The Ramification Problem in Temporal Databases: Concurrent Execution with Conflicting Constraints

Nikos Papadakis, Grigoris Antoniou, Dimitris Plexousakis
Department of Computer Science, University of Crete, and
Institute of Computer Science, FORTH, Greece
npapadak,antoniou,dp@ics.forth.gr

Abstract

In this paper we study the ramification problem in the setting of temporal databases. Standard solutions from the literature on reasoning about action are inadequate because they rely on the assumptions that fluents persist and actions have effects on the subsequent situation only. We provide a solution based on an extension of the situation calculus and the work of McCain and Turner. More specifically, we study the case where there are conflicting effects of actions which executed concurrently, and we distinguish between hard and soft integrity constraints.

1 Introduction

Many domains about which we wish to reason are dynamic in nature. *Reasoning about action* [8, 3] is concerned with determining the nature of the world (what holds in a world state) after performing an action in a known world state, and has found application in areas such as cognitive robotics [8]. In this context, the *ramification problem* [6] is concerned with the indirect effects of actions in the presence of constraints. Standard solutions to this problem [3, 2, 4, 5] rely on the assumptions that fluents persist and that actions have effects on the subsequent situation only.

In our work, we consider the ramification problem in a temporal setting. In this context, actions are allowed to have effects which may commence at a time other than the next time point, and the effects may hold only for certain time. For example, a certain misdemeanor may result in the suspension of an employee for a certain period of time, after which the employee is again active and may get a salary. So, in a temporal setting the main assumptions of previous solutions to the ramification problem are inadequate, and we need new techniques. In [7], the problem has been addressed for the cases in which actions result in changes in the future (e.g., the promotion of an employee takes effect in two months).

In this paper, we consider the case where there can exist inconsistencies between two or more constraints. Clearly, these constraints cannot be satisfied simultaneously. However, conflicts can be resolved if we consider different strengths of integrity constraints. Such a situation may arise in a legal database where a specific law predominates a generic one. Here is an example:

$$public\_worker(p,t) \wedge experience(p,D,t) \wedge D < 25 \rightarrow \neg manager(p,t)$$

$$public\_worker(p,t) \wedge bachelor(p) \wedge experience(p,D,t) \wedge D \geq 20 \rightarrow manager(p,t)$$

We propose the distinction of two types of constraints: (a) strict constraints that must always be satisfied, and (b) defeasible (soft) constraints which can be compared using a priority relation. Intuitively, a defeasibe constraint must be satisfied if and only if it does not contradict some strict constraint or a defeasible constraint of higher priority.

Our approach is based on the situation calculus [6] and the work of McCain and Turner [5]. We extended the approach of [5] by introducing duration to fluents, and by considering temporal aspects. As we have explained in [7], in a temporal database we need to describe the direct and indirect effects of an action not only in the immediately resulting next situation, but possibly for many future situations as well.

This means that we need a solution that seperates the current effects (dynamic rules) from the futue effects (static rules). This is neccessary beacause another action may occur between them which cancels the future effects. We adopt the McCain approach. When an action take place a corresponding dynamic rules will be evaluated in order to ensure the direct effects of the action. At each time poit we execute a set of static rules which encapsulate the indirect effects.

The paper is organised as follows. Section 2 describes the technical preliminaries, that are based on previous work by the authors [7]. Section 3 presents a motivating example for conflicting integrity constraints and our approach, while section 4 describes an algorithm for controlling the application of static and dynamic rules.

2 Technical Preliminaries

Our solution to the ramification problem is based on the situation calculus [6]. However it is necessary to extend the situation calculus to capture the temporal phenomena, as done in the previous work [7].

For each fluent f, an argument L is added, where L is a list of time intervals $[a, b], a < b$. $[a, b]$ represents the time points $\{x | a \leq x < b\}$. The fluent f is true in the time intervals that are contained in the list L.

We define functions $start(a)$ and $end(a)$, where a is an action. The former function returns the time moment at which the action a starts while the latter returns the time moment at which it finishes.

Actions are ordered as follows: $a_1 < a_2 < .... < a_n$, when $start(a_1) < start(a_2) < ..... < start(a_n)$.

