The following proposition can be used to improve the search efficiency.

Proposition 4.5 *Assume $N = holds(F_n, T_n)$ is new knowledge. Let δ be any abductive answer to N in $KB_{occurs}(D, H)$. Then, for any $occurs(A, T) \in \delta$ we have $T \prec T_n$.*

Thus it is not necessary to consider predicate instances $occurs(A, T)$ for $T \not\prec T_n$.

In general, when concurrent actions are taken into account, it is very time-consuming to find all possible causes by abductive logic programs. Note that we have assumed that Σ_a is finite, and thus the program can always terminate in a finite time as it is not necessary to consider predicate instances $occurs(A, T)$ for $T \not\prec T_n$. When we don't consider concurrent actions, we may have a few other techniques to improve the efficiency as discussed in [9]. More techniques for efficiency improvement in the domains of concurrent actions are still under development.

Acknowledgements

This work was in part supported by a post-doctoral fellowship from JNICT under PRAXIS XXI/BPD/4165/94 to the first author, and a JNICT PROLOPPE project under PRAXIS/3/31/TIT/24/94.

References

[1] Alchorrón, C., Gärdenfors, P. and Makinson, D., On the logic of theory change: Partial meet contraction and revision functions, *J. of Symbolic Logic*, 50:2, 1985, 510 – 530

[2] Apt, K.R. and Bezem, M., Acyclic programs, *Proc. of ICLP 90*, MIT Press, 579–597

[3] Dalal, M., Investigations into a theory of knowledge base revision: Preliminary report, *AAAI 88*, 475 – 479

[4] Denecker, M. and de Schreye, D., Representing incomplete knowledge in abductive logic programming, *Logic Programming: Proc. of the 1993 Int'l Symposium*, 1993, 147–163

[5] Gelfond, M. and Lifschitz, V., Representing action and change by logic programs, *Journal of Logic Programming*, Vol.17, 1993, 301–322

[6] Ginsberg, M., Counterfactuals, *Artificial Intelligence*, 30:1, 1986, 35 – 79

[7] Kautz, H.A., The logic of persistence, *Proc. of the AAAI86*, 1986, 401–405

[8] Li, R. and Pereira, L. M., Temporal reasoning with abductive logic programming, *ECAI'96, European Conf. on Artificial Intelligence*, Budapest, August 12-16,1996

[9] Li, R. and Pereira, L. M., What is believed is what is explained (sometimes), *AAAI'96-American Thirteenth National Conf. on Artificial Intelligence*, Portland, Oregon, August 4 - 8, 1996

[10] Nebel, B., Belief revision and default reasoning: Syntax-based approaches, *Proc. of KR'91*, 1991

[11] Winslett, M., Reasoning about action using a possible models approach, *AAAI'88*, 1988, 89 – 93

Definition 4.3 *Given a knowledge base $KB(D,H)$, and new facts N of the forms $is_true(F,T)$ and $is_false(F,T)$, all the possible new knowledge bases $KB(D,H) \oplus N$ are defined to be $\{KB(D,H) \dot{+} \delta : \delta \in KB(D,H) \uparrow N\}$.*

In the Stolen Car Problem, since there are three possible causes for $holds(stolen,3)$, then $KB(D_{scp}, H_{scp}) \oplus holds(stolen,3)$ can be any of the following three new knowledge bases: $\{KB(D_{scp}, \{(steal,0)\}), KB(D_{scp}, \{(steal,1)\}), KB(D_{scp}, \{(steal,2)\})\}$.

Note that the new knowledge N is not directly added to the knowledge base. Instead, its causes are added. If we want to add N as well as its causes, then for each fact $is_true(F,T)$ and $is_false(G,T)$ in N, we can add the following two constraints, respectively:

$$\begin{aligned} false &\leftarrow not\ is_true(F,T) \\ false &\leftarrow not\ is_false(G,T) \end{aligned}$$

Then, N is kept permanently in the knowledge base. Note that a possible cause δ also satisfies the condition that $COMP(KB(D,H)) \cup \{E : E \in \delta\}$ is consistent. Thus, any further expansion of initial state conditions and history won't falsify N for it to be a possible cause, since $false$ is always interpreted to be falsity and the above two rules function as integrity constraints.

