Tractable Closed World Reasoning with Updates

(Extended Abstract)

Oren Etzioni Keith Golden Daniel Weld*

Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195
{etzioni, kgolden, weld}@cs.washington.edu

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, but has not addressed computational tractability. Work in planning has traditionally made the closed world *assumption* but has avoided closed world reasoning. We take a middle position, and describe a tractable method for closed world reasoning over a restricted logical theory of the sort utilized by planning algorithms such as STRIPS, NONLIN, and TWEAK. We show the method to be both sound and tractable (linear-time updates), and incorporate it into the UCPOP planner. Experiments utilizing our Internet softbot (software robot) demonstrate that the method can substantially improve the softbot's performance by eliminating redundant information gathering.

*We thank Neal Lesh, Rich Segal, Steve Hanks, Denise Draper, and Mike Williamson for helpful discussions. Tom Dean, Rao Kambhampati, Craig Knoblock, Martha Pollack, and Yoav Shoham made extremely helpful comments on drafts of this paper. This research was funded in part by Office of Naval Research Grants 90-J-1904 and 92-J-1946, and by National Science Foundation Grants IRI-8957302, IRI-9211045, and IRI-9357772. Golden is supported in part by a UniForum Research Award.

1 Introduction and Motivation

In many cases, an agent does not have complete information about its world. For instance, a robot may not know the size of a bolt or the location of an essential tool [29, 21]. Similarly, a software agent, such as the Internet *softbot* (software robot) [14, 13, 9], cannot be familiar with the contents of *all* the bulletin boards, FTP sites, and files accessible through the Internet[1]

Recent work has sketched a number of algorithms for planning with incomplete information (*e.g.*, [1, 29, 21, 32, 11, 13]). These planners are vulnerable to *redundant information gathering* when they plan to "sense" information that is already known to the agent [12]. Since satisfying the preconditions of an information-gathering action can involve arbitrary planning, the cost of redundant information gathering is unbounded in theory and quite large in practice (see Section 4).

To solve this problem we utilize an explicit database of meta-level sentences such as "I know all the files in `/kr94`," which encode *local closed world information* (`LCW`). The information in this database is equivalent to the "closed roles" found in knowledge-representation systems such as CLASSIC [2], LOOM [3], to predicate completion axioms [4, 20], and to circumscription axioms [26, 23]. Indeed, work on autoepistemic logic (*e.g.*, [27, 22]) and circumscription (*e.g.*, [19, 8, 33]) has investigated the *logic* of closed world reasoning over arbitrary first-order theories, but without considering computational complexity. Work on formal theories of action (*e.g.*, [25, 17, 7]) has done the same for theory updates. Formal models of perception have been proposed [5, 6], but not made computational. Finally, work in database theory has yielded intractable approaches such as enumerating the possible logical models corresponding to a database [36] or computing the disjunction of all possible results of an update [18]. In contrast, we have formulated a tractable algorithm for closed world reasoning, with updates, over the restricted representation language utilized by existing planners. Our approach has several novel features:

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

- We present a sound and tractable calculus for querying and updating closed world information as the state of the world changes (Sections 2.2 and 3).
- We incorporate our closed world reasoning machinery into an extended version of the UCPOP partial-order planner [31, 16] enabling it to avoid redundant information gathering despite the absence of complete information. Experiments in the UNIX domain show that the benefit derived by avoiding redundant sensing far outweighs the costs of maintaining the closed world information (Section 4).

2 Local Closed World Information

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 facts entailed by the world forms a complete logical theory; we use $\mathcal{D}_{\mathrm{W}}$ to denote this theory. Following [30, 15], 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 such that neither $\mathcal{S} \models \varphi$ nor $\mathcal{S} \models \neg\varphi$; in this case we say φ is unknown to the agent.

