# Abductive Reasoning with Filtered Circumscription*

**Martin Magnusson and Jonas Kvarnström and Patrick Doherty**
Department of Computer and Information Science
Linköping University, 581 83 Linköping, Sweden
{marma,jonkv,patdo}@ida.liu.se

## Abstract

For logical artificial intelligence to be truly useful, its methods must scale to problems of realistic size. An interruptible algorithm enables a logical agent to act in a timely manner to the best of its knowledge, given its reasoning so far. This seems necessary to avoid analysis paralysis, trying to think of every potentiality, however unlikely, beforehand. These considerations prompt us to look for alternative reasoning mechanisms for filtered circumscription, a nonmonotonic reasoning formalism used e.g. by Temporal Action Logic and Event Calculus. We generalize Ginsberg's circumscriptive theorem prover and describe an interruptible theorem prover based on abduction that has been used to unify planning and reasoning in a logical agent architecture.

## 1 Introduction

The world around us is uncertain. In fact, we have to cope with "pervasive ignorance" [Pollock, 2008] about most things. This is possible by reasoning defeasibly rather than purely deductively. But the world is also dynamic. Even when we *do* have all the relevant knowledge, we may not have time to think through all its consequences before the changing circumstances make our conclusions obsolete. This is most evident when planning our actions. Unless there is great risk involved, we most often carry out our plans after considering only a small subset of their consequences.

If we want to build logical agents that act autonomously to solve real world problems, we have to equip them with similar mechanisms to cope. Moving from simple benchmark problems to problems of realistic size has proven difficult due to the intractability of logical reasoning. An interruptible algorithm enables an agent to act in a timely manner, to the best of its knowledge given its reasoning so far. This seems a necessary feature of any nonmonotonic reasoning mechanism aimed to scale towards solving real world problems involving very large knowledge bases.

These considerations prompt us to look for alternative reasoning mechanisms for filtered circumscription, a nonmonotonic logic formalism for reasoning about action and change used e.g. by Temporal Action Logic (TAL). Regular theorem provers are not directly applicable to TAL's second-order circumscription axiom. This hinder has usually been overcome by applying predicate completion [Doherty and Kvarnström, 2007] to produce a first-order equivalent theory. But predicate completion involves a potentially costly computation applied to the entire knowledge base before any reasoning can begin. Moreover, the transformation must be recomputed whenever the agent's beliefs change, even e.g. when considering the effects of an action while planning. Finally, the reasoning involved is not interruptible. Predicate completion works by turning defeasible reasoning into deductive proof. These proofs must consider all potential objections to a defeasible conclusion before any answer can be given.

We extend Ginsberg's circumscriptive theorem prover [1989] to *filtered* circumscription. This forms the basis for an interruptible theorem prover based on abduction that operates on the Temporal Action Logic formulas directly, without any compilation step. We show how the same reasoning mechanism can be used to perform abductive planning, providing a unified planning and reasoning framework in a logical agent architecture. Such an agent could act in an any-time manner, using tentative answers based on defeasible assumptions if forced to act quickly, while still considering all potential objections given sufficient time for deliberation.

## 2 Preliminaries

While the results in this paper should be interesting for other logics of action and change, such as the Event Calculus, we focus on Temporal Action Logic and hence give a brief introduction to it. Similarly, while different proof systems could be used, our work implements the natural deduction system introduced below.

### 2.1 Temporal Action Logic

Temporal Action Logic is a highly expressive logic for reasoning about action and change. The origins of TAL are found in Sandewall's Features and Fluents framework [1994].

*This work is supported in part by the Swedish Foundation for Strategic Research (SSF) Strategic Research Center MOVIII, the Swedish Research Council Linnaeus Center CADICS, VR project (50405001, 50405002), and CENIIT, the Center for Industrial Information Technology.

Sandewall classified different variants of non-monotonic reasoning according to the types of reasoning problems for which they are applicable. TAL is a stand-alone logic based on one of the most general variant of these.

A central concept in TAL is *occlusion*. It is introduced as a flexible way to deal with the frame problem and related problems. The basic idea is to make fluent values, given by $Holds(time, fluent, value)$, persist over time by minimizing their opportunities for change. A predicate *Occlude(time,fluent)* represents the possibility of a fluent changing its value. This is a key difference from earlier attempts to use circumscription to minimize *change* rather than *potential for change*. Negated occlusion is then the property of a fluent not being able to change its value from the previous time point, i.e. persistence. A fluent $f$'s default persistence at all time points (assuming unbound variables are implicitly universally quantified) is then axiomatized by:

$$\neg Occlude(t+1, f) \rightarrow (Holds(t, f, v) \leftrightarrow Holds(t+1, f, v))$$

Detailed control over fluent persistency can be exercised by adding similar persistence formulas, collectively denoted by $T_{\text{per}}$.

Situations that are known to cause a fluent's value to change must also occlude that fluent. E.g., actions must explicitly occlude affected fluents. We do not wish, however, to enumerate all situations in which a fluent is *not* occluded. The assumption is that, by default, things do not change without a reason. The Features and Fluents framework used *preferential entailment* to enforce this. Logical consequence is defined only w.r.t. models in which the extension of *Occlude* is minimal. Action occurrences, specified by the $Occurs(time_{start}, time_{end}, action)$ predicate, must also be minimized to prevent occlusion by spurious actions. But the question of how to compute the intended consequences was left open.