The PCA operator $\oplus$ can also be used to evaluate counterfactual queries: if F_1 were true at time T_1, would F_2 be true at time T_2? This amounts to evaluate the query "$? - is_true(F_2,T_2)$" in $KB(D,H) \oplus is_true(F_1,T_1)$.

Now consider another example: a door-light domain and history are described as follows:

$$D_{door} = \left\{ \begin{array}{l} \textbf{initially}\ \ \neg Closed \\ \textbf{initially}\ \ On \\ Open\ \textbf{causes}\ \ \neg Closed \\ Close\ \textbf{causes}\ \ Closed \\ Switch\ \textbf{causes}\ \ On\ \textbf{if}\ \ \neg On \\ Switch\ \textbf{causes}\ \ \neg On\ \textbf{if}\ \ On \end{array} \right\}, \qquad H_{door} = \emptyset$$

Suppose that we have $N = \{is_true(Closed,1), is_false(On,1)\}$. Then, there will only be a unique possible cause: the action $Open$ and $Switch$ has happened at the same time, i.e. $\delta = \{happens(\{Open, Switch\}, 0)\}$. This example shows that this paper has enlarged the range of applicability of the PCA approach initially proposed in [9].

The critical step in the process of temporal knowledge update is the computation of the possible causes. Now we show how to compute possible causes with abductive logic programming.

First of all we introduce a new predicate $occurs(A,T)$, for action expression A and time T, and add a new programming rule to $KB(D,H)$:

$$happens(A,T) \leftarrow occurs(A,T) \tag{2}$$

Now we define our abducible predicates to be $initially_true(F), initially_false(F)$ and $occurs(A,T)$. The above rule means that if action A is abduced to occur at time T, it happens at T. Let $KB_{occurs}(D,H) = KB(D,H) \cup \{(2)\}$. It is easy to see that KB_{occurs} is still acyclic. It can also be shown that possible causes for N become abductive answers to N in $KB_{occurs}(D,H)$.

Proposition 4.4 *δ is a possible cause for new facts N in $KB(D,H)$ iff*

$$(\delta \setminus \{happens(A,T) : happens(A,T) \in \delta\}) \cup \{occurs(A,T) : happens(A,T) \in \delta\}$$

is an abductive answer to N in the abductive logic program $KB_{occurs}(D,H)$.

Suppose that $N = is_true(F,T)$ is a newly observed fact. In the process of incorporating N into $KB(D,H)$, first we should check whether N is already stored in $KB(D,H)$, i.e., whether $COMP(KB(D,H)) \models N$. If it is so, we will do nothing. In this case, $KB(D,H) \oplus N \equiv KB(D,H)$. Suppose that N is not stored in the knowledge base, that is, $COMP(KB(D,H)) \not\models N$. In this case, either the fluent F was true and inertial at the last time point or an action has happened at the last time point and a initiates F. After a careful analysis, it can be seen that the v-propositions **initially** F and occurrences of actions which initiate N or recursively initiate preconditions of actions which initiate N are possible causes for N in $KB(D,H)$.

Let (D,H) be a domain evolution. Let Δ be the set of instances of $initially_true(F)$, $initially_false(F)$, and $happens(A,T)$ for fluent F, action expression A and time T. For any finite subset δ of Δ we will write $KB(D,H) \dot{+} \delta$ to stand for $KB(D',H')$, where $D' = D \cup \{$ **initially** $F : initially_true(F) \in \delta\} \cup \{$ **initially** $\neg F : initially_false(F) \in \delta\}$ and H' is defined as follows: Let $X = H \cup \{(A,T) : happens(A,T) \in \delta\}$. Repeat the following process: if there are different actions A and B, and time T such that $(A,T) \in X$ and $(B,T) \in X$, then let $X1 = (X \setminus \{(B,T)\} \setminus \{(A,T)\}) \cup \{(\{A,B\},T)\}$. If $X1 = X$, then stops, otherwise let $X = X1$ and repeat the above process. Since both H and δ are finite, the above process will terminate in a finite number steps. We will often omit $initially_true(F)$ from δ when both **initially** $F \in D$ and $initially_true(F) \in \delta$, and omit $initially_false(F)$ from δ when both **initially** $\neg F \in D$ and $initially_false(F) \in \delta$.

