as follows. Section 2 introduces our calculus for answering `LCW` queries in a static universe. In Section 3 we present our calculus for updating `LCW` as the world changes. Although our approach is sound, it is not complete. Section 4 uses empirical techniques to show that the incompleteness is not burdensome in practice, and that our approach is fast.

After concluding with a discussion of future work, we present additional details in two appendices. Appendix A provides pseudo-code algorithms based on our inference rules, and appendix B proves that our rules, and the algorithms in Appendix A, are sound.

2 Incomplete Information about the World

We've argued that when acting in complex, real-world domains such as the Internet, no agent can have complete information. In this section we present a compact representation for incomplete world models and describe a set of sound and tractable inference rules for reasoning about local closed world information. Subsequent sections explain how to keep the representation consistent as the world changes.

[2]Since we consider formulae with an essentially unbounded number of instances, it is impractical to translate our first-order theories into propositional correlates. Furthermore, as shown in Sections 2.4 and 3, local closed-world reasoning makes essential use of first-order constructs such as unification.

2.1 Semantics

We begin by formalizing the notion of an incomplete world model. At every point in time, the world is in a unique state, w, which may be unknown to the agent. For any ground, atomic sentence φ, either $\mathsf{w} \models \varphi$ or $\mathsf{w} \models \neg\varphi$ hence the set of ground facts entailed by the world forms a complete logical theory, which we denote $\mathcal{W}$. Following [36, 18] and many others, we formalize an agent's incomplete information with a set of possible world states, $\mathcal{S}$, that are consistent with its information. Since we assume what information the agent *does* have is correct, the current world state, w, is necessarily a member of $\mathcal{S}$. We say that φ is known to the agent (written $\mathcal{S} \models \varphi$) just in case $\forall \mathsf{s} \in \mathcal{S},\ \ \mathsf{s} \models \varphi$. We say that the agent possesses complete information when $\mathcal{S}$ and w entail exactly the same set of facts. Incomplete information means that there are facts, φ, such that neither $\mathcal{S} \models \varphi$ nor $\mathcal{S} \models \neg\varphi$; in this case we say φ is unknown to the agent.

Thus, we say that an atomic formula, φ, has truth value `T` if $\mathcal{S} \models \varphi$, has truth value `F` if it is known to be false, or has truth value `U` if it is unknown to the agent.

2.2 Local Closed World Information

We say that an agent has *local closed world information* (`LCW`) relative to a logical formula Φ if every ground sentence which unifies with Φ is either entailed by $\mathcal{S}$ or is provably false:[3]

$$\texttt{LCW}(\Phi) \equiv \forall\theta(\mathcal{S} \models \Phi\theta) \vee (\mathcal{S} \models \neg\Phi\theta) \tag{1}$$

In essence, this definition specifies which *parts* of the logical theory induced by $\mathcal{S}$ are complete (cf. [11] and others). Note that since the theory induced by $\mathcal{S}$ is a subset of the complete theory induced by w, the definition of `LCW` amounts to a limited correspondence between $\mathcal{S}$ and w. If `LCW`(Φ) holds, then all states in $\mathcal{S}$ (including w) agree on the extension of Φ. As a concrete example, suppose that `parent.dir(`f`,`d`)` means "The parent directory of file f is directory d;" then we can encode the fact that an agent knows all the files in the directory `/kr94` with:

$$\texttt{LCW(parent.dir(}f\texttt{, /kr94))}$$

[3]We use *italics* to denote free variables and write $\Phi\theta$ to denote the result of applying the substitution θ to the formula Φ.

If the agent knows that paper.tex and proofs.tex are in /kr94 then this LCW formula is equivalent to the following implication:

$$\forall f\ \texttt{parent.dir}(f,\ \texttt{/kr94}) \rightarrow \\ (f = \texttt{paper.tex}) \vee (f = \texttt{proofs.tex})$$

An LCW formula can also be understood in terms of circumscription [32]. For the example above, one defines the predicate $\texttt{P}(x)$ to be true exactly when $\texttt{parent.dir}(x,\ \texttt{/kr94})$ is true, and circumscribes P in the agent's theory. While our work can be understood within the circumscriptive framework, our implemented agent requires the ability to infer and update[4] closed world information *quickly.* Thus, we have developed computationally tractable closed-world reasoning and update methods, applicable to the restricted representation languages used by modern planning algorithms.

2.3 Representing Closed World Information

Below, we explain how our agent represents its incomplete information about the world, and how it represents LCW in this context. Due to the size of $\mathcal{S}$ (a potentially infinite set of large structures), the agent cannot represent $\mathcal{S}$ explicitly. Instead we represent the facts known by the agent with a database, $\mathcal{M}$, of ground literals. Formally, $\mathcal{M}$ is a subset of the logical theory induced by $\mathcal{S}$; if $\varphi \in \mathcal{M}$ then $\mathcal{S} \models \varphi$. Since the theory induced by $\mathcal{S}$ is incomplete, the Closed World Assumption (CWA) cannot be applied to $\mathcal{M}$. The agent cannot automatically infer that any atomic formula absent from $\mathcal{M}$ is false. Thus, the agent is forced to represent false facts in $\mathcal{M}$, explicitly, as sentences tagged with the truth value F.

This observation leads to a minor paradox: the agent cannot explicitly represent in $\mathcal{M}$ *every* sentence it knows to be false (there is an infinite number of files *not* in the directory /kr94). Yet the agent cannot make the CWA. We adopt a simple solution: we represent local closed world information explicitly as a meta-level database, $\mathcal{L}$, containing localized closure axioms of the form $\texttt{LCW}(\Phi)$; these record *where* the agent has closed world information. Together, the $\mathcal{M}$ and $\mathcal{L}$ databases specify an agent's state of

[4]Following [24, 25] we distinguish between *updating* a database and *revising* it. We assume that our agent's knowledge is correct at any given time point, hence there is no need to revise it. When the world changes, however, the agent may need to update its model to remain in agreement with the world.

incomplete information about the world (*i.e.*, they correspond to the set, $\mathcal{S}$, of states).

When asked whether it believes an atomic sentence φ, the agent first checks to see if φ is in $\mathcal{M}$. If it is, then the agent responds with the truth value (`T` or `F`) associated with the sentence. However, if $\varphi \notin \mathcal{M}$ then φ could be either `F` or unknown (truth value `U`). To resolve this ambiguity, the agent checks whether $\mathcal{L}$ entails $\texttt{LCW}(\varphi)$. If so, the fact is `F`; otherwise it is `U`. Appendix A formalizes this intuitive procedure by providing pseudo code for the Query algorithm. We note, however, that the agent need not perform inference on $\mathcal{M}$ since it contains only ground literals, but it *may* need to perform some deduction on its `LCW` sentences; this inference process is described below.

2.4 Inferring Local Closed World Information

An agent requires information about the external world `w`, but only has direct access to $\mathcal{M}$ and $\mathcal{L}$. The agent needs to answer queries such as "Do I know all the postscript files in `/kr94`?" or, more formally, is the following true:

$$\texttt{LCW}(\texttt{parent.dir}(f, \texttt{/kr94}) \wedge \texttt{postscript}(f))$$

Correctly answering `LCW` queries is not a simple matter of looking up assertions in a database. For instance, suppose the agent wants to establish whether it is familiar with all the files in `/kr94`, and it finds that it is familiar with all the files in *all* directories. Then, *a fortiori*, it is familiar with all the files in `/kr94`. That is:

$$\texttt{LCW}(\texttt{parent.dir}(f, d)) \models \texttt{LCW}(\texttt{parent.dir}(f, \texttt{/kr94}))$$

In general, we have:

Theorem 1 (Instantiation Rule) *If Φ is a logical formula and θ is a substitution, then* $\texttt{LCW}(\Phi) \models \texttt{LCW}(\Phi\theta)$.[5]

Moreover, `LCW` assertions can be combined to yield new ones. For instance, if the agent knows all the group-readable files, and it knows which files are located in `/kr94`, it follows that it knows the set of group-readable files in `/kr94`. In general, we have:

[5]Proofs of the theorems are sequestered in Appendix B.

Theorem 2 (Conjunction Rule) *If Φ and Ψ are logical formulae then* $\texttt{LCW}(\Phi) \wedge \texttt{LCW}(\Psi) \models \texttt{LCW}(\Phi \wedge \Psi)$.

And, also:

Theorem 3 (Disjunction Rule) *If Φ and Ψ are logical formulae then* $\texttt{LCW}(\Phi) \wedge \texttt{LCW}(\Psi) \models \texttt{LCW}(\Phi \vee \Psi)$.

The intuition behind these rules is simple — if one knows the contents of two sets then one knows their intersection. (the Conjunction Rule) and their union (the Disjunction Rule). Note that the converse of the Conjunction Rule is invalid. If one knows the group-readable files in `/kr94`, it does not follow that one knows *all* group-readable files. The rule $\texttt{LCW}(\Phi) \models \texttt{LCW}(\Phi \wedge \Psi)$ is also invalid. For instance, if one knows all the group-readable files, it does not follow that one knows exactly which of these files reside in `/kr94`.

However, if one knows the group-readable files, and for each group-readable file, one knows whether that file is in `/kr94`, then one knows the set of group-readable files in `/kr94`. In general, if we know the contents of set A, and for each member of A, we know whether that member resides in another set B, then we know the intersection of sets A and B. More formally:

Theorem 4 (Composition Rule) *If Φ and Ψ are logical formulae and* $\texttt{LCW}(\Phi)$ *and for all substitutions σ, if $\mathcal{S} \models \Phi\sigma$ implies* $\texttt{LCW}(\Psi\sigma)$*, then we can conclude* $\texttt{LCW}(\Phi \wedge \Psi)$.

2.5 Discussion

To make `LCW` inference and update tractable, we restrict the formulae in $\mathcal{L}$ to conjunctions of positive literals. As a result, we lose the ability to represent `LCW` statements that contain negation or disjunction such as "I know the protection of all files in `/kr94` *except* the files with a `.dvi` extension."[6] Thus $\mathcal{M}$ and $\mathcal{L}$ form a *conservative representation.* For any consistent $\mathcal{M}$, $\mathcal{L}$ pair, there exists an $\mathcal{S}$ that entails the same set of `LCW` sentences, but the converse is false.

This restriction provides significant efficiency gains. To see this, consider a singleton `LCW` query such as $\texttt{LCW}(\texttt{parent.dir}(f, \texttt{/kr94}))$. If $\mathcal{L}$ contains only positive conjunctions, the query can be answered in sub-linear

[6]Thus, we do not implement the Disjunction Rule, which yields disjunctive `LCW` sentences. The Disjunction Rule is the only rule in the paper that is described but not implemented.

time — examining only singleton `LCW` assertions indexed under the predicate `parent.dir`. If negation is allowed, however, then a combination of multiple `LCW` sentences has to be explored. For instance, by the Disjunction Rule, we have that $\texttt{LCW}(\Phi \wedge \Psi) \wedge \texttt{LCW}(\Phi \wedge \neg\Psi) \models \texttt{LCW}(\Phi)$. Introducing disjunctive `LCW` sentences into $\mathcal{L}$ would make matters even worse. In general, answering a singleton query, in the presence of negation and disjunction, is NP-hard. Since our planner makes numerous such queries, we chose to sacrifice completeness in the interest of speed.

Moreover, since $\mathcal{L}$ is restricted to positive conjunctions, `LCW` inference is reduced to the problem of matching a conjunctive `LCW` query against a database of conjunctive `LCW` assertions. Appendix A provides pseudo-code for the actual algorithms, but the intuition is straightforward. A successful match occurs when repeated applications of the Conjunction Rule decompose the query into subconjunctions, which are directly satisfied by the Instantiation Rule applied to $\mathcal{L}$, or by the Composition Rule applied to $\mathcal{L}$ and $\mathcal{M}$.

