Formalising the Common Sense of a Mobile Robot

Paulo E. Santos
Institute for Logic, Language and Computation
University of Amsterdam.

September 25, 1998

Abstract

Looking at the history of Artificial Intelligence, often theoretical rigor seems inversely proportional to practical applicability. Particularly, in the field of robotics, there is a gap between the research on knowledge representation and practical robotics. In the present paper, a solution for bridging this gap is described and discussed: Shanahan's logical account for a mobile robot. Called "A Logical Account for the Common Sense Informatic Situation for a Mobile Robot", this framework involves the assimilation of sensor data, and action planning, through an abduction process on theories about actions and change, and space and shape. An extension of the Event Calculus is taken as the underlying logical system. Actually, these theories are fragments of second-order logic since Circumscription (which is expressed in second-order logic) is used to overcome the frame problem.

However, the solution presented to the frame problem does not cope with all of the issues concerned with this problem (since the solution deals only with the persistence problem); moreover, due to the lack of an algorithm capable of automatically constructing theories about the background knowledge, it is impossible to affirm whether the solution presented solves, in a general way, the problem for which it is proposed.

Finally, the analysis of the computational complexity of the framework in question made us conclude that, in the general case, its complexity is at least complete for the second level of the polynomial hierarchy.

Contents

1 Introduction

Those long chains of reasoning, quite simple and easy, which geometricians use in order to achieve their most difficult demonstrations, had given me occasion to imagine that all things which can be known by man are mutually related in the same way.

In R. Descartes, Discourse on Method.

1.1 Common Sense

One of the main challenges of the Artificial Intelligence (A.I.) is the construction of machines capable of coping with problems in the ordinary (or common sense) world. Many approaches were proposed in order to solve this problem, all of them unsuccessful in one way or another. A description and analysis of the various approaches is far from the scope of the present paper. This work is concerned with one such proposal, the one based in a formal logical account of A.I.; more specifically, of robotics.

Logiscist A.I. was started in the late fifties [24], and it was based on the idea that knowledge could be represented by sentences in a formal language (in particular in first-order logic) and the reasoning of an intelligent tool would be reduced to proving theorems by means of those sentences. In the early eighties, however, the logiscist approach to robotics was almost completely abandoned due to the difficulty of implementing the formal theories in real applications. The literature from the mid eighties to the mid nineties often talks about a *gap* between theoretical rigor and practical applicability. The purpose of the present work is to present and discuss a new logical account to bridge the gap between formal knowledge representation and practical robotics [39].

The first issue that a formal theory about an autonomous agent (immersed in the real world) has to cope is the representation of Common Sense Knowledge. This knowledge involves the basic facts about the ordinary objects, their *properties* and *spatial relations* in the world, facts about *events* (including *actions*) and its *effects* in the state of the world; it involves also facts about *beliefs* and *desires*.

Clearly, in order to solve problems in the common sense world, common sense reasoning is required. The main feature of this reasoning is its use of *defaults*. A default is a proposition that is accepted as true in the absence of information to the contrary. In the case that we are interested to formalise (a robot in the real world), defaults are used to cope with the robot's limited knowledge about its environment.

In this work, a particular form of reasoning with defaults is used: so called Circumscription. This method of default reasoning is constructed on classical logic, and is based on the idea of finding the minimal extensions of predicates [25][22]. More specifically, the circumscription of a predicate P w.r.t.[1] a certain fact A dictates that the objects that can be shown to have the property P by reasoning from certain facts A are **all** the objects that satisfy P. In other words, the circumscription of a predicate (or sequence of predicates) occurring in a first-order formula ϕ is the second-order formula stating that, not only does ϕ

[1]This abbreviation means *with respect to.*

hold, but the extensions of the circumscribed predicates satisfying ϕ are minimal w.r.t. set inclusion. Circumscription is going to be described in more details in section 3.

Two other elements of Common Sense reasoning are central to our work: reasoning about time and events and reasoning about space[2]; to these we reserved some of the following subsections of this introduction.

1.2 Time and Events

The ordinary world is dynamical, its state changes every moment an action takes place; and actions happen in specific *time instants* (e.g. a robot picks a book up from the table) changing properties in the world; and properties hold during determined *time intervals* (e.g. the robot's claw remains busy until it drops the book). This kind of reasoning about time is a central issue in common sense reasoning.

Talking about temporal reasoning would give a book by itself[3]. The issue that interests us in the present paper is formalising time for *reasoning about actions and change*. In other words, one of our interests is to construct a temporal reasoning system for formalising the effects of events (actions)[4] on properties of the ordinary world.

More specifically, we are going to discuss through this paper the formalism to reason about actions and change called Event Calculus. Intuitively, the *Event Calculus* (*EC*), is a temporal (first-order) formalism designed to model and reason about scenarios described as a set of *events*, whose occurrences have the effect of starting or terminating the validity of determined properties of the world (which commonly are called *fluents*).

The first version of the Event Calculus was proposed by Kowalski and Sergot [21]; known in the literature as the *Original Event Calculus*, it is based on the primitive notions of events, fluents, time intervals and time points. However, due to the complexity of the vocabulary (and the use of negation-as-failure) of the original Event Calculus, its axioms present some problems which motivated the proposal of other variants.

In this paper we are going to use the variant of the calculus defined by Shanahan in [37][44] which, instead of negation-as-failure, uses Circumscription as non-monotonic formalism and (only) time points as time units. The Event Calculi are presented in more detail in section 2.2.

The purpose of Circumscription in this framework is to give a non-monotonic solution to the frame problem. This problem is one of the main open issues in reasoning about actions in general, and A.I. planning in particular; it is briefly discussed in the following.

The Frame Problem and the Common Sense Law of Inertia

As we are going to see later (section 4), a general definition of the frame problem is still unclear, since there are many problems identified in the literature under this name. In order to

[2] *Beliefs and desires*, as part of the common sense knowledge, are not discussed in this paper.

[3] Interesting surveys about temporal reasoning can be found in [31] which surveys non monotonic temporal reasoning, [48] about temporal reasoning in A.I. and [15] about temporal reasoning with constraint programming.

[4] In this paper we use interchangeably the words *event* and *action*.

introduce this problem we are going to present the example (proposed by McCarthy [24][26]) that was used to introduce, for the first time, the term *frame problem*; some definitions for this problem, found in the literature, are also presented here in order to give a clearer idea of the meaning of this term. The example follows[5]:

We want to represent a fact like: "Fred is in the kitchen", and the action of "Fred" going to another place; for instance, in order to represent the action "moving to another place" we can state the general fact that "If an object is animate, it will be in a location it goes to".

Unfortunately, it is necessary to provide a large number of axioms stating each property of the world that remains unchanged after the application of an action (axioms of this kind are called *frame axioms*). For instance we would need to state that moving from place to place leaves one's colour unchanged. Since most events leave most facts untouched, most axioms in the system will be of this tedious kind. Most steps of a proof involving dynamical situations like this will be aimed at proving that something is still true in a given state of the world by stringing together such axioms.

Dennett [10] gives the following characterization to the frame problem:

> for this problem concerns how to represent all that hard-won *empirical information*[6] - a problem that arises independently of the truth value, probability, warranted assertability, or subjective certainty of any of it.

The *empirical information* that Dennett refers in the previous quotation is another way to refer to the Searle's concept of *Background Knowledge* [32] [33].

> This is the knowledge that comes before any intention of applying an action, it is all of the fundamental beliefs, capacities and abilities with which we, human beings, interpret reality. It is concerned with non-representative capacities which allow a right representation of the facts.

In conclusion, the frame problem arises due to the impossibility of finding an algorithm capable of computing the Background Knowledge as a whole.

However, (as presented in section 4) given a background theory, the problem of predicting which properties do not change when an action is applied can be solved assuming (by *default*) that *everything remains in the state it is. Any axiom describing the effect of an action on the state of the world represents an exception to this default* [23]. This assumption is called *the common sense law of inertia*; and it is formalised in this framework using Circumscription.

A similar problem arises when formalising logically *space occupancy* as, briefly, discussed below.

[5]This example is reproduced also in [27], but representing the facts with a first-order formalism; however, in order to make it more comprehensible in this introduction, we do not represent it with any specific formalism.

[6]The italic is mine.

1.3 Spatial Reasoning

We said previously that our interest is modeling the reasoning of an autonomous agent immersed in the real world. Obviously, one of the main elements of this reasoning is the interaction of the agent with the external world and, consequently, the formalisation of ordinary space and of the reasoning about the objects located in it.

Even though spatial reasoning is an important field in A.I., it was only during the last five years that fundamental studies on spatial representation and reasoning have appeared in a significant number in the A.I. literature [7].

In the present work (section 5.2) we are going to describe an *axiomatic theory of an absolute space* [39][38]. According to [47], an *absolute space* is the one that exists independently of objects located in it; and, in order to be formalised, it needs to be equipped with a location (or place) function relating each object to its location in space.

In order to logically formalise space, not only do we have to represent the regions occupied by objects but also all the empty regions; moreover it should be done in every state of the world. This problem is similar to the frame problem described above. In the present paper we describe a non-monotonic solution to it, formalising (using Circumscription) the *common sense law of space occupancy*, which intuitively says that *the space is normally empty. Any formula describing an object's location in the world is an exception to this default.* This issue is discussed in section 5 of the present paper.

1.4 Common Sense in a Mobile Robot

The concepts discussed above, taken together, characterize the "Logical Account of the Common Sense Informatic Situation in a Mobile robot" [39].

More specifically, the axioms of Event Calculus (with Circumscription) and the axioms about space are components of a theory denoted by Σ_B. On the other hand, the relations between the robot and the objects in its environment are described in a theory called Σ_E.

The central task of this framework is, then, to assimilate sensor data and to construct action sequences in order to solve a determined goal. Both sensor data assimilation and action planning are done using Abduction w.r.t. the theories Σ_B and Σ_E.

Intuitively, Abduction is the process of explaining a sentence Γ by finding a set of formulae Δ such that, given a theory Σ (background theory), Γ is a logical consequence of $\Sigma \cup \Delta$. This is going to be explained in more detail in section 5.

In the process of assimilating a stream of sensor data Ψ, the task is to find a logical description of the object's initial locations (Δ_M) using Abduction such that

$$\Sigma_B \wedge \Sigma_E \wedge \Delta_N \wedge \Delta_M \models \Psi,$$

where Δ_N is a logical description of the movements of the objects, including the robot itself.

Similarly, planning a sequence of actions in order to satisfy a given goal Γ, given the objects initial locations (Δ_M), is the abduction explanation Δ_N such that

$$\Sigma_B \wedge \Sigma_E \wedge \Delta_N \wedge \Delta_M \models \Gamma.$$

[7] A good overview of the field of spatial representation and reasoning in A.I. can be found in [47].

1.5 The Structure of this Paper

This paper is organised as follows: in the next section we present a brief discussion of the Event Calculus' variants and the Event Calculus as defined by Shanahan [44]; in section 3 Circumscription is discussed together with a survey of the main results about its computational complexity; section 4 presents the Circumscriptive Event Calculus as a solution to the frame problem; the abduction and the theories about space occupancy and about the robot's relation to the world are presented in section 5; and, finally, section 6 concludes this paper.

2 Event Calculus

Event Calculus (*EC*), as already mentioned, is a temporal formalism designed to model and reason about scenarios characterized by a set of *events*, whose occurrence have the effect of starting or terminating determined properties of the world (*fluents*). Trying to be more precise in this definition we are in danger to give a restricted idea about the calculus, since there are many different variants of this formalism.

A better way to talk about Event Calculus is discussing some of its variants. After that, we will be presenting the version of the EC that we are going to use through this paper.

2.1 From Time and Events to Event Calculi

A central idea behind this work is to construct a formal representation of time, not concerned with what time actually is in the real world (its philosophical or physical interpretation) but with a model of time sufficient for reasoning about actions and change in order to implement autonomous systems (more specifically mobile robots). In order to build such a formalism one needs to make a choice about time units, in the case of Event Calculi this choice varies between time intervals [21][2] and time points [21][37][44] [8].

The purpose of these formalisms explicitly linked to the time representation is to model the reasoning of intelligent agents as they plan to act (and reason about the effects of their actions) in the world they are immersed in.

In the present section we are interested in the approaches based in the original Event Calculus [21] which gave birth to the Circumscriptive Event Calculus [37][44], that we are going to discuss through this paper.

Roughly, the *Original Event Calculus* (OEC) is an axiomatisation of events and their effects in terms of PROLOG-like rules, i.e., in terms of Horn clauses extended with the negation-as-failure rule. The main purpose of the negation-as-failure in the OEC is to overcome the frame problem since it carries an intrinsic notion of common-sense law of inertia.

The OEC distinguishes the concepts of time intervals, time points and events as the elementary objects of the language; and axiomatises the relationships between them. Roughly,

[8] Generally, time in Event Calculus is represented on the set of natural numbers; however, when dealing with continuous change, it has to range over the real line (as we are going to see further).

an event happens in a determined time point; an event starts (or ends) a time period during which *fluents* hold [9].

The basic relations used by the OEC are presented below. The event e in the argument of the following predicates is considered as an event occurrence (i.e., an event *token*) and not an event type (as defined in some of the variants of Event Calculus, e.g. [37][44]):

- $After(e, f)$ stands for an interval of time initiated by the event e during which the fluent f holds[10];
- $Before(e, f)$ stands for an interval of time terminated by the event e during which the fluent f holds[11];
- $Holds(p)$ is true of an interval p if for every fluent such that the time interval where f holds has non empty intersection with p, then f is contained in p;
- $HoldsAt(f, t)$ means that the fluent f holds at the time t;
- $Start(p, e)$ represents that the interval p begins with the event e;
- $End(p, e)$ represents that the interval p terminates with the event e;
- $Broken(t, f, t')$ represents that the fluent f ceases to hold between times t and t';
- $Incompatible(f, f')$ represents that the fluents f and f' cannot hold at the same time;
- $Time(e, t)$ says that the event e occurs at a time t;
- $In(t, p)$ represents that the time point t falls within interval p;
- $Initiates(e, f)$ states that the event e initiates the fluent f;
- $Terminates(e, f)$ states that the event e terminates the fluent f;
- $Eq(p, p')$ represents that the two intervals p and p' coincide.

These predicates are constrained by the following axioms (the symbol *not* below refers to negation-as-failure):

[9] There are many definitions for *fluent* in the literature, in this work we understand fluent as a time-dependent property of the world.

[10] Note that the fluent f does not (necessarily) need to start at the same moment that the event e is started.