Let $\delta_1, \delta_2 \in \Delta$. We say that δ_1 is smaller than δ_2 iff either $\delta_1 \subset \delta_2$, or there are $(A,T) \in \delta_1$ and $(B,T) \in \delta_2$ such that (i) $\delta_1 \setminus \{(A,T)\}$ is smaller than $\delta_2 \setminus \{(B,T)\}$ and A is a proper subaction of B.

Definition 4.1 *Let (D,H) be a domain evolution, and N a set of literals of the forms $is_true(F,T)$ and $is_false(F,T)$. A possible cause of N in $KB(D,H)$ is a finite subset δ of Δ such that (i) $KB(D,H) \dot{+} \delta \models N$; (ii) $KB(D,H) \dot{+} \delta$ is consistent; (iii) δ is minimal in the sense that no smaller δ' than δ exists such that δ' satisfies the first two conditions. We will write $KB(D,H) \uparrow N$ to stand for all the possible causes for N in $KB(D,H)$.*

As an example, consider the Stolen Car Problem [7], whose domain description $D_{scp} = \{$ **initially** $\neg Stolen.$ $Steal$ **causes** $Stolen.\}$ and the history $H_{scp} = \emptyset$. Suppose that the car is found missing at time 3, i.e., $is_true(stolen,3)$. Then there are three possible causes of $is_true(stolen,3)$ in $KB(D_{scp}, H_{scp})$:

$$\delta_1 = \{happens(steal,0)\}, \quad \delta_2 = \{happens(steal,1)\}, \quad \delta_3 = \{happens(steal,2)\}$$

In order to incorporate N into $KB(D,H)$, if there is a possible cause δ for N in $KB(D,H)$, then we can simply add δ into the knowledge base to have a new knowledge base. Now we have to question about whether there is always a possible cause δ for N in $KB(D,H)$. The answer turns out to be negative.

Proposition 4.2 *Let D be a domain description, and H a history. Let N be a set of facts of the forms $is_true(F,T)$ and $is_false(F,T)$. It does not hold that there is always a possible cause of N in $KB(D,H)$.*

For incorporation of N into $KB(D,H)$, when there is no possible cause for N in $KB(D,H)$, we will simply discard N because we only believe what is explained.

3.2 Soundness and completeness of knowledge bases

Given a history H, we can have a unique sequence of action expressions $A_1; A_2; \ldots; A_m$ and a sequence of time points $T_1 \prec T_2 \prec \ldots \prec T_m$ such that $H = \{(A_i, T_i) : 1 \leq i \leq m\}$. The sequence of action expressions $A_1; A_2; \ldots; A_m$ will be called *the trajectory generated from H, denoted by $Traj(H)$*. A pre-history of H is defined to be a subset H' of H such that $Traj(H')$ is a prefix of $Traj(H)$. Given an interpretation structure $M = (\sigma_0, \Phi)$, a history H and a time point T, we can derive a state S as follows: if $H' = \{(A_1, T_1), \ldots, (A_k, T_k)\}$ is a pre-history of H such that $T_k \prec T$ and there is no time point Tx and action Ax with $T_k \prec Tx \prec T$ and $(Ax, Tx) \in H$, then $S = \Phi(A_1; A_2; \ldots; A_k, \sigma_0)$, called *the state generated from H and T in M*, denoted by $GS_M(H, T)$. We will often write $GS^+_M(H, T)$ and $GS^-_M(H, T)$ to stand for S^+ and S^-, where $GS_M(H, T) = S = \Phi(A_1; A_2; \ldots; A_k, \sigma_0)$. If T is a time point such that $(A, T) \notin H$ for any action A, and $(A, Tx) \notin H$ for any action A and any time point Tx with $T \prec Tx$, then the state $GS_M(H, T)$ is called *the state generated from H in M*, denoted by $GS_M(H)$. For example, let $M = (\sigma_0, \Phi)$ be an interpretation structure, and $H = \{(\{a, b, c\}, 1), (\{a, c\}, 6), (b, 4), (\{b, c\}, 10)\}$ a history. Then, $GS_M(H, 5) = \Phi(\{a, b, c\}; b, \sigma_0)$ and $GS_M(H) = \Phi(\{a, b, c\}; b; \{a, c\}; \{b, c\}, \sigma_0)$.

