Actions with Duration and Constraints: the Ramification Problem in Temporal Databases

Nikos Papadakis, Dimitris Plexousakis
Department of Computer Science, University of Crete, and
Institute of Computer Science, FORTH, Greece
{npapadak, dp}@csd.uch.gr

January 31, 2003

Abstract

The ramification problem is a hard and ever present problem in systems exhibiting a dynamic behavior. The area of temporal databases in particular is still lacking satisfactory solutions to the ramification problem. In this paper, we address the ramification problem based on causal relationships that take time into account. We study the problem for both instantaneous actions and actions with duration. The proposed solution advances previous work by considering actions with effects occurring in any of the possible future situations resulting from an action's execution.

1. Introduction

The ramification problem is a hard problem, that arises in robotics, software engineering, databases and all systems exhibiting a dynamic behavior. In this paper we consider the case of temporal databases where transaction are a sequences of deterministic actions.

We introduce this problems by mean of examples. Suppose we are interested in maintaining a database that describes a simple circuit, which has two switches and one lamp (figure 1.A).

The circuit's behavior is described by the following integrity constraints. First, when the two switches are up, the lamp must be lit. Second if one switch is down then the lamp must not be lit. The integrity constraints are expressed as the following formulas, employing predicates up and light,

$$up(s_1) \wedge up(s_2) \supset light$$
$$\neg up(s_1) \supset \neg light$$
$$\neg up(s_2) \supset \neg light\,.$$

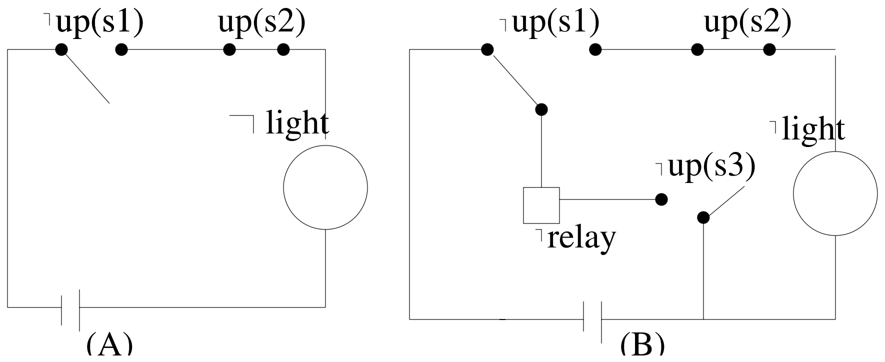

Figure 1: The circuit

Action *toggle_switch* changes the situation of a switch as follows:

$$toggle\_switch(s) \supset up(s) \quad if \quad \neg up(s)$$
$$toggle\_switch(s) \supset \neg up(s) \quad if \quad up(s)\,.$$

The above propositions describe the direct effects of the action *toggle_switch*. A situation is called consistent when it satisfies all integrity constraints. Assume that the circuit is in situation $S = \{up(s_1), \neg up(s_2), \neg light\}$. The situation S is consistent, because it satisfies all integrity constraints. Now assume that we execute the action $toggle\_switch(s_2)$. This action has as direct effect to change the state of switch s_2 from $\neg up(s_2)$ to $up(s_2)$. Now the situation of the circuit is $S_1 = \{up(s_1), up(s_2), \neg light\}$. This situation is inconsistent, because it violates the first integrity constraint. The reasonable conclusion is that the lamp must be lit. So the final situation is $S_2 = \{up(s_1), up(s_2), light\}$. The change of the condition of the lamp is the indirect effect of the action $toggle\_switch(s_2)$. Notice that the indirect effects exists because of the presence of integrity constraints. The ramification problem refers to the concise description of the indirect effects of an action in the presence of constraints.

Several ways for addressing the ramification problem have been suggested in the leterature. The majority of them are based in the situation calculus [12]. The situation calculus is a second-order language that represents the changes which occur in a domain, as results of actions. One possible evolution of the world is a sequence of actions and is represented by a first-order term. The situation at which no action has occurred yet, is called the initial situation (S_0). There is a binary function $do(a, S)$ which yields the new situation resulting from the execution of action a in the situation S. Predicates, called fluents, may change truth value from one situation to another. Similary, one can represent functions whose values are dependent on the situations on which evaluated (functional fluents).

The simplest of the technique suggested in the literature is the *minimal-change* approach [24]. It suggests that, when an action occurs in a situation S, are need to find the consistent situation S' which has the fewer changes from the situation S. S' is that situation that is closer to S than any other situation.

Another solution is the *categorization* of fluents [8, 9, 10]. The fluents are categorized in primary and secondary. A primary fluent may change only as a direct effect of an action, while a secondary one may change only as an indirect effect of an action. After an action takes place, we choose the situation with in the fewer changes in primary fluents. The categorization of fluents solves the ramification problem only if all fluents can be categorized. If some fluents are primary for some actions and secondary for some other this solution is not satisfactory.

As we can observe from the above examples, the change of fluent f's truth value potentially affects the truth-value of some other fluents, while it does not affect others. We define a binary relation I between fluents as follows: if $(f, f') \in I$, then a change in fluent f's value may affect the value of f'. In the above example, $(up(s_1), light) \in I$, whereas $(up(s_1), up(s_2)) \notin I$. A fluent could change or remain unchanged after an action. This depends on the context in which an action take place.

Causal relationships [11, 22, 23] capture this dependence between an action and an indirect effect. Each causal relationship consists of two parts. A causal relationship has the form

$$\epsilon \;\; causes \;\; \rho \;\; if \;\; \Phi$$

where ϵ is an action, ρ is the indirect effect and Φ is the context. The context is a fluent formula. Each causal relation must evaluated, after the execution of the action ϵ, if and only if the context is true. The binary relation I define the dependence that exists between context Φ and fluent ρ.

In the above example, there are four causal relationships:

$$\begin{array}{l} up(s1) \;\; causes \;\; light \;\; if \;\; up(s2) \\ up(s2) \;\; causes \;\; light \;\; if \;\; up(s1) \\ \neg up(s1) \;\; causes \;\; \neg light \;\; if \;\; \top \\ \neg up(s2) \;\; causes \;\; \neg light \;\; if \;\; \top \end{array}$$

The majority of approaches for ensuring consistency of data or knowledge base ignore the **frame** and **ramification** problem. A notable exception is the work described by Plexousakis in [16] and by Plexousakis and Mylopoulos [17] where the problem is addressed in a temporal database context.

The rest of the paper is organized as follows: in section 2, we describe an algorithm that discovers dependencies between fluents. In section 3, we define the ramification problem in temporal databases and in section 4 we present prevalent previous work relevant to this problem. In section 5 , we propose an extension to the situation calculus for addressing the ramification problem in temporal databases. In section 6, we deal with the ramification problem in temporal databases when actions execute sequentially. We examine two cases, namely the case of instantaneous actions and the case of actions with duration. Finally, in section 7, we address the ramification problem in temporal databases when actions execute concurrently.

2. Fluent Dependencies

This section describe an algorithm that discovers dependencies between fluents. Assume that we have two kinds of integrity constraints:

$$\begin{array}{ll} (a) & G_f \supset K_f \\ (b) & G_f \equiv K_f \ , \end{array}$$

where G_f and K_f are fluent propositions. The difference between the two kinds is that, for the second kind, when $\neg G_f$ holds then $\neg K_f$ also holds, whereas this is not necessarily the case for the first. For the first kind of constraints, for each $f \in G_f$ and $f' \in K_f$ we add the pair (f, f') *in* I. Notice that $(f', f) \notin I$ (because $K_f \not\supset G_f$). For the second kind of constraints we make the following hypothesis: The change of the truth value of a fluent belonging to G_f is expected to affect the truth values of some fluents belonging to K_f, while it is not expected to affect the truth values of other fluents which belong to G_f. We make the same hypothesis for the fluents of K_f. Now we can construct the set I. For each pair of fluents f, f', such that $f \in G_f$ and $f' \in K_f$ we add (f, f') *and* (f', f) *to* I. Consider the circuit in figure 1 (B). The integrity constraints specifying the behavior of this system are expressed as the following formulae:

$$\begin{array}{ll} (a) & light \equiv up(s1) \wedge up(s2) \\ (b) & relay \equiv \neg up(s1) \wedge up(s3) \\ (c) & relay \supset \neg up(s2) \end{array}$$

By applying this procedure the set I is constructed as follows: for constraint (a) we conclude that $(up(s1), light)$, $(up(s2), light)$, $(light,(up(s1))$,$(light, (up(s2))$ must be added in I. From rule (b) we obtain $(up(s1), relay)$, $(up(s3), relay)$, $(relay, (up(s1))$,$(relay, (up(s3))$ to be in I and from rule (c) we obtain $(relay, up(s2)) \in I$.

Because of our hypothesis, $(up(s1), light) \in I$, while $(up(s_1), up(s_2)) \notin I$. Assume that the circuit is in the situation that is depicted in figure 1 (B). The action $toggle\_switch(s_1)$ has as indirect effect to light the lamp and not to toggle

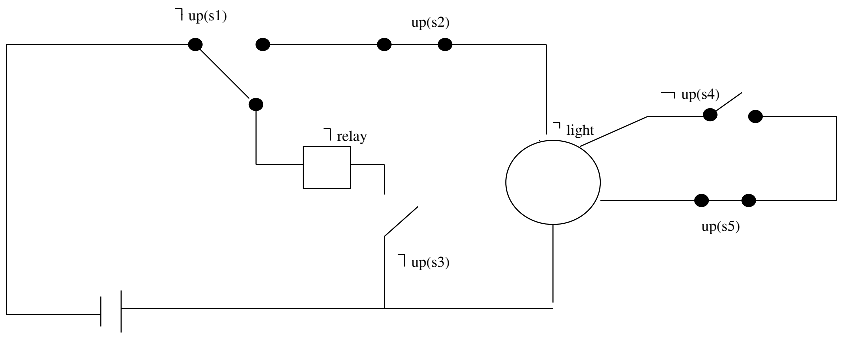

Figure 2: A more complex circuit

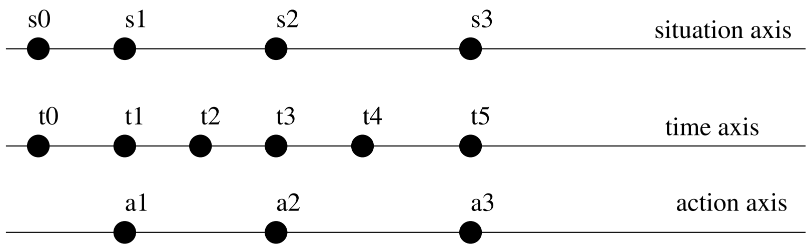

Figure 3: The Time-Situation-Actions Correspondence

the switch s_2. We observe that it is not reasonable to include the fluent pairs $(light, (up(s1))$,$(light, (up(s2))$, $(relay, (up(s1))$,$(relay, (up(s3))$ in I. The truth values of fluents $light$ and $relay$ cannot change as the direct effect of an action, so they cannot affect the truth values of other fluents. We must eliminate them from set I. The algorithm is:

1. For each $f \in G_f, f' \in K_f$, where $G_f \supset K_f$ is a specified constraint add the pair $(f, f') \in I$.

2. For each $f \in G_f, f' \in K_f$, where $G_f \equiv K_f$ is a specified constraint do

   (a) If f can change its truth value as the direct effect of an action, then add (f, f') in I. If f' can change its truth value as direct effect of an action then add (f', f) in I.