2.1 Representing Closed World Information

Due to the size of $\mathcal{S}$ (a potentially infinite set of large structures), the agent cannot represent it explicitly. Instead we represent the facts known by the agent with a database $\mathcal{D}_{\mathrm{M}} \subset \mathcal{D}_{\mathrm{W}}$; if $\varphi \in \mathcal{D}_{\mathrm{M}}$ then $\mathcal{S} \models \varphi$. In the interests of tractability, we restrict $\mathcal{D}_{\mathrm{M}}$ to ground literals. Since $\mathcal{D}_{\mathrm{M}}$ is incomplete, the Closed World Assumption (CWA) is invalid – the agent cannot automatically infer that any sentence that is not in $\mathcal{D}_{\mathrm{M}}$ is false. Thus, the agent is forced to represent false facts in $\mathcal{D}_{\mathrm{M}}$, explicitly, as sentences tagged with the truth value `F`.

Yet, many sensing actions return exhaustive information which warrants limited or "local" closed world information. For example, scanning with a TV camera shows *all* objects in view, and the UNIX `ls` command lists *all* files in a given directory. After executing `ls`, 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 access to an action that yields closed world *information*.

Although the agent now knows that `(parent.dir foo /kr94)` is false, it is impractical for the agent to store this information explicitly, since there are an infinite number of such sentences. This observation leads to a minor paradox: we cannot explicitly represent in $\mathcal{D}_{\mathrm{M}}$ *every* sentence we know to be false, yet we cannot make the CWA. We adopt a simple solution: we represent closed world information explicitly as a meta-level database, $\mathcal{D}_{\mathrm{C}}$, containing sentences of the form `LCW`(Φ) that record *where* the agent has closed world information. For instance, we represent knowing all the files in `/kr94` as:

```
LCW((parent.dir ?f /kr94))
```

We adopt the following procedural semantics for `LCW` sentences. When asked whether it believes in an atomic sentence Φ, the agent first checks to see if it is in $\mathcal{D}_{\mathrm{M}}$. If it is, then the agent responds with the truth value (`T` or `F`) associated with the sentence. However, if $\Phi \notin \mathcal{D}_{\mathrm{M}}$ then Φ could be either `F` or `U`. To resolve this ambiguity, the agent checks whether $\mathcal{D}_{\mathrm{C}}$ entails `LCW`(Φ). If so, the fact is `F` otherwise it is unknown (truth value `U`). Note that the agent need not perform inference on $\mathcal{D}_{\mathrm{M}}$ since it contains only ground literals.

A `LCW` sentence can be understood in terms of circumscription [23]. For the example above, one defines the predicate `(P ?x)` to be true exactly when `(parent.dir ?x /kr94)` is true, and circumscribes `P` in $\mathcal{D}_{\mathrm{M}}$. However, existing work on circumscription is insufficient for our needs. The work does not provide a theory indicating which predicates should be minimized and no way to update what is being minimized when the world changes.[2] Moreover, most work on circumscription is not computational (see [24] for an exception), and none is computationally tractable. Yet, for the purpose of

[2]Following [17, 18] 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.

planning with incomplete information, we require the ability to represent, infer, and update closed world information *quickly*. Below, we describe a computational mechanism that addresses this problem.

2.2 Inferring Local Closed World Information

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

`LCW((and (parent.dir ?f /kr94) (postscript ?f)))`

Correctly answering `LCW` queries is not a simple matter of looking up `LCW` assertions in the $\mathcal{D}_{\mathrm{C}}$ database. For instance, suppose that 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:

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

In general, we have:

Theorem 1 (Instantiation Rule) *If Φ is a logical sentence and θ is a substitution, then* $\texttt{LCW}(\Phi)\models\texttt{LCW}(\Phi\theta)$.[3]

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:

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

The intuition behind the rule is simple — if one knows the contents of two sets then one knows their intersection. Note that the converse 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`.

Given this inference rule, it is natural to inquire whether answering `LCW` queries is tractable. In general, the answer is no:

[3]Proofs of the theorems are in the full paper.