TAL provides a *syntactic* characterization using a form of circumscription called *filtered circumscription* [Doherty and Lukaszewicz, 1994], also referred to as *forced separation* [Shanahan, 1997]. Circumscription is used to minimize *Occlude* and *Occurs*, while fixing *Holds*, but $T_{\text{per}}$ is forcedly separated from the rest of the theory $T$, outside the scope of the circumscription:

$$Circ(T; Occlude, Occurs) \wedge T_{\text{per}}$$

If the persistence formulas $T_{\text{per}}$ had been included in the circumscription, then the extensions of *Occlude* had not been made any smaller. To see this, note e.g. that the contrapositives of formulas in $T_{\text{per}}$ would have said that fluent change is *by itself* a reason for occlusion. Removing these formulas from the circumscription leaves only occlusion with some *explicit* cause, such as being occluded by an action's effects. The filtering occurs when we add $T_{\text{per}}$ to the minimized theory, removing all models in which a fluent changes despite not being occluded.

This short introduction to TAL is intended to aid understanding of the rest of this paper. A more detailed presentation with a complete list of TAL's features is available elsewhere [Doherty and Kvarnström, 2007].

### 2.2 Natural Deduction

Example proofs will be presented in Suppes' style natural deduction [Pelletier, 1999]. Each proof row consists of a premise set, a row number, the formula, and a list of row numbers identifying the previous proof rows used by the current inference step (or empty for given input formulas). The premise set is a convenient bookkeeping device that keeps track of the assumptions that a formula depends on. This is important, not only for natural deduction's ability to construct proofs using temporary assumptions, but also during abductive proofs to label formulas by the set of ground instances of abducibles required. An assumption depends only on itself and thus its premise set only contains its own row number. Inference rules then combine their premises' dependency sets to form the conclusion's premise set, usually by taking the set union.

Another useful device is an explicit notation for proof goals. We write *Show P* when we adopt an interest in proving $P$, either because it is given as the overall proof goal, or as the result of reasoning backwards from the proof goal. Both devices are illustrated by the following simple abductive proof, where the conclusion is allowed to depend on a consistent set of ground instances of the abducible $\neg Ab(x)$:

| | | | |
|---|---|---|---|
| $\{1\}$ | 1 | $Bird(x) \wedge \neg Ab(x) \rightarrow Flies(x)$ | |
| $\{2\}$ | 2 | $Bird(tweety)$ | |
| $\{\}$ | 3 | *Show* $Flies(tweety)$ | |
| $\{\}$ | 4 | *Show* $\neg Ab(tweety)$ | $1,2,3$ |
| $\{5\}$ | 5 | $\neg Ab(tweety)$ | $4$ |
| $\{1,2,5\}$ | 6 | $Flies(tweety)$ | $1,2,5$ |

## 3 Predicate Completion

TAL's syntactic characterization in terms of filtered circumscription produces a second-order theory to which regular theorem provers are not applicable. Fortunately, by placing certain syntactic restrictions on the TAL formulas one can ensure that the second-order circumscription formula can be compiled into an equivalent first-order characterization [Doherty, 1996]. The transformation is equivalent to Clarke's *predicate completion* [1978].

To see how it works, consider the Yale Shooting Problem formulated in TAL. There is an action for loading the gun, an action for firing the gun and killing Fred the turkey just in case the gun was loaded, the observation that Fred is initially alive, and a narrative consisting of the load and fire actions with a small wait in between. Note how the actions explicitly release the affected fluents from persistence by occluding them:

$$\begin{aligned}
&Occurs(t_1, t_2, load) \rightarrow \\
&\quad Occlude(t_2, loaded) \wedge Holds(t_2, loaded, true) \\
&Occurs(t_1, t_2, fire) \rightarrow \\
&\quad Holds(t_1, loaded, true) \rightarrow \\
&\qquad Occlude(t_2, loaded) \wedge Holds(t_2, loaded, false) \wedge \\
&\qquad Occlude(t_2, alive) \wedge Holds(t_2, alive, false) \\
&Holds(0, alive, true) \\
&Occurs(1, 2, load) \\
&Occurs(3, 4, fire)
\end{aligned}$$

Without saying anything about when fluents are *not* occluded, the above formulas do not predict the value of *alive*

at time points other than 0, even if we add the persistence formulas $T_{\text{per}}$. We must first perform the predicate completion transformation step, minimizing fluent change by extending the above theory with additional formulas that correspond to the circumscription of *Occlude* and *Occurs*:

$$\begin{aligned}
&Occlude(t,f) \leftrightarrow \\
&\quad f = loaded \wedge \\
&\quad \exists t_1[Occurs(t_1,t,load) \vee \\
&\qquad Occurs(t_1,t,fire) \wedge Holds(t_1,loaded,true)] \vee \\
&\quad f = alive \wedge \\
&\quad \exists t_1[Occurs(t_1,t,fire) \wedge Holds(t_1,loaded,true)] \\
&Occurs(t_1,t_2,action) \leftrightarrow \\
&\quad t_1 = 1 \wedge t_2 = 2 \wedge a = load \vee \\
&\quad t_1 = 3 \wedge t_2 = 4 \wedge a = fire
\end{aligned}$$

The new theory makes it possible to derive non-occlusion deductively. Adding the $T_{\text{per}}$ filter results in the intended consequences, e.g. $\neg Holds(4, alive, false)$.

