Representing Beliefs in a Situated Event Calculus

François Lévy, J. Joachim Quantz

December 10, 1997

Abstract

This paper proposes an extension of the event calculus [KS86] which allows the representation of alternative situations (states-of-affairs) and beliefs. The basic idea of this *situated event calculus* is to use the notion of situations as developed in Situation Semantics, a linguistically motivated semantic theory [BP83, BE87] . Instead of having events of the form 'Happens(Action,Time)' all events occur relative to a specific situation: 'Happens(Action,Time,Sit)'. In other words, an event can happen in one situation and not happen in an otherwise similar situation.

Situations can thus be viewed as collections of events and are used here to represent beliefs: a person adopting a situation has knowledge of all the events occurring in it and believes all the fluents holding in it (ie all consequences of the events). Changing beliefs are modeled by adopting different situations. By using a circumscribed version of the event calculus [Shanahan97] and modeling 'adopt' and 'believes' as ordinary events and fluents, the frame problem is easily overcome, both wrt beliefs and "physical" fluents.

1 Motivation: processing car crash reports

The motivation for the work presented in this paper stems from an application in Natural Language Processing, namely the automatic processing of car crash reports. These reports are short descriptions of accidents as found in statements for insurance companies. A corpus of 115 reports has been investigated by different research groups wrt various research issues relevant for NLP, such as syntactic and semantic phenomena, discourse structure, etc.

This section briefly presents some sample texts containing explicit or implicit references to the driver's beliefs concerning the actual state-of-affairs. One of these examples will be formally represented in our Situated Event Calculus in Section 4.

The most simple accident reports only contain a description of the situation and an (unexpected) event which causes the accident. However, a substantial number of reports show a more complicated structure. Three types of such complicated structures recur frequently. In the first one, the author describes not only the situation and an unexpected event but also his own unsuccessfull action to prevent the accident. Interestingly, such actions may be described indirectly. In the following text, though no explicit reference is made here to an action of braking in order to avoid the accident, the reader assumes that such an action has taken place.

> We were at Saint-Ouen, I was surprised when the person in front of me braked suddenly and, being unable to change lanes and the road being wet, I couldn't come to a complete stop in time (A15)

The second complex type of reports is similar to the first one, but whereas in the first one the author's action did not prevent the accident, here the action does indeed prevent the impending accident—only to produce some other accident, however.

> I was driving along the Pasteur street when a car suddenly came out on my right; to avoid it I swerved to the left and braked. I did avoid it and my sideview mirror caught his. The car continued undamaged and as for me, I hit a garbage tip parked on the roadside. ... (A13)

Finally, a third complex type of reports makes (sometimes implicit) reference to actions which were expected by the author but did not actually take place.

> I was already in a two-way street when car B came up on my left. Thinking he was going to stop and give me right of way, he cut in front of me so I braked but it was too late, I went into his nearside fender. (B38)

When reading such reports one understands that the author expected certain events to happen or not happen and adjusted his behavior accordingly. An intelligent text processing system might have to represent these expectations formally. This paper presents work along these lines and concentrates on the task of representing the beliefs of the driver wrt actions and their consequences. Deriving these representations automatically from the texts is a rather non-trivial task which is not addressed here.

In order to allow modeling the contents of car crash reports adequately, a representation formalism should meet at least the following conditions.

- the consequences of actions should be derivable;
- it should be possible to represent alternative states-of-affairs (situations);
- it should be possible to represent the beliefs of the driver and his reasoning about consequences of actions.

Since only some particular succession of events are of interest, Event Calculus has been chosen as the basic formalism to represent fluents and events. The remaining task is then to integrate alternative situations and beliefs into the Event Calculus. Considering multiple situations at the same time is obtained simplistically by adding a 'situation' parameter to the basic relations (Happens, Holds_at, ...) of the language. These situations are given a branching situation with the help of the 'Equal_until()' predicate, which states that two situations share the same initial part. Then new actions governing epistemic fluents are introduced.

Note that any version of Event Calculus may be convenient to implement these extensions, although their consequences may differ. In order to describe moving cars, we stick to versions involving continuous change. In this paper, ECF proposed by M. Shanahan ([Shanahan97]) and which relies on Forced circumscription, is considered.

2 Representing situations and beliefs

This section informally introduces the Situated Event Calculus which will be used for representing alternative states-of-affairs and beliefs. A formal presentation will then be given in the next section. A more detailled example will be developed in Section 4.