Theorem 3 *If* $\mathcal{D}_C$ *is an arbitrary set of* `LCW` *sentences, and* Φ *is a distinct* `LCW` *sentence, then computing whether* $\mathcal{D}_C \models \Phi$ *is undecidable.*

To gain tractability, we restrict the sentences in $\mathcal{D}_C$ 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 all the files in `/kr94` *except* the files with a `.dvi` extension," and "If a file is either in `/kr94` or is a lisp file then I know it." However, `LCW` inference is reduced to the problem of matching a conjunctive `LCW` query against a database of conjunctive `LCW` assertions. Although a conjunctive match terminates in time exponential in the length of the query [35], if we accept a bound k on the number of conjuncts in a query, then the running time is a polynomial of order k in the size of $\mathcal{D}_C$ [10]. In the UNIX domain, we have found that `LCW` queries are typically short ($k \leq 3$) which, with the aid of standard indexing techniques, yields reasonably fast `LCW` inference in practice (see Section 4).

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 `LCW`. When a file is compressed, for example, we lose information about its size; when all postscript files are deleted from a directory, we gain the information that it contains no such files. This section presents a linear-time algorithm for updating $\mathcal{D}_C$, the agent's store of `LCW` sentences. The key to our algorithm is the restriction of $\mathcal{D}_M$ to ground literals (Section 2.1). Since we have foregone the ability to write domain axioms relating predicates, we sidestep the ramification problem [25]; instead, we demand that updates to $\mathcal{D}_M$ (such as those caused by action execution) explicitly enumerate changes to *every* predicate that is affected.[4]

Recall that $\mathcal{D}_M$ is a database of ground atomic sentences each explicitly tagged with a `T` or `F` truth value. We signify the atomic update of a single positive literal φ from unknown to true with $\Delta(\varphi, \mathtt{U} \rightarrow \mathtt{T})$ and denote analogous changes in the obvious manner.[5]

[4]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 ?x)` could be defined as `(forall (?y) (not (on ?y ?x)))`.

[5]Note that, as explained in Section 2.1, if a fact is absent from $\mathcal{D}_M$, the agent deter-

$\mathcal{D}_{\text{M}}$ can change due to information gathering on the part of the agent, or when the agent is informed of a change to the state of the external world. We formulate the update policy as a set of rules and state them as theorems since they are sound. Our update rules compute conservative (*i.e.*, sound but incomplete) updates from $\mathcal{D}_{\text{C}}$ to $\mathcal{D}'_{\text{C}}$ based on the changes to `w` and $\mathcal{D}_{\text{M}}$. By distinguishing between transitions to and from `U` truth values, $\mathcal{D}_{\text{C}}$ 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 4 (Information Gain) *If* $\Delta(\varphi, \texttt{U} \rightarrow \texttt{T} \vee \texttt{F})$ *then* $\mathcal{D}'_{\text{C}} \supseteq \mathcal{D}_{\text{C}}$.

This theorem suggests a simple conservative policy: as long as an action gains information, then $\mathcal{D}_{\text{C}}$ 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 the agent knows the cardinality of the set of instances matching a pattern, then it can deduce when it has `LCW` [34]. The most common case involves functional relations. When an information-gathering action yields the unique value of a variable (*e.g.*, `wc` reports the word count of a file), the agent knows it has local closed world information and can update $\mathcal{D}_{\text{C}}$.

To state this update rule in general, we define an instance function, I, that returns the set of sentences matching Φ in a given theory:

$$\text{I}(\Phi, \mathcal{D}) = \{\Psi \in \mathcal{D} \mid \exists \theta \text{ such that } \Phi\theta = \Psi\} \qquad (1)$$

We refer to the set denoted by I(Φ,$\mathcal{D}$) as the *domain* of Φ in $\mathcal{D}$. When the cardinality of the Φ's domain is the same in $\mathcal{D}_{\text{M}}$ and in the theory induced by `w`, we can conclude that we have `LCW`(Φ):

mines whether it is `F` or `U` by consulting $\mathcal{D}_{\text{C}}$. Thus, updates of the form $\Delta(\varphi, \texttt{U} \rightarrow \texttt{F})$ or $\Delta(\varphi, \texttt{F} \rightarrow \texttt{U})$ may occur when $\mathcal{D}_{\text{C}}$ changes, even if $\mathcal{D}_{\text{M}}$ does not.