This transformation works well for research benchmark problems. But the methodology has undesirable properties from the point of view of scalability. The transformation is applied to most of the entire knowledge base and is invalidated whenever some parts of the knowledge base change. A logical agent in a dynamic environment could have even simple queries stymied by potentially expensive computations.

Moreover, while the theorem prover can be used to reason about the consequences of *given* actions, it is not directly applicable to the more fundamental problem of reasoning about which actions to do in the first place. Even considering whether to do an action would require adding that action occurrence to the theory and repeating the transformation.

TALplanner [Kvarnström, 2005] avoids this problem by using special purpose planning algorithms to generate action occurrences. TAL is still used as a semantics for the finished plans, but the planning *process* is metatheoretical. In contrast, the next section will introduce an alternative abductive inference mechanism that naturally extends to planning, resulting in a unified planning and reasoning system without the need for a special purpose planning algorithm.

# 4 Abduction and Filtered Circumscription

Ginsberg [1989] presents a *circumscriptive theorem prover* (CTP) with properties conducive to scalability. The algorithm makes it possible to compute the logical consequences of a circumscribed theory without constructing the second-order circumscription axiom or compiling the theory beforehand. Of course, since circumscription is not even semi-decidable in the general case, some restrictions apply:

1. All formulas are universal, i.e. all its axioms can be written in the form $\forall \vec{x}\, P(\vec{x})$ where $P$ is quantifier free.
2. The theory includes unique names and domain closure axioms.
3. The circumscription policy does not fix predicates.
4. The entire theory is circumscribed.

In the rest of the paper we assume that the theories we are interested in satisfy Restrictions 1 and 2, including only finitely many objects and time points.

Restriction 3 is not satisfied by TAL's circumscription policy as defined in Section 2.1. This, however, is not as troublesome as it might seem. As de Kleer and Konolige have shown [1989], any predicate $P$ can be fixed by simultaneously minimizing both $P$ and $\neg P$. Along with their proof they provide the intuition that this works since any attempt to make $P$ smaller will automatically make $\neg P$ larger, and vice versa. In the end, therefore, the extension of $P$ remains fixed. Using this equivalence we can eliminate the fixation of the *Holds* predicate from TAL's circumscription policy:

$$Circ(T; Occlude, Occurs, Holds, \neg Holds) \wedge T_{\text{per}}$$

Unfortunately, Restriction 4 is not as easily remedied. The formulas belonging to $T_{\text{per}}$ were kept outside the scope of the circumscription for a reason. They were not to affect the minimization of *Occlude* and *Occurs*, while still acting as a filter to remove models in which fluents change without being occluded.

## 4.1 Filtered Circumscription

In order to extend Ginsberg's method to a filtered circumscriptive theorem prover (FCTP) we first note that we can simplify the formulas to which it is applied.

**Lemma 1.** *Regular theorem proving can be used to reserve the FCTP for proving negative literals of minimized predicates, without loss of generality.*

*Proof.* Let $T$ denote a theory and $M$ the set of predicates to be minimized. According to Restriction 1 above, any proof goal $G$ must be universal. If the theory is first put in negation normal form, the following serves as an example of a set of logical equivalences that can be used to reduce the filtered circumscriptive proof goal $G$ to a literal:

$$F, Circ(T;M) \models \forall x P(x) \quad \Leftrightarrow \quad F, Circ(T;M) \models P(c)$$
*Where $c$ does not occur in $P(x)$ nor any premise that $P(c)$ depends on.*

$$F, Circ(T;M) \models P \leftrightarrow Q \quad \Leftrightarrow \quad \begin{cases} F, Circ(T;M) \models P \rightarrow Q \\ F, Circ(T;M) \models Q \rightarrow P \end{cases}$$

$$F, Circ(T;M) \models P \wedge Q \quad \Leftrightarrow \quad \begin{cases} F, Circ(T;M) \models P \\ F, Circ(T;M) \models Q \end{cases}$$

$$F, Circ(T;M) \models P \vee Q \quad \Leftrightarrow \quad F, Circ(T;M) \models \neg P \rightarrow Q$$

$$F, Circ(T;M) \models P \rightarrow Q \quad \Leftrightarrow \quad F, Circ(T;M), P \models Q$$

The only remaining case is when $G$ is a literal $(\neg)P$. Proposition 12 in [Lifschitz, 1994] tells us that if $(\neg)P$ is positive w.r.t. $M$ (or is not one of the predicates in $M$), then $Circ(T;M) \models (\neg)P$ iff $T \models (\neg)P$, in which case we can continue using regular first-order theorem proving. Thus we need only resort to the FCTP when trying to prove literals that are negations of minimized predicates. □

While Ginsberg's implementation is based on an assumption-based truth maintenance system [de Kleer, 1986], the CTP algorithm can now be formulated in terms of abduction. Let $T$ denote our theory, $M$ be the set of predicates

to be minimized, and the goal formula $G$ be the negation of a predicate in M. The CTP then corresponds to the following algorithm [Brewka *et al.*, 1997]:

1. Let the set of abducibles be negations of predicates in $M$.
2. Abduce an explanation $E$ for the goal $G$.
3. Check that there is no counter-explanation of $E$, i.e. that there is no explanation $CE$ for $\neg E$.

This can be reformulated in terms of an inference rule. Explanations $E$ and counter-explanations $CE$ are conjunctions of ground abducible literals from Step 1 of the abductive algorithm. (Since they are the result of abductive proof, we always require them to be consistent with the theory $T$.) Step 2 of the algorithm is represented by the rule's premise, Step 3 by the rule's qualification, and the fact that the algorithm computes circumscription is stated by the rule's conclusion:

$$\frac{T, E \models G}{Circ(T;M) \models G}\ \text{CTP}$$

*Where there is no CE consistent with T such that T,CE* $\models \neg E$.

