DIPARTIMENTO DI INFORMATICA
E SISTEMISTICA ANTONIO RUBERTI

SAPIENZA
UNIVERSITÀ DI ROMA

Switching tasks and flexible reasoning in the Situation Calculus

Alberto Finzi
Fiora Pirri

Technical Report n. 7, 2010

Switching tasks and flexible reasoning in the Situation Calculus

Alberto Finzi and Fiora Pirri
{finzi}@na.infn.it {pirri}@dis.uniroma1.it

March 25, 2010

Abstract

In this paper we present a new framework for modelling switching tasks and adaptive, flexible behaviours for cognitive robots. The framework is constructed on a suitable extension of the Situation Calculus, the Temporal Flexible Situation Calculus (TFSC), accommodating Allen temporal intervals, multiple timelines and concurrent situations. We introduce a constructive method to define pattern rules for temporal constraint, in a language of macros. The language of macros intermediates between Situation Calculus formulae and temporal constraint Networks. The programming language for the TFSC is TFGolog, a new Golog interpreter in the Golog family languages, that models concurrent plans with flexible and adaptive behaviours with switching modes. Finally, we show an implementation of a cognitive robot performing different tasks while attentively exploring a rescue environment.

Keywords: Cognitive robotics, executive control, cognitive control, switching tasks, adaptive and flexible behaviours, Situation Calculus, action perception and change, temporal planning.

1 Introduction

Several approaches have been recently taken for the advances of cognitive robotics. These different viewpoints are foraged by new breakthroughs in different research areas correlated to cognitive control and, mainly, by new experimental settings that have encouraged a better understanding of the cognitive functioning of executive processes. In real-world domains robots have to perform several activities requiring a suitably designed cognitive control, to select and coordinate the operation of multiple tasks.

The ability to establish *the proper mappings between inputs, internal states, and outputs needed to perform a given tasks* [48] is called "cognitive control" or "executive function" in neuroscience studies and it is often analysed with the aid of the concept of *inhibition* (see e.g. [48, 3]), explaining how a subject in the presence of several stimuli responds selectively and is able to resist inappropriate urges (see [77]). Cognitive control, as a general function, explains flexibly switching between tasks, when reconfiguration of memory and perception is required, by disengaging from previous goals or task sets (see [45][55]).

The role of task switching in robot cognitive control is highlighted in many biologically inspired architectures, such as the ISAC architecture [34], ALEC architecture based on state changes induced by homeostatic variables [25], Hammer [15] and the GWT (Global Workspace Theory) [71].

Studies on cognitive control, and mainly on human adaptive behaviours, investigated within the task-switching paradigm, have strongly influenced cognitive robotics architectures since the eighties, as for example the Norman and Shallice [54] ATA schema, the FLE model of Duncan [16] and the principles of goal directed behaviours in Newell [53] (for a review on these architectures in the framework of the task switching paradigm see [67]).

Also the approaches to model based executive robot control, such as Williams [8] and earlier [33, 80], devise runtime systems managing *backward inhibition* via real-time selection, execution and actions guiding, by hacking behaviours. This model-based view postulates the existence of a declarative (symbolic) model of the executive which can be used by the cognitive control to switch between processes within a reactive control loop. Here, the executive model provides a local and detailed representation of the system and monitors the processes engagement and disengagement. In this context, the flexible temporal planning approach (e.g. Constraint-based Interval Planning framework [33]), proposed by the planning community, has shown a strong practical impact in real world applications based on deliberation and execution integration (see e.g. RAX [33], IxTeT [27], INOVA [74], and RMPL [80]). These approaches amalgamate planning, scheduling and resource optimisation for managing all the competing activities involved in many robot tasks. Important examples are the flexible concurrent plan concepts of Jonsson and colleagues [33, 12] and Ghallab and colleagues [27]. The flexible temporal planning approach, underpinning temporal constraint networks, provides a good model for behaviours interaction and temporal switching between different events and processes. However, the extremely complex structure required by the executive robot control has strongly affected the coherence of the whole framework, especially because implementation issues have prevailed in the flexible temporal planning approach over the semantic modelling of the different components integration.