[11] But it does not mean that the fluent f cannot hold after the starting of the time interval $Before(e, f)$.

$$Holds(after(e,f)) \leftarrow Initiates(e,f) \quad \textbf{(OEC1)}$$

$$Holds(before(e,f)) \leftarrow Terminates(e,f) \quad \textbf{(OEC2)}$$

$$Start(After(e,f),e) \quad \textbf{(OEC3)}$$

$$End(Before(e,f),e) \quad \textbf{(OEC4)}$$

$$Start(Before(e_2,f),e_1) \leftarrow Eq(After(e_1,f),Before(e_2,f)) \quad \textbf{(OEC5)}$$

$$End(After(e_1,f),e_2) \leftarrow Eq(After(e_1,f),Before(e_2,f)) \quad \textbf{(OEC6)}$$

$$\begin{aligned}&Eq(After(e_1,f),Before(e_2,f)) \leftarrow Holds(After(e_1,f))\wedge\\&\quad Holds(Before(e_2,f)) \wedge Time(e_1,t_1) \wedge Time(e_2,t_2)\wedge\\&\quad t_1 < t_2 \wedge not\ Broken(t_1,f,t_2)\end{aligned} \quad \textbf{(OEC7)}$$

$$\begin{aligned}&Broken(t_1,f_1,t_3) \leftarrow Holds(After(e,f_2)) \wedge Incompatible(f_1,f_2)\wedge\\&\quad \wedge Time(e,t_2) \wedge t_1 < t_2 \wedge t_2 < t_3\end{aligned} \quad \textbf{(OEC8)}$$

$$\begin{aligned}&Broken(t_1,f_1,t_3) \leftarrow Holds(Before(e,f_2)) \wedge Incompatible(f_1,f_2)\wedge\\&\quad \wedge Time(e,t_2) \wedge t_1 < t_2 \wedge t_2 < t_3\end{aligned} \quad \textbf{(OEC9)}$$

$$HoldsAt(f,t) \leftarrow Holds(After(e,f)) \wedge In(t,After(e,f)) \quad \textbf{(OEC10)}$$

$$HoldsAt(f,t) \leftarrow Holds(Before(e,f)) \wedge In(t,Before(e,f)) \quad \textbf{(OEC11)}$$

$$\begin{aligned}&In(t,p) \leftarrow Start(p,e_1) \wedge End(p,e_2) \wedge Time(e_1,t_1)\wedge\\&\quad Time(e_2,t_2) \wedge t_1 < t \wedge t < t_2\end{aligned} \quad \textbf{(OEC12)}$$

$$In(t,p) \leftarrow Start(p,e_1) \wedge not\ End(p,e_2) \wedge Time(e_1,t_1) \wedge t_1 < t \quad \textbf{(OEC13)}$$

$$In(t,p) \leftarrow End(p,e_1) \wedge not\ Start(p,e_2) \wedge Time(e_1,t_1) \wedge t < t_1 \quad \textbf{(OEC14)}$$

Due to the fact that the OEC deals with incomplete information and with time intervals, the complexity of its vocabulary gives rise to several problems, as follows:

- *Robustness*: different predicates are not always properly related to one another, which implies that some intuitive properties concerning time intervals are impossible to be proved by the axioms. As presented in [30], none of the following properties can be proved from the OEC axioms:
  - The start of a period (P) is before its end. That is

$$Start(P,E_1) \wedge End(P,E_2) \rightarrow E_1 < E_2;$$

– If a property is terminated after it has been initiated then the period of time for which it holds must have an end, i.e.

$$Holds(After(E_1, P)) \wedge Terminates(E_3, P) \wedge E_1 < E_3 \rightarrow \exists E_2 End(After(E_1, P), E2).$$

- *Semantics*: the semantics of negation-as-failure is an unsolved issue in Logic Programming, which plays against the use of this approach in reasoning about actions and change [44].

In order to simplify the Original Event Calculus several versions of it were proposed without using the notion of time interval in its ontology [20][29][35]; however these variations suffered from lack of generality, since they give incorrect results if the event information is incomplete. A *new event calculus* was proposed in [30] joining the generality of the original version to the simplicity of its extensions based in time points.

However, the question concerning the appropriateness of semantics of negation-as-failure w.r.t. action theories remained unanswered. This problem, together with the limited expressive power of the Horn clauses w.r.t. the full first-order logic, makes us wonder about the possibility of a circumscriptive version of an Event Calculus built with first-order sentences (rather than Horn clauses)[12]. This version is presented in [37][44].

An extension of the EC, using Circumscription to cope with the frame problem, was proposed in [37] involving the evaluation of fluents on states rather than on time intervals or time points. We call it *State-Based Event Calculus*. In this version of the calculus a state is defined as being a set of positive (negative) fluents, so that a fluent holds (does not hold) in a state s if it (its negation) is a member of s. Moreover, each state s is associated with a time point t by a formula $State(t, s)$. Circumscription interacts with the state based EC axioms by minimising the extensions of some predicates. However, the resultant axiomatisation has a complex vocabulary which makes its implementation difficult and computationally very complex. In order to cope with these issues, a simplification was proposed [44].

The simplified version of the Circumscriptive Event Calculus, presented in the next subsection, was motivated by the idea of circumscribing *parts* of the theories (as discussed in section 4) and not the whole EC theory. This version is applied in the area of cognitive robotics [39][43] and is the one that we are interested in.

2.2 Event Calculus: A Circumscriptive Flavor

Below we present the axiomatization of the simplified version of the Circumscriptive Event Calculus[13] [44]. After discussing the basic ideas of Circumscription (section 3), we state the circumscription policy considered, in this version of the EC, to overcome the frame problem.

The Event Calculus is a many sorted first-order language comprising, basically:

- A sort of time points $t_1, t_2, \ldots, t_n, t_{n+1}, \ldots$;

[12]In this case, planning is considered to be an abduction task as defined in [14][43]

[13]From now on, when we use the term em Event Calculus to designate the simplified version of the Circumscriptive Event Calculus.

- A sort of fluents $f_1, f_2, \ldots, f_n, f_{n+1}, \ldots$;
- A sort of event types $a_1, a_2, \ldots, a_n, a_{n+1}, \ldots$.

The following are the predicates of the Event Calculus:

- $HoldsAt(f, t)$ means that the fluent f holds at time t;
- $Initially(f)$ means that the fluent f holds from time 0;
- $Happens(a, t)$ means that the event a happens at time t;
- $Initiates(a, f, t)$ means that the fluent f holds after the occurrence of an event a at time t;
- $Terminates(a, f, t)$ means that the fluent f does not hold after the occurrence of an event a at time t;
- $Releases(a, f, t)$ means that the fluent f is not subject to the common-sense law of inertia after event a at time t;

We also introduce two defined notions:

- $Clipped(t_1, f, t_2)$ means that the fluent f ceased to hold (or it is released from the common sense law of inertia[14]) in some time point in $[t_1, t_2]$;
- $Declipped(t_1, f, t_2)$ means that the fluent f started to hold (or it is released from the common sense law of inertia[15]) in some time point in $[t_1, t_2]$.

The predicates described above are constrained by the following axioms:

$$HoldsAt(f, t) \leftarrow \quad \textbf{(EC1)}$$
$$Initially(f) \land \neg Clipped(0, f, t)$$

$$HoldsAt(f, t_2) \leftarrow \quad \textbf{(EC2)}$$
$$Happens(a, t_1) \land Initiates(a, f, t_1) \land t_1 < t_2 \land \neg Clipped(t_1, f, t_2)$$

$$\neg HoldsAt(f, t_2) \leftarrow \quad \textbf{(EC3)}$$
$$Happens(a, t_1) \land Terminates(a, f, t_1) \land t_1 < t_2 \land \neg Declipped(t_1, f, t_2)$$

We have to consider also the following defined notions:

$$Clipped(t_1, f, t_2) := \quad \textbf{(EC4)}$$
$$\exists a, t[Happens(a, t) \land [Terminates(a, f, t) \lor Releases(a, f, t)] \land$$
$$\land t_1 < t \land t < t_2]$$

$$Declipped(t_1, f, t_2) := \quad \textbf{(EC5)}$$
$$\exists a, t[Happens(a, t) \land [Initiates(a, f, t) \lor Releases(a, f, t)] \land$$
$$\land t_1 < t \land t < t_2]$$

[14] This fact is going to be discussed below.
[15] Idem 14.

The predicate $HoldsAt(f,t)$, according to axioms (EC1) and (EC2), says that the fluent f is true from the origin of the time (time zero) to the time point t (EC1); or from a time point t_1 to a time point t_2 if an event initiates f at a time t_1 (EC2). In other words, according to (EC1), $HoldsAt(f,t)$ is true if the fluent f holds at time zero (represented by $Initially(f)$) and no event occurred that terminates (or releases) the fluent f in the time interval $[0,t]$ (r.b.[16] $\neg Clipped(0,f,t)$). According to (EC2), $HoldsAt(f,t)$ is true if an event a happens at time t_1 initiating the fluent f, and no event occurred that terminates (or releases) f during the time interval $(t_1,t_2]$.

The predicate $Releases(a,f,t)$, used in the definitions (EC4) and (EC5), represents the case that action a causes indetermination in the value of fluent f. The use of the *Releases* predicate can be exemplified by the following variant of the Yale Shooting Scenario: consider a gun with only one bullet and, after each *Load* action, the gun's chamber is spun (as in Russia *roulette*). Clearly, after the *Spin* action the value of the *Loaded* fluent is unknown. In order to represent it we can write:

$$Releases(Spin, Loaded, t).$$

After the sequence *Load*, *Spin*, *Shoot* we cannot conclude whether the target is achieved.

The *Releases* predicate turns a *frame fluent* (i.e. the one which value is subject to the common sense law of inertia) into a non-frame fluent, i.e., it allows the fluctuation of the fluent's values. Once a fluent is initiated or terminated by an event, it is once more subject to the common sense law of inertia, and must be considered a frame fluent.

The axiom (EC3) says that a fluent f **does not** hold at a time point t_2 (r.b. $\neg HoldsAt(f,t_2)$) if an event a occurs at a time t_1 (r.b. $Happens(a,t_1)$) that terminates the fluent f, and no other event occurs that initiates f in the time interval $(t_1,t_2]$ (r.b. $\neg Declipped(t_1,f,t_2)$).

The sentence (EC4) defines the predicate $Clipped(t_1,f,t_2)$ in the way that it is true if an event occurs (in the time interval (t_1,t_2)) such that the fluent f is terminated by the event or is released from the common-sense law of inertia (r.b. $Releases(a,f,t)$).

Analogously to (EC4), (EC5) says that the predicate $Declipped(t_1,f,t_2)$ is true if an event a occurs at time point t ($t \in (t_1,t_2)$) such that a initiates the fluent f or releases it from the common-sense law of inertia.

The formalism presented above is just the kernel of the Circumscriptive Event Calculus. In order to incorporate formalisations of *ramification*, *non-deterministic effects of actions*, *concurrent events* and *continuous change*, some extensions of the EC have been proposed introducing new axioms and/or modifying the axioms of the calculus, as can be seen in [37][39][44].

In order to cope with the frame problem, the language of Event Calculus is extended with a second-order formula (Circumscription) allowing reasoning with defaults in the calculus, as we are going to see in section 4.

A last comment about the Event Calculus axioms is worth making. The axiomatisation presented in (EC1) to (EC5) has fewer axioms than the one presented by (OEC1) to

[16] The abbreviation r.b. means *represented by*.

(OEC14), intuitively because the former does not use time intervals (but time points instead). Consequently, there is no need to write the axioms (OEC13) and (OEC14), or to define relations between the predicates $Start$, End, $After$ and $Before$ (axioms (OEC3) to (OEC6)).

We are going now to make some considerations about the semantics of the version of Event Calculus that we are dealing with.

A Model for the Event Calculus

We are going to define a structure M having as universe $U = \mathcal{R} \cup (\mathcal{R} \times \mathcal{R})$, and the following interpretations of the constant and relation symbols:

Constant Interpretation:

- Fluent: finite set of half open intervals $(r, s]$ of maximum length, i.e. disjoint;
- Event(type): the only events of interest are the ones which initiate or terminate fluents. Accordingly we define event types

$$\{a_f^- = \langle f, r\rangle | \exists s((r, s] \in f)\}$$

$$\{a_f^+ = \langle f, s\rangle | \exists r((r, s] \in f)\}$$

Relation Interpretation:

- $Releases := \emptyset$;
- $HoldsAt(f, t) := \exists I \in f(t \in I)$;
- $Happens(a, t) := \exists f((a = a_f^- \vee a = a_f^+) \wedge (\langle f, t\rangle \in a))$;
- $Initiates(a, f, t) := (a = a_f^- \wedge \langle f, t\rangle \in a)$;
- $Terminates(a, f, t) := (a = a_f^+ \wedge \langle f, t\rangle \in a)$;
- $Initially(f) := \exists r, s(r < 0 < s \wedge (r, s] \in f)$.

Definitions (EC4) and (EC5) then force:

- $Clipped(t_1, f, t_2) := \exists t(t_1 < t < t_2 \wedge \langle f, t\rangle \in a_f^+)$;
- $Declipped(t_1, f, t_2) := \exists t(t_1 < t < t_2 \wedge \langle f, t\rangle \in a_f^-)$.

Consistency of the axioms:

Axiom (EC1) is translated as follows:

$$\forall t'(0 < t' < t \to \langle f, t'\rangle \notin a_f^+) \wedge \exists r, s(s > 0 \wedge 0 \in (r, s] \in f) \to \exists r, s(t \in (r, s] \in f).$$

Proof:
Define $s_0 := sup\{s | \exists r < 0\ (r, s] \in f\}$. Then, s_0 exists and $0 < s_0$.

Suppose, reasoning by contraposition, that $s_0 < t$. Then $\langle f, s_0\rangle \notin a_f^+$. However, by definition of s_0,

$$\langle f, s_0\rangle \in a_f^+.$$

$\square$

The translation of (EC2) becomes:

$$\exists f'((a = a_{f'}^- \vee a = a_{f'}^+) \wedge \langle f', t_1\rangle \in a) \wedge a = a_f^- \wedge \langle f, t_1\rangle \in a \wedge (t_1 < t_2) \wedge \forall t(t_1 < t < t_2 \to \\ \to \langle f, t\rangle \notin a_f^+) \to \exists r, s(t_2 \in (r, s] \in f).$$

Proof:
Observe that

$$a = a_f^- \wedge \langle f, t_1\rangle \in a \to \exists f'((a = a_{f'}^- \vee a_{f'}^+) \wedge \langle f', t_1\rangle \in a),$$

hence it is necessary and sufficient to show

$$\forall a t_1[a = a_f^- \wedge \langle f, t\rangle \in a \wedge (t_1, t_2)\forall t(t_1 < t < t_2 \to \langle f, t\rangle \notin a_f^+) \to \exists r, s(t_2 \in (r, s] \in f)].$$

Define $s_0 := sup\{s | \exists r < t_2\ (r, s] \in f\}$.

By definition of s_0, $\langle f, t\rangle \in a_f^+$ and $s_0 > t_1$; we want to prove that $t_2 \leq s_0$. Suppose, reasoning by contraposition, $s_0 < t_2$, then $\langle f, s_0\rangle \notin a_f^+$, contrary to the definition of s_0.

$\square$

The translation of (EC3) is the following:

$$\exists f'((a = a_{f'}^- \vee a = a_{f'}^+) \wedge \langle f', t\rangle \in a) \wedge a = a_f^+ \wedge \langle f, t\rangle \in a \wedge \forall t(t_1 < t < t_2 \to \\ \to \langle f, t\rangle \notin a_f^-) \to \neg(\exists r, s(t_2 \in (r, s] \in f)).$$

Proof:
In order to prove the model satisfies (EC3) we are going to argue by contraposition, as follows:

If $M \models HoldsAt(f, t_2)$ then

$$M \models \exists a t_1[\neg Happens(a, t_1) \vee \neg Terminates(a, f, t_1) \vee (t_2 \leq t_1) \vee Declipped(t_1, f, t_2)].$$

Then, it is enough to show that there exists a and t_1 such that one of the disjunctive terms above is true.

Consider, $M \models HoldsAt(f, t_2)$ iff $\exists r, s((r, s] \in f \wedge r < t_2 \leq s)$; and take $t_1 := s$, then $t_2 \leq s =: t_1$. Consequently we have $\forall a(t_2 \leq t_1)$; in particular it holds for $a = a_f^+$.

$\square$

3 Circumscription

Circumscription is a method for default reasoning constructed on classical logic which involves minimising the extensions of predicates. More specifically, *the circumscription of a property P w.r.t. a certain fact A* dictates that the objects that can be shown to have the property P (by reasoning from certain facts A) are **all** the objects that satisfy P. In other words, as presented by McCarthy [25], Circumscription can be used to conjecture that the tuples $\langle a_1, a_2, \ldots, a_n\rangle$ which can be shown to satisfy a relation $P(x_1, x_2, \ldots, x_n)$ are *all* the tuples satisfying this relation.