Note that $\mathcal{AC}$ has an infinite tree-like time structure. Given a history H, the translation π only takes into account a special path in the tree corresponding to H. In order for π to be correct, we require that π be correct with respect to any history.

Definition 3.3 *The logic program $KB(D, H)$ is said to be sound with respect to H iff for any model $M = (\sigma_0, \Phi)$ of D, any fluent name F and any time point T,*

- *if $COMP(KB(D, H)) \models is_true(F, T)$, then $F \in GS^+_M(H, T)$, and*
- *if $COMP(KB(D, H)) \models is_false(F, T)$, then $F \in GS^-_M(H, T)$.*

The translation π is said to be sound iff, for every history H, $KB(D, H)$ is sound with respect to H.

Definition 3.4 *The logic program $KB(D, H)$ is said to be complete with respect to H iff for every fluent name F, and time point T, if $F \in GS^+_M(H, T)$ for every model M of D, then $COMP(KB(D, H)) \models is_true(F, T)$, and if $F \in GS^-_M(H, T)$ for every model M of D, then $COMP(KB(D, H)) \models is_false(F, T)$. The translation π is said to be complete iff, for every history H, $KB(D, H)$ is complete with respect to H.*

Theorem 3.5 (Soundness Theorem) *Let D be a domain description. For any history H, any model $I = (\sigma_0, \Phi)$ of D, any fluent name F, and any time point T, we have that if $COMP(KB(D, H)) \models is_true(F, T)$, then $F \in GS^+_M(H, T)$, and if $COMP(KB(D, H)) \models is_false(F, T)$, then $F \in GS^-_M(H, T)$.*

Theorem 3.6 (Completeness Theorem) *Let D be a domain description. For every history H, every fluent name F, and time point T,if $F \in GS^+_M(H, T)$ for every model M of D, then $COMP(KB(D, H)) \models is_true(F, T)$, and if $F \in GS^-_M(H, T)$ for every model M of D, then $COMP(KB(D, H)) \models is_false(F, T)$.*

4 Possible causes and updates

For easy reference, we will simply write $KB(D, H) \oplus N$ to informally denote all the possible new temporal knowledge bases obtained by incorporating N into $KB(D, H)$ before we formally define it later. For convenience, we often write $N = L$ if $N = \{L\}$.

- For each effect proposition a **causes** $\neg f$ **if** $p_1, \ldots, p_n$, where f is positive, we have a logic programming rule:

$$immediate_terminates(a, f, T) \leftarrow holds(p_1, T), \ldots, holds(p_n, T).$$

6. For each value proposition **initially** f, or **initially** $\neg g$, where f and g are positive fluents, we have

$$\begin{array}{lcl} false & \leftarrow & not\ is_true(f, init) \\ false & \leftarrow & not\ is_false(g, init) \end{array}$$

The above two logic programming rules function as integrity constraints. Note that we cannot abduce a fluent to be both true and false. For this purpose we add the following domain-independent constraint:

$$false \leftarrow is_true(F, init), is_false(F, init). \quad (1)$$

The literal $false$ is always interpreted as logical falsity, and all rules with $false$ as heads function as integrity constraints.

The above translation will be used to represent temporal knowledge bases about domains of (concurrent) actions, but it can also be used for temporal prediction, explanation and planning.

3 Representation of temporal knowledge

3.1 Domain evolutions and temporal knowledge bases

Given a domain description D, we want to determine whether a fluent holds at a particular time point after some actions have happened in the past. We start with the concept of histories, then employ abductive logic programming for the representation of temporal knowledge.