On the other hand, from a different perspective, high level executive control has been introduced in the *qualitative* Cognitive Robotics[1] community, within the realm of theories of actions and change, such as the Situation Calculus [46, 61, 41, 65], Fluent Calculus [68, 17, 76], Event Calculus [72, 73, 22], the Action language [26] and their built-in agents programming languages such as the Golog family (ConGolog, INDIGolog, Readylog, etc. see [38, 64, 11, 30]), FLUX [75], and similarly APL [1]. In the theory of action and change framework the problem of executive control has been regarded mainly in terms of action properties, their effects on the world (e.g. the frame problem) and the agent's ability to decide on a successful action sequence basing on its desire, intentions and knowledge [40, 4, 29, 42]. These both for off-line and online action execution. In this sense high level executive control is intended as the reasoning process underlying the choice of actions.

Nonetheless reactive behaviours have been considered from the view point of the interleaving properties of the agent's actions and external exogenous actions, induced by nature [63]. Reiter grasped the concept of

[1] The term has been earliest introduced by Reiter IJCAI 93, see also [39]

inhibition through that of "bad situations" [65]. Bad situations, however, were proposed in the perspective of actions effects achievements, although his considerations where more deeply immersed in the human behaviour and also concerned with task switching. Analogously, in Decision Theoretic Golog the stochastic structure of actions served to achieve the most successful plan in uncertain domains [9].

Real world robot applications are increasingly concerned not just (and not only) with properties of actions but also with the system reaction to a huge amount of stimuli, requiring to handle response timing. Therefore, the need to negotiate the multiplicity of reactions in tasks switching (for vision, localisation, manipulation, exploration, etc.) is bearing a different perspective on action theories. An example is the increasing emphasis on agents programming languages or on multiple forms of interactions leading to the extraordinary explosion of multi agent systems.

Indeed, the control of many sources of information, incoming from the environment, likewise arbitration of resource allocation for perceptual-motor and selection processes, had become the core challenge in actions and behaviours modelling.

The complexity of executive control under the view of adaptive, flexible and switching behaviours, in our opinion, requires the design of a grounded and *interpretative* framework that can be accomplished only within a coherent and strong qualitative model of action, perception and interaction. This with the proviso to offer sound transformations of the underlying constructs into structures that can be treated quantitatively (e.g. temporal networks, Bayes networks, graphical models, etc.).

The main contributions of this paper can be briefly summarised as follows:

- we extend the framework of the Situation Calculus to represent heterogeneous, concurrent, and interleaving flexible behaviours, subject to switching-time criteria. This leads to a new integration paradigm in which multiple parallel timelines assimilate temporal constraints among the activities.
- Temporal constraints and rules for their definition (the compatibilities) implement adaptation and inhibition of behaviours. This is made possible via a specific term that we call *bag of timelines* (also bag of situations), actually a set of concurrent, temporal situations formalising processes on multiple timelines. On the basis of this term we are able to introduce a constructive method for declaring temporal compatibilities, based on a meta-language.
- The compatibilities are rules with a double facet, they are formulae of the Situation Calculus but also the logical counterpart of a temporal network. We show, indeed, that compatibilities can be transformed into temporal constraint networks. We show therefore that, under specific circumstances, logic-based reasoning and constraints propagation can be treated independently still in the same logical framework.
- As usual within the Situation Calculus, the extended framework provides the semantics for specifying a Golog interpreter. We introduce the Temporal Flexible Golog (TFGolog) programming language suitable for representing high-level agent programs for concurrent and temporal switching processes. We show how the TFGolog interpreter transforms high-level programs into temporally flexible plans.
- We prove consistency results about the TFSC and prove several properties of the system.
- We provide several examples that illustrate our approach and show its usefulness. In particular, we show that the framework can foster attention driven exploration. The example has also been used for testing this framework, as reported in [10].