Actions(instatnteous) $a_1, a_2, ..., a_n$ are executed concurrently if $start(a_1) = start(a_2) = ... = start(a_n)$.

The predicate $occur(a, t)$ means that the action a is executed at time moment t.

We define functions $start(S)$ and $end(S)$, where S is a situation. The former function returns the time moment at which the situation S starts while the latter returns the time moment at which it finishes.

The function $FluentHold(S, t)$ returns the set of all fluents that are true in the time moment t.

We define as non-temporal situation S in a time point t the situation $S = FluentHold(S', t)$.

A transformation from a situation to another could happen when the function $FluentHold(S, t)$ returns a differenet set.

We define two types of constraints: (a) strict constraints that must always be satisfied, and (b) defeasible (soft) constraints that can be compared by a priority relation.

We define two types of rules: (a) dynamic $occur(a, t) \rightarrow f([..])$ which are executed when the action a takes place; and (b) static rules $G_f([a, b]) \rightarrow f([...])$ which are executed at time point t if the fluent f is not true in the time interval $[a, b]$.

The execution one dynamic or static rule has as consequence the transformation to a new situation.

We define as *legal* (*consistent*) a situation in which all strict integrity constraints are satisfied, and each defeasibe constraint is satisfied if and only if it does not contradict some strict or defeasible constraint with higher priority. Also, each function fluent has only one value at each time point.

As we have already said, the previous approaches to solve the ramification problem are inadequate in the case of temporal databases. We overcome these difficulties with the time - actions - situations correspondence that appears in figure 1.

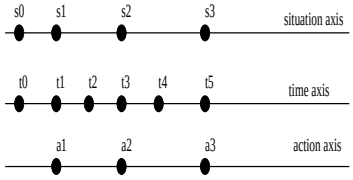

Figure 1. Time-Actions-Situations

There are three parallel axes: the situations axis, the time axis and the actions axis. When an action takes place or a static rule evaluated or cease to hold a fluent, the database changes into a new situation. The database change situation when we have transform from one non-temporal situation to the next situation.

We base our work on the ideas of McCain and Turner [5] who propose to use *static rules* to capture the indirect effects of actions (based on integrity constraints in the particular domain), and *dynamic rules* to represent the direct effects of actions. In our approach, for each action A there is a dynamic rule of the form $A \rightarrow \bigwedge F_i(L'_i)$, where each $F_i(L'_i)$ is $f_i(L'_i)$ or $\neg f_i(L'_i)$ for a fluent f. These rules describe the direct effects of an action. Here is an example:
$occur(misdemeanor(p), t) \rightarrow illegal(p, [[t, t + 5]])$

In addition, for each fluent f we define two rules, $G(L) \rightarrow f(L)$ and $B(L) \rightarrow \neg f(L)$. $G(L)$ is a fluent formula which, when is true (at list L), causes fluent f to become true at the time intervals contained in the list L (respectively for $B(L)$). These rules encapsulate the indirect effects of an action. Here is an example: $illegal(p, [[2, 7]]) \rightarrow suspended(p, [[2, 7]])$

One cornestone of our previous work was the production of the static rules from integrity constraints. We do not repeat its description here, but refer the reader to [7]. The main property of the resulting static rules is given the following theorem that has been proved.

Theorem 2.1 *When a static rule is executable at least one integrity constraint is violated.*

3 A Motivating Example

Assume that if a public employee commits a misdemeanor, then for the next five months she is considered illegal. When a public employee is illegal, then she must be suspended and cannot take promotion for the entire time interval over which she is considered illegal. Also, when a public employee is suspended, she cannot receive a salary until the end of the suspension period. Each public employee is graded for her work. If she receives a bad grade, then she is considered a bad employee. If she receives a good grade, then she is considered a good employee and she may take a bonus if not suspended. Each public employee receives an increase and a promotion every two and five years, respectively, if not illegal.