In the worst case, we have to consider all possible decompositions, which is exponential in c, the number of conjuncts in the query. The worst-case time complexity is, therefore, $O(|\mathcal{L}|^c)$. However, if the number of conjuncts c is bounded by a constant b, then the match time is a polynomial of order b in the size of $\mathcal{L}$. In the UNIX domain, we have found that `LCW` queries are typically short ($c \leq 2$) which, with the aid of standard indexing techniques, yields extremely fast `LCW` inference in practice. In our experiments, `LCW` queries averaged about 2 milliseconds (see Section 4 for the details).

3 Updating Closed World Information

As the agent is informed of the changes to the external world — through its own actions or through the actions of other agents — it can gain and lose information about the world. For example, after executing the UNIX command `finger weld@june`, the agent should update $\mathcal{M}$ with the newly observed truth value for `(active.on weld june)`. Similarly, an agent's actions can cause it to gain or lose `LCW`. When a file is compressed, for example, the agent loses information about its size; when all postscript files are deleted from a directory, the agent gains the information that the directory contains no such files. This section presents a sound and efficient method for updating $\mathcal{L}$, the agent's store of `LCW` sentences.

Recall from Section 2.5 that together $\mathcal{M}$ and $\mathcal{L}$ approximate the incomplete theory induced by $\mathcal{S}$: some facts are known `T`, some `F`, and some are `U`. Execution of an action can change both the world state (from `w` to `w'`) and the agent's model (from $\mathcal{S}$ to $\mathcal{S}'$). Since it is useful to refer concisely to the changes between the theories entailed by $\mathcal{S}$ and $\mathcal{S}'$, we introduce a new notation: the Δ function. For example, suppose that initially the agent doesn't know whether `weld` is active on the machine called `june`, so it executes a `finger` action which `observe`s that `weld` *is* active. We describe the resulting update to the model as $\Delta($`(active.on weld june)`$, \mathtt{U} \rightarrow \mathtt{T})$.

In this case, only a single atomic formula had its truth value changed. However, in some cases an *unbounded* number of atoms change their truth values; for example, if $\mathcal{S} \models$`size(paper.tex, 4713)` then numerous atoms change their truth value when `compress paper.tex` is executed and the size of `paper.tex` becomes unknown: `size(paper.tex, 4713)` changes from `T` to `U`, while `size(paper.tex, 4712)` (and many similar atoms) change from `F` to `U`. In this case, we summarize the change with the expression $\Delta(\mathtt{size(paper.tex,}\ x\mathtt{)}, (\mathtt{T} \vee \mathtt{F}) \rightarrow \mathtt{U})$. In general, we define Δ in terms of its six primitive cases: $\mathtt{T} \rightarrow \mathtt{F}$, $\mathtt{T} \rightarrow \mathtt{U}$, $\mathtt{F} \rightarrow \mathtt{T}$, $\mathtt{F} \rightarrow \mathtt{U}$, $\mathtt{U} \rightarrow \mathtt{T}$, and $\mathtt{U} \rightarrow \mathtt{F}$.[7] Equation 2 provides the definition of one of these cases; the others are analogous.

$$\Delta(\varphi, \mathtt{U} \rightarrow \mathtt{T}) \equiv \forall\theta(\mathcal{S} \not\models \varphi\theta) \wedge (\mathcal{S} \not\models \neg\varphi\theta) \wedge (\mathcal{S}' \models \varphi\theta) \tag{2}$$

$\mathcal{L}$ can change when the agent executes an action or when the agent is informed of an exogenous change to the state of the world. We formulate the update policy as a set of rules and state them as theorems since they are sound: *i.e.*, they preserve the *conservative model* invariant: for any sentence Φ if $\mathcal{M} \cup \mathcal{L} \models \Phi$ then $\mathcal{S} \models \Phi$. The general pattern of these theorems starts with the assumption that the agent has a conservative model of $\mathcal{S}$ and claims that a particular value for $\mathcal{L}'$ results in a conservative model of $\mathcal{S}'$, given an atomic change of a particular Δ format. While these update rules are sound, they are often incomplete; fortunately, they can be computed in polynomial time as we show in Section 3.5.

We sidestep the ramification problem [21] by demanding that updates to $\mathcal{M}$ (such as those caused by action execution) explicitly enumerate changes to *every* predicate that is affected. Note that this is standard in the planning literature. For example, a STRIPS operator that moves block `A` from `B` to `C` must delete `on(A, B)` and also add `clear(B)` even though `clear(B)` can be defined as $\forall y\ \neg$`on(`y`,B)`.

[7] A change such as $\Delta(\varphi, (\mathtt{T} \vee \mathtt{F}) \rightarrow \mathtt{U})$ is defined as $\Delta(\varphi, \mathtt{T} \rightarrow \mathtt{U}) \vee \Delta(\varphi, \mathtt{F} \rightarrow \mathtt{U})$.

By distinguishing between transitions to and from U truth values, $\mathcal{L}$ updates can be divided into four mutually exclusive and exhaustive cases which we call Information Gain, Information Loss, Domain Growth, and Domain Contraction. Below, we consider each case in turn.

3.1 Information Gain

An agent gains information when it executes an information-gathering action (*e.g.*, `wc` or `ls`), or when a change to the world results in Information Gain. In general, if the information in the agent's world model increases, the agent cannot lose `LCW`.

Theorem 5 (Information Gain Rule) *If the agent has a conservative model and an atomic change has no effects other than* $\Delta(\varphi, \mathtt{U} \rightarrow \mathtt{T} \vee \mathtt{F})$ *then* $\mathcal{L}' \mathtt{:=} \mathcal{L}$ *yields a conservative model.*

This theorem suggests a simple conservative policy: when an action gains information, $\mathcal{L}$ need not be modified. However, by analyzing the form of the information gained and exploiting the assumption of correct information, it is possible to do better. For example, when $\mathcal{M}$ is updated to contain information about the unique value of a variable (*e.g.*, the word count of a file), the agent can infer that it has local closed world information. In many cases, predicates implicitly represent functions. For example, the predicate `word.count` is a function from a file to its word count. In general, we have:

Theorem 6 (Function Rule (after Smith [45])) *If the agent has a conservative model and an atomic change has no effects other than* $\Delta(P(c), \mathtt{U} \rightarrow \mathtt{T})$ *for some constant c and some predicate P where*

$$\forall z, y P(z) \wedge P(y) \models (z = y) \qquad (3)$$

then $\mathcal{L}' \mathtt{:=} \mathcal{L} \cup P(x)$ *yields a conservative model.*

To use the Function Rule in practice, our agent relies on explicit axioms, of the form in Equation 3, that indicate which predicates are functional (and in which arguments). Examples abound in the UNIX domain including file properties such as word count, parent directory, user properties such as logname, home machine, and more. The Function Rule turns out to be very useful (cf. [35]).

An agent can obtain local closed world information, even when the cardinality of a domain is variable (*e.g.*, the files in a directory) by executing an action with universally quantified effects. For instance, the execution of UNIX `chmod g+r *` in the directory `/kr94` provides information on the group read protection of *all* files in that directory, since after the command is executed, all the files will be group readable

Theorem 7 (Causal Forall Rule) *If the agent has a conservative model and an atomic change has no effects other than* $\Delta(\mathtt{P}(x), \mathtt{F} \vee \mathtt{T} \vee \mathtt{U} \rightarrow \mathtt{T})$ $\forall x$ *satisfying* $\mathtt{Q}(x)$ *and* $\mathcal{L} \models \mathtt{LCW}(\mathtt{Q}(x))$ *then* $\mathcal{L}' := \mathcal{L} \cup \{\mathtt{P}(x) \wedge \mathtt{Q}(x))\}$ *yields a conservative model.*

Universally quantified observational effects provide similar information as summarized by the following proposition.

Theorem 8 (Observational Forall Rule) *If the agent has a conservative model and an atomic change has no effects other than* $\Delta(\mathtt{P}(x), \mathtt{U} \rightarrow \mathtt{T} \vee \mathtt{F})$[8] $\forall x$ *satisfying* $\mathtt{Q}(x)$ *and* $\mathcal{L} \models \mathtt{LCW}(\mathtt{Q}(x))$ *then* $\mathcal{L}' := \mathcal{L} \cup \{\mathtt{P}(x) \wedge \mathtt{Q}(x)\}$ *yields a conservative model.*

This rule can be extended to handle observational actions which provide `LCW` on the universe of discourse (*i.e.* the extension of $\mathtt{Q}(x)$). In addition it can be combined with the Function Rule to give `LCW` in situations where neither rule alone suffices. For example, since each file has exactly one size, after executing the UNIX action `ls -la bin`, we are able to deduce that we know the size of each file in `bin`:

$$\mathtt{LCW}(\mathtt{parent.dir}(o, \mathtt{bin}) \wedge \mathtt{size}(o, l))$$

3.2 Information Loss

An agent loses information when a literal, previously known to be true (or false), is asserted to be unknown. When a UNIX file is compressed, for example, information about its size is lost. In general, when information is lost about some literal, all `LCW` statements "relevant" to that literal are lost. To make our notion of relevance precise, we begin by defining the set $\mathtt{PREL}(\varphi)$ to denote the `LCW` assertions *potentially relevant* to a positive literal φ:[9]

[8]It is not required that $\mathtt{P}(x)$ be initially unknown for all x, merely that its value afterward, whether `T` or `F`, be known.

[9]Since the sentences in $\mathcal{L}$ are conjunctions of positive literals, we use the notation $\varphi \in \Phi$ to signify that φ is one of Φ's conjuncts, and the notation $\Phi - \varphi$ to denote the conjunction Φ with φ omitted.

$$\texttt{PREL}(\varphi) \equiv \{\Phi \in \mathcal{L} \mid \exists x \in \Phi, \exists \theta, x\theta = \varphi\}$$

For example, if an agent has complete information on the size of all files in `/kr94`, and a file `lcw.tex` in `/kr94` is compressed ($\varphi = \texttt{size}(\texttt{lcw.tex}, n)$), then the sentence

$$\texttt{LCW}(\texttt{parent.dir}(f, \texttt{/kr94}) \wedge \texttt{size}(f, c)) \tag{4}$$

is in $\texttt{PREL}(\varphi)$ and should be removed from $\mathcal{L}$. Unfortunately, when a file in the directory `/bin` is compressed, the above `LCW` sentence is still in $\texttt{PREL}(\varphi)$ ($x = \texttt{size}(f, c)$) even though the agent retains complete information about the files in `/kr94`. Clearly, `LCW` sentence 4 ought to remain in $\mathcal{L}$ in this case. To achieve this behavior, we check whether the agent has information indicating that the `LCW` sentence does not "match" the compressed file. If so, the `LCW` sentence remains in $\mathcal{L}$. In general, we define the set of `LCW` assertions *relevant* to a positive literal φ to be the following subset of $\texttt{PREL}(\varphi)$:

$$\texttt{REL}(\varphi) \equiv \{\Phi \in \texttt{PREL}(\varphi) \mid \forall \phi_i \in (\Phi - x), \neg(\mathcal{L} \wedge \mathcal{M} \models \neg\phi_i\theta)\}$$

where, as in the definition of $\texttt{PREL}(\varphi)$, $\exists x \in \Phi, \exists \theta,$ such that $x\theta = \varphi$.

We can now state our update policy for Information Loss:

Theorem 9 (Information Loss Rule) *If the agent has a conservative model and an atomic change has no effects other than* $\Delta(\varphi, \texttt{T} \vee \texttt{F} \rightarrow \texttt{U})$ *then* $\mathcal{L}' := \mathcal{L} - \texttt{REL}(\varphi)$ *yields a conservative model.*

Note that compressing a file `foo` in `/bin` does not remove `LCW` sentence 4. To see this, let $x = \texttt{size}(f, c)$, $\theta = (\texttt{foo}/f)$, and $\phi_i = \texttt{parent.dir}(f, \texttt{/kr94})$. Since `foo` is known to be in `/bin`, $\mathcal{L} \wedge \mathcal{M}$ entails that $\neg\phi_i\theta$. Hence, $\neg(\mathcal{L} \wedge \mathcal{M} \models \neg\phi_i\theta)$ is false and Φ is not included in $\texttt{REL}(\varphi)$. Note also that, given our assumptions (correct information, *etc.*), information is only lost when the world's state changes.