2.1 Situations

The basic idea underlying the Situated Event Calculus developed below (called SECF) is to represent alternative situations, such that an event may happen in one situation but not in another. This is achieved by adding a situation parameter to all statements and allows us to compare a situation 's1' in which the braking event does not occur with another one, 's2', in which it does occur.

Situation s1	*Situation s2*
Happens(start(v1),0,s1)	Equal_until(s1,s2,1)
Happens(start(v2),0,s1)	Happens(brake(v2),2,s2)

All the course of events starts at time 0. For the sake of simplicity, discrete time is used. The predicate Equal_until(s1,s2,1) states that situations 's1' and 's2' are identical until time 1 – but they differ at time 2 when the breaking event occurs in s2. When answering queries concerning the status of fluents it is also necessary to specify the situation, for example:

Holds_at(moving(v2),4,s1) $\neg$ Holds_at(moving(v2),4,s2)

2.2 Beliefs as fluents

The integration of a situation parameter allows beliefs to be represented by introducing the fluent 'believes(p,f,t)'. The intuitive meaning of this fluent is that the person 'p' believes that fluent 'f' holds at time point 't'. The action 'adopt(p,s)' is also introduced, the basic idea being that a person 'p' adopting a situation 's' believes in all the fluents – or possibly all the fluents of a given type – holding in that situation. This behavior is governed by belief axioms. An example of a belief axiom follows :

$$\text{Initiates}(\text{adopt}(p,s_f),\text{believes}(p,f,t_f),t_b,s_b) \leftarrow \text{Holds\_at}(f,t_f,s_f) \qquad \textbf{(RBF1)}$$

Note that here 't_f' is internal time wrt belief (it is believed that the fluent holds at t_f), while external time t_b (the agent has this belief at t_b) is represented in the same way as for ordinary fluents. Continuing the previous example, let us suppose two 'adopt' events.

Happens(adopt(p,s1),0,s1)   Happens(adopt(p,s2),3,s2)

Given some domain axioms detailed in 4, SECF has the following properties.

Holds_at(believes(p,moving(v2),5),1,s2) is provable
Holds_at(believes(p,accident(v1,o),t),1,s2) is not provable
$\neg$ Holds_at(believes(p,moving(v2),5),4,s2) is provable
Holds_at(believes(p,accident(v1,v2),5),4,s2) is provable

A few remarks seem in order.

First, an agent adopting a situation has knowledge of *all* the events happening in that situation. Thus 'p' who adopts 's2' at time point 3 "expects" the consequences of the breaking event which happens at time 2 and consequently believes at time point 4 that there will be an accident at time point 5.

It could seem at a first glance that the agent has acquired some divinatory power. This is not the case, thanks to the plurality of situations. The agent's actions may depend on his expectations, and he is only perfectly aware of his own expectations, gathered in a particular situation. So full knowledge of the adopted situation does not imply a perfect knowledge of the future. This would only be the case if the agent necessarily adopted the actual situation at time 0. Rather, in our car crash reports the agent generally adopts at the begining of the story a situation which differs from the actual one.

Second, changing beliefs are modeled by adopting different situations. As in the case of ordinary fluents, this affects the beliefs *after* the adoption but does not change the beliefs before the adoption. Representing both the internal and external time of beliefs makes a clear difference between believing the world has changed (believing that different fluents hold at different internal times) and changing one's beliefs (two inconsistent beliefs may be adopted by the agent at different external times). This avoids the paradoxes of the agent modifying the real world as a consequence of beliefs which themselves depend on the real world.

Third, since 'adopt' and 'believes' behave like ordinary actions and fluents, we easily obtain nested beliefs. Thus 'p' is aware of his own beliefs, since 'p' adopting 's2' happens in 's2'. Infinitely nested beliefs are the consequence of adopting 's2' in 's2'. In fact, the belief axiom (RBF1) entails a reflexive belief. With this axiom, nesting can only be avoided if the agent adopts situations where no 'adopt' action occurs. Another schema of belief axioms is conceivable in order to yield a non reflexive belief : adopting in 's2' a situation 's1' initiates only believing in non-belief fluents of's1'. In the following, reflexive belief will be considered.

3 Forced Situated Event Calculus

As an example of situated event calculus, a formal definition of Situated Event Calculus with Forced cicumsciption (SECF) is now given. It involves three kinds of basic axioms. Persistence axioms embody the general behavior of fluents. A new axiom is needed to manage initial equality of situations. Last, epistemic axioms, which are part of the domain description, are devoted to the notion of belief.