We can identify five actions, $misdemeanor$, $good\_grade$, bad_grade, $take\_promotion$ and $take\_increase$ and seven fluents $good\_employee$, $illegal$, $take\_salary$, $take\_bonus$, $position$, $suspended$ and $salary$. The fluent $position(p, l, t_1)$ means that the public worker is at position l for the last t_1 months while $salary(p, s, t_1)$ means that the public worker has been receiving salary s for the last t_1 months. The direct effects of the six actions are expressed by the following rules:

$$occur(misdemeanor(p), t) \rightarrow illegal(p, [t, t+5]) \quad (1)$$

$$occur(bad\_grade(p), t) \rightarrow \neg good\_employee(p, [t, \infty)) \quad (2)$$

$$occur(good\_grade(p), t) \rightarrow good\_employee(p, [t, \infty)) \quad (3)$$

$$occur(take\_increase(p), t) \wedge salary(p, s, 24) \rightarrow salary(p, s + 1, 0) \quad (4)$$

$$occur(take\_promotion(p), t) \wedge position(p, l, 60) \rightarrow position(p, l + 1, 0) \quad (5)$$

where t is a temporal variable and the predicate $occur(misdemeanor(p), t)$ denotes that the action $misdemeanor(p)$ is executed at time t. The former four rules are dynamic and executed every time that the corresponding actions take place. The remaining two are also dynamic but are executed periodically because the corresponding actions take place periodically. Also we have and the following integrity constraints which give rise to indirect effects of the six actions.

$$illegal(p, t_1) \rightarrow suspended(p, t_1) \quad (6)$$

$$suspended(p, t_1) \rightarrow \neg take\_salary(p, t_1) \quad (7)$$

$$\neg suspended(p, t) \wedge good\_employee(p, t) \rightarrow take\_bonus(p, t) \quad (8)$$

$$\neg good\_employee(p, t_1) \rightarrow \neg take\_bonus(p, t_1) \quad (9)$$

$$\neg suspended(p, t_1) \rightarrow take\_salary(p, t_1) \quad (10)$$

The rules (6-10) are static and executed every time. This happens because there are effects of the actions which hold for a time interval (e.g. the effect $illegal$ of the action $misdemeanor$ which hold 5 time point after the execution of the action). After the end of the time intervals the effects pause to hold without an action taking place. In the next section we describe how we extended the solution of McCain and Turner in order to address the problem in temporal databases.

We extend the example as follows: **1)**If she has experience more than 25 years then she can be a manager. Otherwise she cannot be a manager. **2)** If she has a bachelor then she can be a manager if she has experience of more than 20 years. **3)** If she has a master then she can be a manager if she has experience of more than 15 years. **4)** If she has a PhD then she can be a manager if she has experience of more than 10 years. **5)**If she suspended then she cannot be a manager in any case.

Now we have five new fluents $manager$,$bachelor$, $master$, phd and $experience$. The fluent $experience(p, D, t)$ means that the public worker p have experience D at the time moment t. The above constraints are equivalent with the follow

$$public\_worker(p, t) \wedge experience(p, D, t) \wedge D \geq 25 \rightarrow manager(p, t) \quad (11)$$

$$public\_worker(p, t) \wedge experience(p, D, t) \wedge D < 25 \rightarrow \neg manager(p, t) \quad (12)$$

$$public\_worker(p, t) \wedge bachelor(p) \wedge experience($$

$p, D, t) \wedge D \geq 20 \rightarrow manager(p, t)$ (13)

$public\_worker(p, t) \wedge master(p) \wedge experience($
$p, D, t) \wedge D \geq 15 \rightarrow manager(p, t)$ (14)

$public\_worker(p, t) \wedge phd(p) \wedge experience(p, D, t) \wedge$
$D \geq 10 \rightarrow manger(p, t)$ (15)

$public\_worker(p, t) \wedge suspended(p, t) \rightarrow$
$\neg manager(p, t)$ (16)

Assume that we have the public workers P_1, P_2, P_3, P_4, P_5 and the following khoweldge:

$public\_worker(P_1, 50), public\_worker(P_2, 50),$
$public\_worker(P_3, 50), public\_worker(P_4, 50),$
$public\_worker(P_5, 50), bachelor(P_2), master(P_3),$
$phd(P_4), phd(P_5), experience(P_1, 24, 50),$
$experience(P_2, 21, 50), experience(P_3, 16, 50),$
$experience(P_4, 11, 50), experience(P_5, 11, 50)$

From the rule (12) there is no manager. From the rule (13) P_2 is a manager. From the rule (14) P_3 is a manager. From the rule (15) P_4 is a manager. From the rule (16) P_5 is not a manager

As we observe in the first four cases there is contradiction between rule (12) and the rules (13),(14),(15) and in the latter case there is contradiction between (13),(14),(16). It is obvious that we require an order between these rules in order to avoid the inconstistency in the conclusions.