3.3 Changes in Domain

Finally, we have the most subtle cases: an agent's model changes without strictly losing or gaining information. For example, when the file `ai.sty` is moved from the `/tex` directory to `/kr94`, we have that the updated $\mathcal{M}' \neq \mathcal{M}$ but neither database is a superset of the other. When the model changes in this way, the domain of sentences containing $\texttt{parent.dir}(f, \texttt{/kr94})$ grows whereas the domain of sentences containing $\texttt{parent.dir}(f, \texttt{/tex})$ contracts. LCW information may be lost in sentences whose domain grew. Suppose that, prior to the file move, the agent knows the word counts of all the files in /kr94; if it does not know the word count of ai.sty, then that LCW assertion is no longer true. As with Information Loss, we could update $\mathcal{L}$ by removing the set $\texttt{REL}(\varphi)$. However, this policy is overly conservative. Suppose, in the above file move, that the agent *does* know the word count of ai.sty. In this case, it retains complete information over the word counts of the files in /kr94, even after ai.sty is moved.

More generally, when the domain of an LCW sentence grows, but the agent has LCW on the new element of the domain, the LCW sentence can be retained. To make this intuition precise, we define the following "minimal" subset of $\texttt{REL}(\varphi)$:

$$\texttt{MREL}(\varphi) = \{\Phi \in \texttt{REL}(\varphi) \mid \neg(\mathcal{L} \models (\Phi - x)\theta)\}$$

where, as in the definition of $\texttt{PREL}(\varphi)$, $\exists x \in \Phi, \exists\theta$, such that $x\theta = \varphi$. We can now state our update policy for Domain Growth:

Theorem 10 (Domain Growth Rule) *If the agent has a conservative model and an atomic change has no effects other than* $\Delta(\varphi, \texttt{F} \rightarrow \texttt{T})$ *then* $\mathcal{L}' \texttt{:=} \mathcal{L} - \texttt{MREL}(\varphi)$ *yields a conservative model.*

When the domain of a sentence contracts, no LCW information is lost. For instance, when a file is removed from the directory /kr94, we will still know the size of each file in that directory.

Theorem 11 (Domain Contraction Rule) *If the agent has a conservative model and an atomic change has no effects other than* $\Delta(\varphi, \texttt{T} \rightarrow \texttt{F})$ *then* $\mathcal{L}' \texttt{:=} \mathcal{L}$ *yields a conservative model.*

Our update rules cover all possible truth-value transitions. The rules guarantee that $\mathcal{L}$ does not contain invalid LCW assertions, so long as the agent is appraised of any changes to the world state. However, for the sake of tractability, the rules are conservative — $\mathcal{L}'$ may be incomplete. For example, when the word count of ai.sty is unknown in the above example, we might wish to say that we know the word counts of all the files in /kr94 *except* ai.sty. However, we refrain from storing negated sentences in $\mathcal{L}$because such sentences would slow down LCW inference, as shown in Section 2.5.

3.4 Example

Below, we provide an extended example of our LCW update machinery in action. We will see how the LCW database ($\mathcal{L}$) is updated, relying on the rules shown in Section 3, as the following command sequence is executed:

```
ls -al /kr94
ls -al /papers
mv /kr94/kr.ps /papers
compress /papers/kr.ps
```

Initially, both the model ($\mathcal{M}$) and the LCW database are empty. The execution of `ls -al` in the directory `/kr94` reveals the files in the directory and their size in bytes. Suppose that the files are `kr.tex` and `kr.ps`, and their sizes are 100 and 300 respectively. In this case, $\mathcal{M}$ contains:

parent.dir(kr.tex, /kr94)
size(kr.tex, 100)

parent.dir(kr.ps, /kr94)
size(kr.ps, 300)

The agent is familiar with all the files in `/kr94` and with each file's size. This information is recorded in the LCW database as follows:

$\mathcal{L}$ = {parent.dir(f, /kr94),
parent.dir(f, /kr94) $\wedge$ size(f, l)}

In addition, because the parent directory and size of each file are unique, the Function Rule implies that we have LCW on the size and parent directory of each file. For brevity, we omit LCW assertions derived from the Function Rule from the snapshots of $\mathcal{L}$ shown here.

The directory `/papers` is initially empty. Thus, after executing `ls -al` in the directory `/papers`, the agent records LCW information for the directory `/papers`, but no updates are made to $\mathcal{M}$.

$\mathcal{L}$ = {parent.dir(f, /kr94),
parent.dir(f, /kr94) $\wedge$ size(f, l),

`parent.dir`(f, `/papers`),
`parent.dir`(f, `/papers`) $\wedge$ `size`(f, l)}

Moving the file `kr.ps` from the directory `/kr94` to the directory `/papers`, results in both Domain Contraction to the directory `/kr94`, and Domain Growth to the directory `/papers`. There is no change to $\mathcal{L}$ due to Domain Contraction. However, Domain Growth could potentially result in statements being retracted from $\mathcal{L}$. This example illustrates the advantage of having the Domain Growth Rule retract the set of `MREL` sentences from $\mathcal{L}$, rather than naively retracting the set of `REL` sentences. There are two statements in `REL`:

`REL`(`parent.dir`(`kr.ps`, `/papers`)) = {`parent.dir`(f, `/papers`),
`parent.dir`(f, `/papers`) $\wedge$ `size`(f, l)}

In contrast, the `MREL` of the update is empty, due to the fact that we have `LCW` on the size of `kr.ps`:
`MREL`(`parent.dir`(`kr.ps`, `/papers`)) = {}

As a result, $\mathcal{L}$ remains unchanged after the `mv` command is executed. However, note that if we did not know the size of `kr.ps` when it was moved, we would have lost `LCW` on the size of the files in the directory `/papers`.

The last action in our example is compressing the file `kr.ps`. This action illustrates the advantage of retracting `REL` rather `PREL` in the Information Loss Rule. After the file `kr.ps` is compressed, its size becomes unknown. The set of `PREL` statements is:

`PREL`(`size`(`kr.ps`, l)) = {`parent.dir`(f, `/kr94`) $\wedge$ `size`(f, l),
`parent.dir`(f, `/papers`) $\wedge$ `size`(f, l)}

In contrast, because we know that `kr.ps` is now in the directory `/papers`, the set of `REL` statements contains only the following:

`REL`(`size`(`kr.ps`, l)) = `parent.dir`(f, `/papers`) $\wedge$ `size`(f, l)

Thus after the `compress` action is executed, we remove the `REL` statement from $\mathcal{L}$ obtaining:

$$\begin{aligned}\mathcal{L} = \{&\texttt{parent.dir}(f, \texttt{/kr94}),\\ &\texttt{parent.dir}(f, \texttt{/kr94}) \wedge \texttt{size}(f, l),\\ &\texttt{parent.dir}(f, \texttt{/papers})\}\end{aligned}$$

3.5 Computational Complexity of Updates

As stated above, our motivation for formulating conservative update rules has been to keep `LCW` update tractable. We make good on this promise below. We start by considering the complexity of applying single update rules:

- **Information Gain:** Theorem 5 implies that no sentences have to be retracted from $\mathcal{L}$. `LCW` sentences may be added by the Function Rule in time that is independent of the size of $\mathcal{L}$.[10] The Forall Rules require the agent to compute whether it has `LCW` over the universe of the update. The universe is denoted by a bounded-length conjunction (typically, a singleton such as the files in the directory `/tex`). Thus, Forall Rules take time polynomial in the size of $\mathcal{L}$.

- **Information Loss:** First, the agent computes the set `PREL`(Φ), which takes time linear in the size of $\mathcal{L}$ in the worst case. Next, the agent computes `REL`(Φ) from `PREL`(Φ). The time to identify each element of `REL` is linear in the size of $\mathcal{L}$, since establishing whether $\mathcal{L} \wedge \mathcal{M} \models \neg\phi_i\theta$ may require singleton `LCW` queries. In the worst case, the size of `PREL`(Φ) is linear in the size of $\mathcal{L}$, so computing `REL`(Φ) from `PREL`(Φ) could take time quadratic in the size of $\mathcal{L}$, which dominates the time for the entire update.

- **Domain Growth:** The agent has to compute the set `REL`(Φ) which, as explained above, is quadratic in the size of $\mathcal{L}$. Computing `MREL`(Φ) from `REL`(Φ) is linear in the size of `REL`, but potentially polynomial in the size of $\mathcal{L}$, since additional bounded-length `LCW` queries may be involved. The agent then removes each element of the set from $\mathcal{L}$, which takes time linear in the size of the set `MREL`(Φ). Thus the whole operation is polynomial in the size of $\mathcal{L}$.

[10]The Conjunction, Composition, and Instantiation Rules are applied in response to `LCW` queries, but ignored when $\mathcal{L}$ is updated.

- **Domain Contraction:** $\mathcal{L}$ remains unchanged in this case.

While the application of each individual update rule is reasonably fast, even in the worst case, we have to consider the possibility of a cascade of $\mathcal{L}$ updates. Will the update rules chain on each other? Are such chains guaranteed to terminate? Fortunately, we can show that rule chaining is unnecessary. The intuition is as follows. Chaining could potentially occur in one of two ways. First, when $\mathcal{L}$ shrinks, due to Domain Growth or Information Loss, a potentially infinite number of sentences change from `F` to `U`. Thus one might think that the Information Loss Rule (Theorem 9) has to be applied to further retract sentences from $\mathcal{L}$. However, careful examination of the definition of `REL` shows that this is not the case — all relevant `LCW` sentences have already been excised from $\mathcal{L}$. Second, when $\mathcal{L}$ grows due to Information Gain, a potentially infinite number of sentences changes from `U` to `F`. However, by Information Gain, no statements have to be excised from $\mathcal{L}$, and the Forall Rules do not yield new `LCW` sentences as a consequence.

Thus, in the absence of chaining, the time to perform `LCW` updates is dominated by the time to retrieve `MREL`(Φ) which is polynomial in the size of $\mathcal{L}$ in the worst case, but much faster when standard indexing techniques (*e.g.*, hashing on the predicates in Φ) are used.

3.6 Discussion

The update rules defined above comprise a sound, efficient method for updating $\mathcal{M}$ and $\mathcal{L}$. We believe our rules satisfy the update postulates specified in [24] and generalized in [8], but have not attempted a proof. Since sentences in $\mathcal{L}$ are restricted to positive conjunctions, the algorithm is incomplete. Nevertheless, it is easy to see that our algorithm is better than the trivial update algorithm ($\mathcal{L}'$`:={}`). In our softbot's domain, for example, our Function and Forall Rules enable us to derive `LCW` from a wide range of "sensory" actions, including `pwd, wc, grep, ls, finger`, and many more. Furthermore, our update rules retain `LCW` in many cases. For example, changes to the state of one "locale" (such as a directory, a database, an archive, etc.) do not impact `LCW` on other locales. This feature of our update calculus applies to physical locales as well.

Below, we are able to make a much stronger claim, that the sets of sentences retracted by theorems 9 through 11 are, in fact, minimal. Every sentence retracted is invalid and *must* be removed from $\mathcal{L}$ to maintain soundness. This statement is trivially true for Domain Contraction where no sentences are retracted. Clearly, we cannot do better than that. The following theorem asserts that each `LCW` sentence retracted due to Information Loss is, in fact, invalid.