This idea can be explained by the following example, due to Lifschitz [22]: consider a blocks world such that, **normally, the blocks are on the table (A1)**. Then we just need to add information about the location of a block if it is an exception to the default information (A1); so, if for instance there is a block that does not obey this rule, we have to make explicit that this supposed **block B_1 is not on the table (A2)**. However, if we consider another block B_2, we must conclude (according to the default (A1)) that B_2 **is on the table (A3)**. We can represent this problem in first-order logic as follows:

$$Block(x) \wedge \neg Ab(x) \rightarrow Ontable(x), \tag{1}$$

$$\neg Ontable(B_1). \tag{2}$$

The formulae 1 and 2 represent respectively the rules (A1) and (A2). The predicate $Ab(x)$ represents that the object x is an exception to the default (A1); i.e., x is **abnormal**.

We have to write also the fact that B_1 and B_2 are two different blocks:

$$Block(B_1), Block(B_2), B_1 \neq B_2. \tag{3}$$

With this formalisation we want to conclude that the following formula holds:

$$Ontable(B_2). \tag{4}$$

However, this fact cannot be derived in first-order logic, as we can see in the following counter-example: consider a classical model M as composed of a universe $U = \{B_1, B_2\}$, such that the predicates *Block* and *Ab* are universally true and *Ontable* is universally false (i.e., a model such that all the blocks are abnormal). Thus, it is clear that the axioms 1, 2 and 3 are true in M while 4 is false.

The model M, presented in the last paragraph, is not desirable since we should not consider the block B_2 as abnormal if it is not stated to be; the formalism has to constrain the extensions of the predicate *Ab* only to the objects explicitly mentioned to be exceptions to the default (such as the block B_1).

From the last paragraph we can conclude that our interest is a certain minimality condition of the models, which is going to be formalised further.

Continuing the example, we can create another model M' such that the block B_2 is on the table and, consequently, the set of abnormal objects is the singleton $\{B_1\}$. The axioms

1, 2 and 3 clearly are satisfied by M'. Then we can say that the extensions of Ab in M' is a proper subset of its extensions in M and, in this sense, we **cannot** create another model M'' such that the extensions of Ab in M'' is a proper subset of its extensions in M'. Clearly, the axioms 1, 2 and 3 logically imply $Ab(B_1)$. In conclusion, the extensions of Ab in M' are minimal.

This is the main idea of Circumscription, i.e., to consider only the models that satisfy a certain minimality condition (w.r.t. the *abnormal* predicates) instead of arbitrary models. Formally, Circumscription is a syntactic transformation of logical formulae, in the sense that it transforms a first-order sentence A into a second-order sentence A^*, such that the models of A^* are precisely the minimal models of A.

In order to give a formal definition for Circumscription, it is useful to introduce the following notation: for any predicate symbols $\mathcal{P}$, $\mathcal{Q}$ of the same arity; and $\overline{x}$ a tuple of distinct variables, we say that:

- $\mathcal{P} = \mathcal{Q}$ stands for $\forall \overline{x}(\mathcal{P}(\overline{x}) \leftrightarrow \mathcal{Q}(\overline{x}))$;
- $\mathcal{P} \leq \mathcal{Q}$ stands for $\forall \overline{x}(\mathcal{P}(\overline{x}) \rightarrow \mathcal{Q}(\overline{x}))$;
- $\mathcal{P} < \mathcal{Q}$ stands for $(\mathcal{P} \leq \mathcal{Q}) \wedge \neg(\mathcal{P} = \mathcal{Q})$.

Definition 1 (Circumscription I) *Let $A(P)$ be a sentence containing a predicate constant P. The circumscription of $A(P)$ minimising P is the following second-order sentence $A^*(P)$:*

$$A(P) \wedge \neg\exists p[A(p) \wedge p < P],$$

where p is a predicate variable of the same arity as P.

We denote the circumscription of a formula A w.r.t. the predicate P by $CIRC[A; P]$.

The subformula $\neg\exists p[A(p) \wedge p < P]$ states that the extension of P is minimal given $A(P)$.

Another simple example is worth mentioning: let Σ be the conjunction of the following formulae: $P_1(a)$, $P_1(b)$ and $P_2(c)$. Then we have that $\Sigma \models P_1(a)$, but we cannot conclude that $\Sigma \models P_1(c)$ nor that $\Sigma \models \neg P_1(c)$. The application of the circumscription operator in Σ minimising P_1 (say $CIRC[\Sigma; P_1]$) makes P_1 true for only those objects that satisfy P_1 and are explicitly stated in Σ. This gives us $CIRC[\Sigma; P_1] \models \neg P_1(c)$, assuming that $a \neq c$ and $b \neq c$. More generally we have:

$$CIRC[\Sigma; P_1] \models \forall x[P_1(x) \leftrightarrow [x = a \vee x = b]].$$

The definition of Circumscription just stated is not strong enough to deal with more complex problems. For situations where we need to specify that some of the objects (functions or predicate constants) occurring in the circumscribed formula are allowed to vary, while minimising a predicate (or some predicates), we must consider the following extended definition for Circumscription:

Definition 2 (Circumscription II) *Let $A(P, Z_1, \ldots, Z_m)$ be a sentence, where P is a predicate constant and $Z_1, \ldots, Z_m$ objects, functions, and/or predicate constants. The circumscription of P in A varying $Z_1, \ldots, Z_m$ is the sentence*

$$A(P, Z_1, \ldots, Z_m) \wedge \neg\exists p z_1 \ldots z_m [A(p, z_1 \ldots z_m) \wedge p < P], \tag{5}$$

where p is a predicate variable of the same arity as P; if Z_i is an object constant, then z_i is an object variable, and if Z_i is a function or predicate constant then z_i is a function/predicate variable of the same arity of Z_i.

We denote the circumscription of a formula A minimising the predicate P and letting Z vary by $CIRC[A; P; Z]$ (where Z is an abbreviation of the tuple $Z_1, \ldots, Z_m$).

The predicates occurring in P are called *minimised* while those not occurring in $P \cup Z$ are denoted by Q and are called *fixed*.

The models of $CIRC[A; P; Z]$ are the models of A in which the extension of P cannot be made smaller without loosing the property A, even at the price of changing the interpretations of the constants Z [22]. This notion can be made precise in the following way:

Let $|M|$ denote the universe of the structure M and $[|C|]_M$ the interpretations of all constant, function and predicate constants C in $\mathcal{L}$.

Consider two structures M_1 and M_2 for a one-sorted language $\mathcal{L}$ [17], and let $CIRC[A; P; Z]$ denote the circumscription of the formula A w.r.t. the predicate P letting Z vary. Then we write $M_1 \leq_{P,Z} M_2$ if

- $|M_1| = |M_2|$;
- $[|C|]_{M_1} = [|C|]_{M_2}$ for every predicate constant C which is different from P and does not belong to Z;
- $[|P|]_{M_1} \subset [|P|]_{M_2}$.

In other words $M_1 \leq_{P,Z} M_2$ means that M_1 and M_2 differ only on how they interpret P and Z, and the extension of P in M_1 is a subset of its extension in M_2.

In particular we can say that a model M of A is minimal w.r.t. $\leq_{P,Z}$ if there is no model M' of A such that $M' \leq_{P,Z} M$.

An important result is the following proposition (due to Lifschitz [22]):

Proposition 1 *A structure M is a model of $CIRC[A; P; Z]$ if and only if M is a model of A which is minimal w.r.t. $\leq_{P,Z}$.*

Proof:

Specifying values of p, z for which $A(p, z) \wedge p < P$ is true in M is equivalent to specifying a model M' of A such that $M' \leq_{P,Z} M$.

$\square$

Up to this point we gave a general idea of Circumscription. Now we are going to consider how to compute circumscription in specific cases.

[17] For simplicity we are dealing only with the one-sorted case, the definitions for the many-sorted one are straightforward.

3.1 Some Aspects of Circumscription

Let's start reproducing some rules for computing circumscription, as presented in [22].

3.1.1 Rules for Computing Circumscription

In order to simplify a Circumscription formula we use the following rules:

Proposition 2 (Predicate completion) *If $F(x)$ does not contain P, then the circumscription*

$$CIRC[\forall x(F(x) \to P(x)); P]$$

is equivalent to $\forall x(F(x) \leftrightarrow P(x))$.

Proof:

In order to make the proof of the equivalence above simpler let's use the, already introduced, notation $P = Q$, $P \leq Q$ and $P < Q$, where P or Q is a formula. Thus, the proposition above can be stated as follows:

$$CIRC[E \leq P; P] \equiv (E = P)$$

for any formula E which does not contain P.

Applying the definition of Circumscription (5) in the left hand side of the expression just stated we get:

$$E \leq P \wedge \neg\exists p(E \leq p \wedge p \wedge p < P). \tag{6}$$

However,

$$\exists p(E \leq p \wedge p \wedge p < P) \equiv E < P.$$

Consequently formula (6) can be written as

$$E \leq P \wedge \neg(E < P)$$

which is equivalent to $E = P$.

$\square$

The following proposition shows how Circumscription with varied predicate constants can be reduced to the basic case:

Proposition 3 (Elimination of varied constants) *If Z does not contain function constants or predicate constants, the circumscription $CIRC[A(P,Z); P; Z]$ is equivalent to $A(P,Z) \wedge CIRC[\exists z A(P,z); P]$.*

Proof:

The right hand side of the equivalence, namely

$$A(P,Z) \wedge CIRC[\exists z A(P,z); P]$$

can be written (by the definition of Circumscription) as

$$A(P, Z) \wedge \exists z A(P, z) \wedge \neg \exists p[\exists z A(p, z) \wedge p < P],$$

which is equivalent to

$$A(P, Z) \wedge \neg \exists p z[A(p, z) \wedge p < P].$$

This expression is exactly the definition of $CIRC[A(P, Z); P; Z]$.

□

If the list Z contains function or predicate constants then $CIRC[\exists z A(P, z); P]$ is a circumscription of a second order formula and, in order to apply the simplification stated in proposition 3, we need to reduce $\exists z A(P, z)$ to a first-order formula.

For the next two propositions we need to recall that an occurrence of a predicate symbol in a formula is positive (negative) if it is in the scope of an even (odd) number of negations.

Proposition 4 (Negative Formulae) *If $B(P)$ is negative, then the circumscription*

$$CIRC[A(P) \wedge B(P); P]$$

is equivalent to

$$CIRC[A(P); P] \wedge B(P).$$

Proof:

By the definition of Circumscription we have that the formula $CIRC[A(P) \wedge B(P); P]$ stands for

$$A(P) \wedge B(P) \wedge \neg \exists p[A(p) \wedge B(p) \wedge p < P]. \tag{7}$$

Recalling that

$$\forall p q[B(p) \wedge q \leq p \rightarrow B(q)]$$

is a tautology in second-order logic for $B(p)$ negative, we have that $B(P)$ and $p < P$ imply $B(p)$. Hence the expression 7 can be replaced by the expression

$$A(P) \wedge B(P) \wedge \neg \exists p[A(p) \wedge p < P],$$

which is $B(P) \wedge CIRC[A(P); P]$.

□

Proposition 5 (Positive Formula) *If $A(P, Z)$ is positive relative to P then the circumscription $CIRC[A(P, Z); P; Z]$ is equivalent to*

$$A(P, Z) \wedge \neg \exists x z[P(x) \wedge A(\lambda y(P(y) \wedge x \neq y), z)].$$

Proof:

It is sufficient to prove that

$$\exists p[A(p,z) \wedge p < P]$$

is equivalent to

$$\exists x[P(x) \wedge A(\lambda y(P(y) \wedge x \neq y), z)]$$

($\rightarrow$) Suppose p such that $A(p,z)$ and $p < P$, and x such that $P(x)$ and $\neg p(x)$. Clearly, $p \leq \lambda y(P(y) \wedge x \neq y)$. Since $A(P,Z)$ is positive w.r.t. P, knowing that

$$\forall pq[B(p) \wedge p \leq q \rightarrow B(q)]$$

is a tautology in second-order logic for $B(p)$ positive, we have that $A(\lambda y(P(y) \wedge x \neq y), z)$ holds.

($\leftarrow$) Suppose x such that $P(x)$ and $A(\lambda y(P(y) \wedge x \neq y), z)$ hold, and take p to be $\lambda y(P(y) \wedge x \neq y)$, then clearly $\exists p[A(p,z) \wedge p < P]$ holds.

□

The following is an important result [22]:

Theorem 1 *Let P be a tuple of predicate symbols $P_1, P_2, \ldots, P_n$, and $A(P,Z)$ be any formula. If all occurrences in $A(P,Z)$ of the predicate symbols in P are positive, then*

$$CIRC[A(P,Z); P; Z]$$

is equivalent to

$$CIRC[A(P,Z); P_1; Z] \wedge CIRC[A(P,Z); P_2; Z] \wedge \ldots \wedge CIRC[A(P,Z); P_n; Z].$$

Proof:

The circumscription $CIRC[A(P,Z); P, Z]$ can be written as

$$A(P,Z) \wedge \neg \exists pz \left(A(p,z) \wedge (p \leq P) \wedge \bigvee_i \neg(p_i = P_i) \right),$$

or

$$A(P,Z) \wedge \bigwedge_i \neg \exists pz(A(p,z) \wedge (p \leq P) \wedge \neg(p_i = P_i)).$$

Thus, it is sufficient to show that the second conjunctive term of the sentence

$$CIRC[A(P,Z); P_i; Z]$$

is equivalent to

$$\neg \exists pz(A(p,z) \wedge (p \leq P) \wedge \neg(p_i = P_i)).$$

To simplify the notation, take i=1; we need to prove that

$$\neg \exists p_1 z(A(p_1, P_2, \ldots, P_n, z) \wedge (p_1 < P_1)) \tag{8}$$

is equivalent to

$$\neg\exists pz(A(p,z) \wedge (p \leq P) \wedge \neg(p_1 = P_1)).$$

The last formula can be written as

$$\neg\exists p_1 z(\exists p_2 \ldots p_n(A(p_1, p_2, \ldots, p_n, z) \wedge \\ \wedge(p_2 \leq P_2) \wedge \ldots \wedge (p_n \leq P_n)) \wedge (p_1 < P_1)) \tag{9}$$

To show that (8) is equivalent to (9), observe that $A(p_1, P_2, \ldots, P_n, z)$ and

$$\exists p_2 \ldots p_n(A(p_1, p_2, \ldots, p_n, z) \wedge (p_2 \leq P_2) \wedge \ldots \wedge (p_n \leq P_n)),$$

are equivalent since $A(P, Z)$ is positive relative to $P_2, \ldots, P_n$.

□

A more detailed discussion of these rules can be found in [22]. We now discuss the computational complexity of Circumscription.

3.2 A Note on Complexity

The original purpose of non-monotonic reasoning (NMR) was the development of a formalism capable to represent knowledge more compactly than first-order logic. A consequence of this is that non-monotonic knowledge bases would be much less space consuming than the ordinary ones, increasing in this way the performance of theorem provers. In other words, the objective of non-monotonic reasoning was twofold: to give a compact representation of knowledge and to allow faster inference. However, the results on the computational complexity of the present NMR formalisms do not reflect these initial goals.