Theorem 5 (Counting Rule (after [34])) *If* $|I(\Phi, \mathcal{D}_{\mathrm{M}})| = |I(\Phi, \mathcal{D}_{\mathrm{W}})|$ *then* $\mathcal{D}'_{\mathrm{C}} \leftarrow \mathcal{D}_{\mathrm{C}} \cup \{\Phi\}$

To utilize the Counting Rule in practice, our agent relies on explicit axioms such as $\forall$`?f`, $|$I(`(word.count ?f ?c)`$, \mathcal{D}_{\mathrm{W}})| = 1$ which are used to expand the effects of action schemata to include `LCW` updates to $\mathcal{D}_{\mathrm{C}}$.

An agent can obtain local closed world information, even when the cardinality of a domain is variable (*e.g.*, the number of files in a directory) by executing an action with universally-quantified effects. For instance, the execution of `ls` in the directory `/bin` provides information on *all* files in that directory: `(LCW (parent.dir ?o /bin))` In general,[6] we have:

Theorem 6 (Forall Rule) *If an action has a universally quantified observational effect of the form* `(forall (?V1..?Vn) (pred ?V1..?Vn)))` *then after execution of that action,* $\mathcal{D}'_{\mathrm{C}} \leftarrow \mathcal{D}_{\mathrm{C}} \cup \{$`(pred ?V1..?Vn)`$\}$

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 this precise, we introduce the set REL(φ) to denote the `LCW` assertions relevant to a literal φ as follows:[7]

$$\mathrm{REL}(\varphi) \equiv \{\Phi \in \mathcal{D}_{\mathrm{C}} \mid \exists x \in \Phi, \exists \theta, x\theta = \varphi\}$$

For example, if an agent has complete information on the sizes of all files in `/kr94`, and a file in `/kr94` is compressed, then the sentence

`LCW((and (parent.dir ?f /kr94) (size ?f ?c)))`

is "relevant" and must be removed from $\mathcal{D}_{\mathrm{C}}$.

Theorem 7 (Information Loss) *If* $\Delta(\varphi, \mathtt{T} \vee \mathtt{F} \rightarrow \mathtt{U})$ *then* $\mathcal{D}'_{\mathrm{C}} \leftarrow \mathcal{D}_{\mathrm{C}} - REL(\varphi)$.

Note that, given our assumptions (correct information, *etc.*), information is only lost when the world's state changes.

[6]Space considerations in this extended abstract preclude a rigorous statement of this theorem; the full paper defines precisely *what* a universally quantified observational effect *is* and generalizes this result to universally quantified conditional and causal effects.

[7]Since the sentences in $\mathcal{D}_{\mathrm{C}}$ are conjunctions of positive literals, we use the notation $\varphi \in \Phi$ to signify that φ is one of Φ's conjuncts.

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 `kr.sty` is moved from the `/tex` directory to `/kr94`, we have that $\mathcal{D}'_{\mathrm{M}} \neq \mathcal{D}_{\mathrm{M}}$ but neither database is a superset of the other. When the model changes in this way, the domain (denoted by I) of the patterns containing `(parent.dir ?f /kr94)` grows whereas the domain of the patterns containing `(parent.dir ?f /tex)` contracts. `LCW` information may be lost in patterns whose domain grew. Suppose that, prior to the file move, we knew the word counts of all the files in `/kr94`; if we do not know the word count of `kr.sty`, then that `LCW` assertion is no longer true. In general, we have:

Theorem 8 (Domain Growth) *If* $\Delta(\varphi, \mathtt{F} \rightarrow \mathtt{T})$ *then* $\mathcal{D}'_{\mathrm{C}} \leftarrow \mathcal{D}_{\mathrm{C}} - \mathit{REL}(\varphi)$

When the domain of a pattern contracts, no `LCW` information is lost. For instance, when a file is removed from the directory `/kr94`, we will still know the lengths of all the files in that directory.

Theorem 9 (Domain Contraction) *If* $\Delta(\varphi, \mathtt{T} \rightarrow \mathtt{F})$ *then* $\mathcal{D}'_{\mathrm{C}} = \mathcal{D}_{\mathrm{C}}$.