A domain history H is a finite set of pairs (A, T) for action A and time T. By $(A, T) \in H$ we mean that A happens at T. If both $(A, T) \in H$ and $(B, T) \in H$, we say that A and B happens concurrently at time T, and in this case we can alway write $(\{A, B\}, T) \in H$. And thus from now on we assume that for any $T \in TP$ there is at most one action expression A such that $(A, T) \in H$. A domain evolution E is a pair (D, H), where D is a domain description and H a history. Given a domain evolution (D, H), we can have an abductive logic program $KB(D, H)$ defined as follows:

$$KB(D, H) = \pi D \cup \{false \leftarrow not\ happens(a, t) : (a, t) \in H\}$$

The logic program $KB(D, H)$ will be simply called *the knowledge base generated from* D *and* H. The literal $false$ is always interpreted as logical falsity, and all rules with $false$ as heads function as integrity constraints. The predicates $initially_true(F)$ and $initially_false(F)$ are taken as abducible predicates used to capture the incomplete knowledge about the initial time. The semantics of $KB(D, H)$ is defined to be the union of the integrity constraints, the Clark Equality Theory, and the first-order theory obtained by completing all the non-abducible predicates (all predicates except $initially_true(F)$ and $initially_false(F)$). We will simply write $COMP(KB(D, H))$ to denote the semantics of of $KB(D, H)$. The following two results justify the above semantics definition.

Proposition 3.1 *$KB(D, H)$ is an acyclic logic program in the sense of [2].*

Corollary 3.2 *For the abductive logic program $KB(D, H)$, $COMP(KB(D, H))$ coincides with its generalized stable model semantics and generalized well-founded model semantics.*

3. Auxiliary Predicates:

   These auxiliary predicates will be used to define two main predicates $is_true(F,T)$ and $is_false(F,T)$, which indicate whether fluent F is true or false at time T.

$$
\begin{array}{rcl}
initiates(A,F,T) & \leftarrow & immediate_initiates(A,F,T) \\
initiates(A,F,T) & \leftarrow & subacteq(B,A), immediate_initiates(B,F,T), \\
 & & not\ clip_initiates(F,B,A,T) \\
terminates(A,F,T) & \leftarrow & immediate_terminates(A,F,T) \\
terminates(A,F,T) & \leftarrow & subacteq(B,A), immediate_terminates(B,F,T), \\
 & & not\ clip_terminates(F,B,A,T) \\
causes(F,T,A,B) & \leftarrow & subacteq(B,A), immediate_initiates(B,F,T), \\
 & & not\ clip_initiates(F,B,A,T), \\
 & & not\ clip_Cause1(F,B,A,T) \\
causes(F,T,A,B) & \leftarrow & subacteq(B,A), immediate_terminates(B,F,T), \\
 & & not\ clip_terminates(F,B,A,T), \\
 & & not\ clip_Cause2(F,B,A,T) \\
delta(A,T,F) & \leftarrow & initiates(A,G,T), terminates(A,G,T), \\
 & & causes(G,T,A,B), initiates(B,F,T) \\
delta(A,T,F) & \leftarrow & initiates(A,G,T), terminates(A,G,T), \\
 & & causes(G,T,A,B), terminates(B,F,T)
\end{array}
$$

4. Main Predicates:

   The following predicates are used to determine whether a fluent is true or false in a situation.

$$
\begin{array}{rcl}
is_true(F,succ(T)) & \leftarrow & is_true(F,T), not\ delta(A,T,F), \\
 & & not\ terminates(A,F,T). \\
is_true(F,succ(T)) & \leftarrow & initiates(A,F,T), not\ delta(A,T,F), \\
 & & not\ terminates(A,F,T). \\
is_false(F,succ(T)) & \leftarrow & is_false(F,T), not\ delta(A,T,F), \\
 & & not\ initiates(A,F,T). \\
is_false(F,succ(T)) & \leftarrow & terminates(A,F,T), not\ delta(A,T,F), \\
 & & not\ initiates(A,F,T).
\end{array}
$$

5. Domain-Specific Predicates

   The syntax and semantics of the following predicates depend on domain descriptions. Let $F \in \Sigma_f$ be a fluent name. From now on, we write $holds(F,T)$ and $holds(\neg F,T)$ to stand for $is_true(F,T)$, and $is_false(F,T)$, respectively.