The solution is to define some sets of rules which have different priority. The idea is to seperate the constraints in different sets and to execute the algorithm for production of static rules for any sets seperately. Thus we construct different sets of static rules. Now we must determine how we execute these different sets of rules.

We propose two main categories of constraints: (a) the stricts constraint which must always be satisfied and (b) the defeasibe constraints which must be satisfied if its satisfaction does not violed any strict constraint or defeasibe constraint with higher priority.

In our example the set of strict constraints is the rule (6-10) and (16). The sets of defeasibe constraints are as follows: with priority 1: the rules (11) and (12). with priority 2: the rule (13). with priority 3: the rule (14). with priority 4: the rule (15).

The constraints (6-10) and (16) must always be satisfied. The defeasibe constraints with priority i must be satisfied if there is no contradiction between them and the set of strict constraints and the sets of defeasibe constraints with priority greater than i.In the case concurrence the above exmaple must be examine under the following cases: **A)**When there is contradiction between the direct effects of the actions which executed concurrence. **B)** When there is contraction between the direct effects and the indirect effects of the actions which executed concurrence and the indirect effects emerge by the evaluation strict static rules. In that case we reject the execution beacuse the evaluation of the strict integrity constraint is necessary(thus there is no consistent situation). **C)** When there is contradiction between the direct effects and the indirect effects of the actions which executed concurrence and the indirect effects emerge by the evaluation defeasible static rules. In that case we do not evaluate the deafeasible rules which create the contradiction. **D)** When there is contraction between the indirect effects of the action which executed concurrece and the contraction emerge from integrity constraint with differenet priority. In that case we do not evaluate the static rule with smaller priority. **E)** When there is contraction between the indirect effects of the action which executed concurrece and the contraction emerge from static rules with same priority. In that case we reject the execution.

4 An Algorithm for Controlling the Execution of Static and Dynamic Rules

Algorithm 1

1. Before the concurrent execution of the actions $a_1, ..., a_n$ check whether the set $E = \{f_i([t, t']) : \exists a_i \; s.t \; occur(a_i, t) \rightarrow f_i([t, t'])\}$ is satisfiable. If it is not, reject the concurrent execution.
2. After the execution of concurrent actions, evaluate the dynamic rules which refer to those actions.
3. Evaluate the algorithm of the evaluation of static rules (algorithm 2)
4. If the algorithm of the evaluation of static rules returns inconsistency, then reject the execution, else continue.
5. Until some other action executes, use the situations which have been produced by the algorithm of the evaluation of static rules.

Algorithm 2: evaluation of static rules

1. At time point t if there is a strict static rule $G_f \rightarrow f(L_1)$ evaluated then

(a) if $\neg f(L_2) \in E$ and $L_1 \cap L_2 \neq \{\}$ then return inconsistency.
(b) else set $L_2 = L_2 \setminus (L_1 \cap L_2)$ and evaluate the rule $K_f \rightarrow \neg f(...)$.

2. Repeat step 1 until L_1 and L_2 do not change or until they take previous values. In the first case add the the$f(L_1), \neg f(L_2)$ in the set E. In the latter case, return inconsistency.
3. Repeat the step 1 and 2 for all strict static rules.
4. If there are sets of static rules $R_1, ...R_N$ of defeasible rules with the corresponding priority then: For $i = N \ to \ 1$ do
   (a) set $E' = \emptyset$
   (b) While there is a static rule $r \in R_i$ which is executable
   (c) evaluate this rule $r : A(L) \rightarrow f(L)$
   (d) if $\neg f(L_1) \in E$ and $L \cap L_1 \neq \{\}$ then deseable this rule for this execution of algorithm and ignore its conclusion.
   (e) else if $\neg f(L_1) \in E'$ return inconsistency. Else change the set $E' = E' \cup \{f[L)\}$.
5. Repeat the step 1,2,3 and 4 for all time moments at which there are references.