The rest of the paper is organised as follows. In the next section we give an intuition of the proposed work with an example about cognitive robot control. In Section 3 we recall some preliminaries properties of the Situation Calculus and Golog and we introduce the Temporal Flexible Situation Calculus (TFSC). The TFSC as an extension of the Situation Calculus, including timelines and bag of timelines, is used also to define processes and constraints between processes, these issues are discussed in Section 4 and in Section 5. The language for the

construction of constraints and flexible behaviours is presented in Section 5 and it is shown, in Section 6, how it maps to a Constraint Temporal Network. In Section 7 we introduce the Temporal Flexible Golog interpreter showing several results and examples. In Section 8 we illustrate the example of an attentive robot controller based on the Temporal Flexible Golog interpreter. Finally we dedicate Section 9 to the related works and to hint future works. All the proofs are collected in the Appendices A and B at the end of the paper.

2 Why Flexible Planning and why modelling multiple behaviours

Robotic systems, whether they are mobile robots, camera networks or sensor networks, are composed of several heterogeneous hardware and software components operating *concurrently* and smoothly interacting with the external world. In these systems, complexity increases exponentially with the number of possible interactions among components. Each component can perform a set of *activities* whose *duration* might be controllable or not, as exogenous events can disturb allocated time lags. The range and variety of components possible interactions is quite extended, although limited by both structural temporal constraints (such as timeouts, time priorities, time windows), and resource constraints, such as terrain features for locomotion, light features for vision, and speed time for sensor networks.

A control plan suitable for these systems should be *flexible*, in order to be robust and avoid deadlocks. In other words it should be able to make available, *at any time*, a set of possible behaviours, so that the actual behaviour can be decided on-line.

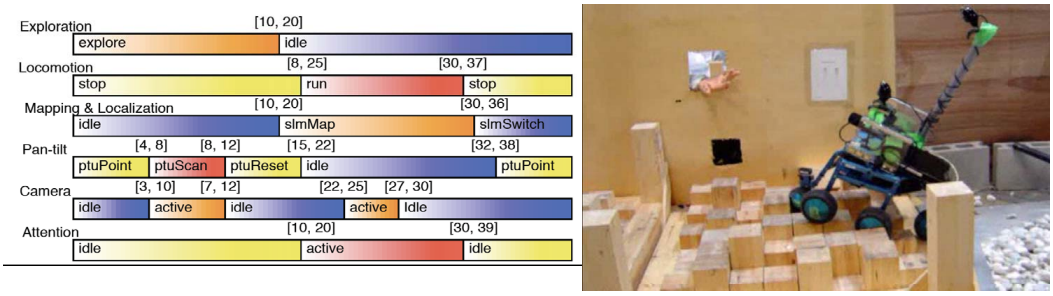

Figure 1: Planned activities for the rescue mobile robot (see the robot looking at a victim hand waving from a hole, in the wall, on the right). Each component is represented by a timeline where the planned activities are sequenced. Starting and ending time of these activities are bound just at the execution time.

Example 1 (Cognitive Robot) A mobile robotic system performs some basic tasks such as exploring the environment (possible a rescue environment). The robot control system is composed of several functional components, some typical are: *Mapping and Localisation*, *Navigation* (for path-planning), *Pan-tilt* unit (for head and gaze control), *Camera* (for vergence, zooming, etc.), *Locomotion* (low level engine controllers), *Sound processing*, *Visual processing*, *Attentional system*, *Exploration* (taking care of search strategies), and possible other components related to other sensors and other adaptation needs. These concurrent activities may have several causal, temporal, and resource constraints. For example, the *Pan-tilt* should look ahead while the robot is moving. The *Camera* should be continuously pointed in the correct direction that might be earlier detected by sound and, at the same time, the robot engine vibrations should be compensated by some stabilisation process, likewise ego-motion for suitable tracking. *Camera* and *Pan-tilt* components might start a tracking activity during

a task requiring to explore and search for something, or to follow someone. However while the starting time of the *ptuScan* process is controllable, the ending time of this process is nondeterministic, as it depends on the response of the scanned object/person. In turn, the ending time of the tracking process affects the starting and ending times of other activities, for example to pinpoint where exactly to go, operating the necessary strategies to achieve that. Figure 1 illustrates on the left a flexible temporal plan for a rescue robot (on the right looking at a hand waving from a hole in a wall). The plan stipulates that the *explore* process, given that it ends within an interval of $[10, 20]$, should commit the the end-time of the *stop* process to be greater than 8, but less than 25 seconds, while *ptuReset* should end between 15 and 22 seconds. Now, *ptuReset* can be active just during the locomotion component process *stop*; the *stop* process, in turn, can end only after the end of *ptuReset*. On the other hand *explore* is not directly affected by *ptuReset* and *stop*, hence it can end before, after or during these activities, and its ending time can switch w.r.t. the ending times of *ptuReset* and *stop*. Whenever a set of planned activities is executed, the associated activation times are actually bound; hence, the enforced constraints can be suitably propagated.