Theorem 12 (Minimal Information Loss) *Let φ be a positive literal and let A be an atomic change whose only effect is $\Delta(\varphi, \mathtt{T} \vee \mathtt{F} \rightarrow \mathtt{U})$. If $\Phi \in$ $\mathtt{REL}(\varphi)$ then `LCW`(Φ) does not hold after A has occurred.*

Remarkably, the same holds for Domain Growth.

Theorem 13 (Minimal Domain Growth) *Let φ be a positive literal and let A be an atomic change whose only effect is $\Delta(\varphi, \mathtt{F} \rightarrow \mathtt{T})$. If $\Phi \in \mathtt{MREL}(\varphi)$ then `LCW`(Φ) does not hold after A has occurred.*

Are the update rules for Information Loss and Domain Growth the best possible? At first blush, the answer to this question would seem to be yes, since the rules are sound and they retract the minimal set of sentences from $\mathcal{L}$. So what more could we want? However, this observation overlooks the key fact that inference in our framework is lazy so that when the sentence φ is retracted we effectively also retract $\varphi\sigma$ for any variable substitution σ. Above, we claimed that the sentence φ really ought to be retracted, but we didn't claim that the sentence $\varphi\sigma$ (which is weaker!) is invalid. In fact, there are cases where such sentences are valid. For example, consider the case where we have `LCW` on the size of all the files in the directory `/bin`, but the file `a.out` in that directory is compressed. Our update rule for Information Loss would retract the `LCW` statement, when, in fact, a weaker statement that we know the size of all the files in `/bin` — except `a.out` — is true. Since the sentences in $\mathcal{L}$ are conjunctions of positive literals, we have no way of expressing the above statement.

Ultimately, the test of any mechanism for closed-world reasoning – conservative or not – is its impact on the agent's performance. In the next section we describe preliminary experiments that suggest ours is effective in practice, speeding up execution by a factor of ten in some cases.

4 Experimental Results

In the previous sections we argued that our `LCW` mechanism is computationally tractable, but incomplete. However, asymptotic analysis is not always a good predictor of real performance, and incompleteness is a matter of degree. To evaluate our LCW machinery empirically, we incorporated LCW into the XII planner [22] and measured its impact on the performance of the Internet Softbot [13]. In this section, we address the following questions experimentally:

- What is the speed of LCW queries and LCW updates as a function of the size of the LCW database and the size of the LCW formulae?
- Because our LCW database is incomplete, a query may result in the truth value U whereas in fact, its truth value is F (See the definition of Query in Appendix A). How often does this occur as the database processes a sequence of queries and updates issued by the XII planner?
- The LCW machinery adds a time overhead to the XII planner, but also reduces redundant sensing operations. In theory, LCW could speed up the planner or slow it down. What happens in practice?

4.1 The Experimental Framework

Below, we describe the experimental set-up and address each question in turn. The goal of our experiment is to measure the performance of our LCW machinery in a real-world setting. All of our evaluations of LCW are through queries and updates generated by the XII planner in the course of satisfying randomly-generated goals in the Softbot domain. We randomly generate both goals and initial worlds. To make our experiments easier to control, vary, and replicate, we built a simulation environment that allows us to generate arbitrary UNIX *worlds*, which behave exactly as UNIX behaves in response to actions executed by the softbot. Additionally, the simulation greatly simplifies the task of evaluating LCW, as we will discuss in Section 4.5. Nearly all the of results we report using simulated UNIX worlds are identical to the results we would obtain if XII were executing in an equivalent, real UNIX environment.[11]

[11]The one exception is the report of total time in Figure 2, which does not reflect the time required to execute actions in a UNIX shell. However, the purpose of Figure 2 is to evaluate the impact of LCW on planning, not to measure the performance of the Internet Softbot. Based on earlier experiments in this domain (see [22]), it seems likely that accurately reporting execution time would only make our results stronger, since, without LCW, XII spends a greater percentage of its time executing actions, and execution is expensive.

4.2 The Simulation Environment

The simulation environment consists of a current world state w_s, represented as a database, which completely specifies the state of all files and directories in the simulation, and an execution procedure that translates an action to be executed into the appropriate queries and updates on w_s. In our experiments, w_s contains up to 80 directories, each directory holding between 5 and 20 files. The topology of the directory tree is random, each directory containing at most five other directories. Filenames are all of the form `dir1`, `file2`, etc. Other file attributes, such as `size` and `file.type`, are chosen randomly. Although w_s doesn't model *every* aspect of UNIX, it is complete relative to every action that could be executed in service of the test suite.

The execution procedure simply computes a mapping from an action to database operations on w_s. This mapping is trivial; all the required information is contained in the effects of the action. For example, `ls -la dir3` determines, among other things, the size of each file in `dir3`, so the execution procedure handles the execution of `ls -la dir3` by querying w_s for

$$\texttt{parent.dir}(f,\ \texttt{dir3}) \wedge \texttt{size}(f,\ n)$$

and updating $\mathcal{M}$ with the results. Similarly, since `cd dir11` has the effect `current.dir(dir11)`, this update is done to w_s as well as $\mathcal{M}$.

4.3 The Goal Distribution

The test suite consists of a series of *runs*. At the beginning of each run, a simulated world w_s is randomly generated, and $\mathcal{M}$ and $\mathcal{L}$ are reinitialized. A sequence of 30 goals is then randomly generated, and XII is given the goals to solve one by one. $\mathcal{M}$ and $\mathcal{L}$ are left intact between goals, so for each goal, XII has the benefit of knowledge obtained in solving the previous goals. After the 30 goals are completed, a new world is generated, $\mathcal{M}$ and $\mathcal{L}$ are emptied, and the process is repeated.

Our goal generator creates either universally-quantified or existentially-quantified goals. Quantification aside, the two sets of goals are essentially equivalent, and consist of finding files meeting certain properties, and performing certain operations on them. The properties include `filename`, `parent.dir`, `word.count`, `file.type`, and others. The updates include compressing the files, moving them to a different directory, and finding out their size. A typical goal is "Compress all postscript files in the directory `/dir0/dir1/dir21`." In XII's goal language, we have:

```
find-out (pathname(pd, "/dir0/dir1/dir21")) ∧
∀ file(f) in file.type(f, "postscript") ∧
             parent.dir(f, pd)
      satisfy (is.compressed (f))
```

4.4 LCW Speed

The interesting questions regarding LCW speed are "How fast are queries and updates on average?" and "How does the time vary as a function of the length of the LCW formula and the size of $\mathcal{L}$?" The answer to the first question is that across the experiments we've run, queries have consistently averaged about 2 milliseconds, while updates have been about 1.2 milliseconds. The average is over 390,000 queries generated by XII in response to the randomly generated problems described earlier.

In answer to the second question, Figure 1 shows query time as a function of the length of the query and the size of the $\mathcal{L}$ database. The graph shows the results for query sizes up to four conjunctions; larger queries don't occur in our experiments. In fact, large queries are uncommon in our domain; even queries with four conjuncts occur only as a result of user-supplied $\forall$ goals. The slow growth of query time as a function of $|\mathcal{L}|$ is due to the use of hashing, as opposed to the more expensive linear-time search assumed in our complexity analysis (Section 3.5). As mentioned earlier, updates are even faster than queries on average. This is not surprising, as most updates take time quadratic in $|\mathcal{L}|$ whereas queries are polynomial in $|\mathcal{L}|$ in the worst case.

4.5 Completeness

Because our LCW machinery is incomplete, QueryLCW(Φ) may return "No" when the agent does in fact have LCW(Φ). We refer to this event as an LCW *miss*. Below, we explain how we measured the percentage of LCW queries that result in LCW misses.

The problem of detecting LCW misses raises a thorny issue. LCW is a semantic notion defined in terms of $\mathcal{S}$, the infinite set of possible world states that are consistent with $\mathcal{M}$. How can we measure, experimentally, when the agent ought to have LCW, but does not? Comprehending the answer to this question requires a deep understanding of the formal basis for LCW. The definition of LCW in Section 2.2, combined with the fact that if $\varphi \in \mathcal{M}$

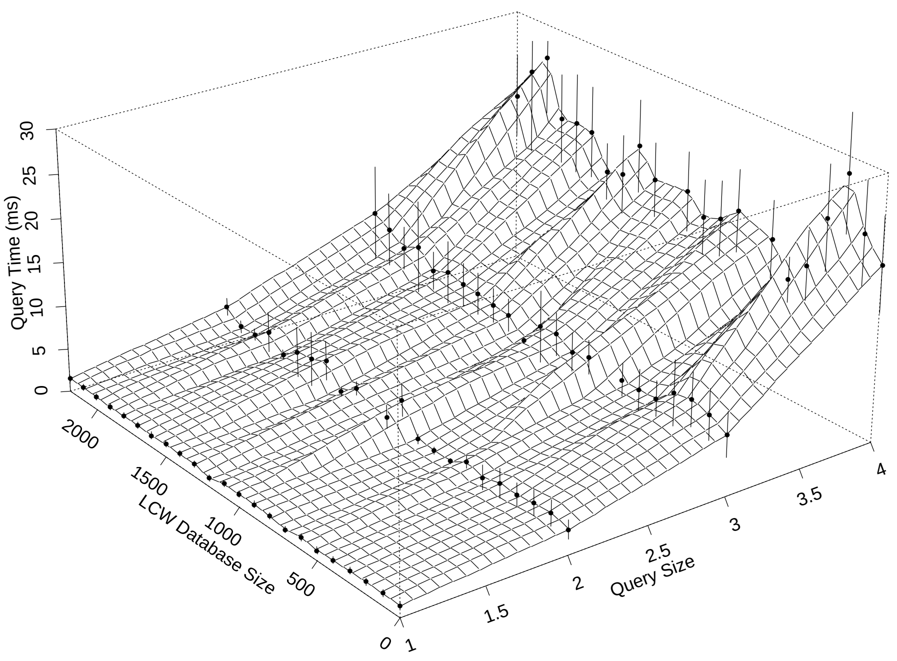

Figure 1: CPU Time for LCW queries as a function of the size of the LCW database $\mathcal{L}$, and the number of conjuncts in the query. Experiments were run on a Sun SPARCstation 20; vertical bars indicate 95% confidence intervals. Note that even as $\mathcal{L}$ grows large, the average query time is approximately 2 milliseconds. Over 90% of the 390,000 queries measured fewer than three conjuncts.

then $\mathcal{S} \models \varphi$, implies that if LCW($\Phi$) then there is a one-to-one correspondence between instances of Φ in w and in $\mathcal{M}$. This one-to-one correspondence is important because it can be tested experimentally.

Thus, to check whether QueryLCW(Φ) has resulted in an LCW miss, we do the following: When QueryLCW returns "No," we check whether every instance of Φ in the w_s database in fact appears in $\mathcal{M}$. If so, LCW is possible, and we report that an LCW miss has occurred. Of course, this mechanism can over-report LCW misses. Although LCW(Φ) is *possible*, and QueryLCW(Φ) failed, it may be that no sensing of Φ has taken place and we cannot expect the agent's LCW database to imply LCW(Φ).

For example, if directory dir1 is empty, then both $\mathcal{M}$ and w_s will agree on the extension of parent.dir(dir1, f), even if ls dir1 was never executed. But not knowing whether there are any files in a directory that happens to be empty is not the same as knowing that there aren't any, so this case would be a *false miss*. We are able to eliminate some of these false misses, but not all of them. However, since we are trying to demonstrate the success of our LCW machinery, we are content to be conservative and overstate the number of LCW misses. In practice, this conservative approximation had negligible impact on the results of our experiments.

In our experiments, fewer than 1% of the queries generated by XII result in misses. The percentage of misses does not vary significantly with the amount of dynamism (measured as updates to w_s/queries), or with the percentage of Domain Growth or Information Loss updates that occur. We varied the problem distribution so as to test the extreme points of each, meaning the number of queries and updates of each type were each allowed to range from 0% to 100% of all operations performed. However, dynamism is not a measure of how much the world changes on its own (it doesn't), but a measure of how much the softbot changes the world. Adding an adversary that was allowed to make arbitrary changes to the world would no doubt increase the miss rate substantially.