In our example, the above change is right if and only if each of the fluents $light$ and $relay$ appear in one only rule of the form $G_f \equiv K_f$. For example, consider the circuit in figure 2. The integrity constraints specifying the behavior of this system are expressed as the following formulae:

$$\begin{array}{ll} (a) & light \equiv up(s1) \wedge up(s2), \\ (b) & light \equiv up(s4) \wedge up(s5), \\ (c) & relay \equiv \neg up(s1) \wedge up(s3), \\ (d) & relay \supset \neg up(s2) \end{array}$$

Applying the procedure described above yields:

$$((up(s1), light), (up(s2), light),\ (up(s4), light), (up(s5), light),$$
$$(up(s1), relay), (up(s3), relay),\ (relay, up(s2)) \in I\ .$$

Assume that the circuit is in the situation depicted in figure 2. Then, after the execution of action $toggle\_switch(s_4)$, because $(up(s4), light) \in I$, the fluent $light$ changes from $\neg light$ to $light$. Because $(light, up(s1)), (light, up(s2)) \notin I$, the fluents $up(s1), up(s2)$ do not change. This means that the circuit will be in situation $\neg up(s1), up(s2), up(s4), up(s5), \neg up(s3), \neg relay, light$, which violates the rule (a). Assume now that the integrity constraints specifying the behavior of this system are expressed as the following formulae:

$$\begin{array}{ll} (a) & light \equiv (up(s1) \wedge up(s2)) \vee (up(s4) \wedge up(s5)) \\ (b) & relay \equiv \neg up(s1) \wedge up(s3), \\ (c) & relay \supset \neg up(s2) \end{array}$$

In the above specification of constraints, the fluent $light$ is only in one constraint of type $G_f \equiv K_f$ and the modified algorithm behaves correctly. As we observe in the circuit of the figure 2, consists from two smaller circuits. The first consists from the switches s_1, s_2 and the lamp, while the second from the switches s_4, s_5 and the lamp. The reasonable is the lamp light when on of the two circuit is closed. This ensuring by the second set of integrity constraints. The first set of integrity constraints ensuring that when one circuit is closed then must closed and the second. This is not reasonable.

3. Temporal Databases and Ramifications

Most solutions of the ramification problem in conventional databases are based on the situation calculus. In temporal databases we need to incorporate time in the situation calculus. Some works have suggested some ways for incorporate time in the situation calculus by drawing a correspondence between situation calculus and a linear time line [13, 5, 18, 19, 4]. This correspondence is defined between real situations and time. This is not absolutely correct because the situation calculus supports parallel histories of situations. The above weakness can be overcome by defining a correspondence between a branching time structure and situations. The branching time structure creates many parallel histories of situations. As we show in the sequel, we extend this approach for addressing the ramification problem in temporal databases. We describe the problem in this context with an example.

Assume that when a driver p drinks alcohol then s/he is considered drunk for the next five hours. In this time span other actions may occur leading to many different situations. In all these situations the fluent $drunk(p)$ must be true. After five hours the fluent $drunk(p)$ must become false and, thus, the database must change into a new situation without any action taking place. The action $drive(p)$ cannot be executed if $drunk(p)$ holds. The constraints which describe that are:

$$occur(drink\_alcohol(p), t) \supset drunk(p, t') \wedge (t' < t + 5h)$$
$$drunk(p, t) \supset \neg drive(p, t) .$$

In a temporal database, we need to describe the direct and indirect effects of an action not only in the next situation but possible for many future situations. In the above example the action drink has the indirect effect that the driver can not drive during the following four hours. In these four hours, a number of other action may execute leading to many different situations. In all these situations the action "drink alcohol" has the indirect effect $\neg drive$. Also five hours after the execution of the action $drink\_alcohol$ the situation changes without the occurrence of an action (because the person is no longer drunk). This means that the transition from one situation to the next could happen without an action taking place. This means that fluents cannot be assumed to persist until an action changes their truth value.

The causal relationships cannot solve the ramification problem in temporal databases because they determine the direct and indirect effects only for the next situation. The same weakness characterizes all other solutions of the ramification problem in conventional databases.

The above weakness can be alleviated by constructing a correspondence between situations and actions with time. Some proposal for that has been done by Fusuaoka [5], Pinto and Reiter [19]. We adopt the correspondence that appears in figure 3 [13]. There are three parallel axes: the first is the situations axis, the second is the time axis and the third is the actions axis. For now, we assume that all actions are instantaneous.

We assume a discrete model of time in which each timestamp specifies a point in time or moment. Each action occurs at a specific time point. When an action a_2 occurs at time point t_3 in a situation S_1, a new situation $S_2 = do(a_2, S_1)$ results. Hence, at each time moment, we must determine the truth value of fluents. In section 6, we provide a solution to the ramification problem in temporal databases for sequentially executing actions (instantaneous or with duration).

4. Previous Work

The most prevalent previous works are those by Reiter [20], Reiter and Pinto [18, 19] and by Kakas [6, 7]. Reiter has suggested an extension of the situation calculus in order to encapsulate time and axioms which ensure that in each legal situation all natural actions have been executed. A natural action is an action which executes in a predetermined time moment except if some other action has changed the time of execution. Reiter has extended the fundamental axioms of the situation calculus in order to determine which fluent is true at each time moment. The problem addressed is the frame[1] rather than the ramification problem. However the work of Reiter sets the basis for encapsulating time in the situation calculus. In this paper, we propose a further extension of the situation calculus based on Reiter's proposal. Kakas [6, 7] proposed the language E which contains a set ϕ of fluents, a set of actions, and a partially ordered set of time points. E employs the following axiom schemas for the description of the world (assume L and F are fluents, T is a time point, A is an action and C is a set of fluents).

$$\begin{array}{l}
L \quad holds \quad at \quad T \\
A \quad happens \quad at \quad T \\
A \quad initiates \quad F \quad when \quad C, \\
A \quad terminates \quad F \quad when \quad C, \\
L \quad whenever \quad C \\
A \quad needs \quad C\,.
\end{array}$$

As we may observe, the third and fourth axioms are dynamic because they evaluate when an action executes, while the last two are static because they evaluate at each time moment.

In the example with the circuit we have the following dynamic axioms:

$$\begin{array}{llll}
toggle\_switch(s_1) & initiates & up(s_1) & when \ \neg up(s_1) \\
toggle\_switch(s_1) & initiates & \neg up(s_1) & when \ up(s_1) \\
toggle\_switch(s_1) & terminates & up(s_1) & when \ up(s_1) \\
toggle\_switch(s_1) & terminates & \neg up(s_1) & when \ \neg up(s_1) \\
toggle\_switch(s_2) & initiates & up(s_2) & when \ \neg up(s_2) \\
toggle\_switch(s_2) & initiates & \neg up(s_2) & when \ up(s_2) \\
toggle\_switch(s_2) & terminates & up(s_2) & when \ up(s_2) \\
toggle\_switch(s_2) & terminates & \neg up(s_2) & when \ \neg up(s_2)
\end{array}$$

[1] The problem of determining which predicates and functions are not affected when an action is executed is the **frame problem** [12].

Also we have the following static rules

$$\begin{array}{l} light \quad whenever \quad up(s_1) \wedge up(s_2) \\ \neg light \quad whenever \quad \neg up(s_1) \vee up(s_2) \end{array}$$

Assume that the initial situation of circuit at time 0 is

$$S_0 = \{up(s_1), \neg up(s_2), \neg light\}$$

At this time we have that

$$\begin{array}{l} up(s_1) \quad holds \quad at \quad 0 \\ \neg up(s_2) \quad holds \quad at \quad 0 \\ \neg light \quad holds \quad at \quad 0 \end{array}$$

Assume the following execution of actionas:

$$\begin{array}{l} toggle\_switch(s_2) \quad happens \quad at \quad 4 \\ toggle\_switch(s_1) \quad happens \quad at \quad 8 \end{array}$$

The language E is based in the idea of persistence. This mean that no fluent changes its true value until some action causes this change. In our example, the situation does not change until the time point 4. At time point 4 the following axioms will be evaluated

$$\begin{array}{l} toggle\_switch(s_2) \quad initiates \quad up(s_2) \quad when \quad \neg up(s_2) \\ toggle\_switch(s_2) \quad terminates \quad \neg up(s_2) \quad when \quad \neg up(s_2) \end{array}$$

Now the new situation is

$$S_1' = \{up(s_1), up(s_2), \neg light\}$$

Now the following static rule is evaluated

$$light \quad whenever \quad up(s_1) \wedge up(s_2)$$

The new situation now is

$$S_1 = \{up(s_1), up(s_2), light\}$$

The fluents which hold are

$$\begin{aligned}&up(s_1) \;\; holds \;\; at \;\; 4\\&up(s_2) \;\; holds \;\; at \;\; 4\\&light \;\; holds \;\; at \;\; 4\end{aligned}$$

The situation does not change until time point 8. Then the situation changes again.

In E, one cannot declare effects that persist over a time span as in the aforementioned example where, if someone drinks then s/he is drunk for the subsequent five hours. In order to achieve this, it is necessary for an action to occur after five hours. This means that the users must explicitly determine all the indirect and direct effects. Also, E cannot represent delayed effects, as e.g., if someone drinks alcohol then s/he becomes drunk half an hour later and remains drunk for the next five hours. We consider these assumptions rather strong and examine the problem in a strictly more general setting. The languages E works satisfactorily only when the world which described is based on the persistence of fluents (like the circuit).

Related also is the language VHDL integrated circuit design used for symbolic temporal behavior. This language is based on the assumption that fluents persist until their truth value is changed. This mean that in the case of driver we must describe the direct and indirect effects as follows:

$$\begin{aligned}&if \;\; (occur(drink(p),t) \quad then\\&\qquad drunk(p)\\&\qquad wait \quad for \;\; 5 \quad month\\&\qquad \neg(drunk(p)\,.\end{aligned}$$

As we observe from this description, the user must determine explicitly all direct and indirect effects. Also, the description becomes more difficult if an action can change the effects of another action. In the above example, assume that there is an action $drink\_glycose$ which has as direct effect $\neg drunk$. In that case, we need to describe an interrupt in order to ensured the effects of the second action. The language VHDL, as E, works satisfactory only when the world which described is based in the persistence of fluents.

5. Extended Situation Calculus

We extend the temporal situation calculus as follows:

- 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.

- We define functions $start(S)$ and $end(S)$ for situations. They return the time moments at which situation S begins and ends respectively.
- We define the functional fluent $f_\alpha(a)$ as $current\_moment - start(a)$, i.e., the duration of execution of action a until the present moment.
- Time is discrete and isomorphic to the set of natural numbers.
- Each fluent f is represented as $f(t')$, which means that the fluent f is true in the time interval $[current\_moment, current\_moment + t']$. $\neg f(t')$ means that the fluent f is false in the time interval $[current\_moment, current\_moment + t']$. As time progresses, the value of t' is decreased by one time unit.
- Actions are ordered as follows:

  For instantaneous actions

  $$\begin{array}{l} a_1 < a_2 < .... < a_n \quad , when \\ start(a_1) < start(a_2) < ..... < start(a_n) \end{array}$$

  Also, for instantaneous actions, $start(a) = end(a)$ holds. In this case, two actions a_1, a_2 will be executed concurrently when $start(a_1) = start(a_2)$ holds.

  For actions with duration, $a_1 < a_2$ when $end(a_1) < start(a_2)$. Two actions a_1, a_2 will be executed concurrently when $start(a_1) \leq start(a_2) \leq end(a_1) \leq end(a_2)$ holds.