3.1 Persistence Axioms

With respect to ECF ([Shanahan97]), the persistence axioms of (SECF) only need the adjunction of a *situation* sort.

To present shortly particular features of these axioms to the unfamiliar reader, fluents may be proven to hold or not to hold, as their status may remain unknown. Two predicates $\mathrm{Initially_P}$ and $\mathrm{Initially_N}$ describe positive or negative status of fluents in the initial state. The predicate 'Releases' cuts the persistence of fluents and returns them to the unknown status. The predicate 'Trajectory' describes continuous change. Last, unique name assumptions are supposed to hold. The basic axioms of SECF are:

$$\begin{aligned}&\mathrm{Holds\_at(f,t1,sit)} \leftarrow && \textbf{(SECF1)}\\&\quad [\mathrm{Initially_P(f,sit)} \wedge \neg\mathrm{Clipped(0,f,t1,sit)}] \vee\\&\quad \exists \mathrm{a,t}\ [\mathrm{Happens(a,t,sit)} \wedge \mathrm{t<t1} \wedge \mathrm{Initiates(a,f,t,sit)} \wedge \neg\mathrm{Clipped(t,f,t1,sit)}]\end{aligned}$$

$$\begin{aligned}&\neg\mathrm{Holds\_at(f,t1,sit)} \leftarrow && \textbf{(SECF2)}\\&\quad [\mathrm{Initially_N(f,sit)} \wedge \neg\mathrm{Declipped(0,f,t1,sit)}] \vee\\&\quad \exists \mathrm{a,t}\ [\mathrm{Happens(a,t,sit)} \wedge \mathrm{t<t1} \wedge \mathrm{Terminates(a,f,t,sit)} \wedge \neg\mathrm{Declipped(t,f,t1,sit)}]\end{aligned}$$

$$\begin{aligned}&\mathrm{Clipped(t1,f,t2,sit)} \leftrightarrow && \textbf{(SECF3)}\\&\quad \exists \mathrm{a,t}\ [\mathrm{Happens(a,t,sit)} \wedge \mathrm{t1<t} \wedge \mathrm{t<t2}\\&\quad \wedge [\mathrm{Terminates(a,f,t,sit)} \vee \mathrm{Releases(a,f,t,sit)}]\ ]\end{aligned}$$

$$\begin{aligned}&\mathrm{Declipped(t1,f,t2,sit)} \leftrightarrow && \textbf{(SECF4)}\\&\quad \exists \mathrm{a,t}\ [\mathrm{Happens(a,t,sit)} \wedge \mathrm{t1<t} \wedge \mathrm{t<t2}\\&\quad \wedge [\mathrm{Initiates(a,f,t,sit)} \vee \mathrm{Releases(a,f,t,sit)}]\ ]\end{aligned}$$

$$\begin{aligned}&\mathrm{Holds\_at(f2,t1,sit)} \leftarrow && \textbf{(SECF5)}\\&\quad \exists \mathrm{a,t}\ [\mathrm{Happens(a,t,sit)} \wedge \mathrm{t<t1} \wedge \mathrm{Initiates(a,f1,t,sit)}\\&\quad \wedge \neg\mathrm{Clipped(t,f1,t1,sit)} \wedge \mathrm{d=t1-t} \wedge \mathrm{Trajectory(f1,t,f2,d,sit)}\ ]\end{aligned}$$

3.2 Equality of situations and the circumscription policy

A theory involves basic axioms and a domain description. The domain description may involve domain constraints. Circumscription is used to compute the effect of events. Belief axioms being part of the domain axioms, the circumscription policy of (SECF) can be defined first. In fact, it is all the same as (ECF)'s one. The only difference is due to the predicate Equal_until, which deserves a brief comment.

Two situations are equal on a period if the same events happen at the same time and the same fluents hold on the same intervals during this period in both of them. Stating cautiousless an equality relation between two situations could introduce unmotivated fluents : if s1 and s2 are equal after t and f is initiated before t in s2, but not in s1, the description of fluents in s1 does not follow from the axioms of Event Calculus. Contradictions could easily arise from this use of equality.