In the next sections we show how these problems can be addressed and solved in the Situation Calculus and Golog providing a clear and sound framework for designing a complex system.

3 Basics for the Temporal Flexible Situation Calculus

In this section, we present the basic ideas and formal structure of the Temporal Flexible Situation Calculus (*TFSC*). The TFSC is conceived for describing a complex dynamic system with a finite number of components, to which a certain amount of resources and processes are assigned. The system should be able to execute interleaving processes, allowing switching between processes threads of different components, by inhibiting active tasks of less demanding components.

3.1 Preliminaries

The Situation Calculus [46, 65] (SC) is a sorted first order language with equality, augmented with a second order induction axiom. The underlying signature of the sorted language is specified by three sorts: Act for actions, Sit for situation and Obj for objects. To simplify reading we usually refer to these sorts as actions, situations and objects.

The terms of sort actions are either constants or functions mapping elements of sort objects and possibly of sort actions into elements of sort actions, e.g. $move(x, y)$.

Terms of sort situation are either the constant symbol S_0 or terms of the form $do(a, s)$, where a is a term of sort action and s is a term of sort situation. The term S_0 denotes the *initial situation*, where no action has yet occurred, while $do(a, s)$ encodes the sequence of actions obtained from executing the action a after the sequence of actions encoded in s.

Properties of objects and their dynamics are described by *fluents*. Thus fluents denote properties that may change when executing an action, and are specified by either predicates or function symbols whose last argument is a situation. A *basic action theory* is defined by the following set of axioms

$$BAT = (\Sigma, \mathcal{D}_{S_0}, \mathcal{D}_{ssa}, \mathcal{D}_{una}, \mathcal{D}_{ap}). \tag{1}$$

Here

- Σ is the set of domain independent foundational axioms for the domain of situations, see Table 1. Situations are kept countably infinite by a second order axiom (see Table 1) assessing that there are no unintended models of the language, in which situations can be strange objects.

- $\mathcal{D}_{una}$ is the set of unique name axioms for actions, which expresses that different action terms, namely different names, stand for different actions:

  $$A(x_1, \ldots, x_n) \neq B(y_1, \ldots, y_n),$$

  and identical action terms have the same arguments

  $$A(x_1, \ldots, x_n) = A(y_1, \ldots, y_n) \rightarrow x_1 = y_1 \wedge \cdots \wedge x_n = y_n.$$

- $\mathcal{D}_{S_0}$ is a set of first-order formulas describing the initial state of the domain (represented by S_0).

- $\mathcal{D}_{ssa}$ is the set of *successor state axioms* [62, 65], one for each fluent symbol $F(\vec{x}, s)$, in the language. A successor state axiom is an explicit definition of a fluent in a successor state $do(a, s)$ as follows:

  $$F(\vec{x}, do(a, s)) \equiv \Phi_F(\vec{x}, a, s).$$

  A successor state axiom provides both a definition of action effects and a solution to the frame problem (assuming deterministic actions).

Given a basic action theory, it is possible to infer the properties of the theory just appealing to the initial theory $\mathcal{D}_{S_0}$. This is done by regressing any formula, taking as argument a situation of the form $do(a_m, \ldots, do(a_1, S_0))$, into an equivalent formula taking as argument the initial situation S_0 and not mentioning other situations different from S_0.