  We assume that all actions which execute at the same time moment will be executed concurrently.
- We define the function do as $do \ : \ action^n \times situation \longrightarrow situation$. $do(\{a_1, a_2, .., a_n\}, S) = S_1$ means that the actions $a_1, a_2, .., a_n$ execute concurrently in the situation S and the result is the situation S_1.
- For two situations S_1, S_2, $S_1 < S_2$, when $end(S_1) \leq start(S_2)$. It is not necessarily the case that $S_2 = do(\{a_{i1}, a_{i2}, ...\}, ..., do(\{a_{j1}, ...\}, S_1)...)$.
- We extend predicate $poss(a, S)$ as follows:
  $poss(\{a_1, a_2, ...a_n\}, S) = \bigwedge_{i=1,\ldots,n} poss(a_i, S)$. This means that the actions $\{a_1, a_2, ...a_n\}$ can execute concurrently if and only if the preconditions of each action are true.
- We define as a legal (consistent) situation, a situation in which all integrity constraints are satisfied.

5.1. *Fundamental Axioms*

We use the axioms which have been defined by Reiter [20]

$$\begin{aligned} &S_0 \neq do(\{a_1, ..., a_n\}, S) && (1)\\ &do(\{a_1, ..., a_n\}, S) = do(\{a_1^{'}, ..., a_n^{'}\}, S^{'}) \supset \\ &\qquad\qquad \{a_1, ..., a_n\} = \{a_1^{'}, ..., a_n^{'}\} \wedge S = S^{'} && (2)\\ &start(a(t)) = t && (3)\,. \end{aligned}$$

Reiter has also defined one other axiom, namely

$$(\forall P).P(S_0) \wedge (\forall \{a_1, ..., a_n\}, S)\ [P(S) \supset\ P(do(\{a_1, ..., a_n\}, S)] \supset (\forall S)P(S)\,.$$

This is an inductive axiom which means that each situation is the result of the execution of a sequence of actions. Thus, if the initial situation is known we can determine which fluent is true in each situation. This axiom does not hold in our case because the transition from one situation to the next does not necessarily happen after the execution of an action. In order for the above axiom to hold, we define a natural action a_f for each fluent f. The only direct effect of the action a_f is that the fluent f becomes false ($f(0)$ [2]). This means that when an action a has as effect $f(10)$ the action a_f will execute 10 time moments later. Natural actions do not affect the world being modeled. They are employed to ensure that the transition from one situation to the next is the result of the execution some action (natural or not). The transition from one situation to the next happens when the truth value of at least one fluent changes. By the inclusion of natural actions no fluent can change its truth value without some action taking place.

6. Sequential execution

In this section we present a solution for the ramification problem in temporal databases, when the actions execute sequentially. This solution extends the solution which has been proposed by McCain and Turner [11].

Each action A is represented as $A(t)$ which mean that the action A executed at time t. Each fluent f represented as $f(t')$, which mean that the fluent f is true in the time interval $[curent\_moment, current\_moment + t']$. $\neg f(t')$ mean that the fluent f is false in the time interval $[curent\_moment, current\_moment + t']$. As time progresses, the value of t' is decreased by one time unit.

For each action A we define one axiom

$$A \supset \bigwedge L_i(t')\,,$$

[2]This mean that the fluent f is true for the 0 next time points. Thus is false

where $L_i(t')$ is $f_i(t')$ or $\neg f_i(t')$. The above axioms describe the direct effects of an action. For each fluent f we define two axioms

$$\begin{aligned} G(t,t') &\supset f(t') \\ B(t,t') &\supset \neg f(t')\,, \end{aligned}$$

where $G(t,t')$ is a formula which when true (at time point t) causes fluent f to become true at the next t' time moments (respectively for $B(t,t')$). These axioms encapsulate the indirect effects of an action. The former of axioms are dynamic because they are evaluated after the execution of an action, while the latter are static because evaluated every time point, at which the fluent is false.

We need $O(A + 2 * F)$ axioms, where A is the number of actions and F the number of fluents. Additional may be necessary to define one default axioms for each pair $(f, \neg f)$. The default axiom has one of the following forms: $f(0) \wedge \neg f(0) \supset f(t)$ or $f(0) \wedge \neg f(0) \supset \neg f(t)$. The default axioms evaluated when $f(0) \wedge \neg f(0)$ holds [3]. It is not necessary to define default axiom for each pair $(f, \neg f)$, if there is no case to hold $(f(0) \wedge \neg f(0))$.

Consider the following example: if a public employee commits a misdemeanor, then for the next five months s/he is considered illegal, except if s/he receive a pardon. When a public employee is illegal, then s/he must be suspended and cannot take promotion for the entire time interval over which s/he is considered illegal. Also when a public employee is suspended cannot take his/her salary until stop to be suspended. Each public employee is evaluated for his/her work. If s/he receive a bad grade, then s/he is assumed to be a bad employee and s/he cannot take promotion until s/he receive good grade. If s/he receive a good grade, then s/he is assumed to be a good employee and s/he a take bonus if s/he not suspended. Also assume that a public worker is not illegal if there does not exist information that proves s/he is illegal, is not suspended if there does not exist information that proves s/he is suspended and takes his/her salary if there does not exist information that proves the opposite. This helps us define the default axioms. As we observe we have four actions $misdemeanor, take\_pardon, good\_grade, bad\_grade$ and seven fluents $good\_employee$, $bad\_employee$, $illegal$, $take\_salary$, $take\_bonus$, $take\_promotion$, $suspended$. The direct effect of the four actions are expressed in propositional form by the following constraints[4]:

$$occur(misdemeanor(p), t) \supset illegal(p, 5m) \quad (1)$$
$$occur(take\_pardon(p), t) \supset \neg illegal(p, \infty) \quad (2)$$

[3]In that case mean that the fluent f and its negation are false. The default axioms determine which of two fluent f or $\neg f$ assumed true in that case

[4]Quantifiers are committed in the expression of these propositions. They are considered to be implicitly universally quantified over their temporal and non-temporal arguments.

$$occur(bad\_grade(p),t) \supset \neg good\_employee(p,\infty) \quad (3)$$
$$occur(good\_grade(p),t) \supset good\_employee(p,\infty) \quad (4),$$

where t is a temporal variable and the predicate $occur(misdemeanor(p),t)$ denotes that the action $misdemeanor(p)$ is executed at time t. Also we have the following integrity constraints which describe the indirect effects of the four actions.

$$illegal(p,t_1) \supset suspended(p,t_1) \quad (5)$$
$$illegal(p,t_1) \supset \neg take\_promotion(p,t_1) \quad (6)$$
$$suspended(p,t_1) \supset \neg take\_salary(p,t_1) \quad (7)$$
$$\neg good\_employee(p,t_1) \supset \neg take\_promotion(p,t_1) \quad (8)$$
$$\neg suspended(p,t_1) \wedge good\_employee(p,t_2) \supset take\_bonus(p,min(t_1,t_2) \; (9)$$
$$\neg good\_employee(p,t_1) \supset \neg take\_bonus(p,t_1) \quad (10)$$
$$\neg suspended(p,t_1) \supset take\_salary(p,t_1) \quad (11).$$

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. In the above example, the action $misdemeanor(p)$ has the indirect effect that the public worker is in suspension in the next five months. In these five months the action $good\_grade$ may occur but even if that happens the employee cannot take promotion. This means that the world changes situations while the direct and indirect effects of some action still hold. In the above example the dynamic axioms are the (1) - (4) while the static axioms are

$$illegal(p,t_1) \supset suspended(p,t_1)$$
$$illegal(p,t_1) \vee \neg good\_employee(p,t_2) \supset \neg take\_promotion(p,max(t_1,t_2)$$
$$suspended(p,t_1) \supset \neg take\_salary(p,t_1)$$
$$\neg suspended(p,t_1) \wedge good\_employee(p,t_2) \supset take\_bonus(p,min(t_1,t_2))$$
$$\neg good\_employee(p,t_1) \supset \neg take\_bonus(p,t_1)$$
$$\neg suspended(p,t_1) \supset take\_salary(p,t_1).$$

We have following default axioms

$$illegal(p,0) \wedge \neg illegal(p,0) \supset illegal(p,\infty)$$
$$take\_salary(p,0) \wedge \neg take\_salary(p,0) \supset take\_salary(p,\infty)$$
$$suspended(p,0) \wedge \neg suspended(p,0) \supset suspended(p,\infty).$$

6.1. *Algorithms for the production of static rules*

The static rules encapsulate the indirect effects of the execution each action. The indirect effect exist because of the presence integrity constraints. It is reasonable to produce the static rules from the integrity constraint. Static rules are produces as follow:

1. Transform each integrity constraint in its CNF form. Now each integrity constraint has the form $C_1 \wedge C_2 \wedge C_3 \ldots\ldots \wedge C_n$, where each C_i is a disjoinet.

2. For each i from 1 to n do

   (a) assume $C_i = f_1 \vee \ldots. \vee f_m$
   For each j from 1 to m do
   For each k from 1 to m, and $k \neq j$, do
   if $(f_j, f_k) \in I$ then
   $R = R \vee (\neg f_j \quad causes \quad f_k \quad if \quad \bigwedge \neg f_l), l = 1, ..m, l \neq j, k.$

3. For each fluent F_k the rules have the following form

$$\bigwedge F_i \quad causes \quad F_k \quad if \quad \Phi$$
$$\bigwedge F_i' \quad causes \quad \neg F_k \quad if \quad \Phi' \, .$$

   We change the static rules from the form:

$$\begin{array}{l} G \supset F_k \\ K \supset \neg F_k \\ to \\ G' \supset F_k \\ K' \supset \neg F_k \\ where \\ G' = G \vee (\bigwedge F_i \wedge \Phi) \\ K' = K \vee (\bigwedge F_i' \wedge \Phi') \, . \end{array}$$

4. For each rule $G_p \supset f_p$, we replace each fluent f with $f(t)$, as we have defined above. The static rule has the form $G_p(t) \supset f_p(1)$.

The proposition $G_{f_p}(t)$ could contain information which permit us to understand that the fluent f_p is true for a time interval greater than one time unit. We change the static rules in order to encapsulate the above observation. The rules change from $G_{f_p}(t) \supset f_p(1)$ to $G_{f_p}(t, t') \supset f_p(t')$, where $G_{f_p}(t, t')$ means that, if G_{f_p} is true at time moment t, then the fluent f_p is true for the next t' time units:

```
1. For each static rule $G(t) \supset f$ do
    (a) let $G = G_1 \vee .... \vee G_n$
    (b) For each j from 1 to n do
        • let $G_i = f_1(t_1) \wedge .... \wedge f_n(t_n)$
        • let $t = min(t_1, ..., t_n)$
        • replace $G_i$ with $G_i(t)$
    (c) $t' = max(t_i : G_i \quad is \quad true)$
    (d) replace $G(t)$ with $G(t, t')$
```

Notice that the first four step are static and execute one time at the start. The five step is executed dynamicaly at each time point at which the static rule must be executed. This happened because the formula $G_{f_p}(t, t')$ can be true for different value of t and t'.