The predicate Equal_until is safe from this problem. Since the equality holds from the begining, only the identity of initial conditions and of occuring events need to be stated. The identity of fluents follows then from the Event calculus itself. So axiom (EU1) is :

$$\begin{aligned}&\text{Equal\_until(sit1,sit2,t1)} \equiv_{def} \qquad \textbf{(EU1)}\\ &\quad \forall\text{f } [[\text{Initially}_{\text{P}}(\text{f,sit1}) \leftrightarrow \text{Initially}_{\text{P}}(\text{f,sit2})] \wedge [\text{Initially}_{\text{N}}(\text{f,sit1}) \leftrightarrow \text{Initially}_{\text{N}}(\text{f,sit2})]] \wedge\\ &\quad \forall\text{a,t } [\text{t<t1} \rightarrow [\text{Happens(a,t,sit1)} \leftrightarrow \text{Happens(a,t,sit2)}]]\end{aligned}$$

To define the circumscription policy, the whole theory Σ is first split into several subparts :

- Φ involves the persistence axioms (SECF1) to (SECF5)
- Δ is the conjunction of ((EU1)) on one hand, Initially$_P$, Initially$_N$, Happens, Equal_until and temporal ordering formulae on the other hand.
- Ξ is the conjunction of Initiates, Terminates and Releases formulae
- Ω is the conjunction of unique name axioms
- Ψ is the conjunction of domain constraints
- Π is the conjunction of trajectory formulae

Now the effect of events is computed applying circumsciption piecewisely on different subparts. Here is the definition of the operator CIRC$_{SECF}$:

$$\begin{aligned}&\text{CIRC}_{SECF}(\Sigma) \equiv_{def}\\ &\quad \text{CIRC}[\Delta \text{ ; Happens}] \wedge \text{CIRC}[\Xi \text{ ; Initiates, Terminates, Releases}] \wedge \Omega \wedge \Psi \wedge \Pi \wedge \Phi\end{aligned}$$

3.3 Belief axioms

Beliefs are represented with the help of an action function *adopt(p,sit)* and a fluent function *believes(p,f,t)*. As discussed in section 2.2, reflexive beliefs are considered here. The 'adopt' event carries the status of fluents in the adopted situation to beliefs in the adopting one : fluents known to hold initiate a belief in the situation where the 'adopt' occurs, and those known not to hold terminate a belief. The (RBF3) is necessary to get free of previous beliefs.

$$\begin{aligned}&\text{Initiates(adopt(p,sit}_f\text{),believes(p,f,t}_f\text{),t}_b\text{,sit}_b) \qquad \textbf{(RBF1)}\\ &\quad \leftarrow \text{Holds\_at(f,t}_f\text{,sit}_f)\end{aligned}$$

$$\text{Terminates(adopt(p,sit}_f\text{),believes(p,f,t}_f\text{),t}_b\text{,sit}_b) \qquad \textbf{(RBF2)}$$

$\leftarrow \neg$Holds_at(f,t_f,sit_f)

Releases(adopt(p,sit_f),believes(p,f,t_f),t_b,sit_b) **(RBF3)**

4 An example

This section illustrates briefly how the SECF developed in the previous sections can be used in our car crash reports processing application. First the axioms used to model background knowledge about accidents will be introduced. Note that the aim is just a rather coarse grained representation in order to illustrate the most important features of the application scenario and the advantages of a SECF formalization. In particular, motion is modeled in discrete time and space. It should be obvious that the modeling can in principle be refined to a much more fine-grained representation.[1] A sketch of the proofs is given in the appendix.

4.1 The Background Model

The location of cars (and other objects) is represented in a two dimensinal space (x,y) by the fluents 'location_x(Car,x)' and 'location_y(Car,y)'. The movement of cars is modeled simplistically by the fluent 'moving(Car)', so all cars move in the same direction y and have the same speed. If the car is not blocked, a starting event initiates the fluent 'moving(Car)'. If it is blocked, the fluent 'accident(Car,Object)' is initiated.

Initiates(start(c),moving(c),t,s) $\leftarrow \neg$Holds_at(blocked(c,o),t,s) **(BGM1)**

Initiates(start(c),accident(c,o),t,s) $\leftarrow$ Holds_at(blocked(c,o),t,s) **(BGM2)**

Since a car can only move in one direction in our modeling, a car is blocked if there is an object located in front of it, as formalized by the following domain constraint.

Holds_at(blocked(c,o),t,s) $\leftarrow$ **(BGM3)**
Holds_at(location_y(c,y1),t,s) $\wedge$ Holds_at(location_y(o,y2),t,s) $\wedge$ y2 = y1 + 1

In the case in which starting initiates moving, the fluent 'location_y(C,Y)' must also be released, since the location of a moving car changes. This changing location is modeled by a *trajectory* which says that a moving car moves one step in the "y-direction" at each time point.