Ginsberg [1989] shows this sound and complete for circumscription. Furthermore, as we prove next, the only explanation we need is the goal itself.

**Lemma 2.** *When G is the negation of a predicate in M, the CTP can use* $E = G$ *without loss of generality.*

*Proof.* Constraining the set of explanations $E$ does not affect soundness. Let us consider completeness. Using a stronger explanation than $E = G$ would gain us nothing since it can only decrease the applicability of the CTP. Assume a weaker explanation $T, E' \models G$. Then $T \models E' \rightarrow G$ and (because $G = E$) $T \models E' \rightarrow E$. Since $E'$ is weaker than $E$, we know $E \rightarrow E'$, and consequently $T \models E \leftrightarrow E'$. □

We want to extend this to filtered circumscription, which adds a *filter* formula $F$. Since the filter is outside the scope of circumscription, it should not invalidate any conclusion drawn from the original theory by the inference rule above. However, it might allow us to draw new conclusions. As an intermediate step, we reformulate the CTP so that any counter-explanations consistent with $T$ are listed explicitly in the conclusion.

**Lemma 3.** *The following inference rule is sound and complete for circumscription:*

$$\frac{\begin{array}{l} T, E \models G \\ T, CE_1 \models \neg E \\ \vdots \\ T, CE_n \models \neg E \end{array}}{Circ(T;M) \models \neg CE_1 \wedge \cdots \wedge \neg CE_n \rightarrow G}$$

*Where* $CE_1, \ldots, CE_n$ *are all minimal counter-explanations consistent with T.*

*Proof.* Completeness follows since if the CTP can be used to prove $G$, there are no consistent counter-explanations, and the implication antecedent collapses to true. To prove soundness, assume that we can use the above rule to prove $G$. It must be the case that $Circ(T;M) \models \neg CE_1 \wedge \cdots \wedge \neg CE_n$. Since each $\neg CE_i$ is a disjunction of minimized predicates, and circumscription never makes the extension of a minimized predicate larger, we have $T \models \neg CE_i$. But the rule assumes every $CE_i$ consistent with $T$. This is only possible if $n = 0$, in which case $G$ follows from the original CTP.

Note that it suffices to consider minimal counter-explanations. Suppose that $CE_i \subset CE_i'$. If we can prove $\neg CE_i$, then we can also prove the weaker condition $\neg CE_i'$.

□

The new rule makes it clear that when there *are* counter-explanations consistent with $T$, these could become inconsistent after adding the filter $F$, and the implication used to conclude $G$ after all. A naïve implementation could simply add all (finitely many) implications produced by the above proof rule to $T$, creating a first-order equivalent[1] of $Circ(T;M)$, and append $F$. By running a sound and complete theorem prover one could derive $G$ using Modus Ponens on the implications whose antecedent counter-explanations are inconsistent with this new filtered circumscription $F$,$Circ(T;M)$:

$$\frac{\begin{array}{l} Circ(T;M) \models \neg CE_1 \wedge \cdots \wedge \neg CE_n \rightarrow G \\ F, Circ(T;M) \models \neg CE_1 \wedge \cdots \wedge \neg CE_n \end{array}}{F, Circ(T;M) \models G}\ \text{MP}$$

It would, however, be very inefficient to add all implications when we only care about those implications that are relevant in a proof of $G$. Instead, we can get exactly the same result by adding an FCTP inference rule that allows us to conclude $G$ directly, whenever all counter-explanations consistent with $T$ are inconsistent with the filtered circumscription $F$,$Circ(T;M)$:

$$\frac{\begin{array}{l} T, E \models G \\ T, CE_1 \models \neg E \\ \vdots \\ T, CE_n \models \neg E \\ F, Circ(T;M) \models \neg CE_1 \\ \vdots \\ F, Circ(T;M) \models \neg CE_n \end{array}}{F, Circ(T;M) \models G}\ \text{FCTP}$$

*Where* $CE_1, \ldots, CE_n$ *are all minimal counter-explanations consistent with T.*

**Theorem 1.** *The FCTP inference rule is sound and complete for filtered circumscription.*

[1] It is always possible to construct a first-order equivalent of $Circ(T;M)$ given Ginsberg's assumption of universal theories with unique names and domain closure axioms.

*Proof.* Each proof produced by the naïve algorithm corresponds to a proof using the FCTP rule, obtained by replacing applications of Modus Ponens on one of the added implications by an application of the FCTP rule. Likewise, any application of the FCTP rule can be replaced by an implication and an application of Modus Ponens of the naïve algorithm. □

## 5 Examples

Let us use some TAL reasoning problems to illustrate the use of the FCTP. The proofs are abbreviated compared to the output of the implementation described in Section 6. E.g., we use the same (incremental and interruptible) consistency checking mechanism as Poole's THEORIST [1991] but do not display these steps below. All of the examples refer to the following filter formula as $F$:

$$\neg Occlude(t+1, f) \rightarrow (Holds(t, f, v) \leftrightarrow Holds(t+1, f, v))$$

Consider first the simplest case of fluent persistency through $F$. The fluent *alive* should persist from 0 to 1:

| | | | |
|---|---|---|---|
| {1} | 1 | $Holds(0, alive, true)$ | |
| {} | 2 | *Show* $Holds(1, alive, true)$ | |