Now we show how the algorithm works in the above example. First, we must construct the set I using the algorithm which has been proposed in section 2. All the integrity constraints(IC) have the form $A \supset B$. We have that

$$(illegal, suspended) \in I \quad (from \quad IC \quad 5)$$
$$(illegal, \neg take\_promotion) \in I \quad (from \quad IC \quad 6)$$
$$(suspended, \neg take\_salary) \in I \quad (from \quad IC \quad 7)$$
$$(\neg good\_employee, \neg take\_promotion) \in I \quad (from \quad IC \quad 8)$$
$$(\neg suspended, \neg take\_bonus) \in I \quad (from \quad IC \quad 9)$$
$$(good\_employee, \neg take\_bonus) \in I \quad (from \quad IC \quad 9)$$
$$(\neg good\_employee, take\_salary) \in I \quad (from \quad IC \quad 10)$$
$$(\neg suspended, take\_salary) \in I \quad (from \quad IC \quad 11)$$

The transformation of integrity constraints in CNF form yields:

$$illegal(p, t_1) \vee \neg suspended(p, t_1) \quad (5)$$
$$illegal(p, t_1) \vee take\_promotion(p, t_1) \quad (6)$$
$$suspended(p, t_1) \vee take_salary(p, t_1) \quad (7)$$
$$\neg good\_employee(p, t_1) \vee take\_promotion(p, t_1) \quad (8)$$
$$\neg suspended(p, t_1) \wedge good\_employee(p, t_2) \vee \neg take\_bonus(p, t_3) \quad (9)$$
$$\neg good\_employee(p, t_1) \vee take\_bonus(p, t_1) \quad (10)$$
$$\neg suspended(p, t_1) \vee take\_salary(p, t_1) \quad (11)$$

This is step 1 of the algorithm. In the step 2 we have

$$
\begin{aligned}
&R = \{illegal(p) \quad causes \quad suspended(p) \quad if \quad T,\\
&illegal(p) \quad causes \quad \neg take\_promotion(p) \quad if \quad T,\\
&suspended(p) \quad causes \quad \neg take\_salary(p) \quad if \quad T,\\
&\neg good\_employee(p) \quad causes \quad \neg take\_promotion(p) \quad if \quad T,\\
&good\_employee(p) \quad causes \quad take\_bonus(p) \quad if \quad \neg suspended(p),\\
&\neg suspended(p) \quad causes \quad take\_bonus(p) \quad if \quad good\_employee(p),\\
&\neg good\_employee(p) \quad causes \quad \neg take\_bonus(p) \quad if \quad T,\\
&\neg suspended(p) \quad causes \quad take\_salary(p) \quad if \quad T\}\,.
\end{aligned}
$$

In the step 2 we estimate all causal relationships. In the step 3 we have that

$$
\begin{aligned}
&R = \{illegal(p) \supset suspended(p),\\
&illegal(p) \vee \neg good\_employee(p) \supset \; \neg take\_promotion(p),\\
&suspended(p) \supset \neg take\_salary(p),\\
&\neg suspended(p) \wedge good\_employee(p) \supset take\_bonus(p),\\
&\neg good\_employee(p) \supset take\_bonus(p),\\
&\neg suspended(p) \supset take\_salary(p)\}
\end{aligned}
$$

In step 3 we construct for each fluent the fluents formula which makes the fluent true. Notice that perhaps there are many causal relationships which affect the same fluents. We integrate these causal relationship in this step. For example see the second and fourth causal relationships from step 2. In step 4

$$
\begin{aligned}
&R = \{illegal(p, t_1) \supset suspended(p, 1),\\
&illegal(p, t_1) \vee \neg good\_employee(p, t_2) \supset \; \neg take\_promotion(p, 1)),\\
&suspended(p) \supset \neg take\_salary(p, 1),\\
&\neg suspended(p, t_1) \wedge good\_employee(p, t_2) \supset take\_bonus(p, 1),\\
&\neg good\_employee(p, t_1) \supset \neg take\_bonus(p, 1),\\
&\neg suspended(p, t_1) \supset take\_salary(p, 1)\}
\end{aligned}
$$

Finally in step 5 (which must executed at each time point at which the static rules evaluate) we have the following four static rules:

$$
\begin{aligned}
&R = \{illegal(p, t_1) \supset suspended(p, t_1),\\
&illegal(p, t_1) \vee \neg good\_employee(p, t_2) \supset \; \neg take\_promotion(p, max(t_1, t_2))\\
&suspended(p) \supset \neg take\_salary(p, t_1),
\end{aligned}
$$

$$\neg suspended(p,t_1) \wedge good\_employee(p,t_2) \supset take\_bonus(p, min(t_1,t_2)),$$
$$\neg good\_employee(p,t_1) \supset \neg take\_bonus(p,t_1),$$
$$\neg suspended(p,t_1) \supset take\_salary(p,t_1)\}$$

In step 5, for each static rule, we find the maximum time that its body is true. For example in the second rule, the body is true when $illegal(t_1) \vee good\_employee(t_2)$ is true. This mean that we take the maximum of times t_1, t_2 for which $good\_employee$ is true. For the fourth rule, the body is true when $\neg suspended(t_1) \wedge good\_employee(t_2)$ is true. This means that we must take the minimum of times t_1, t_2.

6.2. *Algorithms for the evaluation of Dynamical and Static rules*

In this section we presented an algorithm for the evaluation of rules

1. After the execution one action evaluate the dynamic rule which reference at this action.
2. Each time moment do
   (a) Evaluate the default axioms
   (b) Repeat until no change occurs.
      i. Evaluate all static rules.
      ii. If a fluent $f(t)$ becomes true after an evaluation of a static rule, then set $\neg f(0)$. (the negation is false).

Consider the above example with the public worker. We have four dynamic rules (1-4) as we have described in the previous section. Also we have produced the six static rules

$$illegal(p,t_1) \supset suspended(p,t_1),$$
$$illegal(p,t_1) \vee \neg good\_employee(p,t_2) \supset \neg take\_promotion(p, max(t_1,t_2)),$$
$$suspended(p,t_1) \supset \neg take\_salary(p,t_1),$$
$$\neg suspended(p,t_1) \wedge good\_employee(p,t_2) \supset take\_bonus(p, min(t_1,t_2))$$
$$\neg good\_employee(p,t_1) \supset \neg take\_bonus(p,t_1),$$
$$\neg suspended(p,t_1) \supset take\_salary(p,t_1).$$

Assume now that we have a public worker p and the initial situation is

$$S_0 = \{\neg take\_bonus(p,\infty), take\_salary(p,\infty), \neg take\_promotion(p,\infty),$$
$$\neg suspended(p,\infty), \neg good\_employee(p,\infty), \neg illegal(p,\infty)\}\,.$$

Time start at 0 and time has the granularity of month. Assume that the following actions occur at the following time points:

$$
\begin{aligned}
&occur(good\_grade(p), 2)\\
&occur(misdemeanor(p), 4)\\
&occur(bad\_grade(p), 6)\\
&occur(good\_grade(p), 8)\\
&occur(misdemeanor(p), 10)\\
&occur(take\_pardon(p), 12)\,.
\end{aligned}
$$

At time point 2 the action $good\_grade(p)$ executes. From the algorithm for the evaluation dynamic and static rules, after the evaluation of dynamic rule (4) we have the situation

$$
\begin{aligned}
&S'_1 = \{\neg take\_bonus(p, \infty), take\_salary(p, \infty),\ \neg take\_promotion(p, \infty),\\
&\neg suspended(p, \infty), good\_employee(p, \infty), \neg illegal(p, \infty)\}\,.
\end{aligned}
$$

As we observe the following static rule

$$\neg suspended(p, \infty) \wedge good\_employee(p, t_2) \supset\ take\_bonus(p, \infty)\,,$$

will be evaluated. After the evaluation of static rule we have the situation

$$
\begin{aligned}
&S_1 = \{take\_bonus(p, \infty), take\_salary(p, \infty),\ \neg take\_promotion(p, \infty),\\
&\neg suspended(p, \infty), good\_employee(p, \infty), \neg illegal(p, \infty)\}\,.
\end{aligned}
$$

This situation does not change until time point 4, when the second action ($misdemeanor(p)$) take place. From the algorithm for the evaluation of dynamic and static rules, after the evaluation of dynamic rule (1) we have the situation

$$
\begin{aligned}
&S'_2 = \{take\_bonus(p, \infty), take\_salary(p, \infty),\ \neg take\_promotion(p, \infty),\\
&\neg suspended(p, \infty), good\_employee(p, \infty), illegal(p, 5)\}\,.
\end{aligned}
$$

As we observe the following static rules will be evaluated

$$
\begin{aligned}
&illegal(p, 5) \supset suspended(p, 5)\\
&suspended(p, 5) \supset \neg take\_salary(p, 5)
\end{aligned}
$$

After the evaluation of static rules we have the situation

$$
\begin{aligned}
&S_2 = \{take\_bonus(p, \infty), \neg take\_salary(p, 5),\ \neg take\_promotion(p, \infty),\\
&suspended(p, 5), good\_employee(p, \infty), illegal(p, 5)\}\,.
\end{aligned}
$$

This situation does not change until the time point 6, when the third action ($bad\_grade(p)$) executes. From the algorithm for the evaluation of dynamic and static rules, after the evaluation of dynamic rule (3) we have the situation

$$S_3' = \{take\_bonus(p,\infty), \neg take\_salary(p,5),\ \neg take\_promotion(p,\infty),$$
$$suspended(p,5), \neg good\_employee(p,\infty), illegal(p,5)\}\,.$$

The following static rule

$$\neg good\_employee(p,\infty) \supset \neg take\_bonus(p,\infty)$$

will be evaluated, after the evaluation of static rule we have the situation

$$S_3 = \{\neg take\_bonus(p,\infty), \neg take\_salary(p,3),\ \neg take\_promotion(p,\infty),$$
$$suspended(p,3), \neg good\_employee(p,\infty), illegal(p,3)\}\,.$$

This situation does not change until the time point 6, when the fourth action ($good\_grade$) take place. From the algorithm for the evaluation of dynamic and static rules after the evaluation of dynamic rule (4) we have the situation

$$S_4' = \{\neg take\_bonus(p,\infty), \neg take\_salary(p,1),\ \neg take\_promotion(p,\infty),$$
$$suspended(p,1), good\_employee(p,\infty), illegal(p,1)\}\,.$$

None static rule executed. Thus the situation does not change. At time point 9 does not executed any action but the situation change because $take\_salary(p,0) \wedge take\_salary(p,0)$, $suspend(p,0) \wedge \neg suspend(p,0)$ and $illegal(p,0) \wedge \neg illegal(p,0)$ holds. Thus the following default axioms are evaluated

$$\neg suspended(p,0) \wedge suspended(p,0) \supset \neg suspended(p,\infty)$$
$$\neg take\_salary(p,0) \wedge take\_salary(p,0) \supset take\_salary(p,\infty)$$
$$illegal(p,0) \wedge \neg illegal(p,0) \supset \neg illegal(p,\infty)$$

The new situation is

$$S_5' = \{\neg take\_bonus(p,\infty), take\_salary(p,\infty),\ \neg take\_promotion(p,\infty),$$
$$\neg suspended(p,\infty), good\_employee(p,\infty), \neg illegal(p,\infty)\}\,.$$

Now the static rule

$$\neg suspended(p,\infty) \wedge good\_employee(p,\infty) \supset \ take\_bonus(p,\infty)\,,$$

will be evaluated. After the evaluation of the static rule the situation is

$$S_5 = \{take\_bonus(p,\infty), take\_salary(p,\infty),\ \neg take\_promotion(p,\infty),$$
$$\neg suspended(p,\infty), good\_employee(p,\infty), \neg illegal(p,\infty)\}\,.$$

At time point 10 the action $misdemeanor(p)$ executes, thus the situation changes in

$$S'_6 = \{take\_bonus(p,\infty), take\_salary(p,\infty),\ \neg take\_promotion(p,\infty),$$
$$\neg suspended(p,\infty), good\_employee(p,\infty), illegal(p,5)\}\,.$$

Now the following static rules

$$illegal(p,5) \supset suspended(p,5)$$
$$suspended(p,5) \supset \neg take\_salary(p,5)$$

will be evaluated. After the evaluation of static rules the situation is

$$S_6 = \{take\_bonus(p,\infty), \neg take\_salary(p,5),\ \neg take\_promotion(p,\infty),$$
$$suspended(p,5), good\_employee(p,\infty), \neg illegal(p,5)\}\,.$$

The last action ($take\_pardon$) occur at time point 12. The new situation is

$$S'_7 = \{take\_bonus(p,\infty), \neg take\_salary(p,3),\ \neg take\_promotion(p,\infty),$$
$$suspended(p,3), good\_employee(p,\infty), \neg illegal(p,\infty)\}\,.$$

Finally the situation changes again at time point 15, because $\neg suspended(p,0)$ $\wedge suspended(p,0)$ and $\neg take\_salary(p,0) \wedge take\_salary(p,0)$ and holds. Thus the following default axioms are evaluated

$$\neg suspended(p,0) \wedge suspended(p,0) \supset \neg suspended(p,\infty)$$
$$\neg take\_salary(p,0) \wedge take\_salary(p,0) \supset take\_salary(p,\infty)$$

Now the new situation is

$$S_8 = \{take\_bonus(p,\infty), take\_salary(p,\infty),\ \neg take\_promotion(p,\infty),$$
$$\neg suspended(p,\infty), good\_employee(p,\infty), \neg illegal(p,\infty)\}\ .$$

This is the end of execution.