Now we present how the algorithms 1 and 2 run for the case B. Assume the following intial situation at time point 2

$S_0 = \{bachelor(P_1), master(P_1),$
$experience(P_1, 10, 2), \neg suspended(P_1, [0, \infty]),$
$\neg illegal(P_1, [0, \infty]), \neg manager(P_1, [0, \infty])\}$

Assume that at time point 2 the actions $misdemeanor(P_1), take\_increase(P_1)$ are executed concurrencing. As we observe the two above actions have as direct effects the following

$occur(misdemeanor(P_1), 2) \rightarrow illegal(P_1, [2, 7])$

$occur(take_increase(P_1), 2) \wedge salary(P_1, 2, 2) \rightarrow$
$salary(P_1, 3, 2) \wedge take\_salary(P_1, [2, \infty])$.

By the step 1 of the algorithm 1 we have that $E =$ $\{illegal(P_1, [2, 7]), salary(P_1, 3, 2), take\_salary(P_1,$ $[2, \infty])\}$ The set E is satisfiable thus we go on the step 2 of the algorithm 1. we execute the dynamic rules and the new situation is

$S_1 = \{bachelor(P_1), master(P_1), experience(P_1, 10,$ $2), \neg suspended(P_1, [0, \infty]), illegal(P_1, [2, 7]), \neg illegal($ $P_1, [8, \infty]), \neg manager(P_1, [0, \infty]), salary(P_1, 3, 2)\}$

In the step 3 of the algorithm 1 we call the algorithm 2. In the step 1 of the algorithm 2 the following strict static rules will be evaluated

$illegal(P_1, [2, 7]) \rightarrow suspended(P_1, [2, 7])$

$suspended(P_1, [2, 7]) \rightarrow \neg take\_salary(P_1), [2, 7])$

Thus the indirect effects of these actions are $E' =$ $\{suspended(P_1, [2, \infty]), \neg take\_salary(P_1, [2, 7])\}$

From the step 1.a of the algorithm 2 we have that $take\_salary(P_1, [2, \infty]) \in E$ and $[2, 7] \cap [2, \infty] =$ $[2, 7]$ thus the algorithm 2(step 1) return inconsistency and the algorithm 1 reject the concurrence execution (step 4 of algorithm 1). We have proved that:

Theorem 4.1 *The algorithm 1 and 2 always terminate and return a consistent situation.*

5 Conclusion

In this paper we studied the ramification problem in the setting of temporal databases. More specifically, we studied the case where the effects of actions which executed concurrence may be conflicting, and considered the distinction between strict and defeasible (soft) integrity constraints of different strength. This case is particularly complex, thus interesting.

In future work we intend to examine the problem when the effects of the action refered in the past.

References

[1] M. Ginsberg and D. Smith. Reasoning about action I: A possible worlds approach. Art. Int., 35:165-195, 1988.

[2] A. Fusaoka. Situation Calculus on a Dense Flow of Time, Proc. of AAAI-96, pp. 633-638, 1996

[3] Antonis Kakas and Rob Miller, A Simple Declarative Language for Describing Narratives with Actions,Logic Programming, pp. 157-200,1997.

[4] V. Lifshitz. Frames in the space of situations, Artificial Intelligence, 46:365-376, 1990.

[5] N. McCain and H. Turner. A causal theory of ramifications and qualifications. Proc. of IJCAI-95, pp. 1978-1984, 1995.

[6] J. McCarthy and P.J. Hayes. Some philophical problem from the standpoint of artificial intelligence, pp. 463-502. American Elsevier, 1969.

[7] Nikos Papadakis and Dimitris Plexousakis. Action with Duration and Constraints: The Ramification problem in Temporal Databases. IJTAI, pp. 315-353, September 2003.

[8] R. Reiter. Natural Actions, Concurrency and Continous Time in the Situation Calculus, KR 96, pages 2-13, 1996.