The only way to show this is to use the filter $F$'s persistence axiom:

| | | | |
|---|---|---|---|
| {} | 3 | *Show* $\neg Occlude(1, alive)$ | $F, 1, 2$ |

While none of the given formulas entail non-occlusion, we can apply the FCTP. First, $\neg Occlude(1, alive)$ can be used as its own explanation:

| | | | |
|---|---|---|---|
| {4} | 4 | $\neg Occlude(1, alive)$ | 3 |

Next, we must find all counter-explanations consistent with $T$, i.e. all abductive explanations of $Occlude(1, alive)$ using $T$:

| | | | |
|---|---|---|---|
| {} | 5 | *Show* $Occlude(1, alive)$ | 3 |

Given the theory $T$ consisting of Row 1, it is impossible to prove $Occlude(1, alive)$. Consequently there are no counter-explanations and the proof succeeds:

| | | | |
|---|---|---|---|
| {1,4} | 6 | $Holds(1, alive, true)$ | $F, 1, 4$ |

The following example illustrates how the simultaneous minimization of *Holds* and $\neg$*Holds* can provide counter-examples that prevent credulous conclusions in the case of incomplete information of *loaded* in the initial state:

| | | | |
|---|---|---|---|
| {1} | 1 | $Holds(0, alive, true)$ | |
| {2} | 2 | $Holds(0, loaded, true) \rightarrow Occlude(1, alive)$ | |
| {} | 3 | *Show* $Holds(1, alive, true)$ | |
| {} | 4 | *Show* $\neg Occlude(1, alive)$ | $F, 1, 3$ |
| {5} | 5 | $\neg Occlude(1, alive)$ | 4 |
| {} | 6 | *Show* $Occlude(1, alive)$ | 4 |
| {} | 7 | *Show* $Holds(0, loaded, true)$ | 2, 6 |
| {8} | 8 | $Holds(0, loaded, true)$ | 7 |
| {2,8} | 9 | $Occlude(1, alive)$ | 2, 8 |

Since any attempt to apply FCTP recursively to prove $Holds(0, loaded, false)$ fails, we have a consistent Row 8 that counter-explains Row 5, and the proof fails. In other words, since it is possible that the gun is loaded, it is not safe to rely on the persistence of alive.

Here is an example in which Ginsberg's CTP does not give the expected result due to TAL's filtered circumscription:

| | | |
|---|---|---|
| {1} | 1 | $Holds(0, alive, true)$ |
| {2} | 2 | $Holds(0, loaded, false)$ |
| {3} | 3 | $Holds(1, loaded, true) \rightarrow Occlude(2, alive)$ |
| {} | 4 | *Show* $Holds(2, alive, true)$ |

The goal follows if *alive* is not occluded. The first invocation of FCTP comes up with the explanation for $\neg Occlude(2, alive)$ in Row 6, but also a potential counter-explanation for $Occlude(2, alive)$ consistent with $T$ in Row 9:

| | | | |
|---|---|---|---|
| {} | 5 | *Show* $\neg Occlude(2, alive)$ | $F, 1, 4$ |
| {6} | 6 | $\neg Occlude(2, alive)$ | 5 |
| {} | 7 | *Show* $Occlude(2, alive)$ | 5 |
| {} | 8 | *Show* $Holds(1, loaded, true)$ | 3, 7 |
| {9} | 9 | $Holds(1, loaded, true)$ | 8 |

A recursive invocation of FCTP attempts to disprove the counter-explanation by showing that its negation (where $\neg Holds(t, f, true) \leftrightarrow Holds(t, f, false)$) follows from *F,Circ*(*T*;*M*):

| | | | |
|---|---|---|---|
| {} | 10 | *Show* $Holds(1, loaded, false)$ | 8 |
| {} | 11 | *Show* $\neg Occlude(1, loaded)$ | $F, 2, 9$ |
| {12} | 12 | $\neg Occlude(1, loaded)$ | 11 |
| {} | 13 | *Show* $Occlude(1, loaded)$ | 11 |
| {2,12} | 14 | $Holds(1, loaded, false)$ | $F, 2, 12$ |

This proof succeeds in Row 14 since the explanation in Row 12 is not counter-explained. Row 9 is thus not consistent with the *filtered* circumscription and the original conclusion follows after all:

| | | | |
|---|---|---|---|
| {} | 15 | *Show* $\neg Occlude(1, alive)$ | $F, 1, 4$ |
| {16} | 16 | $\neg Occlude(1, alive)$ | 15 |
| {} | 17 | *Show* $Occlude(1, alive)$ | 15 |
| {1,16} | 18 | $Holds(1, alive, true)$ | $F, 1, 16$ |
| {1,2,6,12,16} | 19 | $Holds(2, alive, true)$ | $F, 6, 18$ |

Finally, let us consider a disjunctive example. Suppose that activating a lamp either causes a change to the bulb's lit state or its broken state:

| | | |
|---|---|---|
| {1} | 1 | $Occurs(t_1, t_2, activate) \rightarrow Occlude(t_2, lit) \vee Occlude(t_2, broken)$ |
| {2} | 2 | $Occurs(0, 1, activate)$ |