   - For each effect proposition a **causes** f **if** $p_1,\ldots,p_n$ in D, where f is positive, we have a logic programming rule:

$$immediate_initiates(a,f,T) \leftarrow holds(p_1,T),\ldots,holds(p_n,T).$$

such that $B \subset C \subseteq A$, where C immediately initiates $\neg F$ (or F, resp.) in σ. If $B \in Cause(F, \sigma, A)$, we will say that B is a cause for the change in truth values of the fluent name F in σ when A is done. The truth value of the fluent name F may change in two ways: from *true* to $false$ and from $false$ to *true*. Note that the subaction B is set-inclusion minimal among all subactions which satisfy one of the two conditions above.

For later purpose we also need one more auxiliary functions:

$$\Delta(A, \sigma) \quad = \quad \bigcup_{B \in W} (Initiate(B, \sigma) \cup Terminate(B, \sigma))$$

where

$$W = \bigcup_{F \in Initiate(A,\sigma) \cap Terminate(A,\sigma)} Cause(F, \sigma, A)$$

Intuitively speaking, $\Delta(A, \sigma)$ denotes those fluents influenced by subactions of A which have conflicting effects. All of the fluents influenced by these subactions will be made undefined in the new situation resulting from doing A.

Definition 2.1 (Model) *A structure* (σ_0, Φ) *is a* model *of a domain description* D *iff*

- *Every v-proposition of* D *is satisfied in* (σ_0, Φ).
- *For every action expression* A *and every state* σ, $\Phi(A, \sigma) = \langle S^+, \ S^- \rangle$ *where*

$$\begin{aligned} S^+ &= (\sigma^+ \cup Initiate(A, \sigma)) \setminus Terminate(A, \sigma) \setminus \Delta(A, \sigma) \\ S^- &= (\sigma^- \cup Terminate(A, \sigma)) \setminus Initiate(A, \sigma) \setminus \Delta(A, \sigma) \end{aligned}$$

A domain description is consistent *if it has a model. A domain description is* complete *if it has exactly one model. We will use* $Mod(D)$ *to denote the set of all models of* D.

Definition 2.2 (Entailment) *A v-proposition is* entailed *by a domain description* D *if it is satisfied in every model of* D.

2.3 Translation into logic programs

In the following we present a new translation π from domain descriptions in $\mathcal{A}$ to abductive logic programs, where a time dimension is incorporated so that one can represent and reason about narratives, actions and time. In the following translation, all predicates will be self-explanatory. Let D be a domain description, the translation πD consists of the following logic programming rules:

1. Time dimension: In this paper we assume that time is structured on points, represented by a totally linearly-ordered set $(TP, \prec, succ, init)$, where TP is an infinite set of time points, $init \in TP$, and $succ(T) \in TP$ for any $T \in TP$. We will often use natural numbers which have a straightforward correspondence to terms for time points: $init$ corresponding to 0, $succ(init)$ to 1, etc.

2. Initialization:

$$\begin{aligned} is_true(F, init) \quad &\leftarrow \quad initially_true(F). \\ is_false(F, , init) \quad &\leftarrow \quad initially_false(F). \end{aligned}$$

that B contains A. We will simply write A for $\{A\}$ if A is atomic. An action expression denotes a concurrent action composed of atomic actions which happen at the same time. Fluent expressions and action expressions will simply be called fluents and actions, respectively, if no confusion arises.

In $\mathcal{A}$ there are two kinds of propositions: value propositions and effect propositions, simply called v-propositions, e-propositions. In this paper we only need a special class of v-proposition of the form

$$\textbf{initially} \quad F$$

where F is a fluent. An e-proposition is defined to be a statement of the form

$$A \textbf{ causes } \quad F \textbf{ if } \quad P_1, \ldots, P_n$$

where A is an action expression, and each of $F, P_1, \ldots, P_n (n \geq 0)$ is a fluent expression. If $n = 0$, then we will write it as A **causes** F.

A *domain description* D is a set of v-propositions and e-propositions. For example, a domain description D_{scp} for the Stolen Car Problem [7] is defined as follows: $D_{scp} =$ { **initially** $\neg Stolen$. $Steal$ **causes** $Stolen$}.