In most of the cases studied, the complexity of NMR is much worse compared to the complexity of the first-order logic, for example (as discussed in [5]) the reasoning task of deciding if a literal follows from a propositional Horn formula can be done in linear time in first-order logic [11], while using Circumscription the reasoning task is co-NP-complete [7]. Continuing this game of comparisons between monotonic reasoning and NMR we have that, generally, tractable problems may become intractable (e.g. the complexity increases from polynomial to NP-complete [18]) and intractable problems may become "more intractable" (e.g. from NP-complete to Σ_2^p complete [16]).

Before describing the results of complexity of Circumscription, some basic notions of complexity theory are worth mentioning.

3.2.1 Background Concepts

The complexity results about Circumscription are taken in relation to a Turing machine with oracle (*computation with oracles*). Intuitively oracles are subroutines without cost which are responsible for the solution of a particular class of problems. More specifically, given a class of decision problems C, the class P^C (NP^C) is the class of decision problems that can be solved in polynomial time by a deterministic (non-deterministic) machine that uses an oracle for the problems in C, i.e., a subroutine for any problem in C that can be called several times, spending just one time-unit for each call [8]. The next definitions (based on oracle computations) are worth mentioning:

The classes Σ_k^p, Π_k^p and Δ_k^p of the polynomial hierarchy are defined by

$$\Sigma_0^p = \Pi_0^p = \Delta_0^p = P$$

and for all $k \geq 0$:

$$\Sigma_{k+1}^p = NP^{\Sigma_k^p}, \quad \Pi_{k+1}^p = co - NP^{\Sigma_k^p}, \quad \Delta_{k+1}^p = P^{\Sigma_k^p}.$$

Notice that

$$\Sigma_1^p = NP, \quad \Pi_1^p = co{-}NP, \quad \Delta_1^p = P.$$

The following are conjectures:

$$\Delta_k^p \subset \Sigma_k^p, \quad \Delta_k^p \subset \Pi_k^p, \quad \Sigma_k^p \subset \Delta_{k+1}^p, \quad \Pi_k^p \subset \Delta_{k+1}^p, \quad \Sigma_k^p \neq \Pi_k^p,$$

for all $k \geq 1$.

In particular, many problems in propositional non-monotonic reasoning are Π_2^p-complete or Σ_2^p-complete, and many algorithms designed for solving such problems use satisfiability testers as subroutines [8].

Having these definitions in mind, let's present some complexity results about Circumscription.

3.2.2 Complexity of Circumscription

Many attempts have been made to find classes of first-order formulae whose Circumscription is equivalent to first-order formulae; in these classes the inference was proved to be r.e.[18]. However, the complexity of deciding whether a given formula is in one of these classes is unknown.

Several results in the literature suggest that circumscriptive inference is harder than first order inference; we mention some of them.

The inference of a formula γ from a circumscription was proved to be as hard as testing validity of a formula in second-order logic [6]. This result holds even when γ is a quantifier free sentence, regardless of the presence of varying and/or fixed predicates. In fact, fixed predicates do not contribute to the complexity of Circumscription [9].

When the inferred formula γ is a clause, then the problem is co-NP-complete for very restricted cases; for instance, when the formula A in the expression (5) is both Horn and 2CNF [19] or when A is 2CNF and only positive literals occur in it. A polynomial case arises when A is Horn, 2CNF and with no clauses of the form $\neg x \vee \neg y$. Further polynomial cases

[18] The abbreviation *r.e.* means recursively enumerable.

[19] A 2CNF formula is a conjunction of clauses such that each clause has no more than 2 literals.

were found for other restricted versions, for example, when A is Horn and $Q = \emptyset$, or when A is 2CNF, γ is a literal and $Z = \emptyset$ [8].

An interesting result is the one presented in [12], where the inference of the propositional version of Circumscription (formula (5)) (with $Q = \emptyset$ and $Z = \emptyset$ or not) is proved to be complete for the class Π_2^P of the polynomial hierarchy. Propositional circumscription is a special case of formula (5), defined as follows:

Definition 3 (Propositional Circumscription [7]) *Let A be a propositional formula and $X = \{x_1, \ldots, x_n\}$ its alphabet. The circumscription formula is the following sentence:*

$$A \wedge (\forall Y(A[Y] \rightarrow \neg Y < X))$$

where $Y = \{y_1, \ldots, y_n\}$ is a list of propositional letters disjoint from X, $A[Y]$ is A with all the occurrences of atoms of X substituted by the corresponding ones in Y.

Clearly, for any more complex case (such as the circumscription of a first-order theory, as far as the Event Calculus is concerned), the complexity of circumscription is (at least) Π_2^P-complete.

With respect to satisfiability, it was proved that the task of deciding whether the formula (5) has a countably infinite model is Σ_2^1-complete.

The complexity of Circumscription arises even in the problem of *model checking* (which is the task of deciding whether a finite structure satisfies a given formula Ψ). However, when Ψ is the formula (5) the task is co-NP-complete [19]; this result holds when A is a universal existential first-order sentence and $Z \neq \emptyset$. Actually, co-NP is the upper bound for the model checking of the circumscription of all first-order formulae A in expression (5). This result is strengthened in [4], where co-NP-hardness of the model checking is proved for the case when A is quantifier free, satisfiable, and $Q = Z = \emptyset$. If the formula A to be circumscribed is a logic program the complexity is polynomial [19].

4 Circumscriptive Event Calculus: A Solution to the Frame Problem?

After discussing the Event Calculus (section 2) and Circumscription (section 3), we are ready to go through the circumscription policy used to solve the frame problem in the version of EC considered in this work (section 2.2). This section presents such a circumscription policy and gives the solution to the Hanks-McDermott problem as an example; after this, section 4.4 presents the *continuous change* as an extension of the calculus.

4.1 "Solving the Frame Problem"?

In the final panel of the Actions and Change workshop[20], the following question was proposed by the chair:

[20] Part of the workshop of non-monotonic reasoning of the Conference of Knowledge Representation held in Trento, 1998.

"So... The frame problem is, at last, solved. Isn't it?"

The answers were the following:

According to Ray Reiter:

"We believe we solved it!"

Eric Sandewall completed as follows:

"Let's affirm that it is solved and call the future problems with different names!"

The question was directed to John McCarthy (who coined the term "frame problem" and proposed Circumscription), who absented himself from an answer.

In order to understand this controversy and to analyse the solution given by the Circumscriptive Event Calculus, we have to discuss what frame problem really is. This is briefly done as follows.

Defining the frame problem is itself a problem because many authors refer to different issues under this name. Originally, the frame problem was taken to be the problem of predicting, within a formal language, that most objects of the world remain the same when actions are performed. More specifically, this problem is called the *persistence problem*.

Some definitions of the frame problem take into account, not only the persistence problem, but also the problem of predicting how properties change over time due to the application of actions. This problem is identified as the *temporal projection problem*.

In a more general sense we could think of the frame problem as being the problem of predicting *effects of effects* of actions, i.e., predicting the *indirect effects of actions*; this problem is known as *ramification problem*. For a simple example consider a room with one air duct and many blocks inside; in this room actions to move blocks can be applied. It is clear that the effect of the action *moving* a block on top of the air duct is just placing the block on that position, but doing so the air duct becomes blocked causing the room to be stuffy. The "stuffiness" is an indirect effect of an action, and must be predicted by a formal theory of actions and change.

The counterpart of the temporal projection problem, i.e., the problem of specifying what conditions must be true in the world for a given action to have its intended effects (*qualification problem*) is also considered in some definitions of the frame problem. It is clear that there is an impractical and implausible number of qualifications in order to fully represent the conditions for the successful performance of an action [25].

Other problems are also identified as a frame problem in the literature [31][28]. We call the set of all of these problems *Frame Problems*.

In Shanahan's book "Solving the Frame Problem", the frame problem is taken to be the persistence problem, as can be seen in the following quotation [44]:

> "The frame problem arises when we attempt to describe the effects of actions or events using logic. In a nutshell, the problem is this. If we write our description using classical logic, as well as describing what changes when a particular kind of action is performed or a particular kind of event occurs, we also have to describe what does *not* change. Otherwise, we find that we cannot use the description to draw any useful conclusions."

In the book, the ramification problem is not dealt with at length. A solution to this problem is presented in terms of *domain constraints*, which are formulae expressing temporally invariant facts about the logical relationship among fluents.[21]

Domain constraints, in the Event Calculus, are universally quantified $HoldsAt$ formulae. For instance, in the example of the stuffy room presented above, we can express the fact that the room becomes *stuffy* using the following domain constraint:

$$HoldsAt(Stuffy, t) \leftrightarrow HoldsAt(On(Block_i, Duct), t),$$

where $Block_i$ is a block inside of the room and $on(Block_i, Duct)$ is a term representing that the block i is on top of the duct.

Such formulae are conjoined in the theory outside the scope of the circumscriptions in the circumscription policy stated in formula (10), presented in the next section.

The temporal projection problem and the qualification problem are only mentioned in [44] but no solution is presented.

Let's now turn to Shanahan's solution to the persistence problem.

4.2 Circumscription Policy for Event Calculus

The persistence problem is overcome in the Event Calculus circumscribing first the predicates $Initiates$, $Terminates$ and $Releases$, and then circumscribing $Happens$. By minimising $Initiates$, $Terminates$ and $Releases$ we are imposing that the known effects of actions are the only effects of actions, and minimising $Happens$ we impose that the known event occurrences are the only event occurrences; in doing so, we formalise the common sense law of inertia. More formally:

Given a conjunction Σ of formulae, not including the predicates $Initially$ or $Happens$, describing the effects of actions (*domain description*), a conjunction Δ of formulae, not including the predicates $Initiates$, $Terminates$ or $Releases$, describing the *narrative* of actions and events, a conjunction Ω of uniqueness-of-names axioms for actions and fluents, a conjunction EC of the axioms of the Event Calculus ((EC1) to (EC5)) and a conjunction Ψ of domain constraints, we are interested in[22],

$$CIRC[\Sigma; Initiates, Terminates, Releases] \wedge CIRC[\Delta; Happens] \wedge EC \wedge \Omega \wedge \Psi. \quad (10)$$

[21] An interesting approach to the ramification problem is the one presented by Thielscher in [46]. A description of this issue, however, is outside the scope of the present paper.

[22] In the circumscription policy represented in the sentence 10, the predicate $Initially$ should be circumscribed together with $Happens$ (in the second conjunctive term) in order to guarantee that the facts stated in the initial state of the world are the only facts that hold in in the time 0. Due to lack of time to fix this problem through the present text, we decided to keep the definition stated in [39][44]. Actually, in [44] this problem does not arise since there are two predicates defining the initial state: $Initially_p$ describing the fluents that are positive in the time 0; and $Initially_n$ describing the fluents that are negative in the time 0. In order to solve this problem, in the present paper, we can define the negation of a fluent f as (following the model theoretic counterpart described in section 2.2) $\neg f = \{I | \forall I' \in f\ I' \cap I = \emptyset \textit{ and } I \textit{ is maximal with this property}\}$. A more detailed study of this problem is left for a future work.

The main achievements of this policy are its main source of criticisms. Note that, in the expression (10), the circumscription of a theory is done in two parts (written as $CIRC[\Sigma; Initiates, Terminates, Releases]$ and $CIRC[\Delta; Happens]$); this is assumed *by construction*. This procedure can be criticised by the following arguments: first, if Circumscription is a correct rule of conjecture for default reasoning, then it should be correct for the whole theory and not only for parts of it (or Circumscription should provide a systematic method for choosing what predicates to circumscribe); the second objection is that, circumscribing in parts, we do not take account of the logical relations between the resultant pieces of the theory; so it is impossible to add some information in this version of Event Calculus and, consequently, some conclusions cannot be drawn by this framework. For instance, a formula containing *Initiates* (or $Terminates$ or *Releases*), and $Happens$ (or *Initially*) predicates is not allowed to be written in the theory Σ (or Δ). For a simple example, consider the following example (proposed by Tom Costello [44]): I flick a switch and them enter a room, where I observe the light is on. This observation leads me to conclude that either the light was already on, or that the effect of flicking the switch is to turn the light on. This conclusion can only be expressed as the disjunction of a *Initially* formula and a formula representing that effect to the action *flicking a switch* is initiating the *light is on* (clearly this formula must involve the predicate $Initiates$)[23].

However, if this *separation* is not done, the considered Event Calculus with Circumscription does not solve the Hanks-McDermott problem and, thus, the persistence problem arises [44].

In spite of these objections and the results about the complexity of Circumscription (section 3.2), the simplicity of the Event Calculus axiomatisation, the easy way to extend it in order to cope with issues like continuous change and the elegance of the *circumscriptive* solution to the persistence problem make us conclude that this framework is a very good attempt for bridging the gap between formal work in knowledge representation and practical robotics.

In the following section we show how Circumscription interacts with Event Calculus using the Yale Shooting Problem (or Hanks-McDermott problem) as an example. The proposition 6 states that the Hanks-McDermott problem arises if circumscribing the whole theory; on the other hand, *separating* the circumscriptions solves this problem (as shown in proposition 7)[24].

4.3 Example: The Hanks-McDermott problem

This scenario comprises a gun and a target; a deterministic situation is supposed (i.e., there is no possibility of missing the target). The gun is loaded at time 10 and shot at time 20; and we consider another action at time 15 that does not change the status of any fluent, it should not interfere in the determinism of our reasoning process.

[23] I quote here Shanahan's comment on this limitation of his solution to the frame problem: "Those who are unimpressed by these objections to forced separation can exploit the idea to produce other elegant solutions to the frame problem".

[24] The proofs of these propositions were taken from [44].

Although we supposed a deterministic situation, some of the non-monotonic formalisms conclude that, after the shot, the target is still "alive" [25]. More precisely, the minimal model given by some formalisms in order to cope with scenarios like this makes us conclude that the **Wait** action somehow unloads the gun! Let's see how Event Calculus with Circumscription solves this problem.

Assume that the domain comprises three kinds of events: *Load*, *Wait* and *Shoot*; and two fluents: *Alive* and *Loaded*. The action description is represented by the formulae:

$$Initiates(Load, Loaded, t) \quad \textbf{(CEY 1)}$$

$$Terminates(Shoot, Alive, t) \leftarrow HoldsAt(Loaded, t) \quad \textbf{(CEY 2)}$$

$$Happens(Load, 10) \quad \textbf{(CEY 3)}$$

$$Happens(Wait, 15) \quad \textbf{(CEY 4)}$$

$$Happens(Shoot, 20) \quad \textbf{(CEY 5)}$$

$$Initially(Alive) \quad \textbf{(CEY 6)}$$

$$UNA[Load, Wait, Shoot] \quad \textbf{(CEY 7)}$$

$$UNA[Alive, Loaded]. \quad \textbf{(CEY 8)}$$

The scenario is represented by the narrative formulae (CEY 3) to (CEY 5); the initial situation by (CEY 6); and the formulae (CEY 7) and (CEY 8) are uniqueness-of-names axioms.

The next proposition shows that the Hanks-McDermott problem arises circumscribing the whole theory.

Proposition 6 *Let the theory Σ be the conjunction of (EC1) to (EC5) with (CEY 1) to (CEY 8); then we can prove that*[26]*:*

$$CIRC[\Sigma; Happens, Initiates, Terminates, Initially; HoldsAt] \\ \not\models \neg HoldsAt(Alive, 25).$$

Proof: Consider a model $\mathcal{M}$ of Σ that meets the following conditions:

- (i) $\mathcal{M} \models Happens(a, t) \rightarrow [a = Load \wedge t = 10] \vee [a = Wait \wedge t = 15] \vee [a = Shoot \wedge t = 20]$
- (ii) $\mathcal{M} \models Initiates(Load, Loaded, 10)$

[25] For a more detailed treatment of the Hanks-McDermott problem we suggest [44].