As we observe from the set R, for each pair $(f, \neg f)$ it holds that $G_f \wedge K_f \equiv FALSE$, when $G_f \supset f,\ \ K_f \supset \neg f$. More specifically the set of static rules is

$$\begin{aligned}
&R = \{illegal(p,t_1) \supset suspended(p,t_1),\\
&illegal(p,t_1) \vee \neg good\_employee(p,t_2) \supset\ \neg take\_promotion(p, max(t_1,t_2))\\
&suspended(p,t_1) \supset \neg take\_salary(p,t_1),\\
&\neg suspended(p,t_1) \wedge good\_employee(p,t_2) \supset take\_bonus(p, min(t_1,t_2)),\\
&\neg good\_employee(p,t_1) \supset \neg take\_bonus(p,t_1),\\
&\neg suspended(p,t_1) \supset take\_salary(p,t_1),\\
&good\_employee(p,t_1) \supset take\_promotion(p,t_1),\\
&False \supset \neg suspended(p,t_1),\\
&False \supset take\_promotion(p,t_1),\\
&False \supset illegal(p,t_1),\\
&False \supset \neg illegal(p,t_1), \}
\end{aligned}$$

As we observe, for the fluents that there is not static rule we add the rule $false \supset f$, because the cannot become true by the static rule but only by the dynamic rules (this mean that the truth value change only as direct effect of some action). Now we have

$$\begin{aligned}
&(\neg suspended(p,t_1) \wedge good\_employee(p,t_2)) \wedge \neg good\_employee(p,t_1)\\
&\qquad\qquad for\ \ (take\_bonus, \neg take\_bonus)\\
&suspended(p,t_1) \wedge \neg suspended(p,t_1)\\
&\qquad\qquad for\ \ (take\_salary(p,t_1), \neg take\_salary(p,t_1))\\
&illegal(p,t_1) \wedge False \quad for\ \ (suspended(p,t_1), \neg suspended(p,t_1))\\
&illegal(p) \vee \neg good\_employee(p) \wedge False\\
&\qquad\qquad for\ \ (\neg take\_promotion(p), take\_promotion(p)\\
&False \wedge False\ \ for\ \ (illegal(p), \neg illegal(p))
\end{aligned}$$

This assumption $G_f \wedge K_f \equiv FALSE$ is very important in order to ensure that, always, after the execution of action there is a consistent situation. Now we show with an example that if this assumption does not holds there is no consistent situation after some sequence of some actions execution.

Consider the above example with the public worker and assume that there is another integrity constraints specifying that when a public worker is good employee then s/he take promotion. Now the set of static rules is

$$\begin{aligned}
&R = \{illegal(p,t_1) \supset suspended(p,t_1),\\
&illegal(p,t_1) \vee \neg good\_employee(p,t_2) \supset \ \neg take\_promotion(p, max(t_1,t_2)\\
&suspended(p,t_1) \supset \neg take\_salary(p,t_1),\\
&\neg suspended(p,t_1) \wedge good\_employee(p,t_2) \supset take\_bonus(p, min(t_1,t_2)),\\
&\neg good\_employee(p,t_1) \supset \neg take\_bonus(p,t_1),\\
&\neg suspended(p,t_1) \supset take\_salary(p,t_1),\\
&good\_employee(p,t_1) \supset take\_promotion(p,t_1)\}
\end{aligned}$$

As we observe for the pair $(take\_promotion(p), \neg take\_promotion(p)$ the above assumption does not hold, because

$$good\_employee(p,t_1) \wedge \ (illegal(p,t_1) \vee \neg good\_employee(p,t_2))$$

can be true when $good\_employee(p) \wedge illegal(p)$ holds.

Assume now that we have a public worker p and the initial situation is

$$\begin{aligned}
&S_0 = \{\neg take\_bonus(p,\infty), take\_salary(p,\infty), \neg take\_promotion(p,\infty),\\
&\neg suspended(p,\infty), \neg good\_employee(p,\infty), \neg illegal(p,\infty)\}\ .
\end{aligned}$$

Assume that the following actions occur at the following time points, assuming time starts at 0 and time granularity is that of months.

$$\begin{aligned}
&occur(misdemeanor(p), 4)\\
&occur(good\_grade(p), 6)\ .
\end{aligned}$$

At time point 4 after the execution of the action $misdemeanor(p)$ we have the situation

$$\begin{aligned}
&S_1' = \{\neg take\_bonus(p,\infty),\ take\_salary(p,\infty), \neg take\_promotion(p,\infty),\\
&\neg suspended(p,\infty), \neg good\_employee(p,\infty),\ illegal(p,\infty),\}
\end{aligned}$$

After the evaluation of the static rules we have

$$\begin{aligned}
&S_1 = \{\neg take\_bonus(p,\infty),\ \neg take\_salary(p,\infty), \neg take\_promotion(p,\infty),\\
&suspended(p,\infty), \neg good\_employee(p,\infty),\ illegal(p,\infty),\}
\end{aligned}$$

At time point 6, after the execution of the action $good\_grade(p)$ we have the situation

$$S_2' = \{\neg take\_bonus(p,\infty),\ \neg take\_salary(p,\infty), \neg take\_promotion(p,\infty), \\ suspend(p,\infty), good\_employee(p,\infty),\ illegal(p,\infty),\}$$

Now the static rule $good\_employee(p,\infty) \supset take\_promotion(p,\infty)$ must be evaluated and after that $take\_promotion(p,\infty)$ must hold. But if $take\_promotion(p,\infty)$ holds then we must examine if the static rule $illegal(p,\infty) \vee \neg good\_employee(p,t_1) \supset \neg take\_promotion(p,\infty)$ must be evaluated. We observe that we must evaluated this static rule. As we observe that the two above static rules will be evaluated one after the other for ever (infinitive). This mean that there is no consistent situation. This happened because there is a mistake in the integrity constraints, which has as result the above assumption does not holds.

The algorithm can run without the above assumption but we must determine the preconditions of each action in order to avoid the above problem.

The following theorem establishs the termination of the algorithm.

Theorem 1 *Each time unit, the algorithms terminated at finite number of step.*

Proof: Assume that at time unit t the algorithm does not terminate. Then, there must be an infinite loop. Assume that S_t^0 is the initial situation at time t. Then, there is a non terminating sequence $S_t^0, S_t^1, ...S_t^k, ...$ If F is the number of fluents then there are 2^F different situations. This means that in the above sequence, there are two identical situations because of the infinite loop. Without loss of generality we assume $S_t^l = S_t^k$. In the sequence $S_t^l, ..., S_t^k$ there is at least one fluent f which changes from f to $\neg f$ and eventually becomes f again. Assume that f' is one such fluent.

Assume that

$$G(t,t') \supset f'(t')$$
$$B(t,t'') \supset \neg f'(t'')$$

Assume that first f' holds. Then we must evaluate the rule $B(t,t'') \supset \neg f'$ and after the $G(t,t') \supset f'$. This means that at time t the proposition $G \wedge B$ must be true. But the conditions G and B are mutually exclusive. A contradiction. $\Box$

The following theorem establishes that we always end up with a consistent situation.

Theorem 2 *The above algorithm return always a legal situation.*

Proof: In order to be correct the algorithm must always terminated in a consistent situation. This means that all integrity constraints must be satisfied at this situation.

Assume that integrity constraint Law_j is not satisfied in one situation. Assume that the CNF of this law is $C_1 \wedge .... \wedge C_n$. Then it must be the case that one of the $C_1, ..., C_n$ is false. Assume that $C_i = f_1 \vee .... \vee f_m$ is false. Then all fluents $f_j, j = 1, ..m$ are false. Assume that f_k and f_p are two of these for which $(f_k, f_p) \in I$. Then for f_p it must be the case that: $G_p(t) = G' \vee (\neg \bigwedge f_j(t_j, j = 1, ..m, j \neq p)$. If all fluents $f_j, j = 1, ..m$ are false then $(\neg \bigwedge f_j, j = 1, ..m, j \neq p)$ is true. Thus $G_p(t)$ is true. This means that the static rule $G_p(t) \supset f_p(1)$ must be evaluated and thus, the f_p is true. A contradiction.

The above proof is independent of the set I. Thus, the algorithm can capture all the indirect effects of the actions and constraints.

□

6.3. *The ramification problem when the direct and indirect effects of an action refer only to future situations*

The ramification problem becomes more complex when the direct and indirect effects of an action do not hold for the next time moment but after some time moments. For example, assume that in the above example the action *good_grade* has as direct effect to characterize the public worker as good employee after two months (respectively for the action *bad_grade*). In that case the above representation of fluents cannot encapsulate the direct and indirect effects of the actions *good_grade*, *bad_grade*.

We change the representation of fluents as follows: each fluent F is represented as $F(L)$, where $L = [[t, t'], ...]$ is a list. Each member of the list is a time interval $[t, t']$, which means that the fluent is true at time interval $[t, t']$. At each time moment t'' we reduce for the list the time intervals which refer in the past($t'' > t'$). Now the rules (dynamic and static) change in order to encapsulate the above change.

The static rules are produced from the same algorithm as previous except from step five which change as follows: At each time moment it which it is necessary to evaluate a static rule, before the evaluation of the rule execute the following algorithm

```
1. At time moment $t$, for the static rule $G(t, t_1) \supset f$ do
   (a) let $G = G_1 \vee .... \vee G_n$
   (b) For each j from 1 to n do

         • let $G_i = f_1([..]) \wedge .... \wedge f_n([..])$
            i. for each fluent $f_i(L)$ take the first element $[t', t'']$ of the list
               $L$.
           ii. if $t' > t$ then $G$ is false and terminated.
          iii. else $t_i = t'' - t$.
```

- let $t_{min} = min(t_1, ..., t_n)$
- replace G_i with $G_i(t, t + t_{min})$

When a static rule $G(t,t^{'}) \supset f(L)$ evaluates the element $[t,t^{'}]$ is added in the list L and is removed from the L', where $\neg f(L')$ [5]. The algorithm for adding an element $[t,t^{'}]$ add in the list L and removing it from L' is:

1. Assume f(L). L is the list at which we add the element $[t,t^{'}]$.
2. if there is an element $[t_{i1}, t_{i2}]$ for which $t_{i1} < t < t_{i2} < t^{'}$ hold then remove it and add $[t_{i1}, t^{'}]$.
3. if there is an element $[t_{i1}, t_{i2}]$ for which $t < t_{i1} < t^{'} < t_{i2}$ hold then remove it and add $[t, t_{i2}]$
4. else add $[t,t^{'}]$.
5. Assume $\neg f(L')$. L' is the list from which we remove the element $[t,t^{'}]$.
6. if there is an element $[t_{i1}, t_{i2}]$ for which $t < t_{i1} < t_{i2} < t^{'}$ hold then remove it.
7. if there is an element $[t_{i1}, t_{i2}]$ for which $t_{i1} < t < t^{'} < t_{i2}$ then remove it and add $[t_{i1}, t]$ and $[t^{'}, t_{i2}]$.
8. if there is an element $[t_{i1}, t_{i2}]$ for which $t < t_{i1} < t^{'} < t_{i2}$ then remove it and add $[t^{'}, t_{i2}]$.
9. if there is an element $[t_{i1}, t_{i2}]$ for which $t_{i1} < t < t_{i2} < t^{'}$ then remove it and add $[t_{i1}, t]$.