Answering the question of how often misses occur independent of the XII planner and the Softbot domain is problematic, since we could construct cases in which all LCW queries are misses, or none are. For example, suppose we have a directory containing only postscript and TEX files, and we have LCW on the size of all files in that directory. Suppose we then compress one of the postscript files. By the Information Loss Rule, the LCW we had on the size of all the files will be removed from $\mathcal{L}$, whereas if our LCW machinery were *complete*, it would *retain* LCW on the size of all *TEX* files in the directory. Now if all queries are of the form "Do I know all TEX files in this directory?" then every query will be a miss. Perverse cases like this one don't come up in practice in our domain.[12] However, cases that are perverse in one domain may be common in another.

4.6 Impact on Planning

We have shown that individual LCW queries are fast, but given that a significant number of LCW queries are performed during planning, it is still conceivable that LCW might slow the planner down. We show that this is not the case, and that in fact it speeds planning considerably by reducing redundant sensing operations. Figure 2 shows the performance of the XII planner with and without LCW, solving a sequence of randomly generated goals, with $\mathcal{M}$ and $\mathcal{L}$ initially empty. The planner runs faster with LCW even

[12]This is due, in part, to the fact that failed LCW queries are likely to be followed by actions that achieve the desired LCW.

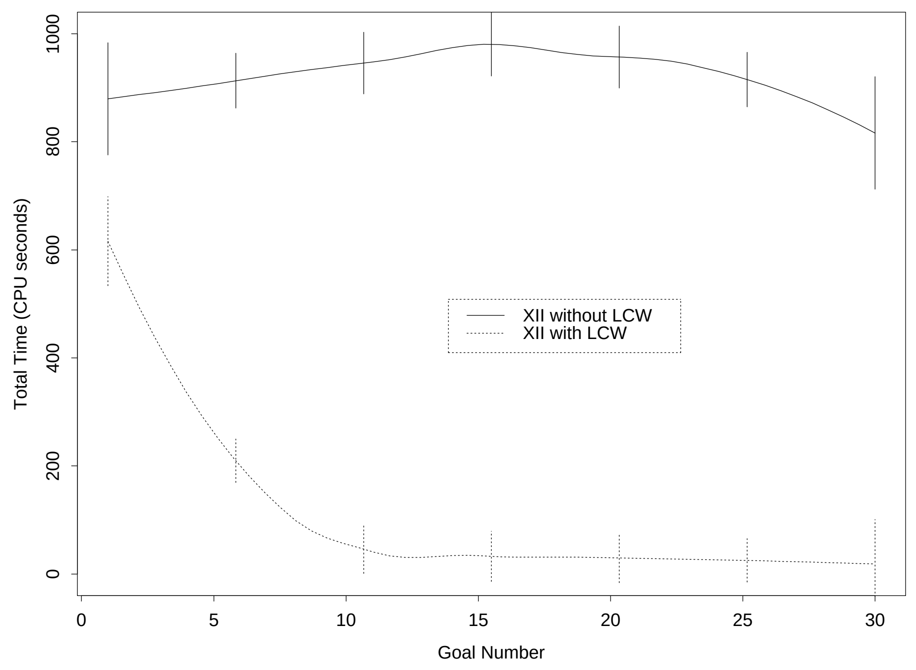

Figure 2: The massive improvement in the speed of the XII planner due to LCW reasoning. Times indicated are CPU seconds on a Sun SPARCstation 20; vertical bars indicate 95% confidence intervals. Each point on the X-axis represents an average of 10 trials on randomly generated initial worlds. Progression to the right along the X-axis shows the averaged planning time on 30 randomly generated goals, given in sequence.

on the first goal, since it leverages the LCW information which it gains in the course of planning. In subsequent goals, XII can take advantage of LCW gained in previous planning sessions for an even more pronounced speedup. Without LCW, the planner wastes an enormous amount of time doing redundant sensing. The version of XII without LCW completed only 8% of the goals before hitting a fixed time bound of 1000 CPU seconds. In contrast, the version with LCW completed 94% of the goals in the allotted time. The hypothesis tests for analyzing censored data, described in [15], demonstrate that our results are statistically significant.

5 Future Work

Although we have relaxed the assumption of complete information, we still assume correct information. Since we want our agents to cope with exogenous events, we are in the process of relaxing this assumption as well. We are investigating two complementary mechanisms to solve this problem. The first mechanism associates *expiration times* with beliefs. If an agent has a belief regarding φ, which describes a highly dynamic situation (e.g. the idle time of a user on a given machine), then the agent should not keep that belief in $\mathcal{M}$ for very long. Thus, after an appropriate amount of time has elapsed, $\Delta(\varphi, \mathtt{T} \vee \mathtt{F} \to \mathtt{U})$ occurs automatically. Note that by the Information Loss Rule, this update will cause `LCW` to be retracted as well. This mechanism is effective when the belief about φ expires before φ changes in the world. However, unless we have extremely short expiration times, we cannot guarantee this to be the case in general.

Thus, an additional mechanism is required that enables the agent to detect and recover from out-of-date beliefs. This is a harder problem, because it involves belief revision, rather than mere update. If executing an action fails, and the action's preconditions are known, it follows that one or more of the preconditions of the action were not satisfied — but which ones? A conservative approach would retract the ground literals satisfying the action's preconditions from the agent's model. However, this approach could discard a great deal of valuable information. We are investigating more efficient mechanisms.

Finally, we need to investigate increasing the expressive power of $\mathcal{M}$ and $\mathcal{L}$. The introduction of negation into $\mathcal{L}$, would enable us to express sentences such as "I know the size of each file in `/kr94` except `paper.tex`," which would make `LCW` update less conservative. For another example, suppose that an agent was unfamiliar with the contents of the `/kr94` directory, yet executed `chmod g+r *` while in that directory. The reasoning mechanism described in this paper is incapable of inferring that all the files in `/kr94` are group-readable.[13] The `LCW` sentence

$$\mathtt{LCW}(\mathtt{parent.dir}(f, \mathtt{/kr94}) \wedge \mathtt{group.protection}(f, \mathit{readable}))$$

is not warranted because it implies that the agent is familiar with all the group-readable files in `/kr94`, which is false by assumption.

[13] Indeed, this inference is only licensed when the agent is authorized to change the protection on each of the files in `/kr94`; suppose this is the case.

We *could* represent the information gained from the execution of `chmod g+r *` in `/kr94` by introducing the following Horn clause into $\mathcal{M}$:

$$\texttt{parent.dir}(f, \texttt{/kr94}) \rightarrow \texttt{group.protection}(f, \mathit{readable})$$

The Horn clause represents the fact that all the files in the directory `/kr94` are group-readable, even though the agent may be unfamiliar with the files in `/kr94`. Although the mechanisms described in this paper don't allow Horn clauses in the $\mathcal{L}$, this example demonstrates that such an extension would provide increased expressiveness. Future work should determine whether this expressiveness comes at great computational cost in terms of `LCW` efficiency.

6 Conclusions

This paper described a sound and efficient method for representing, inferring, and updating *local closed world information* (`LCW`) (*e.g.*, "I know the size of each file in `/kr94`") over a restricted first-order theory of the sort used by planning algorithms such as NONLIN, TWEAK, and UCPOP. To evaluate our `LCW` machinery empirically, we incorporated `LCW` into the XII planner [22] and measured its impact on the performance of the Internet Softbot [13] under a wide range of experimental settings.

Building on the rich body of earlier work on circumscription, autoepistemic logic, and database theory, we have provided sound theoretical foundations for closed-world reasoning in planners. Yet we have not had to sacrifice computational efficiency. As our experiments in the Softbot domain show, `LCW` queries require approximately 2 milliseconds, while `LCW` updates require only 1.2 milliseconds on average. Our method has been tested inside the XII planner, but it may be incorporated readily into other planning systems. We hope our results will persuade planning researchers to do so.

A Algorithms

The theorems presented in Section 2 illustrate the kinds of inference that an agent can make concerning the closure of its partial model of the world. Below, we provide concise pseudo-code descriptions of the actual algorithms based on these theorems. We start with Query, a fast algorithm for determining the agent's belief in a ground conjunction.

```
function Query($\Phi$, $\mathcal{M}$, $\mathcal{L}$): 3-Boolean
    let Result := T
    let LCW := QueryLCW($\Phi$, $\mathcal{M}$, $\mathcal{L}$)
    for each atomic conjunct $\varphi \in \Phi$ do begin
        if $\neg\varphi \in \mathcal{M}$ then return F
        else if $\Phi \notin \mathcal{M}$ then
            if LCW then return F
            else let Result := U
    end(* for *)
    return Result
```

The next algorithm, QueryLCW, determines whether a conjunctive `LCW` statement follows from the agent's beliefs as encoded in terms of the $\mathcal{M}$ and $\mathcal{L}$ databases.

```
function QueryLCW($\Phi$, $\mathcal{M}$, $\mathcal{L}$): Boolean
    QLCW*($\Phi$, {}, $\mathcal{M}$, $\mathcal{L}$)
```

```
function QLCW*($\Phi$, Matches, $\mathcal{M}$, $\mathcal{L}$): Boolean
    if $\Phi = \{\}$ then return T
    else if $\Phi$ is ground and $\mathcal{M} \models \Phi$ or $\mathcal{M} \models \neg\Phi$ then return T
    else for $C \in \mathcal{L}$ do
        for $\Phi' \subseteq (\Phi \cup Matches)$ such that $\exists\theta, \Phi' = C\theta$ do
                begin
            let $NewMatches$ :=$Matches \cup \Phi'$
            if $|NewMatches| > |Matches|$ then
                if QLCW*($\Phi - \Phi'$, $NewMatches$,
                            $\mathcal{M}$, $\mathcal{L}$) then return T
        end (* for *)
    if $Matches \neq \{\}$ and
            $\forall\theta \in$ ConjMatch($Matches$, $\mathcal{M}$) QueryLCW($\Phi\theta$, $\mathcal{M}$, $\mathcal{L}$) then
        return T
    else return F
```

Note that unlike Query, the QueryLCW algorithm allows variables in its

Φ input. QueryLCW calls the QLCW* helper function which calls ConjMatch in turn. ConjMatch(C, $\mathcal{M}$) performs a standard conjunctive match, returning a set of bindings, θ, such that $\mathcal{M} \models C\theta$.

We illustrate the operation of QueryLCW with the following simple example. Suppose we have:

$\mathcal{L}$ = {`parent.dir` (*f1*, *d1*),
    `length` (*f2*, *l2*)}

$\mathcal{M}$ = {`parent.dir (foo, root)`,
    `parent.dir (bar, root)`,
    `length (small, 5)`,
    `length (big, 90000)`,
    `protection(small, readable)`}

There are only two files, `small` and `big`, and one directory called `root`. The agent knows the lengths of all files, and also knows that file `small` is readable. We pose the query to the agent: "Do you know the parent directory and read permission of all files of length 5?"