Releases(start(c),location_y(c,y),t,s) $\leftarrow \neg$ Holds_at(blocked(c,o),t,s) **(BGM4)**

Trajectory(moving(c),t,location_y(c,y2),d,s) $\leftarrow$ **(BGM5)**
Holds_at(location_y(c,y1),t) $\wedge$ y2 = y1 + d

It is now possible to model accidents via the *triggered event* bump, which occurs if a car is moving though it is blocked. The event 'bump' in turn has a considerable impact on several fluents.

Happens(bump(c,o),t,s) $\leftarrow$ **(BGM6)**
Holds_at(blocked(c,o),t,s) $\wedge$ Holds_at(moving(c),t,s)

[1] See [Shanahan96] for a rather detailed model of movement in space, which also served as the basis for our own modeling.

Initiates(bump(c,o),accident(c,o),t,s) **(BGM7)**

Initiates(bump(c,o),location_y(c,y),t,s) $\leftarrow$ Holds_at(location_y(c,y),t,s) **(BGM8)**

Terminates(bump(c,o),moving(c),t,s) **(BGM9)**

Terminates(bump(c,o),braking(c),t,s) **(BGM10)**

The axioms should be obvious, except for (BGM8) which might need some explanation. While moving, the location of a car changes continuously, i.e. 'location_y(c,y)' is not a frame fluent (is not persistent) at this time. After a bump, the car ceases moving. The effect of (BGM8) is to make 'location_y(c,o)' become a frame fluent after a bump.

Finally two actions which are possible to avoid accidents, namely braking and changing lanes, will be integrated. Braking is modeled in a rather simplified way: if the road is not wet, (BGM11) fixes the location of the car immediately after braking. Otherwise, (BGM13) initiates the fluent 'braking(Car)'. This fluent introduces a delay before triggering the event 'stop(Car)'[2].

Initiates(brake(c),location_y(c,y),t,s) $\leftarrow$ **(BGM11)**
Holds_at(location_y(c,y),t,s) $\wedge$ $\neg$ Holds_at(wet,t,s)

Terminates(brake(c),moving(c),t,s) $\leftarrow$ $\neg$ Holds_at(wet,t,s) **(BGM12)**

Initiates(brake(c),braking(c),t,s) $\leftarrow$ Holds_at(wet,t,s) **(BGM13)**

Happens(stop(c),t,s) $\leftarrow$ Holds_at(braking(c),t,s) **(BGM14)**

The effects of 'stop' are similar to those of 'bump' without having the accident, however.

Initiates(stop(c),location_y(c,y),t,s) $\leftarrow$ Holds_at(location_y(c,y),t,s) **(BGM15)**

Terminates(stop(c),moving(c),t,s) **(BGM16)**

Terminates(stop(c),braking(c),t,s) **(BGM17)**

Finally,changing lanes is modeled again in a simplified way.

Initiates(move_left(c),location_x(c,x2),t,s) $\leftarrow$ **(BGM18)**
$\neg$ Holds_at(blocked_left(c)) $\wedge$ location_x(c,x1) $\wedge$ x2 = x1 -1

Initiates(move_left(c),accident(c,left),t,s) $\leftarrow$ Holds_at(blocked_left(c)) **(BGM19)**

Releases(move_left(c),location_x(c,x),t,s) $\leftarrow$ $\neg$ Holds_at(blocked_left(c)) **(BGM20)**

and similarly for 'move_right(c)'.

[2] Note this works since the effect of an action holds after one time unit delay. Thus if the road is wet, 'Happens(brake(c),t,s)' gives 'Holds_at(braking(c),t+1,s)', which triggers 'Happens(stop(c),t+1,s)' whose effects hold at 't+2'. If continuous time were in use, a 'trajectory' would do the work, allowing various times before stopping.