The above algorithm can be used for the evaluation of dynamic rules, too. The algorithm for evaluating the dynamic and static rules does not change, except that now there is no need to evaluate the default axioms. Notice that the algorithm which estimates the indirect effects of an action, does that at the time that they start to hold. This means that if some action has direct effects which refer only in the future, the indirect effects are not estimated at the time of action execution but at the time that the direct effects start to hold. This have the advandange that if in the interval time (between the execution of the action and the time point at which the effects start to hold) occur some action which cancel the direct effects of the first action it is not necessary to estimate the indirect effects twice.

Theorem 3 *The algorithm for evaluating the dynamic and static rules return a legal situation when the direct and indirect effects refer only to future situation.*

[5]The list L' is the list, which contains the time intervals, in which the fluent $\neg f$ is true

Proof: The proof is similar with the proof of the theorem 2. The proof is similar because the algorithm which estimate the indirect effects of an action, does that at the time that they start to hold.

□

Consider the example with the public worker but with the new assumptions (which we do at the begin of the section) for the actions $good\_grade, bad\_grade$. Assume that the direct effect of the actions $misdemeanor, take\_pardon$ hold for the current moment of the execution of actions. Now the dynamic rules are:

$$\begin{aligned}
&occur(misdemeanor(p), t) \supset illegal(p, [t, t+5m]) \quad (1')\\
&occur(take\_pardon(p), t) \supset \neg illegal(p, [t, \infty]) \quad (2')\\
&occur(bad\_grade(p), t) \supset \neg good\_employee(p, [t+2, \infty]) \quad (3')\\
&occur(good\_grade(p), t) \supset good\_employee(p, [t+2, \infty]) \quad (4')\,.
\end{aligned}$$

Now assume that the initial situation is:

$$\begin{aligned}
&S_0 = \{\neg take\_bonus(p, [[0, \infty]]), take\_salary(p, [[0, \infty]]),\\
&\neg take\_promotion(p, [[0, \infty]]),\ \neg suspended(p, [[0, \infty]]),\\
&\neg good\_employee(p, [[0, \infty]]), \neg illegal(p, [[0, \infty]])\}\,.
\end{aligned}$$

Assume that the following actions occur at the following time points, assuminig time starts at 0 and time granularity is that of months.

$$\begin{aligned}
&occur(good\_grade(p), 2)\\
&occur(misdemeanor(p), 4)\\
&occur(bad\_grade(p), 8)\,.
\end{aligned}$$

At time point 2 the first action will be executed. After the execution the new situation is

$$\begin{aligned}
&S_1 = \{\neg take\_bonus(p, [[0, \infty]]), take\_salary(p, [[0, \infty]]),\\
&\neg take\_promotion(p, [[0, \infty]]),\ \neg suspended(p, [[0, \infty]]),\\
&\neg good\_employee(p, [[2, 4]]),\ good\_employee(p, [[4, \infty]]),\ \neg illegal(p, [[0, \infty]])\}\,.
\end{aligned}$$

As we observe at time point 2 no static rule will evaluate (while in the previous section example executed) because the effects of action $good\_grade(p)$ will holds two time points latter.

At time point 4 the second action will be executed and the effects of the first action start to hold. Now the new situation is

$$\begin{aligned}&S_2' = \{\neg take\_bonus(p, [[0,\infty]]),\ take\_salary(p, [[0,\infty]]),\\ &\neg take\_promotion(p, [[0,\infty]]),\ \neg suspended(p, [[0,\infty]]),\\ &good\_employee(p, [[4,\infty]]),\ illegal(p, [[4,9]]),\ \neg illegal(p, [[9,\infty]])\}\,.\end{aligned}$$

As we observe the $\neg good\_employee(p, [[2,4]])$ does not hold at time point 4 and thus we remove it. [6] The following static rules will be evaluated

$$\begin{aligned}&illegal(p, [4,9]) \supset suspended(p, [4,9])\\ &suspended(p, [4,9]) \supset \neg take\_salary(p, [4,9])\end{aligned}$$

After the evaluation of the static rules we have the situation

$$\begin{aligned}&S_2 = \{\neg take\_bonus(p, [[0,\infty]]), \neg take\_salary(p, [[4,9]]),\\ &take\_salary(p, [[9,\infty]]),\ \neg take\_promotion(p, [[0,\infty]]),\\ &suspended(p, [[4,9]]),\ \neg suspended(p, [[9,\infty]]),\\ &good\_employee(p, [[4,\infty]]),\ illegal(p, [[4,9]]),\ \neg illegal(p, [[9,\infty]])\}\,.\end{aligned}$$

As we observe the fluents $suspended(p, [[4,9]])$, $\neg suspended(p, [[9,\infty]])$ and $\neg take\_salary(p, [[4,9]]), take\_salary(p, [[9,\infty]])$ and $illegal(p, [[4,9]])$, $\neg illegal(p, [[9,\infty]])$ encapsulate the default axioms.

At time point 8 the third action will be executed. After the execution the new situation is

$$\begin{aligned}&S_3 = \{\neg take\_bonus(p, [[0,\infty]]), \neg take\_salary(p, [[4,9]]),\\ &take\_salary(p, [[9,\infty]]),\ \neg take\_promotion(p, [[0,\infty]]),\\ &suspended(p, [[4,9]]), \neg suspended(p, [[9,\infty]]),\\ &good\_employee(p, [[4,10]]),\ \neg good\_employee(p, [[10,\infty]]),\\ &illegal(p, [[4,9]]),\ \neg illegal(p, [[9,\infty]])\}\,.\end{aligned}$$

At time point 9 no action takes place but the situation changes, because the fluent $illegal(p, [[4,9]])$, $suspended(p, [[4,9]])$ and $\neg take_salary(p, [[4,9]])$ cease to hold.

$$\begin{aligned}&S_4' = \{\neg take\_bonus(p, [[0,\infty]]), take\_salary(p, [[9,\infty]]),\\ &\neg take\_promotion(p, [[0,\infty]]),\ \neg suspended(p, [[9,\infty]]),\\ &good\_employee(p, [[4,10]]),\ \neg good\_employee(p, [[10,\infty]]),\ \neg illegal(p, [[9,\infty]])\}\,.\end{aligned}$$

[6] We observe with this representation of fluents we can capture the meaning of default axioms. For example the fluent $illegal$, $\neg illegal$.

Now the static rule

$$\neg suspended(p, [9, \infty]) \wedge good\_employee(p, [9, 10]) \supset \ take\_bonus(p, [9, 10])$$

must evaluated. Thus the new situation at time point 9 is

$$\begin{aligned}
&S_4 = \{take\_bonus(p, [[9, 10]]), \neg take\_bonus(p, [[10, \infty]]),\\
&take\_salary(p, [[9, \infty]]),\ \neg take\_promotion(p, [[0, \infty]]),\\
&\neg suspended(p, [[9, \infty]]),\ good\_employee(p, [[4, 10]]),\\
&\neg good\_employee(p, [[10, \infty]]), \neg illegal(p, [[9, \infty]])\}\,.
\end{aligned}$$

At time point 10 no action will execute but the situation changes as follows:

$$\begin{aligned}
&S_5 = \{\neg take\_bonus(p, [[10, \infty]]), take\_salary(p, [[9, \infty]]),\\
&\neg take\_promotion(p, [[0, \infty]]),\ \neg suspended(p, [[9, \infty]]),\\
&\neg good\_employee(p, [[10, \infty]]),\ \neg illegal(p, [[9, \infty]])\}\,.
\end{aligned}$$

The transition from the situation S_4 to S_5 happens because $good\_employee(p, [[4, 10]])$ and $take\_bonus(p, [9, 10])$ cease to hold at time point 10.

6.4. *The ramification problem when actions have duration*

In the case that the actions have duration then all effects must be determined with reference to the start, the end and the duration of the actions. If all direct and indirect effects can be described by reference to the start and the end of the action then we can assume that one action with duration is equivalent with two instantaneous actions: one for the start and one for the end. In that case the dynamic rules must be defined for the instantaneous actions. The above algorithms solve the ramification problem without change. But as we show below this is a very strict assumption.

In the case that the effects of an action depende on its duration, the above approach cannot address the ramification problem. Consider the example with the public worker and assume that if the action *good_grade* has duration more than two time moments then it has as effect the promotion of the employee.

Usually the duration of an action is unknown before its end. So we cannot describe the direct and indirect effects of an action with reference to the start and to the end. We must change the dynamic and static rules.

The fluents representation does not change. For each action a we define a new functional fluent $f_\alpha(a)$ which returns the duration of the execution of action a until the current moment. If this fluent return 0 the action does executed at current moment.

The fluent f_α helps us to determine the indirect effects of an action which depend on the duration of the action a. All direct effects of an action do not depended from the duration of execution. We must change the static rules in order to encapsulate the fluents f_a. The following algorithm implement this change.

```
1. At each static rule $G(t,t') \supset f$ do
    (a) let $G = G_1 \vee .... \vee G_n$
    (b) For each action $a$ which can affect the fluent $f$ if executed for more
        than one time do
        i. set $G' = G \vee (f_\alpha(a) \geq b))$
        ii. set $G = G'$
    (c) let $G = G_1 \vee .... \vee G_m$
    (d) For each j from 1 to m do
        • let $G_i = f_1(t_1) \wedge .... \wedge f_n(t_n)$
        • let $t = min(t_1, ..., t_n)$
    (e) set $t'' = max(t_i : G_i \quad is \quad true)$
    (f) replace $G(t,t')$ with $G(t,t'')$
```

The above algorithm "adds" at each static rule the effect which depends on the duration of an action. The algorithm of evaluation of the dynamic and static rules does not change.