We call QueryLCW on `parent.dir` (*f*, *d*) $\wedge$ `length` (*f*, 5) $\wedge$ `protection` (*f*, `readable`). QLCW* is called with Φ = `parent.dir` (*f*, *d*) $\wedge$ `length` (*f*, 5) $\wedge$ `protection` (*f*, `readable`) and Matches = {}. In the first iteration of the loop at line 3, C = `parent.dir` (*f1*, *d1*). At line 4, Φ' = `parent.dir` (*f*, *d*), and at line 5, $NewMatches$ = {`parent.dir` (*f*, *d*)}. $|NewMatches| > |Matches|$, so QLCW* is called recursively, with $\mathcal{M}$ and $\mathcal{L}$ unchanged, but Φ = `length` (*f*, 5) $\wedge$ `protection` (*f*, `readable`) and $Matches$ = {`parent.dir` (*f*, *d*)}. The first iteration through the loop at line 3 doesn't result in any additional matches. The second iteration through the loop, C = `length` (*f2*, *l2*), Φ' = `length` (*f*, 5). NewMatches = {`parent.dir` (*f*, *d*), `length` (*f*, 5)}. $|NewMatches| > |Matches|$, so QLCW* is once again called recursively, this time, with Φ = `protection` (*f*, `readable`) and $Matches$ = {`parent.dir` (*f*, *d*), `length` (*f*, 5)}. There are no new matches in $\mathcal{L}$ to the query, so the loop at line 3 terminates without any new calls to QLCW*. The test at line 9 is true, since $Matches$ is non-empty and a conjunctive match between $Matches$ and $\mathcal{M}$ results in θ = {f/`small`, d/`root`}. The call to QueryLCW(`protection (small, readable)`, $\mathcal{M}$, $\mathcal{L}$) succeeds (line 2 of QLCW*) because the query is ground and entailed by $\mathcal{M}$. The return value of `T` is propagated through all the previous calls to QLCW*, and returned by the original call to QueryLCW, indicating that the parent directories of all readable files of length 5 are known.

Theorem 14 (Soundness) *Let $\mathcal{M}$ be a set of consistent ground literals and let $\mathcal{L}$ be a set of* `LCW` *formulae such that every $\Psi \in \mathcal{L}$ is a positive conjunction. If* `QueryLCW(`Φ`, `$\mathcal{M}$`, `$\mathcal{L}$`)` *returns* `T` *then* `LCW(`Φ`)`.

Since the pseudo code for the update algorithms is substantially more complex we omit it here, but it is available from the authors upon request.

B Proofs

In many of the following proofs we rely on the following two facts:

- $\mathcal{L}$ contains only positive sentences.
- The variable substitution θ maps a sentence Φ to a ground sentence $\Phi\theta$. Thus, once the truth value of $\Phi\theta$ is known, we have `LCW(`$\Phi\theta$`)`.

Proof of Theorem 1 (Instantiation Rule) Let Φ be a logical sentence and suppose `LCW(`Φ`)` holds. Let θ be an arbitrary substitution; we need show that `LCW(`$\Phi\theta$`)` holds. *I.e.*, by definition of `LCW` (Equation 1) we need show that for all substitutions, σ, either $\mathcal{S} \models \Phi\theta\sigma$ or $\mathcal{S} \models \neg\Phi\theta\sigma$. But since the composition $\theta\sigma$ of substitutions is a substitution, and since `LCW(`Φ`)` we conclude `LCW(`$\Phi\theta$`)`. $\square$

Proof of Theorem 2 (Conjunction Rule) Let Φ and Ψ be logical sentences and suppose `LCW(`Φ`)` and `LCW(`Ψ`)`. Let θ be an arbitrary substitution. We need show $[\mathcal{S} \models (\Phi \wedge \Psi)\theta] \vee [\mathcal{S} \models \neg(\Phi \wedge \Psi)\theta]$ Now, if $\mathcal{S} \models (\Phi \wedge \Psi)\theta$, then the proof is complete; so instead assume that $\mathcal{S} \not\models (\Phi \wedge \Psi)\theta$. This implies that either $\mathcal{S} \not\models \Phi\theta$ or $\mathcal{S} \not\models \Psi\theta$. Without loss of generality, assume that $\mathcal{S} \not\models \Phi\theta$. Since `LCW(`$\Phi$`)`, we conclude $\mathcal{S} \models \neg\Phi\theta$. But then clearly $\mathcal{S} \models \neg\Phi\theta \vee \neg\Psi\theta$ which means that `LCW(`$\Phi \wedge \Psi$`)`. $\square$

Proof of Theorem 4 (Composition Rule) Let Φ and Ψ be logical formulae and suppose $\texttt{LCW}(\Phi)$ and $\forall \sigma, \mathcal{S} \not\models \Phi\sigma \vee \texttt{LCW}(\Psi\sigma)$. Let θ be an arbitrary substitution. We need to show $[\mathcal{S} \models (\Phi \wedge \Psi)\theta] \vee [\mathcal{S} \models \neg(\Phi \wedge \Psi)\theta]$. If $\mathcal{S} \models (\Phi \wedge \Psi)\theta$, then the proof is complete; so instead assume that $\mathcal{S} \not\models (\Phi \wedge \Psi)\theta$. Since $\texttt{LCW}(\Phi)$, either $\mathcal{S} \models \Phi\theta$ or $\mathcal{S} \models \neg\Phi\theta$. If $\mathcal{S} \models \neg\Phi\theta$, then clearly $\mathcal{S} \models \neg\Phi\theta \vee \neg\Psi\theta$, and the proof is complete. If $\mathcal{S} \models \Phi\theta$ then $\mathcal{S} \not\models \Psi\theta$ (otherwise, $\mathcal{S} \models (\Phi \wedge \Psi)\theta$). Furthermore, $\mathcal{S} \models \Phi\theta$ implies $\texttt{LCW}(\Psi\theta)$ (given), so $\mathcal{S} \models \neg\Psi\theta$. Thus $\mathcal{S} \models \neg\Phi\theta \vee \neg\Psi\theta$, which means that $\texttt{LCW}(\Phi \wedge \Psi)$. $\Box$

Proof of Theorem 3 (Disjunction Rule) Follows trivially from the definition of LCW and Theorem 2. $\Box$

Proof of Theorem 5 (Information Gain Rule) It suffices to prove that for any formula, Φ, and literal, φ, if $\texttt{LCW}(\Phi)$ holds before action A is executed and the sole effect of A is $\Delta(\varphi, \texttt{U} \rightarrow \texttt{T} \vee \texttt{F})$, then $\texttt{LCW}(\Phi)$ still holds. Suppose $\texttt{LCW}(\Phi)$ holds and let θ be an arbitrary substitution. By Equation 1, we know that $[\mathcal{S} \models \Phi\theta] \vee [\mathcal{S} \models \neg\Phi\theta]$. Since A has only observational effects, $\mathcal{S} \subseteq \mathcal{S}'$. As a result, for any formula Ψ if $\mathcal{S} \models \Psi$ then $\mathcal{S}' \models \Psi$. Thus, clearly $[\mathcal{S}' \models \Phi\theta] \vee [\mathcal{S}' \models \neg\Phi\theta]$. $\Box$

Proof of Theorem 6 (Function Rule) Let A be an action that has no effects other than $\Delta(P(c), \texttt{U} \rightarrow \texttt{T})$ for some constant c and some predicate P where

$$\forall z, y P(z) \wedge P(y) \models (z = y) \tag{5}$$

We need show that $\texttt{LCW}(P(x))$; in other words, we need show that for an arbitrary substitution θ, $[\mathcal{S} \models P(x)\theta] \vee [\mathcal{S} \models \neg P(x)\theta]$. If θ maps x to c, then $\mathcal{S} \models P(x)\theta$ because $\mathcal{M} \models P(c)$ and $\mathcal{M}$ is correct. If θ does not map x to c then, by Equation 5 $\mathcal{S} \models \neg P(x)\theta$. Either θ maps x to c or not, so $\texttt{LCW}(P(x))$. $\Box$

Proving the next two theorems requires commitment to a formal action semantics. Here we sketch an extension to ADL, Pednault's [39, 38, 37] state-transition model of conditional and universally quantified actions, to handle incomplete information. Formally, each action is modeled as a pair, $\langle \mathsf{C}, \mathsf{O} \rangle$, denoting its causational and observational aspects. Following ADL [38, p. 357], we define the causational effects, C, of an action as a set of pairs $\langle \mathsf{s}_i, \mathsf{s}_j \rangle$ — execution of the action in world state s_i yields state s_j.[14]

[14]If executed in a state which does not appear as the left member of a C pair, the result of execution is an error.

As mentioned earlier, we model an agent's incomplete knowledge of the actual world state, w, with the set of states $\mathcal{S}$. Since we assume correct information, $\mathsf{w} \in \mathcal{S}$. If an agent has incomplete information, then after executing the action it can only conclude that the world is a member of the image of C on $\mathcal{S}$: $\{\mathsf{s}_j \mid \langle \mathsf{s}_i, \mathsf{s}_j \rangle \in \mathsf{C} \wedge\ \mathsf{s}_i \in \mathcal{S}\}$.

When the C pairs denote a function, then the effect of execution is unique, but if C specifies a relation (*i.e.*, two pairs share a first element), then the effect of execution is uncertain. Even if the exact initial state is known, precise prediction of the unique final state resulting from execution is impossible with this action model. Flipping a coin and executing `compress`) are good examples of actions that require relational C. In contrast to actions which increase uncertainty, if an action's C pairs denote a nonsurjective function, then execution can decrease uncertainty. For example, executing `rm *` reduces the size of $\mathcal{S}$ if the contents of the current directory were not known at the time of execution.

Pednault's theory also needs to be extended to handle information gathering actions, *i.e.* effects which change $\mathcal{S}$ without changing w. For example, suppose that Φ=`turned.on(light53)` denotes that the light is on, but $\mathcal{S}$ neither entails Φ nor $\neg\Phi$. The act of scanning the room with a TV camera and finding Φ is false doesn't change the world state, but it does allow discarding from $\mathcal{S}$ any state s which entails Φ. Syntactically, we describe this action with a UWL `observe` postcondition [12], but semantically we model the observational effects, O, of an action as a partition over all possible world states. If two states are in different equivalence classes, then the action's observational effects distinguish between them. In other words, O specifies the discriminatory power provided by the execution system — at run time, the execution system reports which equivalence class resulted. Actions with no observational effects can be modeled with O specifying a single equivalence class. The action of detecting whether the light is on (described above) yields a partition with two classes: states entailing Φ and those entailing $\neg\Phi$. More generally, if $A = \langle \mathsf{C}, \mathsf{O} \rangle$ is an action and $\mathcal{S}$ denotes the agent's knowledge of the world state, and $\mathsf{w}\ \in \mathcal{S}$ is the actual world state, then after executing A the actual world state will be w' where $\langle \mathsf{w}, \mathsf{w}' \rangle \in \mathsf{C}$ and the agent's state set, $\mathcal{S}'$ will be:

$$\{\mathsf{s}_j \mid \langle \mathsf{s}_i, \mathsf{s}_j \rangle \in \mathsf{C} \wedge \mathsf{s}_i \in \mathcal{S} \wedge \exists O \in \mathsf{O},\ \mathsf{w}', \mathsf{s}_j \in O\} \qquad (6)$$

For example, if $\mathcal{S} = \{\mathsf{s}_1, \mathsf{s}_2\}$, $\mathsf{C} = \{\langle \mathsf{s}_1, \mathsf{s}_3 \rangle, \langle \mathsf{s}_2, \mathsf{s}_4 \rangle\}$ and $\mathsf{O} = \{\{\mathsf{s}_1, \mathsf{s}_4\}, \{\mathsf{s}_2, \mathsf{s}_3\}\}$ then by executing the action, the agent should be able to deduce complete information: either $\mathcal{S}'$ will equal $\{\mathsf{s}_3\}$ or $\mathcal{S}'$ will equal $\{\mathsf{s}_4\}$.