2.2 Semantics

The semantics of $\mathcal{AC}$ is defined by using states and transitions. A *state* σ is a pair of sets of fluent names $\langle \sigma^+, \sigma^- \rangle$ such that σ^+ and σ^- are disjoint, i.e., $\sigma^+ \cap \sigma^- = \emptyset$. Given a fluent name F and a state σ, we say that F holds in σ if $F \in \sigma^+$, F does not hold in σ if $F \in \sigma^-$, and it is not known whether F holds in σ otherwise. Given a fluent name F, we also say that $\neg F$ holds in σ if F does not hold in σ. Sometimes we also say that F is true in σ if $F \in \sigma^+$, F is false in σ if $F \in \sigma^-$, and F is undefined in σ otherwise.

A *transition function* Φ is a mapping from the set of pairs (A, σ), where A is an action expression and σ is a state, into the set of states.

A *structure* is a pair (σ_0, Φ), where σ_0 is a state, called the *initial state* of the structure, and Φ is a transition function. For any structure $M = (\sigma_0, \Phi)$ and any sequence of action expressions $A_1; \ldots; A_m$ in M, by $\Phi(A_1; \ldots; A_m, \sigma_0)$ we denote the state

$$\Phi(A_m, \Phi(A_{m-1}, \ldots, \Phi(A_1, \sigma_0) \ldots))$$

A v-proposition of the form **initially** F is satisfied in a structure $M = (\sigma_0, \Phi)$ iff F holds in the initial state σ_0.

We say that the execution of an action A in a state σ *immediately initiates* a fluent expression F if there is an e-proposition A **causes** F **if** $P_1, \ldots, P_m$ in the domain description such that for each $1 \leq i \leq m$, P_i holds in σ. Moreover, we say that the execution of A in σ *initiates* a fluent expression F if A immediately initiates F, or there is a $B \subseteq A$ such that execution of B in σ *immediately initiates* F and there is no C such that $B \subset C \subseteq A$, where execution of C in σ immediately initiates $\neg F$.

Let F be positive. When A initiates $\neg F$, we will say that A terminates F. We define two set-ranged functions as follows:

$$\begin{aligned} Initiate(A, \sigma) &= \{F \ : \ F \in \Sigma_f \text{ and } A \text{ initiates } F \text{ in } \sigma\} \\ Terminate(A, \sigma) &= \{F \ : \ F \in \Sigma_f \text{ and } A \text{ initiates } \neg F \text{ in } \sigma\} \end{aligned}$$

Note that $Initiate(A, \sigma)$ and $Terminate(A, \sigma)$ are not necessarily disjoint. We also need a set-ranged function $Cause(F, \sigma, A)$, (for fluent name F, state σ, and action expression A), which is defined to contain and only contain all the *set-inclusion minimal* subactions B of A satisfying: B immediately initiates F (or $\neg F$, resp.) in σ and there is no C

Among all these belief revision operations there are two common criteria: Consistency should be preserved and change should be *minimal*. Different approaches vary with different definitions of the minimality in change. In this paper we advocate another criterion: A possible cause explaining the new fact and all the effects of the possible cause should be added and all cause-effect relations are maintained. Before a new fact N is added to the knowledge base K, the agent should look for all possible causes of N. If the agent admits some cause C of N, then the agent should also add all the effects of C besides N. Hence, our approach will be called *possible causes approach* (PCA), in contract to the possible worlds approach (PWA) [6] and the possible models approach (PMA) [11]. Conceptually, the possible causes approach is different from all the above mentioned approaches. In the AGM theory, for example, if $Cn(K) \cup N$ is consistent, then the new belief set will simply be $Cn(K \cup N)$. In our approach, however, if a chosen cause C of N has another effect, say X, then the agent should also add X to her knowledge base. The PCA approach is especially suitable to domains of actions, where there is a clear cause-effect relation. In domains of actions, new knowledge is about changes in truth values of fluents. The possible causes for changes in truth values of fluents are the occurrences of actions.