Note that our update rules cover all possible truth-value transitions. The rules guarantee that $\mathcal{D}_{\mathrm{C}}$ does not contain invalid `LCW` assertions, so long as the agent is appraised of any changes to the world state. For the sake of tractability, the rules are conservative— the $\mathcal{D}_{\mathrm{C}}$ may be incomplete. For example, when the word count of `kr.sty` is unknown in the above example, we could say that we know the word counts of all the files in `/kr94` *except* `kr.sty`. However, that would require us to store negated and disjunctive sentences in $\mathcal{D}_{\mathrm{C}}$, which would make `LCW` inference intractable. To see this, consider a singleton `LCW` query such as `LCW((parent.dir ?f /kr94))`. If $\mathcal{D}_{\mathrm{C}}$ contains only positive conjunctions, the query can be answered in sub-linear time—examining only singleton `LCW` assertions indexed under the predicate `parent.dir`. If disjunction is allowed, however, then the combination of multiple `LCW` sentences has to be explored. For instance, by the Conjunction Rule, we have that $\mathtt{LCW}(\Phi \vee \Psi) \wedge \mathtt{LCW}(\Phi \vee \neg\Psi) \models \mathtt{LCW}(\Phi)$. In general, answering a singleton query, in the presence of negation and disjunction, is NP-hard.

3.4 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 4 implies that no sentences have to be retracted from $\mathcal{D}_{\mathrm{C}}$. `LCW` sentences may be added by the Counting Rule (constant time for functional relations–see Section 3.1) or by the Forall Rule (constant time).

- **Information loss & domain growth:** the agent has to retrieve the set REL(Φ) which, in the worst case, takes time linear in the size of $\mathcal{D}_{\mathrm{C}}$. The agent then removes each element of the set from $\mathcal{D}_{\mathrm{C}}$, which takes time linear in the size of the set REL. Typically, REL(Φ) is a small fraction of $\mathcal{D}_{\mathrm{C}}$.

- **Domain contraction:** $\mathcal{D}_{\mathrm{C}}$ remains unchanged in this case.

While each individual update rule application is reasonably fast, even in the worst case, we have to consider the possibility of a cascade of $\mathcal{D}_{\mathrm{C}}$ updates. Will the update rules chain on each other? Are such chains guaranteed to terminate? Fortunately, we can prove that rule chaining is unnecessary. The intuition is as follows. Chaining could potentially occur in one of two ways. First, when $\mathcal{D}_{\mathrm{C}}$ 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 7) has to be applied to further retract sentences from $\mathcal{D}_{\mathrm{C}}$. However, careful examination of the definition of REL shows that this is not the case — all relevant `LCW` sentences have already been excised from $\mathcal{D}_{\mathrm{C}}$. Second, when $\mathcal{D}_{\mathrm{C}}$ 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 $\mathcal{D}_{\mathrm{C}}$, and the Forall and Counting Rules do not yield new `LCW` sentences as a consequence.[8]

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

[8]The Conjunction and Instantiation Rules are applied in response to `LCW` queries, but ignored when $\mathcal{D}_{\mathrm{C}}$ is updated.

3.5 Discussion

The update rules defined above form a sound, linear-time algorithm for updating $\mathcal{D}_{\mathrm{M}}$ and $\mathcal{D}_{\mathrm{C}}$. We believe our rules satisfy the update postulates specified in [17] and generalized in [7], but we have not yet attempted a proof. Since sentences in $\mathcal{D}_{\mathrm{C}}$ are restricted to positive conjunctions and since our Information Gain and Domain Growth rules are conservative, the algorithm is incomplete. Nevertheless, it is easy to see that our algorithm is better than the trivial update algorithm ($\mathcal{D}'_{\mathrm{C}} \leftarrow \{\}$). In the UNIX domain, for example, our Counting 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, given the restriction to positive conjunctions, our information loss and domain contraction rules are complete. Ultimately, the test of any mechanism for closed world reasoning – complete or not – is its impact on the agent's performance. Below, we describe preliminary experiments that suggest ours is effective in practice.

4 Experimental Results

While we have shown that `LCW` inference and update are tractable, computational complexity is not always a good predictor of real performance. To test whether our rules perform well in practice (*i.e.*, run quickly and with sufficient completeness for useful results) we added them to a version of the UCPOP planner [31] extended to interleave planning and execution (following [1]) and incorporated into our Internet Softbot [16].