Predicate completion requires that the theory can be put in the form $\Phi(t, f) \rightarrow Occlude(t, f)$ where $\Phi(t, f)$ does not contain occurrences of *Occlude*. But this is not possible given the above disjunctive action effect. TAL's predicate completion has been applied to actions with non-deterministic effects, but never when the occlusion itself is non-deterministic. The FCTP, however, has no problems proving e.g. that one of the fluents will not be occluded:

| | | | |
|---|---|---|---|
| $\{\}$ | 3 | *Show* $\neg Occlude(1,lit) \vee \neg Occlude(1,broken)$ | |
| $\{4\}$ | 4 | $Occlude(1,lit)$ | 3 |
| $\{4\}$ | 5 | *Show* $\neg Occlude(1,broken)$ | 3 |
| $\{4,6\}$ | 6 | $\neg Occlude(1,broken)$ | 5 |
| $\{4\}$ | 7 | *Show* $Occlude(1,broken)$ | 5 |

The equivalences in Section 4.1 reduce the problem using the assumption in Row 4 and the new goal in Row 5, which has an explanation in Row 6.

From the action and its occurrence in Row 1 and 2 we know that one of the fluents are occluded. A counter-explanation to Row 6 is therefore $\neg Occlude(1,lit)$:

| | | | |
|---|---|---|---|
| $\{1,2\}$ | 8 | $Occlude(1,lit) \vee Occlude(1,broken)$ | $1,2,7$ |
| $\{4\}$ | 9 | *Show* $\neg Occlude(1,lit)$ | $7,8$ |

However, the assumption in Row 4 is not part of the abductive explanation. It was introduced by the previous goal reduction and is still in force when trying to prove the counter-explanation. Consequently, assuming $\neg Occlude(1,lit)$ would be inconsistent, the only counter-explanation fails, and the conclusion follows:

| | | | |
|---|---|---|---|
| $\{1,6\}$ | 10 | $\neg Occlude(1,lit) \vee \neg Occlude(1,broken)$ | $3,6$ |

Given the same action specification and action occurrence, one of the fluents has to be occluded at time 1. Attempting to prove both not occluded fails:

| | | | |
|---|---|---|---|
| $\{\}$ | 3 | *Show* $\neg Occlude(1,lit) \wedge \neg Occlude(1,broken)$ | |
| $\{\}$ | 4 | *Show* $\neg Occlude(1,lit)$ | 3 |
| $\{5\}$ | 5 | $\neg Occlude(1,lit)$ | 4 |
| $\{\}$ | 6 | *Show* $Occlude(1,lit)$ | 4 |
| $\{1,2\}$ | 7 | $Occlude(1,lit) \vee Occlude(1,broken)$ | $1,2,6$ |
| $\{\}$ | 8 | *Show* $\neg Occlude(1,broken)$ | $6,7$ |
| $\{9\}$ | 9 | $\neg Occlude(1,broken)$ | 8 |
| $\{1,2,9\}$ | 10 | $Occlude(1,lit)$ | $7,9$ |

This time $\neg Occlude(1,broken)$ is a consistent counter-explanation to Row 5 since there is no way to prove that its negation follows. The proof of the conjunction fails since the proof of the first conjunct fails. The failure would repeat if trying to prove the second conjunct first.

# 6 Unified Planning and Reasoning

Let us turn now to the task of implementing a planning and reasoning system based on the theory presented above. A commonly used implementation tool is logic programming. Indeed, earlier work with TAL made planning and reasoning possible through a compilation of TAL formulas into Prolog programs [Magnusson, 2007]. Proofs were *deductive* and instantiated a plan variable, similarly to the instantiation of the situation term in deductive Situation Calculus planning. Other work extends Prolog's inference mechanism to *abduction* by means of a meta-interpreter. This has been the de facto standard in work on abductive planning in Event Calculus, e.g. in [Shanahan, 2000; Denecker *et al.*, 1992; Missiaen *et al.*, 1995].

## 6.1 Pattern-Directed Inference System

We have explored a different avenue with a theorem prover based on *natural deduction*, inspired by similar systems by Rips [1994], Pelletier [1998], and Pollock [2000]. This is an interesting alternative to the more common resolution method used by most theorem provers, including Prolog. A natural deduction prover works with the formulas of an agent's knowledge base in their "natural form" directly, rather than first compiling them into clause form. This fits perfectly with the algorithm in Section 4 that has already eliminated the need for a compilation step for nonmonotonic reasoning.

The system uses pattern-directed inference similar to Forbus and de Kleer's fast tiny rule engine [1993]. To see how this works let us look at the inference rules. Applicable rules are added to a queue. By controlling which rule application the prover selects next we can implement e.g. depth-first, breadth-first, or best-first search.

Rules are divided into forward and backward rules. Forward rules are triggered whenever possible. They are therefore designed to be convergent, so as not to generate new inferences forever. An example is the modus ponens rule, which concludes $Q$ whenever both $P$ and $P \rightarrow Q$ are present in the knowledge base. The results in this paper generalizes our previous work that relied on forward rules to implement an incomplete consistency check [Magnusson, 2007; Magnusson and Doherty, 2008a; 2008b]. By explicitly trying to counter-explain abductive assumptions we no longer have to rely on forward rules being strong enough to detect inconsistent assumptions.

Backward rules, in contrast, are used to search backwards from the current proof goal and thus exhibit goal direction. An example is the goal chaining rule, which adds *Show* $P$ as a new goal whenever both *Show* $Q$ and $P \rightarrow Q$ are present in the knowledge base.