The main problem of the actions with duration is that we cannot assume that each of them is equivalent with two instantaneous actions one at the start and one at the end of the duration. We show that with an example. Consider again the example with the *public_worker*. Assume that the action *misdemeanor(p)* has duration and the direct effect is that the *public_worker p* is *illegal* for time of the duration of the action *misdemeanor(p)* and also for 5 months after the end of the action. Because may not know the duration of the action *misdemeanor(p)* from the start we must define the following dynamic rules

$$occur(start(misdemeanor(p)),t) \supset illegal(p,[t,\infty]) \quad (a)$$
$$occur(end(misdemeanor(p)),t) \supset illegal(p,[t,t+5m]) \wedge \neg illegal(p,[t+5,\infty]) \qquad (b)$$

The justification for the introduction of the two rules comes from the fact that when the execution of action *misdemeanor(p)* starts, the public worker is *illegal* idetinitely, because we do not know when the action ends. After the end of the action and the public worker is considered *illegal* for 5 months. This is right if the fluent $illegal(p,[t,\infty])$ does not has as indirect effect to change the truth value of another fluent. For example assume the initial situation is

$$\begin{aligned}&S_0 = \{\neg take\_bonus(p,[0,\infty]),\ take\_salary(p,[0,\infty]),\\&\neg take\_promotion(p,[0,\infty]),\ \neg suspended(p,[0,\infty]),\\&\neg good\_employee(p,[0,\infty]),\ \neg illegal(p,[0,\infty])\}\,.\end{aligned}$$

Also assume the following execution of actions

$$\begin{aligned}&occur(start(misdemeanor(p)),2)\\&occur(end(misdemeanor(p)),4)\end{aligned}$$

At time point 2 the execution of action $misdemeanor(p)$ starts and thus the dynamic rule (a) is evaluated. The new situation is

$$\begin{aligned}&S_1' = \{\neg take\_bonus(p,[0,\infty]),\ take\_salary(p,[0,\infty]),\\&\neg take\_promotion(p,[0,\infty]),\ \neg suspended(p,[0,\infty]),\\&\neg good\_employee(p,[0,\infty]),\ illegal(p,[2,\infty])\}\,.\end{aligned}$$

After the evaluation of the static rules

$$\begin{aligned}&illegal(p,[2,\infty]) \supset suspended(p,[2,\infty])\\&suspended(p,[2,\infty]) \supset \neg take\_salary(p,[2,\infty])\end{aligned}$$

we have the situation

$$\begin{aligned}&S_1 = \{\neg take\_bonus(p,[0,\infty]),\ \neg take\_salary(p,[2,\infty]),\\&\neg take\_promotion(p,[0,\infty]),\ suspended(p,[2,\infty]),\\&\neg good\_employee(p,[0,\infty]),\ illegal(p,[2,\infty])\}\,.\end{aligned}$$

At time point 4 the execution of action $misdemeanor(p)$ ends and the dynamic rule (b) is evaluated. The new situation is

$$\begin{aligned}&S_2 = \{\neg take\_bonus(p,[0,\infty]),\ \neg take\_salary(p,[2,\infty]),\\&\neg take\_promotion(p,[0,\infty]),\ suspended(p,[2,\infty]),\\&\neg good\_employee(p,[0,\infty]),\ illegal(p,[4,9]),\neg illegal(p,[9,\infty])\}\,.\end{aligned}$$

As we can observe the fluent $illegal$ changes but the fluent $suspended(p,[2,\infty])$ does not change. Thus at time point 9 the fluent $suspended(p,[2,\infty])$ still holds and thus the fluent $\neg take\_salary(p,[2,\infty])$ holds. This is wrong because one expects the fluent *suspended* to cease to hold when illegal ceases to hold. This does not happen. The problem is caused by the dynamic rule which refer to the start of some action. More specificaly, the problem is caused by the assumption that each fluent that the specific action makes true, is assumed to be true for indefinitely, because we do not known the end of the action. Thus we must change the dynamic rule which refer to the start and similary with the dynamic rule which refers to the end of the action. Now there is the problem of how to refresh the direct effects of the action as time progress.

In order to solve this problem, we define a natural action $natural(a)$ for each action a with duration. This natural action is instantaneous and will be executed periodically. The direct effects of a natural action for action a are exactly the same with the effects of action a. This means that if $occur(a,t) \supset \bigwedge f_i([t,t'_i])$ then $occur(natural(a),t) \supset \bigwedge f_i([t,t'_i])$ The period is equal with the minimal time that some fluent become trues (as direct effect of action a) after the execution of this action. For example,

$$occur(a,t) \supset \; f_1([t,t+3]) \wedge f_2([t,t+4]) \wedge f_3([t,t+7]) \,,$$

then the period of execution is 3. The precondition of the action $natural(a)$ is the continuation of the execution of action a. This means

$$Poss(natural(a)) \equiv f_\alpha(a) > 0 \,.$$

For each action a with duration, we assume that there are three instantaneous actions $start(a)$, $end(a)$ and $natural(a)$. The dynamic rules must refer to the instantaneous actions so we change the dynamic rules as follows:

1. if the dynamic rule has the form $occur(A,t) \supset \bigwedge f_i([t,t'_i])$ and $t' \neq \infty$ for some f_i then
   (a) replace it with
   $$occur(start(A),t) \supset \bigwedge f_i([t,t'_i])$$
   $$occur(end(A),t) \supset \bigwedge f_i([t,t'_i])$$
   (b) let $t'' = min\{t'_i : f_i([t,t'_i])\}$
   (c) set natural(A) to execute periodically with period t''.
2. if the dynamic rule has the form $occur(A,t) \supset \bigwedge f_i([t,\infty])$ then replace it with
   $$occur(start(A),t) \supset \bigwedge f_i([t,\infty])$$

The algorithm for the evaluation of dynamic and static rules does not change. As we observe if one action has "permanent" direct effects then it is not necessary to define natural action for this action, because there is no need to refresh.

Theorem 4 *The algorithm address the ramification problem in case that the effect of an action depend on its duration.*

Proof: In order to be correct the algorithm must alway terminated in a consistent situation. This mean that all integrity constraint satisfied at this situation.

Assume that integrity constraint Law_j is not satisfied in one situation at time point t. If Law_j does not refer to the effects that depend on the duration then the proof is the same with that of theorem 2. If Law_j refers to the effects that depend on the duration. This mean that there is a fluent f which truth value depend on the duration of an action a. Without loss of generality we assumed that the fluent f becomes true if the action a executed for more than b time points. Assumed that the action a executed for more than b time points and the fluent f is false. Because the truth value of fluent f depend on the duration of an action a, the static rule which refer to its has the form

$$\begin{aligned}&G_f \supset f(...)\\&where\\&G_f \equiv (\bigvee(\bigwedge fi)) \vee \ (f_\alpha(a) \geq b)\end{aligned}$$

This means that the formula G_f is true at time point t because the second part $((f_\alpha(a) \geq b)$ is true. Thus the rule $G_f \supset f(...)$ evaluates. This means that the fluent f becomes true. A contradiction.

□

Consider the example with the public worker with the new assumption. The dynamic rules are

$$\begin{aligned}&occur(misdemeanor(p),t) \supset illegal(p,[t,t+5m]) \quad (1)\\&occur(take\_pardon(p),t) \supset \neg illegal(p,[t,\infty]) \quad (2)\\&occur(bad\_grade(p),t) \supset \neg good\_employee(p,[t,\infty]) \quad (3)\\&occur(good\_grade(p),t) \supset good\_employee(p,[t,\infty]) \quad (4)\,,\end{aligned}$$

After the execution of the algorithm which productes the dynamic rules we have

$$\begin{aligned}&occur(start(misdemeanor(p)),t) \supset illegal(p,[t,t+5m]) \quad (1a)\\&occur(end(misdemeanor(p)),t) \supset illegal(p,[t,t+5m]) \quad (1b)\\&occur(natural(misdemeanor(p)),t) \supset illegal(p,[t,t+5m]) \quad (1c)\end{aligned}$$

$$occur(start(take\_pardon(p)),t) \supset \neg illegal(p,[t,\infty]) \quad (2)$$
$$occur(start(bad\_grade(p)),t) \supset \neg good\_employee(p,[t,\infty]) \quad (3)$$
$$occur(start(good\_grade(p)),t) \supset good\_employee(p,[t,\infty]) \quad (4),$$

The period of the execution of action $natural(misdemeanor(p))$ is 5. The static rules as they are productes by the algorithm in section 6.1 are:

$$\begin{aligned}
&R = \{illegal(p,t_1) \supset suspended(p,t_1),\\
&illegal(p,t_1) \vee \neg good\_employee(p,t_2) \supset \neg take\_promotion(p,max(t_1,t_2)),\\
&suspended(p) \supset \neg take\_salary(p,t_1),\\
&\neg suspended(p,t_1) \wedge good\_employee(p,t_2) \supset take\_bonus(p,min(t_1,t_2)),\\
&\neg good\_employee(p,t_1) \supset \neg take\_bonus(p,t_1),\\
&\neg suspended(p,t_1) \supset take\_salary(p,t_1)\\
&False \supset take\_promotion(p,\infty)\}
\end{aligned}$$

Now we must perform the above algorithm at the set R in order to encapsulate the indirect effects which depend on the duration of some action. The only rule which is effected is the last and changes from $False \supset take\_promotion$ to $f_\alpha(good\_grade) \geq 2) \supset take\_promotion(p,\infty)$

Assume the following sequence of execution

$$\begin{aligned}
&occur(start(good\_grade(p)),2)\\
&occur(end(good\_grade(p)),5)\\
&occur(start(misdemeanor(p)),6)\\
&occur(end(misdemeanor(p)),13)
\end{aligned}$$

Assume now that we have a public worker p and the initial situation is

$$\begin{aligned}
&S_0 = \{\neg take\_bonus(p,[0,\infty]),\ take\_salary(p,[0,\infty]),\\
&\neg take\_promotion(p,[0,\infty]),\ \neg suspended(p,[0,\infty]),\\
&\neg good\_employee(p,[0,\infty]),\ \neg illegal(p,[0,\infty])\}\,.
\end{aligned}$$

At time point 2 the execution of the action $good\_grade(p)$ starts and the dynamic rule (4) is evaluated. Now the new situation is

$$\begin{aligned}
&S_1' = \{\neg take\_bonus(p,[0,\infty]),\ take\_salary(p,[0,\infty]),\\
&\neg take\_promotion(p,[0,\infty]),\ \neg suspended(p,[2,\infty]),\\
&good\_employee(p,[2,\infty]),\ \neg illegal(p,[0,\infty])\}\,.
\end{aligned}$$

After the evaluation of the static rule

$$\neg suspended(p,[2,\infty]) \wedge good\_employee(p,[2,\infty]) \;\supset take\_bonus(p,[2,\infty])$$

the new situation is

$$\begin{aligned}&S_1 = \{take\_bonus(p,[2,\infty]),\; take\_salary(p,[0,\infty]),\\ &\neg take\_promotion(p,[0,\infty]),\; \neg suspended(p,[2,\infty]),\\ &good\_employee(p,[2,\infty]),\; \neg illegal(p,[0,\infty])\}\,.\end{aligned}$$

At time point 4 the action $good\_grade(p)$ executes for more than 2 time points thus the static rule $f_\alpha(good\_grade) \geq 2) \supset take\_promotion(p,\infty)$. is evaluated Now the new situation is

$$\begin{aligned}&S_2 = \{take\_bonus(p,[2,\infty]),\; take\_salary(p,[0,\infty]),\\ &take\_promotion(p,[4,\infty]),\; \neg suspended(p,[0,\infty]),\\ &good\_employee(p,[2,\infty]),\; \neg illegal(p,[0,\infty])\}\,.\end{aligned}$$

At time point 5 the action $good\_grade(p)$ ends but the situation does not change. At time point 6 the execution of the action $misdemeanor(p)$ starts and the dynamic rule (1a) is evaluates. Now the new situation is

$$\begin{aligned}&S_3' = \{take\_bonus(p,[2,\infty]),\; take\_salary(p,[0,\infty]),\\ &take\_promotion(p,[4,\infty]),\; \neg suspended(p,[0,\infty]),\\ &good\_employee(p,[2,\infty]),\; illegal(p,[6,11]),\; \neg illegal(p,[11,\infty])\}\,.\end{aligned}$$