The possible causes approach was originally proposed in [9] in a simple domain of actions. The advantage of the PCA approach over the PWA and PMA approaches are also discussed in [9]. In this paper we will generalize the results in [9] to accomodate concurrent actions. We will follow [8] to discuss fluents in a three-valued setting but logic programs are still viewed as two-valued. The rest of this paper is organized as follows: In Section 2 we introduce the action description language $\mathcal{AC}$ of [8] for domain descriptions of concurrent actions. In this section we also present a new transformation π from domain descriptions in $\mathcal{AC}$ to abductive logic programs. The translation is different from that of [8] in that a time dimension is incorporated so that one can represent and reason about narratives. In Section 3 we use the translation π to represent temporal knowledge bases, and give the soundness and completeness theorems for the representation. In Section 4 we discuss how to update a temporal knowledge base by using the possible causes approach. In this section we also consider some aspects of implementation. For the space limitation, proofs for all the propositions and theorems, and other more theoretical results are omitted.

2 Domain description

In this section we briefly review the action description language $\mathcal{AC}$ of [8], which is obtained by extending Gelfond and Lifschitz' action description language $\mathcal{A}$ [5] with concurrent actions, and present a new translation from domain descriptions in $\mathcal{AC}$ to abductive logic programs, where a time dimension is incorporated.

2.1 Syntax

We begin with the alphabet Σ of a domain, which consists of two disjoint finite non-empty sets of symbols Σ_f and Σ_a, called *fluent names* and *atomic action names*, respectively. We will often use self-explanatory fluent names and atomic action names when no confusion arises.

A *fluent expression* is defined to be a fluent name possibly preceded by $\neg$. A fluent expression is also called a *positive fluent* if it only consists of a fluent name; otherwise it is called a *negative fluent*.

An *action expression* is defined to be a non-empty finite set of atomic action names. We say that an action expression A is a *subaction expression* (*proper subaction expression*) of B iff $A \subseteq B$ ($A \subset B$). If A is a subaction expression of B, we often say

Updating Temporal Knowledge Bases with the Possible Causes Approach*

Renwei Li and **Luís Moniz Pereira**
Department of Computer Science
Universidade Nova de Lisboa
2825 Monte da Caparica, Portugal
{renwei | lmp}@fct.unl.pt

Abstract This paper presents a formal and computational methodology for updating temporal knowledge bases about actions and changes. Technically we employ the action description language $\mathcal{AC}$ of [8] to describe cause-effect relations on domains of (concurrent) actions. The knowledge bases on domains of (concurrent) actions are defined and obtained by a new translation from domain descriptions in $\mathcal{AC}$ into abductive normal logic programs, where a time dimension is incorporated. The knowledge bases are shown to be both sound and complete with respect to their domain descriptions. In particular, in contrast to Ginsberg's possible worlds approach (PWA) and Winslett's possible models approach (PMA) we propose a possible causes approach (PCA) to updates of temporal knowledge based on the slogan: What is believed is what is explained, advocated in [9]. A possible cause of new knowledge consists of abduced occurrences of actions and value propositions about the initial state of the domain of actions, that would allow to derive the new knowledge, and can be computed with abductive logic programming.

1 Introduction

Suppose that we have a temporal knowledge base K about a dynamic domain. Let N represent a new fact. Now we want to update K with N. This kind of tasks are generally called belief revision/update or knowledge assimilation.

A representative of the work on belief revision is the so-called AGM theory, characterized by eight postulates. As is well-known, belief revision operators satisfying the AGM postulates select the minimal models of the new knowledge N according to a suitable ordering relation based on the deductive closure of K [1]. However, it has been noted that some belief revision operators satisfying the AGM postulates give intuitively undesirable results. In the literature, some other belief revision/update operators have been proposed to solve practical problems. These belief revision/update operators can be roughly classified into two types: the syntax-based approach [6, 10] and the semantics-based approach [3, 11]. The syntax-based approach selects set-inclusion maximal subsets of K that do not allow for the derivation of the negation of the new sentence. The semantics-based approach selects those models satisfying the new sentence and differing minimally from the models of K.

*Published in: A. M. Ramsay (ed.), *Artificial Intelligence: Methodology, Systems, Applications*, IOS Press, 1996, pp. 148 - 157. This paper received the Best Paper Award from the Program Committee of AIMSA'96.