4.2 Representing a text

First, consider the text A15 as cited in Section 1. The initial situation can be described as follows:

$\text{Initially}_{\text{P}}$(location_x(v1,0),s1) **(A15.1)**
$\text{Initially}_{\text{P}}$(location_y(v1,0),s1) **(A15.2)**
$\text{Initially}_{\text{P}}$(location_x(v2,0),s1) **(A15.3)**
$\text{Initially}_{\text{P}}$(location_y(v2,2),s1) **(A15.4)**
$\text{Initially}_{\text{P}}$(blocked_left(v1),s1) **(A15.5)**
$\text{Initially}_{\text{P}}$(blocked_right(v1),s1) **(A15.6)**
$\text{Initially}_{\text{P}}$(blocked_left(v2),s1) **(A15.7)**
$\text{Initially}_{\text{P}}$(blocked_right(v2),s1) **(A15.8)**
$\text{Initially}_{\text{P}}$(wet,s1) **(A15.9)**
Happens(start(v1),0,s1) **(A15.10)**
Happens(start(v2),0,s1) **(A15.11)**
Happens(adopts(p,s1),0,s1) **(A15.12)**

ie we have two cars, one 2 units in front of the other, both moving, the road is wet, and both sides are blocked. Note that it is necessary to have Objstart events at time 0 to initialize the moving fluents which then cause the continuous modification of the location fluents (cf. (SECF5)).

What we will be interested in here are the driver's beliefs concerning the occurrence of an accident which can be derived from the modeling. They are consequences of general axioms, the background model and the particular events involved by the text, wrt the circumscription policy described in Section 3.2

After adopting situation 's1' the driver 'p' does not believe in an accident.

$\text{CIRC}_{\text{SECF}}(\Sigma) \not\models \exists$o,t Holds_at(believes(p,accident(v1,o),t),1,s1) **(Cons1)**

Next happens the unexpected braking of car 'v2'. creating a new situation which differs at this point from 's1'.'p' being surprised, he adopts the new situation with a delay:

Equal_until(s1,s2,0) **(A15.13)**
Happens(brake(v2),1,s2) **(A15.14)**
Happens(adopts(p,s2),2,s2) **(A15.15)**

If no further actions occur, there will be an accident.

$\text{CIRC}_{\text{SECF}}(\Sigma) \models$ Holds_at(believes(p,accident(v1,v2),5),3,s2) **(Cons2)**

The SECF now allows us to consider what would happen if the driver were to change lanes, ie two alternative possibilities.

Equal_until(s2,s3,2) **(A15.16)**
Happens(move_left(v1),3,s3) **(A15.17)**
Equal_until(s2,s4,2) **(A15.18)**
Happens(move_right(v1),3,s4) **(A15.19)**

Since both sides are blocked, however, there will be accidents in both situations.

$\text{CIRC}_{\text{SECF}}(\Sigma) \models$ Holds_at(accident(v1,left),4,s3) **(Cons3)**

$\text{CIRC}_{\text{SECF}}(\Sigma) \models$ Holds_at(accident(v1,right),4,s4) **(Cons4)**

What happens instead is the following.

Equal_until(s2,s5,2) **(A15.20)**
Happens(brake(v1),3,s5) **(A15.21)**
Happens(adopts(p,s5),3,s5) **(A15.22)**

But the accident is only delayed by the braking.

$\mathrm{CIRC}_{\mathrm{SECF}}(\Sigma) \models$ Holds_at(accident(v1,v2),6,s5) **(Cons5)**

5 Implementation

SECF applied to the model of the road has been implemented in Prolog. To gain efficiency, the implementation is based on two ideas:

1. to normalize the representation of the "situation structure";
2. to *precompute some fluents*, such that queries can be answered more or less by look-ups instead of by complex backward reasoning.

The constraints imposed on the situation structure are the following:

- *No multiple inheritance:* for each situation 's' there is at most one statement of the form 'Equal_until(s',s,t)'.
- *No cycles* of the form 'Equal_until(s1,s2,t1)', 'Equal_until(s2,s3,t2)', ... 'Equal_until(sn,s1,tn)'.
- *Unidirectionality:* if there is a statement of the form 'Equal_until(s1,s2,t1)' then there are no statements of the form '$\mathrm{Initially}_{\mathrm{P}}$(f,s2)' or of the form 'Happens(e,t2,s2)' with 't2 $\leq$ t1'.

Note that, beause of the symetry of 'Equal_until', these constraints do not reduce the expressive power of the theory. Rather the situation structure has to be described as a directed acyclic graph in which information flows from parents to children.

The *compilation of events* is performed up to a specified time point 'until' and is implemented as follows:

- determine all situations 's' for which there is no statement of the form 'Equal_until(s',s,t)'.
- for each of these situations 's':
  - collect all statements '$\mathrm{Initially}_{\mathrm{P}}$(f,s)';
  - compute fluents for all time points between 0 and 'until';
  - process all situations 's" with 'Equal_until(s,s',t)'.