Table 1 quantifies the impact on the planner's performance: `LCW` inference yields a significant performance gain. The tests mostly consist of simple file searches (*e.g.*, find a file with word count greater than 5000, containing the string "theorem," etc.) and relocations. The actions executed in the tests include `mv` (which can destroy `LCW`), observational actions such as `ls, wc` and `grep`, and more. Each experiment was started from a new lisp session. $\mathcal{D}_{\mathrm{M}}$ and $\mathcal{D}_{\mathrm{C}}$ start out empty, but they are not purged between problems, so for each problem the agent benefits from the information gained in solving the previous problems.

Maintaining $\mathcal{D}_{\mathrm{C}}$ introduced less than 15% overhead per plan explored, and reduced the number of plans explored substantially. In addition, the plans produced are often considerably shorter, since redundant sensing steps are

PROBLEM SET	PLANNER	PLANS EXPLORED	ACTIONS EXECUTED	TOTAL TIME
22 PROBLEMS, 13 SOLVABLE	With LCW	420	55	109
	Without	3707	724	966
14 PROBLEMS, ALL SOLVABLE	With LCW	373	55	94
	Without	1002	140	160

Table 1: Reasoning about local closed world information (LCW) improves the performance of the softbot on two suites of UNIX problems. Times are in CPU seconds on a Sun Microsystems SPARC-10. Without LCW inference the softbot fails to solve eight of the problems in the first set, and one of the problems in the second set, before reaching a 100 CPU second time bound. With LCW, the softbot solves all the problems. The mean size of $\mathcal{D}_C$ (the agent's store of LCW information) is 155 sentences. The maximum size is 167.

eliminated. Without LCW, the softbot performed 16 redundant `ls` operations, and 6 redundant `pwd`s in a "typical" file search. With LCW, on the other hand, the softbot performed no redundant sensing. Furthermore, when faced with unachievable goals, the softbot with LCW inference was able to fail quickly; however, without LCW it conducted a massive search, executing many redundant sensing operations in a forlorn hope of observing something that will satisfy the goal. While more experimentation is necessary, these experiments (combined with our analytic results) suggest that LCW inference has the potential to substantially improve performance.

5 Related Work

Since we have already discussed the connection between our work and the broad spectrum of research on autoepistemic logic, circumscription, database theory, and formal theories of actions, we now focus on related work in the planning literature. Our research has its roots in the SOCRATES planner, where the problem of redundant information gathering was initially discovered [12]. Like our current planner, SOCRATES utilized the UNIX domain as its testbed, supported the UWL representation, and interleaved planning with execution. SOCRATES supported a restricted representation of LCW, which enabled it to avoid redundant information gathering in many cases. Our advances over SOCRATES include an improved semantics for the notion of local closed world information, the ability to satisfy universally quantified goals, and our sound and tractable calculi for `LCW` inference and update.

Some planners count the number of relevant ground propositions in their model, before inserting information-gathering steps into their plans, to check whether the desired information is already known [29]. However, this heuristic, which corresponds directly to the Counting Rule (Section 3 and [34]), is only effective when the number of sought-after facts is known in advance. For example, a bolt has exactly one width, but the number of files in `/kr94` is unknown.

Genesereth and Nourbakhsh [15] share our goal of avoiding redundant information gathering, but do so using radically different mechanisms, and in the context of state-space search. They derive completeness-preserving rules for pruning the search as well as rules for terminating planning and beginning execution. However, they do not have notions that correspond to `LCW`, a database like $\mathcal{D}_C$, or our inference and update calculi.

6 Conclusions

Our work was motivated by the problem of eliminating redundant information gathering in planners with incomplete information. In this paper, we presented a sound and computationally tractable method for representing, inferring, and updating *local closed world information* (`LCW`) (*e.g.*, "I know the lengths of all the files in `/kr94`") over a restricted logical theory of the sort utilized by planning algorithms such as STRIPS, NONLIN, and TWEAK. To demonstrate the utility of our approach, we have incorporated our closed world reasoning machinery into the Internet softbot [14, 13, 9]. Preliminary experiments indicate that `LCW` inference can significantly speed the softbot and drastically reduce the number of actions executed, but more extensive experiments are needed to fully evaluate the utility of our approach.