We close by noting that the $\langle \mathsf{C}, \mathsf{O} \rangle$ pairs are only a semantic construct. Since there may be an infinite number of these pairs, pragmatics dictates that we describe the actions with a convenient (*e.g.* finite) syntax. A precise definition of the mapping between the syntactic constructs of UWL [12] and the $\langle \mathsf{C}, \mathsf{O} \rangle$ pairs is lengthy, but straightforward. For example, suppose $\mathcal{B}$ denotes the extension of $\texttt{block}(x)$ and a `spray-paint` action has a universally quantified causational effect, $\forall x \in \mathcal{B}$ $\texttt{green}(x)$ then `green` must be true of every block in every state s_j present in a $\langle \mathsf{s}_i, \mathsf{s}_j \rangle$ pair in C. Universally quantified observational effects have a similar interpretation. The UNIX `ls -a /kr94` command, for example, provides complete information about *all* files in the `/kr94` directory. This corresponds to a $\langle \mathsf{C}, \mathsf{O} \rangle$ pair in which each equivalence class in O contains states that agree on the extension of the `parent.dir` predicate so long as `/kr94` is given as the second argument:

$$\begin{array}{r} \forall O \in \mathsf{O} \quad \forall s_1, s_2 \in O \quad \forall f \\ \text{if } s_1 \models \texttt{parent.dir}(f, \texttt{/kr94}) \\ \text{then } s_2 \models \texttt{parent.dir}(f, \texttt{/kr94}) \end{array} \tag{7}$$

More generally, suppose action A has a single effect, namely a universally quantified UWL observational effect of the form "$\forall x$ when $\texttt{Q}(x)$ then observe whether $\texttt{P}(x)$ holds or whether $\neg\texttt{P}(x)$ holds." Action A corresponds to the pair $\langle \mathsf{C}, \mathsf{O} \rangle$ in which C denotes the identity relation (since the action has only observational effects, and each $O \in \mathsf{O}$ satisfies the following equation:

$$\begin{array}{r} \forall s_1, s_2 \in O \quad \forall c \\ \text{if } s_1 \models \texttt{Q}(c) \wedge \ s_2 \models \texttt{Q}(c) \\ \text{then } (s_1 \models \texttt{P}(c)) \Leftrightarrow (s_2 \models \texttt{P}(c)) \end{array} \tag{8}$$

Proof of Theorem 7 (Causal Forall Rule) Let $A = \langle \mathsf{C}, \mathsf{O} \rangle$ be an action whose execution leads solely to an update of the form $\Delta(\mathtt{P}(x), \mathtt{F} \vee \mathtt{T} \vee \mathtt{U} \rightarrow \mathtt{T})$ $\forall x$ such that $\mathtt{Q}(x)$ and suppose that $\mathtt{LCW}(\mathtt{Q}(x))$. Suppose that A can be legally executed in every state in $\mathcal{S}$, and let $\mathtt{w}'$ and $\mathcal{S}'$ denote the result of executing A. Since the only effect of A is on $\mathtt{P}$, forall θ if $s_i \models \mathtt{Q}(x)\theta$ and $\langle s_i, s_j \rangle \in \mathsf{C}$ then $s_j \models \mathtt{Q}(x)\theta$. Combining this with the assumption $\mathtt{LCW}(\mathtt{Q}(x))$ yields

$$[\mathcal{S}' \models \mathtt{Q}(x)\theta] \vee [\mathcal{S}' \models \neg\mathtt{Q}(x)\theta] \tag{9}$$

Note that only a universally quantified *causational* effect of A can account for the quantified update $\Delta(\mathtt{P}(x), \mathtt{F} \vee \mathtt{T} \vee \mathtt{U} \rightarrow \mathtt{T})$. Thus, forall $\langle s_i, s_j \rangle \in \mathsf{C}$ if $s_i \models \mathtt{Q}(x)\theta$ then $s_j \models \mathtt{P}(x)\theta$ and hence $s_j \models (\mathtt{P}(x) \wedge \mathtt{Q}(x))\theta$. Combining this with Equation 9 yields

$$[\mathcal{S}' \models (\mathtt{P}(x) \wedge \mathtt{Q}(x))\theta] \vee [\mathcal{S}' \models \neg\mathtt{Q}(x)\theta]$$

Since $\neg\mathtt{Q}(x)$ entails $\neg(\mathtt{P}(x) \wedge \mathtt{Q}(x))$, we conclude that $\mathtt{LCW}(\mathtt{P}(x) \wedge \mathtt{Q}(x))$ holds and the update $\mathcal{L}' := \mathcal{L} \cup \{\mathtt{P}(x) \wedge \mathtt{Q}(x)\}$ is sound. $\square$

Proof of Theorem 8 (Observational Forall Rule) Let $A = \langle \mathsf{C}, \mathsf{O} \rangle$ be an action whose execution leads solely to an update of the form $\forall x$ such that $\mathtt{Q}(x)$ $\Delta(\mathtt{P}(x), \mathtt{U} \rightarrow \mathtt{T} \vee \mathtt{F})$. Furthermore, assume that $\mathtt{LCW}(\mathtt{Q}(x))$. Suppose that A can be legally executed in every state in $\mathcal{S}$, and let $\mathtt{w}'$ and $\mathcal{S}'$ denote the result of executing A. As before, we conclude that Equation 9 holds. In this case, however, note that since the only Δ for $\mathtt{P}$ was from truth value $\mathtt{U}$, only a universally quantified *observational* effect can account for the Δ. By Equation 6 all states in $\mathcal{S}'$ must be a subset of one equivalence class $O \in \mathsf{O}$. Combining this with Equation 8 and Equation 9, allows the conclusion that forall substitutions θ of x, either $\mathcal{S}' \models (\mathtt{P}(x) \wedge \mathtt{Q}(x))\theta$ or else $\mathcal{S}' \models \neg\mathtt{Q}(x)\theta$. But if $\mathcal{S}' \models \neg\mathtt{Q}(x)\theta$ for a specific θ then $\mathcal{S}' \models \neg(\mathtt{P}(x) \wedge \mathtt{Q}(x))\theta$ for that specific substitution. Hence we have $\mathtt{LCW}(\mathtt{P}(x) \wedge \mathtt{Q}(x))$ and thus the update is sound. $\square$

Proof of Theorem 9 (Information Loss Rule) Let Φ be a conjunction of positive literals and suppose that $\mathtt{LCW}(\Phi)$. Let φ be a positive literal and let A be an action whose execution leads solely to an update of the form $\Delta(\varphi, \mathtt{T} \vee \mathtt{F} \rightarrow \mathtt{U})$. To prove that the Information Loss Rule is sound in this case, we need to show that if $\mathtt{LCW}(\Phi)$ no longer holds after executing A then $\Phi \in \mathtt{REL}(\varphi)$ (the set of beliefs removed from $\mathcal{L}$), hence the update correctly recognizes that LCW has been lost, and $\mathcal{L}$ remains conservative. Suppose that $\mathtt{LCW}(\Phi)$ *doesn't* hold after executing A; then there exists a substitution, θ such that $[\mathcal{S}' \not\models \Phi\theta] \wedge [\mathcal{S}' \not\models \neg\Phi\theta]$ even though $[\mathcal{S} \models \Phi\theta] \vee [\mathcal{S} \models \neg\Phi\theta]$. Note that since Φ is conjunctive, $\Phi = \phi_1 \wedge \ldots \wedge \phi_n$. There are two cases:

1. $(\mathcal{S} \models \Phi\theta)$. So forall $\phi_i \in \Phi$ we know that $\mathcal{S} \models \phi_i\theta$. But since $\mathcal{S}' \not\models \Phi\theta$ there exists ϕ_j such that $\mathcal{S}' \not\models \phi_j\theta$. Hence execution of A caused $\Delta(\phi_j\theta, \mathtt{T} \rightarrow \mathtt{U})$. But since we assumed that the only updates produced by A were of a specific form, $\phi_j\theta = \varphi$. We conclude that $\Phi \in \mathtt{PREL}(\varphi)$.

2. $(\mathcal{S} \models \neg\Phi\theta)$. In this case we know that $\exists\phi_j \in \Phi$ such that $\mathcal{S} \models \neg\phi_j\theta$ yet $\mathcal{S}' \not\models \neg\phi_j\theta$. As above, the restriction on Δ allows us to conclude that $\phi_j\theta = \varphi$ and $\Phi \in \mathtt{PREL}(\varphi)$.

To show $\Phi \in \mathtt{REL}(\varphi)$, we now need argue that $\forall\phi_i \in (\Phi - \phi_j), \mathcal{M} \cup \mathcal{L} \not\models \neg\phi_i\theta$. Suppose that this is *not* the case and $\mathcal{M} \cup \mathcal{L} \models \neg\phi_k\theta$ for some $k \neq j$. Since $\mathcal{M}$ and $\mathcal{L}$ are conservative, $\mathcal{S} \models \neg\phi_k\theta$ as well. Furthermore, since the only change affected by action A had Δ restricted to φ_j, we know that $\mathcal{S}' \models \neg\phi_k\theta$. But since the falsity of a single conjunct entails the falsity of the whole conjunction (and $\Phi = \phi_1 \wedge \ldots \wedge \phi_n$), we conclude that $\mathcal{S}' \models \neg\Phi\theta$. But this contradicts our assumption that A destroyed $\mathtt{LCW}(\Phi)$. So it must be the case that $\forall\phi_i \in (\Phi - \phi_j), \mathcal{M} \cup \mathcal{L} \not\models \neg\phi_i\theta$. Thus $\Phi \in \mathtt{REL}(\varphi)$. $\square$

Proof of Theorem 10 (Domain Growth Rule) Let Φ be a conjunction of positive literals and suppose that $\texttt{LCW}(\Phi)$. Let φ be a positive literal and suppose A is an atomic action whose only effect is $\Delta(\varphi, \texttt{F} \rightarrow \texttt{T})$. Suppose that $\texttt{LCW}(\Phi)$ no longer holds after executing A; then there exists a substitution, θ such that $[\mathcal{S}' \not\models \Phi\theta] \wedge [\mathcal{S}' \not\models \neg\Phi\theta]$ even though $[\mathcal{S} \models \Phi\theta] \vee [\mathcal{S} \models \neg\Phi\theta]$. A case analysis on the these disjuncts (as in the proof of Theorem 9) yields that $\exists\phi_j \in \Phi$ such that $\phi_j\theta = \varphi$ and that $\Phi \in \texttt{PREL}(\varphi)$. The contradiction argument from that proof also extends to show that $\Phi \in \texttt{REL}(\varphi)$. Now note that after execution of A, we have $\texttt{LCW}(\phi_j\theta)$ (since we know that φ changed to $\texttt{T}$), but by assumption not $\texttt{LCW}(\Phi\theta)$. Therefore, by the contrapositive of Theorem 2 (Conjunction Rule), $\neg\texttt{LCW}((\Phi - \phi_j)\theta)$. This leads to $\Phi \in \texttt{MREL}(\varphi)$. $\square$

Proof of Theorem 11 (Domain Contraction Rule) Let φ be a positive literal and suppose A is an action whose only effect is $\Delta(\varphi, \texttt{T} \rightarrow \texttt{F})$. To show that the update rule is sound, it is sufficient to prove that for any conjunction of positive literals, $\Phi = \phi_1 \wedge \ldots \wedge \phi_n$, if $\texttt{LCW}(\Phi)$ holds before executing A then $\texttt{LCW}(\Phi)$ holds after executing A. If $\texttt{LCW}(\Phi)$ holds before execution then, for arbitrary θ, we know that $[\mathcal{S} \models \Phi\theta] \vee [\mathcal{S} \models \neg\Phi\theta]$. We need to show that after executing A $[\mathcal{S}' \models \Phi\theta] \vee [\mathcal{S}' \models \neg\Phi\theta]$. Suppose, on the other hand, that $[\mathcal{S}' \not\models \Phi\theta] \wedge [\mathcal{S}' \not\models \neg\Phi\theta]$. But since the Δ effected by A only made *more* atomic formulae false, $\mathcal{S} \not\models \neg\Phi\theta$. Since $\texttt{LCW}(\Phi)$ holds before executing A, it follows that $\mathcal{S} \models \Phi\theta$ which means that $\mathcal{S} \models \phi_i\theta$ forall $\phi_i \in \Phi$. Now if $\varphi \notin \Phi$ then $\mathcal{S}' \models \Phi\theta$ (since the truth will be unchanged). So it must be the case that $\varphi \in \Phi$ but that means $\mathcal{S}' \models \neg\Phi\theta$. Either way there is a contradiction. $\square$