[26] As, in the scenario of the Yale Shooting problem, the *Releases* predicate is not used we omitted it it the propositions 6 and 7.

We want $\mathcal{M}$ to be a minimal model for Σ such that:

$$\mathcal{M} \models HoldsAt(Alive, 25)$$

In order to get this fact we can use (EC1); if we have

$$\mathcal{M} \models Initially(Alive) \wedge \neg Clipped(0, Alive, 25)$$

then we get:

$$\mathcal{M} \models HoldsAt(Alive, 25)$$

But, by (CEY6), we know that

$$\mathcal{M} \models Initially(Alive);$$

so we need to build up $\mathcal{M}$ so that $\mathcal{M} \not\models Clipped(0, Alive, 25)$.

However, we know that $\mathcal{M} \not\models Clipped(0, Alive, 25)$ iff

$$\mathcal{M} \models \forall a\, t(\neg Happens(a, t) \vee \neg Terminates(a, Alive, t) \vee \neg(0 < t) \vee \neg(t < 25)).$$

Because of (i) we have just to check when $M \models Happens(a, t)$; then we have three possible cases:

1. T) $a = Load$ and $t = 10$. In this case we need that

$$\mathcal{M} \not\models Terminates(Load, Alive, 10);$$

2. T) $a = Wait$ and $t = 15$. In this case we need that

$$\mathcal{M} \not\models Terminates(Wait, Alive, 15);$$

3. T) $a = Shoot$ and $t = 20$. In this case we need that

$$\mathcal{M} \not\models Terminates(Shoot, Alive, 20);$$

From 3.T) and (CEY2) we need

$$\mathcal{M} \models \neg HoldsAt(Loaded, 20) \tag{11}$$

In order to minimise $Initiates$ in $\mathcal{M}$ assume (considering (ii)):

$$\mathcal{M} \models \forall a\ t\ f\ [Initiates(a, f, t) \rightarrow (a = Load \wedge f = Loaded \wedge t = 10)]. \tag{12}$$

Thus, in order to verify that still $\mathcal{M} \models (EC2)$ we need only to check the cases where the antecedent of (EC2) is true in $\mathcal{M}$. From 12 we have to consider the case $Initiates(a, f, t_1)$, when $a = Load$, $f = Loaded$ and $t_1 = 10$; then, from (i) we have

$$\mathcal{M} \models Happens(Load, 10).$$

Let's take into consideration $Clipped(10, Loaded, 25)$.

From the definition of $Clipped$ (EC4) we have

$$Clipped(10, Loaded, 25) \leftrightarrow \exists at[Happens(a,t) \wedge Terminates(a, Loaded, t) \wedge (10 < t) \wedge (t < 25)].$$

We ask ourselves when $\mathcal{M} \models \neg Clipped(10, Loaded, 25)$ that is when

$$\mathcal{M} \models \forall at(\neg Happens(a,t) \vee \neg Terminates(a, Loaded, t) \vee \neg(10 < t < 25)).$$

So, we need to know when, $\forall at$

$$\mathcal{M} \models \neg Happens(a,t) \vee \neg Terminates(a, Alive, t) \vee \neg(10 < t < 25). \qquad (13)$$

If $\neg(10 < t < 25)$, then (13) trivially holds.

Therefore let's analyse the case when $10 < t < 25$. Under this assumption, we have that (13) holds for every a iff

$$\mathcal{M} \models \neg Happens(a,t) \vee \neg Terminates(a, Alive, t); \qquad (14)$$

from (i) and the assumption $10 < t < 25$ we just need to take care of the cases when $(a = Wait \ \ and \ \ t = 15)$ or $(a = Shoot \ \ and \ \ t = 20)$, but we have already considered these cases in 2.T) and 3.T). Consequently, we have that the antecedent of (EC2) is false, so $\mathcal{M} \models (EC2)$.

Consider the axiom (EC1).

We minimise $Initially$ so that

$$\mathcal{M} \models \forall f(Initially(f) \rightarrow f = Alive). \qquad (15)$$

Then, we just need to check that, under this minimisation, $\mathcal{M} \models (EC1)$. However, since $HoldsAt$ is defined as a varying predicate in the circumscription policy in question (proposition 6), when $t \neq 25$ we have nothing to consider.

In conclusion, the structure for which (i), (ii), 1.T), 2.T), 3.T), (11), (12) and (15) hold is a model of Σ, is minimal w.r.t. the predicates $Happens$, $Initiates$, $Terminate$ and $Initially$, and

$$\mathcal{M} \models HoldsAt(Alive, 25)$$

$\square$

The next proposition shows how the Yale Shooting scenario can be solved separating the circumscriptions.

Proposition 7 *Let Δ be the conjunction of (CEY 3) to (CEY 6), Σ the conjunction of (CEY1) and (CEY2), Ω the conjunction of (CEY7) and (CEY8). And let Φ be the conjunction of (EC1) to (EC5). Then we can prove that:*

$$\begin{aligned}&CIRC[\Delta; Happens] \wedge CIRC[\Sigma; Initiates, Terminates, Releases] \wedge\\ &\wedge \Omega \wedge \Phi \models \neg HoldsAt(Alive, 25).\end{aligned}$$

Proof: Using the fact that any ground atom $P(t_1, \ldots, t_n)$ can be written as $\forall x(F(x) \rightarrow P(x))$, where $F(x)$ is $x_1 = t_1 \wedge \ldots \wedge x_n = t_n$, we can apply the rule of predicate completion (proposition 2) on

$$CIRC[\Delta; Happens]$$

to obtain that

$$Happens(a,t) \leftrightarrow [a = Load \wedge t = 10] \vee [a = Wait \wedge t = 15] \vee [a = Shoot \wedge t = 20]. \quad (16)$$

Applying theorem 1 and the predicate completion rule (similarly as done above) to

$$CIRC[\Sigma; Initiates, Terminates, Releases]$$

gives

$$Initiates(a, f, t) \leftrightarrow [a = Load \wedge f = Loaded \wedge t = 10] \quad (17)$$

$$Terminates(a, f, t) \leftrightarrow [a = Shoot \wedge f = Alive \wedge HoldsAt(Loaded, t) \wedge t = 20] \quad (18)$$

$$\neg \exists a, f, t[Releases(a, f, t)]. \quad (19)$$

From definition (EC4) and formulae (16), (18) and (19) we get $\neg Clipped(10, Loaded, 20)$. Using this fact together with (CEY 1), (CEY 5), and axioms (EC1) and (EC2) we have

$$HoldsAt(Loaded, 20).$$

From definition (EC5) and formula (16) we get $\neg Declipped(20, Alive, 25)$. Given this, (EC3), formulae (16), (18) and $HoldsAt(Loaded, 20)$ we conclude

$$\neg HoldsAt(Alive, 25).$$

$\square$

A more detailed proof involving Circumscription and Event Calculus is going to be given when we talk about continuous change (in the next section).

We saw in the previous example that the Circumscriptive Event Calculus solves the Hanks-McDermott problem. But does it solve the Frame Problems (or even the persistence problem)? We cannot answer it affirmatively since, up to now, there is no algorithm capable of choosing the theories Σ and Δ in order to give correct answers about the ordinary world; this fact recalls the impossibility of computing the background knowledge, as briefly discussed in the introduction to this work.

A complete discussion of the Frame Problems, however, is orthogonal to the purpose of this paper. Let's now turn our attention to the formalisation of continuous change.

4.4 Continuous Change

Continuously varying properties are commonly found in every day life. Examples of these are the level of the water when filling a vessel, the velocity of an accelerating car, the dislocation of a moving object, the height of a falling object, etc. Together with such *continuous change* problem we have to take into account the issue of *triggered events*, i.e., events that are started when a property that varies continuously (*continuous property*) achieves a determined value, for instance when the temperature achieves zero degrees the water solidifies or (the classical example) when the level of water in a vessel achieves the vessel's rim, the level stops rising and the water spills over. In order to give a formal account of common sense knowledge we have to formalise such issues.

In the present work we are going to describe the formalisation of continuous change as proposed in [36][44][39].

In order to formalise continuous change in Event Calculus we extend the calculus in the following ways:

- consider a sort for reals;
- represent time in the real line;
- introduce the predicate $Trajectory(f_1, t_1, f_2, d)$, which states that, if the fluent f_1 is initiated at time t, then the fluent f_2 (which represents the continuously variating property) holds at time $t + d$;
- introduce one more axiom :

$$\begin{aligned} &HoldsAt(f_2, t_2) \leftarrow && (EC6)\\ &\quad Happens(a, t_1) \wedge Initiates(a, f_1, t_1) \wedge t_1 < t_2 \wedge t_2 = t_1 + d \wedge \\ &\quad Trajectory(f_1, t_1, f_2, d) \wedge \neg Clipped(t_1, f_1, t_2) \end{aligned}$$

  which states that there exists event a initiating the fluent f_1 then the fluent f_2 (the continuous property) will hold at time t_2 ($t_2 = t_1 + d$); in this formalisation, the fluent (f_1) has to be chosen such that it holds in the time interval $[t_1, t_2]$, since $\neg Clipped(t_1, f_1, t_2)$ must be true.

Consequently, a domain description must include a collection of trajectory clauses (the conjunction of which we denote by Λ). In the circumscription policy, these trajectory formulae are included outside the scope of the circumscriptions, as we are going to see below (in proposition 8 and formula (34)).

Triggered events can be incorporated in Event Calculus including, in the domain description, formulae of the form:

$$Happens(\alpha, t) \leftarrow HoldsAt(\beta_1, t) \wedge HoldsAt(\beta_2, t),$$

where (at time t) β_1 and β_2 are fluents. Such formulae are considered in the circumscription minimising *Happens* (in proposition 8 and formula (34)).

We have to point out here that *continuity* is a relative concept in this framework; for instance, the velocity of an accelerating car is a *continuous* fluent if the acceleration is different from zero, otherwise the velocity can be represented by a discrete quantity. Actually, talking about *continuous* fluents, we are dealing with *continuously varying* properties; if such properties stop varying they can be taken as discrete.

Let's exemplify continuous change and triggered events presenting the *water spilling vessel scenario* [44]. This is the scenario where a vessel of depth 10 units of distance (u.d.) is being filled by a tap (continuously) with water at rate one unit of distance per unit of time (u.d./u.t.). In the initial state (at the starting of the time) the vessel is empty (i.e. the level of the water is null); the vessel remains in this state until the tap is turned on at time 5 u.t.. We want to prove the common sense fact that (considering the rate already mentioned, and that the tap is not turned off in any instant) the vessel's level does not increase after the water reaches the vessel's rim.

Consider the events $TapOn$ and $TapOff$ as having their obvious meaning. The fluent $Filling$ represents the property that the vessel is being filled; $Height$ is a function symbol ($Height$: *real* $\rightarrow$ *fluent-term*), such that $Height(x)$ is a fluent term denoting that the water inside of the vessel is at the level x; *Overflow* is the event that is triggered when the level of the water reaches the value 10 u.d. (i.e. the vessel's rim), at this point the water starts to spill over (this property is represented by the fluent *Spilling*). This problem can be formalised as follows[27] [28]:

$$
\begin{array}{ll}
Initiates(TapOn, Filling, t) & (SkF1) \\
Terminates(TapOff, Filling, t) & (SkF2) \\
Releases(TapOn, Height(x), t) & (SkF3) \\
Initiates(TapOff, Height(x), t) \leftarrow HoldsAt(Height(x), t) & (SkF4) \\
Terminates(Overflow, Filling, t) & (SkF5) \\
Initiates(Overflow, Height(10), t) & (SkF6) \\
Initiates(Overflow, Spilling, t) & (SkF7) \\
Trajectory(Filling, t, Height(x_2), d) \leftarrow & (SkF8) \\
\quad HoldsAt(Height(x_1), t) \wedge x_2 = x_1 + d & \\
x_1 = x_2 \leftarrow HoldsAt(Height(x_1), t) \wedge HoldsAt(Height(x_2), t) & (SkF9) \\
Happens(Overflow, t) \leftarrow HoldsAt(Height(10), t) \wedge & (SkF10) \\
\quad HoldsAt(Filling, t) & \\
Initially(Height(0)) & (SkF11) \\
Happens(TapOn, 5) & (SkF12)
\end{array}
$$

[27] In this example the predicate $Releases$ is used to represent the transition from a discretely varying property to a continuously varying property; in the present case the fluent $Height(x)$, after the tap is turned on, is not subject to the common sense law of inertia but it starts varying continuously. If $Terminates$ were used to represent this fact, it would mean that the fluent stopped holding (and not that it started varying continuously).

[28] The three axioms denoted by (SkF14) are uniqueness-of-names axioms; we decided to write them explicitly in order to make clear the form of the UNA w.r.t. a function (in the case $Height(x)$).

$$
\begin{array}{ll}
UNA[TapOn, TapOff, Overflow] & (SkF13) \\
Filling \neq Spilling & (SkF14) \\
\forall x Height(x) \neq Filling & (SkF14) \\
\forall x Height(x) \neq Spilling & (SkF14)
\end{array}
$$

With the formalisation of the spilling vessel scenario presented above, we can prove that at time 20 u.t. the water level in the vessel is 10 u.d. (i.e. an overflow occurred):

Proposition 8 *Let Φ be the conjunction of (EC1) to (EC6); Δ be the conjunction of (SkF10) to (SkF12); Σ be the conjunction of (SkF1) to (SkF7); Ψ be (SkF9); Ω be the conjunction of (SkF13) and (SkF14), and Π be (SkF8). Then,*

$$
\begin{array}{l}
CIRC[\Delta; Happens] \wedge CIRC[\Sigma; Initiates, Terminates, Releases] \wedge \\
\quad \wedge \Psi \wedge \Pi \wedge \Phi \wedge \Omega \models HoldsAt(Height(10), 20).
\end{array}
$$

Proof:

Consider the first conjunct, $CIRC[\Delta; Happens]$. We can rewrite the formulae Δ in the following way:

$$
\begin{array}{l}
\forall x, t\{Happens(x,t) \leftarrow x = Overflow \wedge HoldsAt(Height(10), t) \wedge HoldsAt(Filling, t)\} \\
\forall x\{Initially(x) \leftarrow x = Height(0)\} \\
\forall x, t\{Happens(x,t) \leftarrow x = TapOn \wedge t = 5\}.
\end{array}
$$

We can also write the rules which have the predicate $Happens$ in the head as the rule:

$$
\begin{array}{r}
\forall x, t\{Happens(x,t) \leftarrow (x = TapOn \wedge t = 5) \vee \\
(x = Overflow \wedge HoldsAt(Height(10), t)) \wedge HoldsAt(Filling, t)\}.
\end{array} \tag{20}
$$

Then, applying proposition 2 in the rule (20) we get:

$$
\begin{array}{r}
\forall x, t\{Happens(x,t) \leftrightarrow (x = TapOn \wedge t = 5) \vee (x = Overflow \\
\wedge HoldsAt(Height(10), t) \wedge HoldsAt(Filling, t))\}.
\end{array} \tag{21}
$$

As every formula is universally quantified we are going to omit the universal quantifier in the rest of this proof.