In the present version, only "physical" fluents are precomputed in this way. In order to determine whether a (nested) 'believes' fluent holds, the algorithm performs backward chaining which is ultimately reduced to looking up a precomputed "physical" fluent.

6 Related Work

As branching time is an inherent feature of the situation tree, Situation Calculus (SC) basically involves the possibility to consider different hypotheses, while genuine Event Calculus (EC) copes with one single line of time. There has been in recent years a trend to explore the gap between EC and SC. [PR93] and [MS94] introduce time in SC and distinguish one single sequence of *actual* situations in accordance with the persistence of fluents. [Reiter96] builds on *natural* actions, i.e. actions resulting from laws of physics as opposed to the free will of an agent. There may then be several *legal* situations, i.e. situations where only the closest future natural actions may have occurred at each step of the history but free-will actions are temporally unconstrained (provided that their prerequisites are fulfilled).

A similar distinction appears in EC in [Miller96]: natural actions *Occur* but agent's actions are *Performed*, and only occurring actions are minimized. After a standard treatment of narratives in a logic programming version of EC, [Provetti96] attaches situation names to sets of fluents holding on a given time interval. Then hypothetical timeless situations are generated similarly to SC, with the help of a standard 'Result()' function.

[BGP95] proposes the action description language L_1 to describe the effects of actions according to the situation. Here a formula stating that the fluent *f* would hold if the sequence $a_1,\ldots,a_n$ of actions was executed in situation *s* allows hypothetical reasoning.

With respect to our aim, the forms of hypothetical reasoning proposed in these papers consider the whole set of possible actions and prune only physically impossible situations. This is certainly adequate for a lot of applications, including planning tasks which most of them consider. But in our texts one course of events is the real one, some others are mentioned as possible or impossible, and the remaining ones don't appear at all. The SECF is adequate here because only mentioned hypothetical courses of events need to be represented.

The same need to represent events which never occur is present in [Ortiz93]. To represent sentences such as "agent p1's action a1 prevented agent p2 from performing action a2", he advocates the pattern "before a1, a2 was possible; after a1, it became impossible". He extends to this aim a modal Belief, Desire, Intention (BDI) logic, in which each world involves linear time: branching time is introduced with the help of an operator $\Diamond_B$, the scope of which is *all the possible worlds with pasts identical to the current one*. Various contextual notions of possibility are considered to instantiate the pattern, for instance "believing (im)possible", etc. The '$\Diamond_B$' operator is certainly very close to our 'Equal_until' predicate. Unhappily, the definitions of worlds and possibility remain at a very abstract level.

Knowledge-producing actions have been considered in [SL93] to model sensing or reading actions [3] in the framework of SC. They define Knowledge similarly to a modal 'Necessary' operator, taking situations as worlds and modeling the accessibility relation as a fluent. In short, performing the same ordinary action in related worlds carries the accessibility relation to the resulting worlds, while a knowledge-producing action carries it only if the sensed values are the same in both worlds where the action is performed.

With respect to the 'believes' fluent considered here, this definition of knowledge involves no internal time: only the current state of affairs may be known, but neither the past nor the future. Moreover, due to reflexivity of the accessibility relation, false (w.r.t. the current situation) knowledge is impossible to represent. This is clearly inadequate to model the driver's beliefs and expectations which explain his behavior.

7 Conclusion

It should be obvious that the work presented in this paper is only a small contribution to processing car crash reports. The problem of automatically deriving the above representations from Natural Language texts has not been addressed. Much work remains to be done in order to build knowledge representation formalisms adequate to capture the semantic content of texts. Amongst the main challenges for A.I. are deriving those parts of the representation which are implicit in the text, and providing a semantic account of mentioned non-occurring events - two tasks performed easily every day by human beings.

[3] i.e. acquiring the truth value of a fluent or the value of a parameter.

The structure of situated event calculus offers an adequate framework for representing several courses of events and efficiently deriving the consequences of actions and events. Moreover, as changing beliefs are modeled by adopting different situations, it is easy to consider only mentioned situations and to overcome the frame problem wrt fluents and beliefs. This allows to represent the driver's beliefs and their changes in time as they are stated in the reports, and can form a basis to explain the driver's behavior.