Combining forward and backward rules results in a bidirectional search for proofs that is pattern-directed since the prover's current goals are explicitly represented (by *Show* formula "patterns") in the knowledge base. This further contributes to the incremental nature of the reasoner. Inference can be interrupted at any time and later resumed since the knowledge base keeps track of what the prover was about to do.

## 6.2 Abductive Planning

The same reasoning mechanism can be used for abductive planning. Instead of reasoning about the effects of a given narrative, we reason about what narrative would have the desired effects.

Given a domain, we define a planning goal $G$ as a conjunction of ground *Holds* literals. A plan for $G$ is then a set $T_{\text{occ}}$ of ground atomic *Occurs* formulas such that:

$$T_{\text{per}} \wedge Circ(T \wedge T_{\text{occ}}; Occlude, Occurs, Holds, \neg Holds) \models G$$

where the left hand side is consistent.

To generate $T_{\text{occ}}$ we simply add *Occurs* to the set of abducibles for the proof goal. (Note however that we do not add *Occurs* as an abducible to explanations and counter-explanations. Doing so would amount to planning to thwart

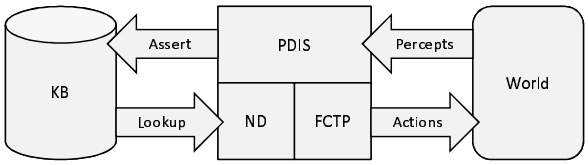


Figure 1: A logical agent architecture.

our own plans!) The soundness of this planning method follows directly from the soundness of the FCTP. Its implementation is already implicit in the theorem prover described above.

### 6.3 Logical Agents

Using the same reasoning mechanisms for many problems faced by autonomous agents results in a particularly simple agent architecture, as illustrated by Figure 1. The agent is equipped with a knowledge base containing formulas encoding knowledge about actions, world laws, and memory of recent events and percepts. The knowledge is used by the pattern-directed inference system (PDIS), with the help of natural deduction (ND) and the abductive algorithm from Section 4, to plan its actions.

But when plans meet the World they often fail. Executing a plan will achieve the goal only if the plan's assumptions hold up. The agent can detect some failures early through execution monitoring. In particular, persistence assumptions are represented in the plan by non-occlusion assumptions and can be continually evaluated. When a failure is perceived, that percept constitutes a counter-explanation to the assumption. Neither the assumption nor the planning goal derived from it are justified conclusions given the new percept. This immediately makes the pattern-directed goal-chaining inference rules applicable in trying to find an alternative proof of the goal. The result is an automatic plan revision and failure recovery process as the agent uses abductive planning to reestablish goals that lost their justification and execute an alternative plan.

## 7 Discussion

Predicate completion and the filtered circumscriptive theorem prover (FCTP) are two methods for automated reasoning in logics that use filtered circumscription. E.g., they both satisfy the circumscriptive characterization of TAL and produce the same end result, given the restrictions in Section 4. But in practical problem solving we believe the FCTP to have a number of desirable properties and advantages.

### 7.1 Abductive Planning and Reasoning

The FCTP uses abductive proof methods to reason with the original TAL formulas directly. As noted by Brewka, Dix, and Konolige [1997]:

> Abduction offers several benefits from a knowledge representation viewpoint. It does not require the assumption of complete knowledge of causation, and it is not necessary to assert the explanatory closures (which can lead to inconsistency and is computationally discouraging since it is performed globally on the theory).

Since the method does not presume complete knowledge of action occurrences, applications in planning open up. We exemplified this by using the reasoner for abductive planning. Furthermore, this unification of planning and reasoning forms the basis of a logical agent architecture that is highly capable despite its simplicity.

### 7.2 Reasoning Without Compilation

Predicate completion works by compiling the theory into a first-order equivalent with which reasoning proceeds. Lifschitz [1994] comments:

> But it should be observed that this approach to the automation of circumscription is not the only one possible. In fact, it may be unattractive, in view of the fact that it requires preprocessing the circumscription in a manner that is not related in any way to the goal formula.

The FCTP reasons with the TAL formulas directly, without first transforming the theory. Its efforts are spent only on those formulas of the knowledge base that are potentially relevant to the goal. The resulting proofs are also easier to comprehend since they refer directly to the formulas that were given to the system as input. Comprehension is also improved by the mechanism's similarity to argumentation and thereby to human reasoning. It would be interesting to further investigate the relation between FCTP and argumentation-theoretic systems.

### 7.3 Doubly Defeasible Reasoning

Finally, let us adopt a long term view and consider logical agents with commonsense knowledge. With the manual development or automated learning of very large knowledge bases, which are presumably needed for commonsense reasoning, it will be impractical or even impossible to search through all conceivable counter-explanations to a defeasible inference before taking action. It becomes necessary to consider what Pollock [2008] refers to as "doubly defeasible" reasoning. Not only can the reasoner change its mind with new information, it can also change its mind with more time to reason with its current information.

Predicate completion forces the agent to prove conclusions deductively in a first-order equivalent to the circumscribed theory. There is no way to interrupt the reasoning and act to the best of one's current knowledge. The incremental nature of the FCTP makes this possible. If counter-explanations are tested in the order of their likelyhood, it implements, in effect, an any-time algorithm that is always able to respond with the current best answer.

## 8 Conclusion

We are interested in building logical agents that use knowledge and reasoning to achieve goals. Observing that the world is both uncertain and dynamic motivates our choice of reasoning mechanisms that are incremental in nature. The

computational effort of pondering a question should be related to the extent of relevant knowledge and the time available, not the total size of the knowledge base nor a potentially unbounded time requirement. Only then will the technology have the potential to scale up to very large knowledge bases.