After the execution of static rules

$$\begin{aligned}&illegal(p,[6,11]) \supset suspended(p,[6,11])\\ &suspended(p,[6,11]) \supset \neg take\_salary(p,[6,11])\\ &illegal(p,[6,11]) \vee \neg good\_employee(...) \supset\; \neg take\_promotion(p,[6,11])\end{aligned}$$

the situation at time point 6 is:

$$\begin{aligned}&S_3 = \{take\_bonus(p,[2,\infty]),\; \neg take\_salary(p,[6,11]), take\_salary(p,[11,\infty]),\\ &\neg take\_promotion(p,[6,11]),\; take\_promotion(p,[11,\infty]),\; suspended(p,[6,11]),\\ &\neg suspended(p,[11,\infty]),\; good\_employee(p,[2,\infty]),\\ &illegal(p,[6,11]), \neg illegal(p,[11,\infty])\}\,.\end{aligned}$$

The situation change at point 11 because the fluent $\neg take\_salary(p,[6,11])$, $\neg take\_promotion(p,[6,11])$, $suspended(p,[6,11])$ and $illegal(p,[6,11])$ ceases to hold. At time point 11 the situation is

$$\begin{aligned}&S_4'' = \{take\_bonus(p,[2,\infty]),\ take\_salary(p,[11,\infty]),\\&take\_promotion(p,[11,\infty]),\ \neg suspended(p,[11,\infty]),\\&good\_employee(p,[2,\infty]),\ \neg illegal(p,[11,\infty])\}\,.\end{aligned}$$

At time point 11 the action $natural(misdemeanor(p))$ executes (because $f_\alpha(misdemeanor(p)) > 0$ holds) and refreshes the effect of action $misdemeanor(p)$. The dynamic rule (1c) executes. Now the new situation is

$$\begin{aligned}&S_4' = \{take\_bonus(p,[2,\infty]),\ take\_salary(p,[11,\infty]),\\&take\_promotion(p,[11,\infty]),\ \neg suspended(p,[11,\infty]),\\&good\_employee(p,[2,\infty]),\ illegal(p,[11,16]), \neg illegal(p,[16,\infty])\}\,.\end{aligned}$$

After the execution of the static rules

$$\begin{aligned}&illegal(p,[11,16]) \supset suspended(p,[11,16])\\&suspended(p,[11,16]) \supset \neg take\_salary(p,[11,16])\\&illegal(p,[11,16]) \vee \neg good\_employee(...) \supset\ \neg take\_promotion(p,[11,16])\end{aligned}$$

we have

$$\begin{aligned}&S_4 = \{take\_bonus(p,[2,\infty]),\ \neg take\_salary(p,[11,16]), take\_salary(p,[16,\infty]),\\&\neg take\_promotion(p,[11,16])),\ take\_promotion(p,[16,\infty]),\\&suspended(p,[11,16]),\ \neg suspended(p,[16,\infty]),\ good\_employee(p,[2,\infty]),\\&illegal(p,[11,16]), \neg illegal(p,[16,\infty])\}\,.\end{aligned}$$

If we compare the situations S_3 and S_4 we can clearly obsrve the ruslt of refreshing of the effects. At time point 13 the action $misdemeanor(p)$ ends and thus the dynamic rule (1b) will be evaluated. Now the new situation is

$$\begin{aligned}&S_5 = \{take\_bonus(p,[2,\infty]),\ \neg take\_salary(p,[11,16]), take\_salary(p,[16,\infty]),\\&\neg take\_promotion(p,[11,16]),\ take\_promotion(p,[16,\infty]),\\&suspended(p,[11,16]),\ \neg suspended(p,[16,\infty]),\ good\_employee(p,[2,\infty]),\\&illegal(p,[13,18]), \neg illegal(p,[18,\infty])\}\,.\end{aligned}$$

At time point 13 no static rule evaluates because the fluents $\neg take\_salary(p, [11,16])$, $suspended(p,[11,16])$ are true at time point 13. But at time point 16 the situation change because the fluent $suspended(p,[11,16])$, $\neg take\_promotion(p, [11,16])$ and $\neg take\_salary(p,[11,16])$ cease to hold. Thus

$$\begin{aligned}
&S_6' = \{take\_bonus(p,[2,\infty]),\ take\_salary(p,[16,\infty]),\\
&take\_promotion(p,[16,\infty]),\ \neg suspended(p,[16,\infty]),\\
&good\_employee(p,[2,\infty]),\ illegal(p,[13,18]), \neg illegal(p,[18,\infty])\}\,.
\end{aligned}$$

Now the following static rules evaluates

$$\begin{aligned}
&illegal(p,[16,18]) \supset suspended(p,[16,18])\\
&suspended(p,[16,18]) \supset \neg take\_salary(p,[16,18])\\
&illegal(p,[16,18]) \vee \neg good\_employee(...) \supset\ \neg take\_promotion(p,[16,18])
\end{aligned}$$

and the new situation is

$$\begin{aligned}
&S_6 = \{take\_bonus(p,[2,\infty]),\ \neg take\_salary(p,[16,18]),\\
&take\_salary(p,[18,\infty]),\ \neg take\_promotion(p,[16,18]),\\
&take\_promotion(p,[18,\infty]),\ suspended(p,[16,18]),\\
&\neg suspended(p,[18,\infty]),\ good\_employee(p,[2,\infty]),\\
&illegal(p,[13,18]), \neg illegal(p,[18,\infty])\}\,.
\end{aligned}$$

Finally the situation changes again at time point 18 because the fluent $illegal(p,[13,18])$, $suspended(p,[16,18])$, $\neg take\_salary(p,[16,18])$ and $\neg take\_pro$ $motion(p,[16,18])$ cease to holds. The new situation is

$$\begin{aligned}
&S_7 = \{take\_bonus(p,[2,\infty]),\ take\_salary(p,[18,\infty]),\\
&\neg take\_promotion(p,[6,\infty]),\ \neg suspended(p,[18,\infty]),\\
&good\_employee(p,[2,\infty]),\ \neg illegal(p,[18,\infty])\}\,.
\end{aligned}$$

7. Summary and Future Research

The ramification problem in temporal databases is a complex and many-faceted problem. We have addressed problem for the cases that the effects(direct and indirect) of instantaneous actions refer to the current and future situations. Also, we have described a solution for these cases when the actions have durations.

Further research includes the study of the problem for concurrent actions in the case of instantaneous actions or actions with duration and non-deterministic actions, as well as the problem of changing time granularities. Consider the case that two or more instantaneous actions can execute concurrently. The direct and indirect effects of an action do not start necessarily from the next time moment. This means that two or more actions cannot necessarily be executed concurrently if the preconditions holds. It must be determined that the direct and indirect effects of these actions are consistent not only in the next time moment but in the future, as well.

For example a person cannot work in the public and in the private sector at the same time. Suppose the actions *hire_in_public* and *hire_in_company* are defined:

$$\begin{aligned}&hire\_in\_public(p,t) \supset public\_worker(p,[t+10,\infty])\\&hire\_in\_company(p,t) \supset private\_employee(p,[t+10,\infty])\end{aligned}$$

This means that the employment started ten time moments after the action took place. The integrity constraints

$$\begin{aligned}&public\_worker(p,L) \supset \neg private\_employee(p,L)\\&private\_employee(p,L) \supset \neg public\_worker(p,L)\,,\end{aligned}$$

denote that the actions *hire_in_public* and *hire_in_company* cannot execute concurrently.

Another direction is the study of the problem in the case of actions changing our beliefs about the past. In this case, effects may be periodically recursive and one needs to be able to determine what is allowed to change in the past and what isn't. The related qualification problem, which refers to determininig the preconditions which must hold prior to the execution of an action, is a topic of current research. We are studying the extension of the proposed framework for solving the qualification problem by defining static rules specifying when actions become disqualified.

8. References

[1] A. Borgida, J. Mylopoulos and R. Reiter. On the Frame Problem in Procedure Specifications. IEEE Transactions on Software Engineering, 21(10), Oct. 1995, pp.785-798.

[2] C. Elkan. Reasoning about action in first order logic. Proceedings of the Conference of the Canadian Society for Comptutational Studies in Intelligence (CSCSI), pp 221-227, Vancouver, May 1992.

[3] M. Ginsberg and D. Smith. Reasoning about action I: A possible worlds approach. Artificial Intelligence, 35:165-195, 1988.

[4] J. Gustafon. Extending Temporal Action Logic for Ramification and Concurency, Thesis No 719 of Linkoping Studies in Science and Technology, 1998.

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

[6] A.C. Kakas, R.S. Miller and F. Toni, E-RES: Reasoning about Actions, Events and Observations, in Proceedings of LPNMR2001, pp. 254-266, Springer Verlag, 2001.

[7] Antonis Kakas and Rob Miller, A Simple Declarative Language for Describing Narratives with Actions, The Journal of Logic Programming, Vol 31(1-3) (Special Issue on Reasoning about Action and Change), pages 157-200, Elsevier, 1997.

[8] V. Lifshitz. Towards a metatheory of action. In J.F. Allen, R. Fikes, and E. Sandewall, editors, Proceedings of the International Conference on Principles of Knowledge Representation and Reasoning, pages 376-386, Cambridge, MA, 1991.

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

[10] V. Lifshitz. Restricted monotonicity. In Proceedings of the AAAI National Conference on Artifical Intelligence, pages 432-437, Washington DC, July 1993.

[11] N. McCain and H. Turner. A causal theory of ramifications and qualifications. Proceedings of IJCAI-95, pp. 1978-1984, Montreal, Canada, August 1995.

[12] J. McCarthy and P.J. Hayes. Some philophical problem from the standpoint of artificial intelligence. In B. Meltzer and D. Mitchie, editors, Machine Intelligence 4, pp. 463-502. American Elsevier, 1969.

[13] Nikos Papadakis and Dimitris Plexousakis. Action Theories in Temporal Databases. Proceedings of the 8th Panhellenic Conference on Informatics, pp. 254-264, Cyprus, Nov. 2001.

[14] Nikos Papadakis and Dimitris Plexousakis. The Ramification and Qualification Problems in Temporal Databases. Proceedings of the 2th hellenic Conference on AI, pp. 18-30 Lecture Notes on Artificial Intelligent vol 2308, 10-11 April 2002, Thessaloniki, Greece.

[15] Nikos Papadakis and Dimitris Plexousakis. Action with Duration and Constraints: The Ramification problem in Temporal Databases. 14th IEEE International Conference on Tools with Artificial Intelligent, 4-6 November 2002 , Washington D.C.

[16] Dimitris Plexousakis. Maintenance of Integrity Constraints in Temporal Deductive Knowledge Bases. Phd Thesis, Dept. of Computer Science, Univ. of Toronto, Jan. 1996.

[17] Dimitris Plexousakis, John Mylopoulos. Accomodating Integrity Constraints During Database Design. Proceedings of EDBT 1996, pp. 497-513, Avignon, France, 1996.

[18] J. Pinto. Temporal Reasoning in the Situation Calculus. Ph.D. Thesis, Dept. of Computer Science, Univ. of Toronto, Jan. 1994.

[19] J. Pinto and R. Reiter. Temporal Reasoning in Logic Programming: A Case for the Situation Calculus, Proceedings of 10th Int. Conf. on Logic Programming, Budapest, Hungary, June 21-24, 1993.

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

[21] R. Reiter. Khowledge in Action: Logical Foundation for specifying and implemending Dynamical Systems, MIT Press, 2001.

[22] M. Thielscher. Ramification and causality. Artifical Intelligence, 89(1-2):317-364,

1997.

[23] M. Thielscher. Reasoning about actions: Steady versus stabilizing state constraints. Artifical Intelligence, 104:339-355, 1988.

[24] M. Winslett. Reasoning about action using a possible models approach. Proceedings of AAAI-88, pp. 89-93, Saint Paul, MN, August 1988.