A Sketch of a proof of (Cons1) to (Cons5)

- Φ involves (SECF1) to (SECF5)
- Δ is the conjunction of (EU1),(BGM6), (BGM14), (A15.1) to (A15.22)
- Ξ is composed of the BGM axioms, but (BGM3), (BGM5), (BGM6), (BGM14).
- Ω is the conjunction of unique name axioms
- Ψ involves (BGM3)
- Π involves (BGM5)

Now CIRC(Δ:Happens) can be computed :

CIRC[Δ ; Happens] $\models$ Happens(a,t,s) $\leftrightarrow$
[(a=adopts(p,s1) $\wedge$ t=0 $\wedge$ (s=s1 $\vee$ s=s2 $\vee$ s=s3 $\vee$ s=s4 $\vee$ s=s5)) $\vee$
(a=adopts(p,s2) $\wedge$ t=2 $\wedge$ (s=s2 $\vee$ s=s3) $\vee$ s=s4 $\vee$ s=s5) $\vee$
(a=adopts(p,s5) $\wedge$ t=3 $\wedge$ s=s5) $\vee$
(a=bump(c,o) $\wedge$ Holds_at(blocked(c,o),t,s) $\wedge$ Holds_at(moving(c),t,s)) $\vee$
(a=stop(c) $\wedge$ Holds_at(braking(c),t,s) $\vee$
(a=start(v1) $\wedge$ t=0 $\wedge$ (s=s1 $\vee$ s=s2$\vee$ s=s3) $\vee$ s=s4 $\vee$ s=s5) $\vee$
(a=start(v2) $\wedge$ t=0 $\wedge$ (s=s1 $\vee$ s=s2$\vee$ s=s3) $\vee$ s=s4 $\vee$ s=s5)) $\vee$
(a=brake(v2) $\wedge$ t=1 $\wedge$ (s=s2 $\vee$ s=s3 $\vee$ s=s4 $\vee$ s=s5)) $\vee$
(a=move_left(v1) $\wedge$ t=3 $\wedge$ s=s3) $\vee$
(a=move_right(v1) $\wedge$ t=3 $\wedge$ s=s3) $\vee$
(a=brake(v1) $\wedge$ t=3 $\wedge$ s=s5)]

All the same, circumscibing Initiates, Terminates and Releases in Ξ yields a completion of their description : each Initiates(action_parameter, fluent_parameter, time_parameter, situation_parameter) is equivalent to the disjunction of its preconditions in the different rules of which it is the head.

The consequences (Cons1) to (Cons5) now follow from standard logic.

References

[BE87] Barwise (J.) et Etchemendy (J.). – *The Liar: an Essay on Truth and Circularity.* – Oxford University Press, 1987.

[BGP95] Baral (C.), Gelfond (M.) et Provetti (A.). – *Representing Actions, Laws and Hypotheses.* – Rapport technique, El Paso, Univ. of Texas, 1995.

[BP83] Barwise (J.) et Perry (J.). – *Situations and Attitudes.* – MIT Press, 1983.

[KS86] Kowalski (R. A.) et Sergot (M. J.). – A logic-based calculus of events. *New Generation Computing*, vol. 4, 1986, pp. 67–95.

[Miller96] Miller (R.). – Notes on deductive and abductive planning in the event calculus. – Working notes of the Temporal Reasoning Group., July 1996.

[MS94] Miller (R.) et Shanahan (M.). – Narratives in the situation calculus. *Journal of Logic and Computation*, vol. 4, 1994, pp. 513–530.

[Ortiz93] Ortiz (C. L.). – The semantics of event prevention. *In : AAAI93*, pp. 683–688.

[PR93] Pinto (J.A.) et Reiter (R.). – Adding a time line to the situation calculus. *In : Logical Formalizations of Commonsense Reasoning*, pp. 172–177. – Austin, Texas, 1993.

[Provetti96] Provetti (A.). – Hypothetical reasoning about actions: from situation calculus to event calculus. *Computational Intelligence*, vol. 12, n° 3, 1996, pp. 478–498.

[Reiter96] Reiter (R.). – Natural actions, concurrency and continuous time in the situation calculus. *In : KR96*, pp. 2–13.

[Shanahan96] Shanahan (M.). – Robotics and the common sense informatic situation. *In : ECAI96*, pp. 684–688.

[Shanahan97] Shanahan (M.). – *Solving the Frame Problem. A Mathematical Investigation of the Common Sense Law of Inertia.* – MIT Press, 1997. (to appear).

[SL93] Scherl (R. B) et Levesque (H. J.). – The frame problem and knowledge-producing actions. *In : AAAI93*, pp. 689–695.