Considering the second conjunctive term of the formula in proposition 8,

$$CIRC[\Sigma; Initiates, Terminates, Releases],$$

we note that all occurrences of the predicates $Initiates, Terminates, Releases$ in Σ are positive, so that we can apply theorem 1 to get

$$CIRC[\Sigma; Initiates] \wedge CIRC[\Sigma; Terminates] \wedge CIRC[\Sigma; Releases].$$

By applying proposition 2 to each conjunctive term in this formula (in the same way as presented above for the *Happens* predicate) we get the completion of the predicates *Initiates*, *Terminates*, *Releases* as follows:

$$\begin{aligned} Initiates(x_1, x_2, t) \leftrightarrow (x_1 = TapOn \wedge x_2 = Filling) \vee \\ (x_1 = TapOff \wedge x_2 = Height(x) \wedge HoldsAt(Height(x), t)) \vee \\ (x_1 = Overflow \wedge [x_2 = Height(10) \vee x_2 = Spilling]). \end{aligned} \tag{22}$$

$$\begin{aligned} Terminates(x_1, x_2, t) \leftrightarrow (x_1 = TapOff \wedge x_2 = Filling) \vee \\ (x_1 = Overflow \wedge x_2 = Filling) \end{aligned} \tag{23}$$

$$Releases(x_1, x_2, t) \leftrightarrow x_1 = TapOn \wedge x_2 = Height(x) \tag{24}$$

According to (21) the first event happens at time 5 u.t.. Thus, we conclude that

$$\neg \exists a, t \; Happens(a, t) \wedge 0 < t \wedge t < 5.$$

From axiom (SkF11), definition (EC4), formulae (20) (23) and (24) we conclude that

$$\neg Clipped(0, Height(0), 5).$$

from this, axioms (SkF11) and (EC1), we get

$$HoldsAt(Height(0), 5).$$

Using this fact and the axiom (SkF8) we have

$$Trajectory(Filling, 5, Height(d), d).$$

Given this, (SkF1) and (SkF12), the axiom (EC6) gives

$$\begin{aligned} & HoldsAt(Height(d), t_2) \leftarrow \\ & \quad Happens(TapOn, 5) \wedge Initiates(TapOn, Filling, 5) \wedge \\ & \quad 5 < t_2 \wedge t_2 = 5 + d \wedge \\ & \quad Trajectory(Filling, 5, Height(d), d) \wedge \neg Clipped(5, Filling, t_2), \end{aligned}$$

This implies

$$HoldsAt(Height(t_2 - 5), t_2) \leftarrow 5 < t_2 \wedge \neg Clipped(5, Filling, t_2). \tag{25}$$

From formula (21) we extract that:

$$HoldsAt(Height(10), t) \wedge HoldsAt(Filling, t) \leftarrow Happens(Overflow, t) \wedge t > 5. \tag{26}$$

Considering that $Filling$ must hold at the time of an $Overflow$ (cf. expression (26)) and that (according to axiom (SkF5)) $Overflow$ events terminate $Filling$, we conclude that between any two $Overflow$ events there must be a $TapOn$ event, since only such events initiate $Filling$. However, we know that the only $TapOn$ event occurs at time 5 u.d. (cf. formula (21)). Then we can say that there can be at most one time point such that an $Overflow$ event happens in the time interval [5,15]. In other words, there can be only one t such that

$$Happens(Overflow, t) \wedge 5 < t < 15.$$

Now we are going to prove, reasoning by contraposition, that $\neg Clipped(5, Filling, 15)$. Suppose that $Clipped(5, Filling, 15)$ holds; then, from (EC4) and 20 we get that there is a time τ, $5 < \tau < 15$ such that

$$Happens(Overflow, \tau) \wedge 5 < \tau < 15.$$

Since, as mentioned in the previous paragraph, at most one such a time can exist we conclude

$$\neg Clipped(5, Filling, \tau).$$

Thus, from (25) we have $HoldsAt(Height(\tau - 5), \tau)$. Since $\tau < 15$, this implies that the height of water at τ is less than 10 u.d.. However, the formula (26) states that

$$HoldsAt(Height(10), \tau)$$

, which is a contradiction. Therefore,

$$\neg Clipped(5, Filling, 15).$$

Given $\neg Clipped(5, Filling, 15)$, the formula (25) gives

$$HoldsAt(Height(10), 15). \tag{27}$$

From this fact and (SkF6) we conclude

$$Initiates(Overflow, Height(10), 15). \tag{28}$$

Since no further $TapOn$ event occur, and $TapOn$ is the only event that can terminate or release the $Height$ fluent, we conclude

$$\neg Clipped(15, Height(x), 20). \tag{29}$$

Finally, applying axiom (EC2) with the formulae (26)(28)(29) we derive

$$HoldsAt(Height(10), 20).$$

$\square$

Let's turn our attention now to the goal of the present paper which is to describe and discuss Shanahan's work on the "Common Sense Informatic Situation for a Mobile Robot"[39].

5 Common Sense in a Mobile Robot

The goal of this section is to describe the logical account for perception and action of an autonomous agent according to the framework proposed in [39][41][40]. Roughly, we are going to discuss how to represent knowledge about the objects in a robot's environment, how such knowledge is acquired and how to reason about it.

Talking about a logical formalism applied to an autonomous agent we are recalling McCarthy's definition of a *deliberative agent* (or *sentential processing automaton*) [49]. A deliberative agent is an agent whose design is based on the manipulation of logic sentences.

The purpose of a logic-based approach to an autonomous agent is three fold, according to Shanahan [42]: first, having an agent designed logically we can supply a rigorous mathematical account of the success of its actions. On the other hand, because its knowledge and goals are expressed in a universal declarative language, such an agent can easily be modified and maintained. The third point refers to the facility of incorporating high-level cognitive skills in a logic based agent (such as planning and concurrency).

Let's briefly introduce this framework.

Perceiving Logically

The task of sensor data assimilation is done by hypothesizing the existence of objects with locations and shapes which would explain the data. This process is the product of the following three steps:

- Firstly, the design of a logic-based formalism for reasoning about actions and space;
- With this formalism, construct a theory which captures the agent's relationship to the world;
- Finally, we have to consider the task of sensor data assimilation as a form of Abduction with respect to this theory.

In order to reason about actions and change in a robot agent, the formalism must deal with several issues such as the formalisation of *continuous change*. The formalism must also cope with *concurrent events* since changes in the world can be simultaneous; last but not least, it has to handle actions and events with non-deterministic effects. The Event Calculus presented in section 4 is the chosen logic-based formalism to reason about actions and change; as can be seen in [44], it can be easily extended to cope with the issues just mentioned.

A formal language to reason about space also has to be developed, as we are going to see below. In this language, *space* is represented as the plane $\mathcal{R}^2$, while the objects in the space (such as the robot and obstacles) are path-connected subsets of $\mathcal{R}^2$. The axioms about space, actions and change (including continuous change) are components of a theory named Σ_B. The effects of the robot's motor activity on the world and the consequent effect of the world on the robot's sensors are described in a theory called Σ_E. The axioms of the Event Calculus presented in sections 2.2 and 4 must be included in the theories Σ_B and Σ_E.

Let Σ_B and Σ_E be background theories as described above. And let Δ_N be a logical description of the movements of objects, including the robot itself.

If a stream of sensor data is represented by Ψ, the task is to find an explanation of Ψ in the form of a logical description Δ_M of the initial locations and shapes of a number of objects, such that:

$$\Sigma_B \wedge \Sigma_E \wedge \Delta_N \wedge \Delta_M \models \Psi.$$

The symbol $\models$ stated in the formula above refers to second order consequence, since the theories Σ_B and Σ_E comprises circumscriptions as we are going to discuss below.

Acting Logically

Together with the assimilation of the sensor data, we should have a machine to create sequences of actions based on the sensor information. In the present work this resembles planning in Event Calculus [14] [43]. Roughly, the task is to find a sequence of robot actions Δ_N such that given a goal Γ

$$\Sigma_B \wedge \Sigma_E \wedge \Delta_N \wedge \Delta_M \models \Gamma,$$

where Σ_B, Σ_E and Δ_M are defined above. Thus, the symbol $\models$ refers to second order inference as already discussed.

In the rest of this section we are going to make these notions, briefly presented above, more precise.

5.1 Explanation as Abduction

We saw above that the tasks of sensor data assimilation and action planning are taken to be explanation processes of, respectively, a stream of sensor data (Ψ) and a goal (Γ). These explanations are done, in the framework in question, by *Abduction*.

Abduction is the process of explaining a sentence Γ by finding a set of formulae Δ such that, given a background theory Σ, Γ is a logical consequence of $\Sigma \cup \Delta$. In order to avoid trivial explanations, a set of predicates is distinguished such that every acceptable explanation must contain only these predicates (*abducible predicates*). The choice of abducible predicates is called *abduction policy*.

An additional condition to Abduction (hardly made explicit in the literature) is that, given a theory Σ and a formula Γ to be explained,

$$\Sigma \not\models \Gamma \quad and \quad \Sigma \not\models \neg\Gamma.$$

I.e., in order to explain a fact by Abduction, it is necessary to guarantee that neither Γ follows directly from the theory Σ nor its negation. In other words, we have to guarantee (as pre-condition to the abduction process) that the set of all explanations Δ is non empty, and non trivial.

We cannot forget that a circumscription policy has also to be taken into account in the abduction process in order to overcome the persistence problem.

Considering all these points, Abduction in Circumscriptive Event Calculus can be defined as follows [44] [29]:

Let a *state description*[30] be defined as a conjunction of predicates $HoldsAt(f,t)$ or $\neg HoldsAt(f,t)$; where f is a fluent and t is a time point[31]

Let Δ_1 be a conjunction of formulae not including the predicates $Initiates$, $Terminates$ or $Releases$; Σ be a conjunction of formulae not including the predicates $Happens$ or $Initially$; and Φ a conjunction of axioms (EC1) to (EC5), domain constraints and uniqueness-of-names axioms. Then,

Definition 4 *A formula Δ_2 is an explanation of the state description Γ in the Circumscriptive Event Calculus if*

- $CIRC[\Delta_1 \wedge \Delta_2; Happens]$ *is consistent;*
- Δ_2 *mentions only the predicates* $Happens$ *and* $Initially$ *(abducible predicates); and*
- $CIRC[\Delta_1 \wedge \Delta_2; Happens] \wedge CIRC[\Sigma; Initiates, Terminates, Releases] \wedge \Phi \models \Gamma$.

Clearly, there can be many abduction explanations Δ_2; in such a case the solution to the abduction process is taken to be the disjunction of all minimal explanations obtained (as stated in definition 6 below). A simple minimality condition is presented in the next definition:

Definition 5 *A formula Δ is a* minimal *explanation of Γ if*

- Δ *is an explanation of* Γ*; and*
- *There is no explanation Δ' of Γ such that $\Delta \models \Delta'$ and $\Delta' \not\models \Delta$*

The definition 5 is just an example of a preference relation. It is possible to define other minimality criteria in order to choose the *best* explanation [1]; however, a description of them is outside the scope of this work.

The operator ASM_{EC}, below, defines the result of assimilating a state description Γ into a background theory Σ in Circumscriptive Event Calculus language:

[29] In order to simplify this presentation, we are going to describe abduction in Event Calculus in a general way, i.e., without specifying abduction for sensor data assimilation and action planning.

[30] Shanahan uses the term *observation sentence* instead of state description.

[31] In [44] there is no definition for state description in Event Calculus (although Abduction in this calculus is presented), it is only defined for Situation Calculus (where the fluents hold on states not on time points). Defining state description in Event Calculus using $HoldsAt$ formulae excludes explanations of *miracles* (or unknown actions) since we are not able to create an explanation of a $HoldsAt$ formula if the time point is not known.

Definition 6 *Suppose that there is a non empty set E of all explanations that satisfy the minimality criterion stated in definition 5*[32]*. Then the operator $ASM_{EC}[\Gamma;\Phi;\Delta_1;\Sigma]$ is defined as,*

1. *if Γ is a state description,*

$$CIRC[\Delta_1 \wedge \bigvee_{\Delta\in E} \Delta; Happens] \wedge CIRC[\Sigma; Initiates, Terminates, Releases] \wedge \Phi, \quad (30)$$

   where E is the set of all explanations (Δ_2) given by definition 4;

2. *if Γ is a Initiates, Terminates or Releases formula,*

$$CIRC[\Delta_1; Happens] \wedge CIRC[\Sigma \wedge \Gamma; Initiates, Terminates, Releases] \wedge \Phi; \quad (31)$$

3. *if Γ is a Happens formula,*

$$CIRC[\Delta_1 \wedge \Gamma; Happens] \wedge CIRC[\Sigma; Initiates, Terminates, Releases] \wedge \Phi; \quad (32)$$

4. *otherwise,*

$$CIRC[\Delta_1; Happens] \wedge CIRC[\Sigma; Initiates, Terminates, Releases] \wedge \Phi \wedge \Gamma. \quad (33)$$

It is easy to justify the choice for the operator just defined. As already stated, the interest of the abduction process is to construct explanations of state descriptions, rather than just any formula in Event Calculus. As a consequence of this fact, in the definition of the operator ASM_{EC} the explanations of the formula Γ are only considered in formula (30), which is concerned with assimilating a state description (in the other cases there is no abduction process). Therefore, as we are dealing with the construction of action sequences, the explanation $\bigvee_{\Delta\in E}\Delta$ has to be given only in terms of the abducibles *Initially* and *Happens* (intuitively, the pre-conditions to the actions and the actions themselves, respectively). In order to avoid the frame problem, the extensions of these formulae have to be minimised according to the separation assumption (section 4), that is why the explanations $\bigvee_{\Delta\in E}\Delta$ are included in the scope of the circumscription in the first conjunctive term of formula (30).

It is also a consequence of the separation assumption that formulae (31), (32) and (33) have the definitions above.

The previous description of Abduction is enough for the purpose of the present work. Up to the state of art of Abduction, there is no complete work about Abduction in circumscriptive theories; but several interesting issues can be found about Abduction in logic programming [17].

[32] We did not find in the literature any mention about the case of an empty E; this is an essential issue when dealing with sensor data assimilation, since we should expect possible sensor data such that the background theory comes up to be too weak to handle. If we allow an empty E in the definition 6 the disjunction $\bigvee_{\Delta\in E}$ (in formula (30)) is a contradiction which is a very non intuitive explanation (the consideration of a non empty set E in this definition is not done in [44]). According to Shanahan [34], if the robot faces a situation when E is empty "then there really is no explanation. The robot is stuck". The study of this problem, however, is left for future works.

A clear introduction about Abduction, in general, can be found in [1].

Before talking about the role of the theory about actions and change, space and shape in the common sense robotics, a brief note about the computational complexity of Abduction is worth mentioning.

5.1.1 A note on the complexity of Abduction

The results about the complexity of the logic-based abduction found in the literature are all related to the propositional case, since very little is known about the first-order case [8].

According to [13], there are three main decision problems in the context of logic-based abduction. They are:

1. to determine whether an explanation for the given data exist at all;

2. to determine whether an individual hypothesis is *relevant*; that is, whether it is part of at least one acceptable explanation;

3. to determine whether an individual hypothesis is *necessary*, that is whether it occurs in all acceptable explanations.

The complexity of these problems was presented in [3] for a particular case of Abduction (also in this reference, some tractable classes of abduction problems were defined). However, the complexity of logic-based abduction was only partially investigated.

The most complete work about logic-based abduction is presented in [13] where many complexity results were proved w.r.t. various minimality criteria and considering both general theories (i.e. where Σ is an arbitrary propositional theory) and the propositional Horn case.

In particular, for the general case (under the minimality criteria presented in definition 5), it was proved that the problem is complete for the second level in the polynomial hierarchy (Π_2^P-complete).

If the underlying theory is Horn, then it was proved that the complexity of the abduction problems is always lowered by one level of the polynomial hierarchy.

5.2 The Theory Σ_B: Reasoning about Actions and Change, Space and Shape

Up to this point we only considered one facet of common sense reasoning, the *common sense law of inertia* (section 2). However, the common sense used in the every day life is much more complex. In the present work we are not interested on a full logic of common sense knowledge, or on how we (human beings) reason, but on what the relevant points are for applying such a theory to a mobile robot. This is basically what the theory Σ_B is about.