One step in this direction is reported here in our investigation of Temporal Action Logic and its application in a logical agent architecture. By extending Ginsberg's circumscriptive theorem prover we have made it applicable to logics defined in terms of filtered circumscription. The abduction-based filtered circumscriptive theorem prover reasons directly with the input formulas, removing the need for a compilation step involving the entire knowledge base. Its interruptible nature enables an agent to act to the best of its knowledge given only limited time for reasoning. Finally, its double duty as an abductive planner makes possible a particularly simple agent architecture.

An agent architecture based exclusively on logical reasoning will necessarily suffer somewhat in efficiency compared to less general methods, despite being designed with scalability in mind. But achieving satisfactory performance in certain domains is already possible. E.g., we have applied the architecture to the control of computer game characters that require real-time interaction [Magnusson and Doherty, 2008b]. We believe computer games to be an excellent domain for empirical studies of logical agents on the road from tiny benchmark problems towards larger real world applications.

## Acknowledgements

We thank Andrzej Szalas for helpful suggestions.

## References

[Brewka *et al.*, 1997] Gerhard Brewka, Jürgen Dix, and Kurt Konolige. *Nonmonotonic Reasoning: An Overview*, volume 73 of *CSLI Lecture Notes*. CSLI Publications, 1997.

[Clark, 1978] Keith L. Clark. Negation as failure. In *Logic and Data Bases*, pages 293–322, 1978.

[de Kleer and Konolige, 1989] Johan de Kleer and Kurt Konolige. Eliminating the fixed predicates from a circumscription. *Artificial Intelligence*, 39(3):391–398, 1989.

[de Kleer, 1986] Johan de Kleer. An assumption-based TMS. *Artificial Intelligence*, 28(2):127–162, 1986.

[Denecker *et al.*, 1992] Marc Denecker, Lode Missiaen, and Maurice Bruynooghe. Temporal reasoning with abductive event calculus. In *Proc. of ECAI'92*, pages 384–388, 1992.

[Doherty and Kvarnström, 2007] Patrick Doherty and Jonas Kvarnström. Temporal action logics. In *Handbook of Knowledge Representation*. Elsevier, 2007.

[Doherty and Lukaszewicz, 1994] Patrick Doherty and Witold Lukaszewicz. Circumscribing features and fluents: A fluent logic for reasoning about action and change. In *Proc. of ISMIS'94*, pages 521–530, 1994.

[Doherty, 1996] Patrick Doherty. PMON+: A fluent logic for action and change. Formal specification, version 1.0. Technical report, Linköping University, 1996.

[Forbus and de Kleer, 1993] Kenneth D. Forbus and Johan de Kleer. *Building Problem Solvers*. MIT Press, 1993.

[Ginsberg, 1989] Matthew L. Ginsberg. A circumscriptive theorem prover. In *Nonmonotonic reasoning*, pages 100–114. Springer, 1989.

[Kvarnström, 2005] Jonas Kvarnström. *TALplanner and Other Extensions to Temporal Action Logic*. PhD thesis, Linköping University, 2005.

[Lifschitz, 1994] Vladimir Lifschitz. Circumscription. In *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 298–352. Oxford University Press, New York, NY, USA, 1994.

[Magnusson and Doherty, 2008a] Martin Magnusson and Patrick Doherty. Deductive planning with inductive loops. In *Proc. of KR 2008*, pages 528–534, 2008.

[Magnusson and Doherty, 2008b] Martin Magnusson and Patrick Doherty. Logical agents for language and action. In *Proc. of AIIDE-08*, 2008.

[Magnusson, 2007] Martin Magnusson. *Deductive Planning and Composite Actions in Temporal Action Logic*. Licentiate thesis, Linköping University, 2007.

[Missiaen *et al.*, 1995] Lode Missiaen, Maurice Bruynooghe, and Marc Denecker. CHICA, an abductive planning system based on event calculus. *Journal of Logic and Computation*, 5(5):579–602, 1995.

[Pelletier, 1998] Francis Jeffry Pelletier. Natural deduction theorem proving in THINKER. *Studia Logica*, 60:3–43, 1998.

[Pelletier, 1999] Francis Jeffry Pelletier. A brief history of natural deduction. *History and Philosophy of Logic*, 20:1–31, 1999.

[Pollock, 2000] John Pollock. Rational cognition in OSCAR. In *Intelligent Agents VI: Agent Theories, Architectures, and Languages*, volume 1757 of *Lecture Notes in AI*, pages 71–90. Springer Verlag, 2000.

[Pollock, 2008] John L. Pollock. OSCAR: An architecture for generally intelligent agents. In *Proc. of AGI 2008*, pages 275–286. IOS Press, 2008.

[Poole, 1991] David Poole. Compiling a default reasoning system into prolog. *New Generation Computing*, 9(1):3–38, 1991.

[Rips, 1994] Lance J. Rips. *The psychology of proof: deductive reasoning in human thinking*. MIT Press, 1994.

[Sandewall, 1994] Erik Sandewall. *Features and Fluents: The Representation of Knowledge about Dynamical Systems*, volume 1. Oxford University Press, 1994.

[Shanahan, 1997] Murray Shanahan. *Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia*. MIT Press, 1997.

[Shanahan, 2000] Murray Shanahan. An abductive event calculus planner. *The Journal of Logic Programming*, 44:207–239, 2000.