References

[1] J Ambros-Ingerson and S. Steel. Integrating planning, execution, and monitoring. In *Proceedings of AAAI-88*, pages 735–740, 1988.

[2] R. Brachman. "Reducing" CLASSIC to Practice: Knowledge Representation Theory Meets Reality. In *Proceedings of KR-92*, 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] 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.

[5] E. Davis. Inferring ignorance from the locality of visual perception. In *Proceedings AAAI-88*, pages 786–790. AAAI, 1988.

[6] E. Davis. *Representations of Commonsense Knowledge*. Morgan Kaufmann Publishers, Inc., San Mateo, CA, 1990.

[7] 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.

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

[9] Oren Etzioni. Intelligence without robots (a reply to brooks). *AI Magazine*, December 1993.

[10] Oren Etzioni. A structural theory of explanation-based learning. *Artificial Intelligence*, 60(1):93–140, March 1993.

[11] Oren Etzioni, Steve Hanks, Daniel Weld, Denise Draper, Neal Lesh, and Mike Williamson. An approach to planning with incomplete information. In *Proceedings of KR-92*, October 1992. Available via annonymous FTP from `~ftp/pub/ai/` at `cs.washington.edu`.

[12] 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.

[13] Oren Etzioni, Neal Lesh, and Richard Segal. Building softbots for UNIX (preliminary report). Technical Report 93-09-01, University of Washington, 1993.

[14] Oren Etzioni and Richard Segal. Softbots as testbeds for machine learning. In *Working Notes of the AAAI Spring Symposium on Knowledge Assimilation*, Menlo Park, CA, 1992. AAAI Press.

[15] M. Gensereth and I. Nourbakhsh. Time-saving tips for problem solving with incomplete information. In *Proceedings of 11th Natl. Conf. on Artifical Intelligence (AAAI-93)*, pages 724–730. MIT Press(AAAI), July 1993.

[16] K. Golden, O. Etzioni, and D. Weld. XII: Planning for Universal Quantification and Incomplete Information. Technical report, University of Washington, Department of Computer Science and Engineering, To Appear 1993.

[17] H. Katsuno and A. Mendelzon. On the difference between updating a knowledge base and revising it. In *Proceedings of KR-91*, pages 387–394, 1991.

[18] 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.

[19] K. Konolidge. Circumscriptive ignorance. In *Proceedings of AAAI-82*, pages 202–204, 1982.

[20] 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.

[21] K. Krebsbach, D. Olawsky, and M. Gini. An empirical study of sensing and defaulting in planning. In *Proceedings of the First International Conference on AI Planning Systems*, pages 136–144, June 1992.

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

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

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

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

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

[27] R.C. Moore. A formal theory of knowledge and action. In J. Hobbs and R. Moore, editors, *Formal Theories of the Commonsense World*. Ablex, 1985.

[28] N. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing Company, Palo Alto, CA, 1980.

[29] 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.

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

[31] J.S. Penberthy and D. Weld. UCPOP: A sound, complete, partial order planner for ADL. In *Proceedings of KR-92*, pages 103–114, October 1992. Available via annonymous FTP from `~ftp/pub/ai/` at `cs.washington.edu`.

[32] Mark A. Peot and David E. Smith. Conditional nonlinear planning. In *Proceedings of the First International Conference on AI Planning Systems*, pages 189–197, June 1992.

[33] R. Reiter. Circumscription implies predicate completion (sometimes). *Proceedings of AAAI-82*, pages 418–420, 1982.

[34] D. Smith. Finding all of the solutions to a problem. In *Proceedings of AAAI-83*, pages 373–377, 1983.

[35] Milind Tambe and Paul Rosenbloom. Eliminating expensive chunks by restricting expressiveness. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 1989.

[36] M. Winslett. Reasoning about action using a possible models approach. In *Proceedings of AAAI-88*, page 89, August 1988.