A robot immersed in the real world has to deal not only with actions and change, but also with properties that change continuously (such as the robot's motion, for example) and objects' shape and location. Therefore the theory Σ_B has to cope with the issue of continuous change as well as to include axioms about actions and change, space and shape (an extension of the Event Calculus capable of coping with continuous change was already presented in section 4.4).

A theory of actions and change was already described in section 2.2. In order to formalise spatial reasoning, Circumscriptive Event Calculus has to be extended in the following way.

5.2.1 Reasoning about Spatial Occupancy

In this part of the paper we are going to describe the theory of space and shape as proposed in [38] and [39].

It is clear that any action, whose effect is changing an object's position, has as precondition that the object's destination must be unoccupied. We must be careful when using the expression "it is clear", since it is related to our common sense knowledge about the world. In order to give a logical account of such a reasoning we need to assume that the space is empty by *default*, and to consider object's locations as abnormalities that must be minimised in a theory about space occupancy. Again we are appealing to the idea of Circumscription.

Of course we can try a brute force description of which regions are not occupied. However, in doing so, several problems arise, for instance:

- A description has to be supplied for every state of the world;
- A description has to be supplied for every unoccupied place;
- It may not be the case that we have always a complete description of the objects and their locations in the world;
- It may happen that the agent has *wrong* information about the objects' description in a given state of the world. Thus, we have to admit that the information may be *revised* due to the acquisition of new data.

Considering these problems, the definition of a default rule about space occupancy is the best option at hand. This rule we name *common sense law of space occupancy*, and it will be presented after the description of a language to talk about space and shape.

A language for space and shape extends Event Calculus with the following sorts:

- A sort of regions $g_1, g_2, \ldots, g_n, g_{n+1}, \ldots$;
- A sort of points $p_1, p_2, \ldots, p_n, p_{n+1}, \ldots$;
- A sort of objects $ob_1, ob_2 \ldots, ob_n, ob_{n+1}, \ldots$.

As mentioned above, it is assumed that *space* is two dimensional (represented by the plane $\mathcal{R}^2$), while a *region* is a path-connected subset of $\mathcal{R}^2$ and a *point* is an element of $\mathcal{R}^2$. In this framework the $\in$ predicate has its intended meaning.

Shape is represented as a region. The regions are defined according to a fixed coordinate axes system.

We can define the following functions and predicates, useful to describe shapes of objects:

- a function that gives the *distance* between two points (*Distance:point term* $\times$ *point terms* $\rightarrow$ *reals*):

$$Distance(\langle x_1, y_1\rangle, \langle x_2, y_2\rangle) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (Sp1)$$

- a function *bearing* that gives the smallest angle between the line defined by the points (x_1, y_1) and (x_2, y_2), and the vertical line that passes through the point (x_i, y_i), where i is the index of $y_i = max(y_1, y_2)$ (*Bearing:point terms* $\times$ *point terms* $\rightarrow$ *reals*):

$$\begin{array}{l} Bearing(\langle x_1, y_1\rangle, \langle x_2, y_2\rangle) = r \leftarrow \qquad (Sp2) \\ \quad z = Distance(\langle x_1, y_1\rangle, \langle x_2, y_2\rangle) \wedge z \neq 0 \wedge \\ \quad Sin(r) = \frac{x_2 - x_1}{z} \wedge Cos(r) = \frac{y_2 - y_1}{z} \end{array}$$

- the predicate $Displace(g, \langle x, y\rangle)$ represents the result of displacing the region g by x unit east and y units north:

$$\langle w, z\rangle \in Displace(g, \langle x, y\rangle) \leftrightarrow \langle w - x, z - y\rangle \in g \quad (Sp3)$$

- the term $Line(p_1, p_2)$ denotes the straight line with end points p_1 and p_2[33]:

$$\begin{array}{l} p \in Line(p_1, p_2) \leftrightarrow \qquad (Sp4) \\ \quad Bearing(p_1, p) = Bearing(p_1, p_2) \wedge Distance(p_1, p) \leq Distance(p_1, p_2) \end{array}$$

- A function that gives the property representing that an object w *occupies* a region g; *Occupies*: *object term* $\times$ *region term* $\rightarrow$ *fluent term*.

  Clearly, *Occupies* has to obey two domain constraints stating the common sense facts that no object (w) occupies two regions (g_1 and g_2) at the same time (rule (Sp6)) and that no two different objects (w_1 and w_2) occupy the same region at the same time (rule (Sp7)).

$$\begin{array}{l} HoldsAt(Occupies(w, g_1), t) \wedge HoldsAt(Occupies(w, g_2), t) \rightarrow g_1 = g_2 \quad (Sp5) \\ HoldsAt(Occupies(w_1, g_1), t) \wedge \qquad (Sp6) \\ \quad HoldsAt(Occupies(w_2, g_2), t) \wedge w_1 \neq w_2 \rightarrow \\ \quad\quad \neg\exists p[p \in g_1 \wedge p \in g_2] \end{array}$$

With such a language we can formalise objects' shapes and locations expressing, as logical rules, geometrical descriptions. For instance we can formalise an open circle, of radius z units centred in the origin, in the following way:

$$p \in Disc(z) \leftrightarrow Distance(p, \langle 0, 0\rangle) < z \quad (Sp7)$$

[33]It is useful in defining shapes with straight line boundaries.

Finally, we need to define a rule guaranteeing that the space is empty by default, this rule, called the *common sense law of space occupancy*, (rule (Sp8)) can be written as follows:

$$AbSpace(w) \leftarrow Initially(Occupies(w, g)) \quad (Sp8)$$

The axiom (Sp8) states that the objects existent at time $t = 0$ are *abnormalities* in the space. Consequently, in order to guarantee that the only models of the theory are the ones whose extensions of the *AbSpace* consist only of the explicitly given object locations (i.e., space is empty by default, existent objects are exceptions), we have to circumscribe the theory minimising the extensions of the predicate *AbSpace*, allowing *Initially* to vary.

Thus, considering

- the conjunction O of axioms (Sp1) to (Sp8);
- a conjunction M of *Initially* formulae which mention only *Occupies*;
- a conjunction Δ of *Happens* formulae and *Initially* formulae which do not mention *Occupies*;
- EC^*, which is the conjunction of the axioms (EC1) to (EC5) and the axiom for continuous change (EC6); and
- conjunctions Σ, Ω and Ψ as described for the formula (10),

the circumscription policy, taking into account the common sense law of inertia and the common sense law of space occupancy, is the following[34]:

$$CIRC[O \wedge M; AbSpace; Initially] \wedge CIRC[\Sigma; Initiates, Terminates, Releases] \wedge \quad (34)$$
$$CIRC[\Delta; Happens] \wedge \Lambda \wedge EC^* \wedge \Omega \wedge \Psi.$$

The previous formula represents the theory Σ_B.

5.3 The Theory Σ_E: Reasoning with Sensors and Effectors

With the formalism developed so far, we are able to describe the formal theory about the relationship between robot's motors, sensor data and world: the theory Σ_E.

Σ_E includes the axioms for actions and change, space and shape (presented above), together with axioms relating object's shapes and movements to robot's sensor data.

When building the theory Σ_E we have to take into account a specific robot, since each different kind of robot sensors gives a different axiomatisation for Σ_E. In the present case, we

[34]Idem 22.

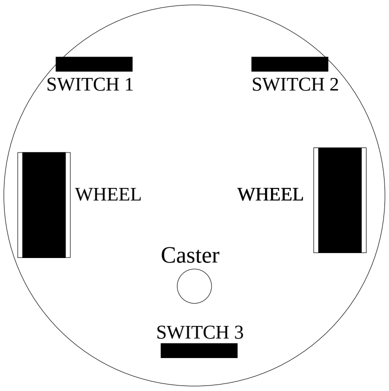

Figure 1: The rug warrior robot.

are considering a circular robot with two drive wheels and three bump sensors, two forward and one backward (this robot is called *The Rug Warrior* and is represented in figure 1). The Rug Warrior robot can perform three actions: move forward, stop, and rotate by a given number of degrees.

The theory Σ_E is composed of four parts as follows:

Part 1:

Consider a fixed Cartesian coordinate system.

Let $Switch_1$ and $Switch_2$ be fluent constants representing the activation of the robot's forward switches[35].

Assume that the robot travels at a velocity of one unit of distance per unit of time, and consider the following functions:

- A function *Rotate*: *real term* $\rightarrow$ *event term*; the term $Rotate(r)$ represents a robot's action of rotating r degrees about some axis passing through its body;
- A function $Facing$: *real term* $\rightarrow$ *fluent term*; the term $Facing(r)$ represents that the robot is turned to the r direction;
- A function *Blocked*: *object term* $\times$ *object term* $\times$ *real terms* $\rightarrow$ *fluent term*; the term $Blocked(w_1, w_2, r)$ represents that the object w_1 cannot move any distance at all in direction r without overlapping with object w_2;
- A function $Touching$: *object term* $\times$ *object term* $\times$ *point terms* $\rightarrow$ *fluent term*;

$$Touching(w_1, w_2, p)$$

  holds when objects w_1 and w_2 are touching at point p. In other words, this fluent holds if there are two points p_1 and p_2 such that the line segments defined by $\overline{p_1p_2}$ includes p, and for every point p_3 on the line segment $\overline{p_1p}$ ($\overline{pp_2}$), apart from p itself, falls into the region g_1 (g_2) occupied by the object w_1 (w_2).

[35]For simplicity we are not formalising the third sensor and the caster.

Then we have the following axioms[36]:

$$\begin{aligned}&Trajectory(Moving, t, Occupies(Robot, g_2), d) \leftarrow \\ &\quad HoldsAt(Occupies(Robot, g_1), t) \wedge HoldsAt(Facing(r), t) \wedge \\ &\quad g_2 = Displace(g_1, \langle d.sin(r), d.cos(r)\rangle)\end{aligned} \qquad (B1)$$

$$r_1 = r_2 \leftarrow HoldsAt(Facing(r_1), t) \wedge HoldsAt(Facing(r_2), t) \qquad (B2)$$

$$\begin{aligned}&HoldsAt(Blocked(w_1, w_2, r), t) \leftrightarrow \\ &\quad \exists g_1 g_2[HoldsAt(Occupies(w_1, g_1), t) \wedge HoldsAt(Occupies(w_2, g_2), t) \wedge \\ &\quad w_1 \neq w_2 \wedge \exists z_1[z_1 > 0 \wedge \forall z_2[z_1 \leq z_2 \rightarrow \\ &\quad \exists p[p \in g_2 \wedge p \in Displace(g_1, \langle z_2.sin(r), z_2.cos(r)\rangle)]]]]\end{aligned} \qquad (B3)$$

$$\begin{aligned}&HoldsAt(Touching(w_1, w_2, p), t) \leftrightarrow \\ &\quad \exists g_1, g_2[HoldsAt(Occupies(w_1, g_1), t) \wedge HoldsAt(Occupies(w_2, g_2), t) \wedge \\ &\quad w_1 \neq w_2 \wedge \exists p_1, p_2[p \in Line(p_1, p_2) \wedge p \neq p_1 \wedge p \neq p_2 \wedge \\ &\quad \forall p_3[[p_3 \in Line(p_1, p) \wedge p_3 \neq p] \rightarrow p_3 \in g_1] \wedge \\ &\quad \forall p_3[[p_3 \in Line(p, p_2) \wedge p_3 \neq p] \rightarrow p_3 \in g_2]]]\end{aligned} \qquad (B4)$$

$$UNA[Occupies, Facing, Moving, Blocked, Touching] \qquad (B5)$$

$$UNA[Rotate, Go, Stop, Bump, Switch_1, Switch_2] \qquad (B6)$$

We denote B the conjunction of EC^* with axioms (B1) to (B6) above; this collection of formulae is considered outside any circumscription.

Part 2:

We have to consider now a collection of *Initiates*, *Terminates* and *Releases* formulae. Considering the *Bump* symbol below as the collision of the robot with an object, we have:

$$Initiates(Rotate(r_1), Facing(r_1 + r_2), t) \leftarrow HoldsAt(Facing(r_2), t) \qquad (E1)$$

$$Releases(Rotate(r_1), Facing(r_2), t) \leftarrow HoldsAt(Facing(r_2), t) \wedge r_1 \neq 0 \qquad (E2)$$

$$Initiates(Go, Moving, t) \qquad (E3)$$

$$Releases(Go, Occupies(Robot, g), t) \qquad (E4)$$

[36] In order to make clear the form of the uniqueness-of-names axioms (B5) we write explicitly the UNA relating *Occupies* and *Blocked* as follows:

$$\forall w_1, w_2, r, w, g Occupies(w, g) \neq Blocked(w_1, w_2, r).$$

$$Terminates(a, Moving, t) \leftarrow a = Stop \vee a = Bump \vee a = Rotate(r) \quad (E5)$$

$$\begin{aligned} &Initiates(a, Occupies(Robot, g), t) \leftarrow & (E6)\\ &\leftarrow [a = Stop \vee a = Bump] \wedge HoldsAt(Occupies(Robot, g), t) \end{aligned}$$

We call Σ the conjunction of axioms (E1) to (E6) above.

Part 3:

Now we have to consider a collection of formulae concerning the narrative of actions and events (Δ). In the present case let Δ be composed of two parts, $N_1 \wedge N_2$.

The first part (N_1) refers to the triggered events and is composed of the conjunction of axioms (H1) to (H3) below.

$$\begin{aligned} &Happens(Bump, t) \leftarrow & (H1)\\ &\quad [HoldsAt(Moving, t) \vee Happens(Go, t)] \wedge \\ &\quad HoldsAt(Facing(r), t) \wedge HoldsAt(Blocked(Robot, w, r), t) \end{aligned}$$

$$\begin{aligned} &Happens(Switch_1, t) \leftarrow & (H2)\\ &\quad Happens(Bump, t) \wedge HoldsAt(Facing(r), t) \wedge \\ &\quad HoldsAt(Occupies(Robot, Displace(Disc(z), p_1)), t) \wedge \\ &\quad HoldsAt(Touching(Robot, w, p_2), t) \wedge \\ &\quad r - 90 \leq Bearing(p_1, p_2) < r + 12 \end{aligned}$$

$$\begin{aligned} &Happens(Switch_2, t) \leftarrow & (H3)\\ &\quad Happens(Bump, t) \wedge HoldsAt(Facing(r), t) \wedge \\ &\quad HoldsAt(Occupies(Robot, Displace(Disc(z), p_1)), t) \wedge \\ &\quad HoldsAt(Touching(Robot, w, p_2), t) \wedge \\ &\quad r - 12 \leq Bearing(p_1, p_2) < r + 90 \end{aligned}$$

According to [44] the *bump* event is somewhat redundant. It is kept in the axiomatisation because of the following reasons:

- A collision can occur without the attendant sensor event (for instance if one of the sensors is broken);
- A sensor event can occur without the occurrence of a collision.

The conjunctive term N_2 is a description of the robot's actions, i.e., it is a conjunction of *Happens* formulae.

Part 4:

The final component of Σ_E is related to the theory of space occupancy and to the map from the world to the robot's representation of it. More specifically, Σ_E is comprised of $O \wedge M$, where O is the conjunction of axioms (Sp1) to (Sp8) and M is composed of the conjunction $M_1 \wedge M_2$; such that, M_1 is a description of initial locations, shapes and orientations of known objects, including the robot itself; and M_2 is similar to M_1; however, when assimilating sensor data M_2 is supplied by adduction. In the present discussion M_2 plays the same role as Δ_2 in the introduction to abduction (section 5.1); however, M_2 has to obey a particular form, it is a conjunction of formulae [39]:

$$\exists g[Initially(Occupies(w, g)) \wedge \forall p[p \in g \leftrightarrow \Pi]], \tag{35}$$

where w is an object constant and Π is any formula in which p is free (intuitively, Π represents the description of region g).