Proof of Theorem 12 (Minimal Domain Growth) Let φ be a positive literal and let A be an atomic change whose only effect is $\Delta(\varphi, \mathtt{T} \vee \mathtt{F} \rightarrow \mathtt{U})$. Suppose $\Phi \in \mathtt{REL}(\varphi)$. We need to show that $\mathtt{LCW}(\Phi)$ does not hold after A has occurred. Thus it suffices to show that there exists a θ such that $\mathcal{S}' \not\models \Phi\theta$ and $\mathcal{S}' \not\models \neg\Phi\theta$. Since Φ is conjunctive, the definition of $\mathtt{PREL}(\varphi)$ dictates that there exists $\phi \in \Phi$ such that $\phi\theta = \varphi$. Since the only change from $\mathtt{w}$ to $\mathtt{w}'$ is that φ changed its value from true or false to unknown, and since from the definition of $\mathtt{REL}(\varphi)$, we also have $\forall \phi_i \in (\Phi - \phi), \neg(\mathcal{L} \wedge \mathcal{M} \models \neg\phi_i\theta)$, i.e. all other conjuncts may be true in $\mathtt{w}$, it follows that $\Phi\theta$ may be true in $\mathtt{w}'$. Let $\mathcal{M}'$ denote the state of $\mathcal{M}$ after the update due to A, and let $\mathcal{S}'$ denote the possible states of the world after the update due to A. Since $\Phi\theta$ may be true in $\mathtt{w}'$, we have that $\mathcal{S}' \not\models \neg\Phi\theta$. Furthermore, since $\mathcal{M}' \not\models \varphi$, $\mathcal{M}' \not\models \Phi\theta$, and thus $\mathcal{S}' \not\models \Phi\theta$. Therefore, $\mathtt{LCW}(\Phi)$ does not hold. $\Box$

Proof of Theorem 13 (Minimal Domain Growth) Let φ be a positive literal and let A be an atomic change whose only effect is $\Delta(\varphi, \mathtt{F} \rightarrow \mathtt{T})$. We need show that if $\Phi \in \mathtt{MREL}(\varphi)$ then $\mathtt{LCW}(\Phi)$ does not hold after A has occurred. Since Φ is conjunctive, the definition of $\mathtt{PREL}(\varphi)$ dictates that there exists $\phi \in \Phi$ such that $\phi\theta = \varphi$. Since $\Phi \in \mathtt{REL}(\varphi)$, we know that $\Phi\theta$ *may* be true in $\mathtt{w}'$. So, $\mathcal{S}' \not\models \neg\Phi\theta$. Since $\Phi \in \mathtt{MREL}(\varphi)$, we conclude that $\neg\mathtt{LCW}((\Phi - \phi)\theta)$, meaning that for some $\psi \in \Phi$, $\psi\theta \notin \mathcal{M}'$. Hence $\mathcal{M}' \not\models \Phi\theta$, and since Φ contains only positive literals we can conclude that $\mathcal{S}' \not\models \Phi\theta$. Therefore, $\mathtt{LCW}(\Phi)$ does not hold. $\Box$

Proof of Theorem 14 Soundness of QueryLCW We use induction on the number of conjuncts in Φ.

Case $|\Phi| = 0$: An invocation of QueryLCW induces a call to QLCW* where line 1 returns `T`. This is correct, because the null clause (*i.e.*, a ground query with zero conjuncts) is unsatisfiable by definition. Since every state agrees that the null clause is false, $\mathcal{S} \models \neg\Phi$ and hence `LCW`(Φ).

Case $|\Phi| = k \geq 1$: If QLCW* returns `T`, it must have terminated on line 2, 7 or 10. But line 2 only returns true when all ground instantiations, namely Φ itself, are entailed by $\mathcal{M}$. This corresponds directly to the definition of `LCW`. Line 10 will only return `T` under conditions matched by the Composition Rule which is sound by Theorem 4. Suppose instead that QLCW* terminated on line 7. In this case, Φ equals the conjunction of two formulae: Φ' and $\Phi - \Phi'$. Since line 5 checks the conditions of the Instantiation Rule, Theorem 1 guarantees `LCW`(Φ'). Since Φ' is nonempty, the inductive hypothesis guarantees `LCW`($\Phi - \Phi'$). So the Conjunction Rule (Theorem 2) guarantees `LCW`(Φ). Since these are the only termination points for QueryLCW, the algorithm is sound. $\Box$

References

[1] J Ambros-Ingerson and S. Steel. Integrating planning, execution, and monitoring. In *Proc. 7th Nat. Conf. on A.I.*, pages 735–740, 1988.

[2] R. Brachman. "Reducing" CLASSIC to Practice: Knowledge Representation Theory Meets Reality. In *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, October 1992.

[3] D. Brill. *LOOM Reference Manual.* USC-ISI, 4353 Park Terrace Drive, Westlake Village, CA 91361, version 1.4 edition, August 1991.

[4] M. Cadoli and M. Schaerf. A survey of complexity results for non-monotonic logics. *Journal of Logic Programming*, 17:127–160, November 1993.

[5] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence*, 32(3):333–377, 1987.

[6] K. L. Clark. Negation as failure. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 293–322. Plenum Publishing Corporation, New York, NY, 1978.

[7] Alvaro del Val. Computing Knowledge Base Updates. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, pages 740–750, 1992.

[8] Alvaro del Val and Yoav Shoham. Deriving Properties of Belief Update from Theories of Action (II). In *Proceedings of IJCAI-93*, pages 732–737, 1993.

[9] T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates, and counterfactuals. *Artificial Intelligence*, 57:227–270, October 1992.

[10] D. Etherington. *Reasoning with Incomplete Information.* Morgan Kaufmann Publishers, Inc., Los Altos, CA, 1988.

[11] D. Etherington, S. Kraus, and D. Perlis. Nonmonotonicity and the scope of reasoning: Preliminary report. In *Proc. 8th Nat. Conf. on A.I.*, pages 600–607, July 1990.

[12] O. Etzioni, S. Hanks, D. Weld, D. Draper, N. Lesh, and M. Williamson. An Approach to Planning with Incomplete Information. In *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, October 1992. Available via FTP from `pub/ai/` at `ftp.cs.washington.edu`.

[13] O. Etzioni and D. Weld. A softbot-based interface to the internet. *CACM*, 37(7):72–76, July 1994.

[14] Oren Etzioni. Intelligence without robots (a reply to brooks). *AI Magazine*, 14(4), December 1993. Available via anonymous FTP from `pub/etzioni/softbots/` at `cs.washington.edu`.

[15] Oren Etzioni and Ruth Etzioni. Statistical methods for analyzing speedup learning experiments. *Machine Learning*, 14, March 1994. Technical note.

[16] Oren Etzioni and Neal Lesh. Planning with incomplete information in the UNIX domain. In *Working Notes of the AAAI Spring Symposium: Foundations of Automatic Planning: The Classical Approach and Beyond*, pages 24–28, Menlo Park, CA, 1993. AAAI Press.

[17] Oren Etzioni, Neal Lesh, and Richard Segal. Building softbots for UNIX (preliminary report). Technical Report 93-09-01, University of Washington, 1993. Available via anonymous FTP from `pub/etzioni/softbots/` at `cs.washington.edu`.

[18] M. Genesereth and I. Nourbakhsh. Time-saving tips for problem solving with incomplete information. In *Proc. 11th Nat. Conf. on A.I.*, pages 724–730, July 1993.

[19] M. Ginsberg, editor. *Readings in Nonmonotonic Reasoning.* Morgan Kaufmann, San Mateo, CA, 1987.

[20] M. Ginsberg. A circumscriptive theorem prover. *Artificial Intelligence*, 39(2):209–230, June 1989.

[21] M. Ginsberg and D. Smith. Reasoning about action I: A possible worlds approach. *Artificial Intelligence*, 35(2):165–196, June 1988.

[22] K. Golden, O. Etzioni, and D. Weld. Omnipotence without omniscience: Sensor management in planning. In *Proc. 12th Nat. Conf. on A.I.*, July 1994.

[23] G. Grahne. The problem of incomplete information in relational databases. In *Lecture Notes in Computer Science*, volume 554. Springer Verlag, New York, 1991.

[24] H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proc. 2nd Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 387–394, 1991.

[25] A. Keller and M. Wilkins. On the use of an extended relational model to handle changing incomplete information. *IEEE Transactions on Software Engineering*, SE-11(7):620–633, July 1985.

[26] K. Konolidge. Circumscriptive ignorance. In *Proc. 2nd Nat. Conf. on A.I.*, pages 202–204, 1982.

[27] R. Kowalski. Logic for data description. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 77–103. Plenum Publishing Corporation, New York, NY, 1978.

[28] K. Krebsbach, D. Olawsky, and M. Gini. An empirical study of sensing and defaulting in planning. In *Proc. 1st Int. Conf. on A.I. Planning Systems*, pages 136–144, June 1992.

[29] H.J. Levesque. All I know: A study in autoepistemic logic. *Artificial Intelligence*, 42(2–3), 1990.

[30] A. Levy. Queries, updates, and LCW. Personal Communication, 1994.

[31] A. Levy and Y. Sagiv. Queries independent of updates. In *Proceedings of the 19th VLDB Conference*, 1993.

[32] V. Lifschitz. Closed-World Databases and Circumscription. *Artificial Intelligence*, 27:229–235, 1985.

[33] J. McCarthy. Circumscription - a form of non-monotonic reasoning. *Artificial Intelligence*, 13(1,2):27–39, April 1980.

[34] R. Moore. A Formal Theory of Knowledge and Action. In J. Hobbs and R. Moore, editors, *Formal Theories of the Commonsense World.* Ablex, Norwood, NJ, 1985.

[35] D. Olawsky and M. Gini. Deferred planning and sensor use. In *Proceedings, DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control.* Morgan Kaufmann, 1990.

[36] C. Papadimitriou. Games against nature. *Journal of Computer and Systems Sciences*, 31:288–301, 1985.

[37] E. Pednault. *Toward a Mathematical Theory of Plan Synthesis.* PhD thesis, Stanford University, December 1986.

[38] E. Pednault. Synthesizing plans that contain actions with context-dependent effects. *Computational Intelligence*, 4(4):356–372, 1988.

[39] E. Pednault. ADL: Exploring the middle ground between STRIPS and the situation calculus. In *Proc. 1st Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 324–332, 1989.

[40] J.S. Penberthy and D. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 103–114, October 1992. Available via FTP from `pub/ai/` at `ftp.cs.washington.edu`.

[41] M. Peot and D. Smith. Conditional Nonlinear Planning. In *Proc. 1st Int. Conf. on A.I. Planning Systems*, pages 189–197, June 1992.

[42] O. Raiman and J. de Kleer. A Minimality Maintenance System. In *Proc. 3rd Int. Conf. on Principles of Knowledge Representation and Reasoning*, pages 532–538, October 1992.

[43] R. Reiter. On closed world databases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 55–76. Plenum Press, 1978. Reprinted in [19].

[44] R. Reiter. Circumscription implies predicate completion (sometimes). *Proc. 2nd Nat. Conf. on A.I.*, pages 418–420, 1982.

[45] D. Smith. Finding all of the solutions to a problem. In *Proc. 3rd Nat. Conf. on A.I.*, pages 373–377, 1983.

[46] A. Tate. Generating project networks. In *Proc. 5th Int. Joint Conf. on A.I.*, pages 888–893, 1977.

[47] D. Weld. An introduction to least-commitment planning. *AI Magazine*, pages 27–61, Winter 1994. Available via FTP from `pub/ai/` at `ftp.cs.washington.edu`.

[48] M. Winslett. Reasoning about action using a possible models approach. In *Proc. 7th Nat. Conf. on A.I.*, page 89, August 1988.

[49] M. Winslett. *Updating Logical Databases.* Cambridge University Press, 1990. Cambridge, England.