The theory Σ_E is represented by the following sentence (which includes the common sense law of inertia and the common sense law of space occupancy):

$$\begin{aligned} &CIRC[O \wedge M_1 \wedge M_2; Abspace; Initially] \wedge CIRC[N_1 \wedge N_2; Happens] \wedge \\ &\quad CIRC[\Sigma; Initiates, Terminates, Releases] \wedge B \end{aligned}$$

Clearly, Π must be expressed with the predicates defined in the theory of space and shape, thus the choice of predicates for constructing such theories fixes the possible descriptions Π for regions g. Consequently Π cannot be defined as a precise description of g, as the biimplication in formula (35) suggests, but it has to be an approximation for the region. A solution for this problem is left for future works.

We finish this section with a short note on the computational complexity of the framework presented above.

5.4 A note on the complexity of the Logical Account of the Common Sense Informatic Situation for a Mobile Robot

Recalling the results presented in sections 3.2.2 and 5.1.1, the complexity of the "Logical Account of the Common Sense Informatic Situation for a Mobile Robot" seems to be very hard.

More precisely, we saw in section 3.2.2 that the complexity of the propositional circumscription is Π_2^P-complete, which was the same result obtained for the abduction of a general propositional circumscription (as described in section 5.1.1). Thus, the complexity of the framework presented above is at least Π_2^P-hard, since we are dealing with the circumscription of a first-order theory and operating abduction in a fragment of the second-order logic.

In order to have an exact idea about the computational complexity of the framework presented in this section, several issues must be taken into account:

- understand the complexity of circumscribing a first-order theory;
- understand Abduction in a second-order theory (and its complexity);

- determine how the use of circumscription in the background theory increases the complexity of Abduction.

Problems that, at the time of writing, are far from being solved.

6 Discussion and Open Issues

Will Lifschitz always be here to bail us out from the frame problem?

D. McDermott.

In this paper we presented and discussed the main concepts proposed by Shanahan [39] in order to construct a logical account for a mobile robot capable of bridging the gap between a rigorous logical description of a robotic system and its application in the real world [37].

In this framework, sensor data is assimilated through an abduction process on theories relating concepts of space, time, events and effects of events; an extension of the Event Calculus was assumed as a language for describing these concepts. Actually, the theories mentioned are fragments of the second-order logic since Circumscription was used to overcome the frame problem. In the present work, more emphasis was given to the background concepts than to the application in robotics, which was discussed only in the previous section.

In order to solve the Hanks-McDermott problem (also called the Yale shooting problem), which is an instance of the frame problem, Circumscription had to be applied in parts of theories (rather than in the theory as a whole). As a consequence of this, we had to consider a restriction on the expressivity of the Event Calculus, separation of circumscriptions, since any description of a scenario has to obey this *separation principle*. For instance, suppose that a circumscription of a theory Σ w.r.t. predicates ψ and ξ is given by $CIRC[\Psi;\psi] \wedge CIRC[\Xi;\xi]$, where $\Sigma = \Psi \wedge \Xi$; then, clearly, a formula composed by both ψ and ξ cannot be allowed in Σ because its position in the separation would not be defined. According to Shanahan [45] the justification for separation is that,

> by splitting the Circumscription into parts, we get a more manageable theory, one whose consequences are easier to work out. In fact, the role of Circumscription here [in the Event Calculus] is mainly to complete certain predicates, specially $Initiates$, $Terminates$ and $Happens$. If we circumscribe everything together, we have to do a lot more work to avoid difficulties like the Hanks-McDermott problem.

With the framework presented, the persistence problem is avoided, however we have no means to guarantee that future research will not come up with a new scenario from where this problem comes out to be unsolvable by the existent action theories.

With respect to the whole of the Frame Problems, during the last few years several interesting solutions were presented. We believe that a solution to the other problems related

[37] Due to the restricted time for writing this paper, several issues were not dealt here (such as uncertainty of the robot's trajectory and representation of the objects boundaries). A discussion of these issues can be found in [39].

to the frame problem (such as the qualification problem, the temporal projection problem, ramification problem) is near to be found. However, the creation of an algorithm capable to compute logical theories in order to solve problems in the real world (i.e. to compute the background knowledge and the necessary rules allowing an agent to act in the common sense world) is still a matter of fiction. Actually, the construction of such a machine is not the goal of the research on reasoning about actions and change nowadays; particularly, in robotics, the purpose of the development of action formalisms is the search for a powerful specification language for systems that must deal with common sense information in *specific* application domains.

This is also the answer for one of the critics to the research of reasoning about actions and change, which is the fact that there is no general theory as a basis for the research; on the other hand, there are many proposed formalisms and very few relationships between them. The same argument exposed in the previous paragraph can be repeated here, the purpose of the development of action formalisms is *the application*, if a particular set of axioms can be applied in a particular domain then it is a suitable formalism.

But... can it really be applied?

In the light of the complexity results presented through the text, we conclude that the framework presented is intractable. More specifically, we concluded that, the complexity of the "Logical Account of the Common Sense Informatic Situation for a Mobile Robot" is at least Π_2^P-complete. Shanahan comments this issue as follows [45]:

> My feeling is that complexity results should be used only as a guideline, and a disappointing complexity result rarely justifies the wholesale rejection of an approach. This is because the complexity results are worst-case, while in actual usage, a technique is often confined to a narrow, tractable sub-class of problems that no-one has pinned down yet.
>
> In fact the use of circumscription to overcome the frame problem here is a case in point. The form of the theories we're interested in guarantees that the circumscriptions always reduce to predicate completions, which can be handled efficiently by Prolog.
>
> Moreover, in the context of robotics, I think there's another argument that worst-case complexity results are misleading. No robot should be allowed unlimited computation time for any reasoning task before that task is suspended to sense the world and respond reactively to it. And ultimately, the world always moves on and renders old, unfinished reasoning tasks irrelevant. (...). A robot's designer needs to organise things so that most reasoning tasks the robot sets out to perform can be completed in a short time.(...).

The task of considering tractable sub-classes of abduction problems and classes of circumscription formulae that guarantees the reduction of the circumscriptive formulae to first-order formulae and, considering such classes, determining which (relevant) problems in robotics they represent are important open issues that the future research has to handle.

Another important open issue is the transformation of the logical account presented in section 5 to a logic programming system, in this case the logic becomes more intimately related to the implementation.

Acknowledgments

This thesis grew up under incessant readings, discussions and criticisms from my adviser Michiel van Lambalgen who, on top of all, had always the patience and respect hearing all of my ideas to solve the most diverse problems of the Universe. My most sincere gratitude goes also to Hans Rott for giving me the first lessons of non-monotonic reasoning and believing in my potential; to Peter van EmdeBoas for the interest and the good suggestions for future works; and to Murray Shanahan for always answering my e-mails (that were full of questions about his framework).

I would also like to thank Krzysztof Apt, Bert Bredeweg, Peter D. Grünwald, Michael Masuch and J. J. Meyer for the interest and availability to discuss my ideas.

Thanks to AAAI, in particular to Vladimir Lifschitz, for the grant that allowed me to participate in the Workshop of Non-monotonic Reasoning (part of the Conference of Knowledge Representation) realised in Trento, 1998.

Special thanks to Nikos Massios and Rosella Gennari who read several drafts of this thesis always with a very critical sense (and before the cinema time). Thanks also to Alessandra Palmigiano, Caterina Caracciolo, Carlos Areces, Eyal Hurvitz, Marc Pauly, Mehmet Altinkaya, Renata Wasserman and Rosalie Iemhoff for making happier my days here.

I am also in debt with Renata C. Favalli for all of the background work in my country (getting documents, sending books, calling my mother, etc) and, mainly, for helping me to solve problems that were far from any reasoning that I would be capable to do.

Many thanks to the ILLC staff, in particular to Dick de Jongh, Ingrid van Loon and Marco de Vries who took care of the course administration.

This work was partially supported by NUFFIC scholarship.

References

[1] Atocha Aliseda-Llera. *Seeking explanations: Abduction in Logic, Philosophy of Science and Artificial Intelligence.* PhD thesis, Institute for Logic Language and Computation - University of Amsterdam, 1997.

[2] J. F. Allen and G. Fergunson. Actions and events in interval temporal logic. In Olivero Stock, editor, *Spatial and temporal Reasoning*, pages 205–245. Kluwer Academic Publishers, 1997.

[3] Tom Bylander, Dean Allemang, Michael C. Tanner, and John R. Josephson. The computation complexity of abduction. *Artificial Intelligence*, pages 25–60, 1991.

[4] M. Cadoli. The complexity of model checking for circumscriptive formulae. *Information Processing Letters*, 44:113–118, 1992.

[5] M. Cadoli, F. M. Domini, and M. Schaerf. Is intractability of non monotonic reasoning a real drawback? In *Proceeding of the Twelfth National Conference on Artificial Intelligence*, pages 205–216. AAAI, 1994.

[6] M. Cadoli, T. Eiter, and G. Gottlob. An efficient method for eliminating varying predicates from a circumscription. *Artificial Intelligence Journal*, 54:397–410, 1992.

[7] M. Cadoli and M. Lenzerini. The complexity of propositional closed world reasoning and circumscription. *Journal of Computer Systems and Sciences*, 48:255–310, 1994.

[8] M. Cadoli and M. Schaerf. A survey of complexity results for non-monotonic logics. *Journal of Logic Programming*, 17:127–160, 1993.

[9] J. de Kleer and K. Konolige. Eliminating the fixed predicates from circumscription. *Artificial Intelligence Journal*, 39:391–398, 1989.

[10] Daniel C. Dennett. *Minds, Machines and Evolution: Philosophical Studies*, chapter Cognitive Wheels: The Frame Problem of A.I. Cambridge University Press, 1984.

[11] W.P. Dowling and J.H. Gallier. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *Journal of Logic Programming*, 1:267–284, 1984.

[12] T. Eiter and G. Gottlob. Propositional circumscription and extended closed world assumption are Π_2^P-complete. *Theoretical Computer Science*, 114:231–245, 1993.

[13] T. Eiter and G. Gottlob. The complexity of logic-based abduction. *Journal of the ACM*, 42(1):1278–1304, 1995.

[14] K. Eshghi. Abductive planning with event calculus. In *Proceedings of the Fifth International Conference on Logic Programming*, pages 562–579, 1988.

[15] R. Gennari. Temporal reasoning: A survey. Master's thesis, Institute for Logic Language and Computation, University of Amsterdam, 1998. Author's draft.

[16] G. Gottlob. Complexity results for nonmonotonic logics. *Journal of Logic and Computation*, 2:397–425, 1992.

[17] A.C. Kakas, R. Kowaslki, and F. Toni. The role of abduction in logic programming. In C. J. Hogger D.M.Gabbay and J.A. Robinson, editors, *Handbook of Logic in Artificial Intelligence and Logic Programming*, pages 235–324. Oxford University Press, 1998.

[18] H.A. Kautz and B. Selman. Hard problems for simple default logics. *Artificial Intelligence Journal*, 49:243–279, 1991.

[19] P. G. Kolaitis and C.H. Papadimitriou. Some computational aspects of circumscription. *Journal of the ACM*, 37:1–14, 1990.

[20] R. Kowalski. Database updates in event calculus. *Journal of Logic Programming*, 12(162):121–146, 1992.

[21] R.A. Kowalski and M.J. Sergot. A logic-based calculus of events. *New Generations Computing*, 4:67–95, 1986.

[22] V. Lifschitz. Circumscription. In D.M.Gabbay, C.J.Hogger, and J.A.Robinson, editors, *The Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 3, pages 297–352. Oxford University press, 1994.

[23] V. Lifschitz. The logic of common sense. *ACM Computing Surveys*, 27:343–345, 1995.

[24] J. McCarthy. Programs with common sense. In *Proceedings of the Teddington Conference on the Mechanisation of Thought Processes*, pages 75–91. Her Magesty's Stationary Office, 1959.

[25] J. McCarthy. Circumscription - a form of nonmonotonic reasoning, 1986. http://www-formal.stanford.edu/jmc/.

[26] John McCarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969. reprinted in McC90.

[27] D. McDermott. We've been framed: Or, why ai is innocent of the frame problem. In Zenon W. Pylyshyn, editor, *The Robot's Dilemma: The Frame Problem in Artificial Intelligence*, Theoretical Issues in COgnitive Science. ABLEX Publishing Corporation, 1987.

[28] L. Morgenstern. The problem with solutions to the frame problem. Technical report, IBM T.J. Watson Research Center, 1995.

[29] F. Sadri. Three recent approaches to temporal reasoning. *Temporal Logics and their Application*, 1987.

[30] F. Sadri and R. Kowlaski. Variants of the event calculus. In *Proceeding ICLP 95*, 1995.

[31] E. Sandewall and Y. Shoham. Non-monotonic temporal reasoning. In C.J. Hogger D.M. Gabbay and J.A. Robinson, editors, *The handbook of logic in A.I. and Logic Programming*, volume 4, pages 439–498. Oxford university press, 1995.

[32] J. Searle. *Intentionality*, chapter Background. Cambridge University Press., 1983.

[33] J. Searle. *The Rediscovery of the Mind*, chapter Consciousness, Intentionality, Background. MIT Bradford, 1992.

[34] M. Shanahan. Personal communication.

[35] M. Shanahan. Prediction is deduction but explanation is abduction. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1989.

[36] M. Shanahan. Representing continuous change in the event calculus. In *Proceedings ECAI-90*, pages 598–603, 1990.

[37] M. Shanahan. A circumscriptive event calculus. *Artificial Intelligence*, 77:249–284, 1995.

[38] M. Shanahan. Default reasoning about spatial occupancy. *Artificial Intelligence*, 74(1):147–163, 1995.

[39] M. Shanahan. A logical account of the common sense informatic situation for a mobile robot. Technical report, Department of Computer Science, Queen Mary and Westfield College, 1996.

[40] M. Shanahan. Noise and the common sense informatic situation. In *Proceedings AAAI 96*, pages 1098–1103, 1996.

[41] M. Shanahan. Robotics and the common sense informatic situation. In *Proceeding of the 12th European Conference on Artificial Intelligence*. John Wiley and Sons, 1996.

[42] M. Shanahan. What sort of computation mediates between perception and action? Working notes of the AAAI Fall Symposium on Embodied Cognition, 1996.

[43] M. Shanahan. Event calculus planning revised. In *Proceedings European Conference on Planning*, pages 390–420. Springer-Verlag Lecture notes in Artificial Intelligence 1348, 1997.

[44] M. Shanahan. *Solving the frame problem*. The MIT press, 1997.

[45] Murray Shanahan. Discussion of the paper: "A Logical Account of the Common Sense Informatic Situation for a Mobile Robot". ETAI debates, August 1998. http://www.ida.liu.se/ext/etai/rac/.

[46] M. Thielscher. Ramification and causality. *Artificial Intelligence*, 87:317–364, 1997.

[47] Laure Vieu. Spatial representation and reasoning in artificial intelligence. In Olivero Stock, editor, *Spatial and Temporal Reasoning*, pages 5–41. Kluwer Academic Publishers, 1997.

[48] Luìs Vila. A survey on temporal reasoning in artificial intelligence. *AI Communications*, 7(1):4–28, March 1994.

[49] M. Wooldridge and N. R. Jennings. Intelligent agents: Theory and practice. *Knowledge engineering Review*, 10(2):115–152, 1995.