The regression of a formula $\phi(do(a_m, \ldots, do(a_1, S_0)))$ is defined via a regression operator $\mathcal{R}$ by induction using the definitional structure of the successor state axioms and the properties of $\mathcal{R}$ as follows:

$$\begin{array}{l} \mathcal{R}(F(\vec{x}, do(a, s))) = \Phi_F(\vec{x}, a, s) \\ \mathcal{R}(\neg\phi) = \neg\mathcal{R}(\phi) \\ \mathcal{R}(\phi_1 \wedge \phi_2) = \mathcal{R}(\phi_1) \wedge \mathcal{R}(\phi_2) \\ \mathcal{R}(\exists x.\phi) = \exists x.\mathcal{R}(\phi). \end{array} \tag{2}$$

The simplicity and elegance of making inference and prove properties of situations is, indeed, due to the structure of the axioms, based on explicit definitions of the successor state. This structure would be prejudiced if state constraints are added to the theory, i.e. formulas mentioning situations neither *uniform* in S_0[2] nor in the form of successor state axioms. For example the followings state constraints:

$$\begin{array}{l} \forall s.Raise(sun, s). \\ \forall s On(x, y, s) \wedge On(y, z, s) \rightarrow On(x, z, s). \end{array} \tag{3}$$

lacking a definitional structure, would compromise the inference based on regressing sentences to the initial database $\mathcal{D}_{S_0}$.

Golog, was earlier introduced in [38], is an agent programming language formally based on the SC and usually implemented in Eclipse Prolog. Golog uses Algol-like control constructs to define complex actions from the primitive actions, which are those of a basic action theory *BAT*, see (1):

1. *Action sequences*: $p_1; p_2$.
2. *Tests*: $\phi?$.
3. *Nondeterministic action choices*: $p_1|p_2$.

[2]Formulas uniforms in $\sigma = do(a_1, \ldots, do(a_m, S_0)), m \geq 0$, are formulas either not mentioning situation terms or formulas not mentioning *Poss* nor $\sqsubset$ nor any other situation term than σ [61].

4. *Nondeterministic choices of action argument*: $(\pi x).p(x)$.
5. *Conditionals*: **if** ϕ **then** p_1 **else** p_2.
6. *While loops*: **while** ϕ **do** p.
7. *Nondeterministic iteration*: ${p_1}^*$.
8. *Procedure calls*: {**proc** $P_1(\vec{x}_1)$ p_1 **end**; ... **proc** $P_n(\vec{x}_n)$ p_n; p }

An example of a Golog program is

while $\neg At(1,2)$ **do** $(\pi x, y)moveTo(x,y)$.

Intuitively, the nondeterministic choice $(\pi x, y)moveTo(x,y)$ is iterated until the atom $At(1,2)$ is verified.

The Golog declarative semantics is defined in the language of SC. Given a complex action δ (a Golog program), the abbreviation $Do(\delta, s, s')$ says that situation s' can be reached from situation s by executing some complex action specified by the program δ.

The construct definitions are the following:

1. Primitive actions:

$$Do(a, s, s') \stackrel{def}{=} Poss(a, s) \wedge s' = do(a, s).$$

2. Test actions:

$$Do(\phi?, s, s') \stackrel{def}{=} \phi[s] \wedge s = s'.$$

3. Sequence:

$$Do(p_1; p_2, s, s') \stackrel{def}{=} \exists s'' Do(p_1, s, s'') \wedge Do(p_1, s'', s').$$

4. Non-deterministic choice of two actions:

$$Do(p_1|p_2, s, s') \stackrel{def}{=} Do(p_1, s, s') \vee Do(p_2, s, s').$$

5. Non-deterministic choice of arguments:

$$Do(\pi(x, p(x)), s, s') \stackrel{def}{=} \exists x\, Do(p(x), s, s').$$

6. Non-deterministic iteration:

$$\begin{aligned} Do(p^*, s, s') \stackrel{def}{=} \forall P.\{&P(s_1, s_1) \wedge \\ &\forall s_1, s_2, s_3.P(s_1, s_2) \wedge Do(p, s_2, s_3)]\} \rightarrow P(s_1, s_3). \end{aligned}$$

For procedure call expansion and other important constructs we refer the reader to [38, 29, 28, 65, 31].

3.2 Extensions of the SC

Among several languages for action theories (like the Fluent Calculus [68, 17, 76] the Event Calculus [72, 73, 22] and the Action language [26]) the SC is particularly simple to be extended and adapted to specific domains, such as, for example, cognitive robotics domains.

In fact, being an axiomatic theory, an extension of the SC, requires two simple steps:

1. extend the set of foundational axioms Σ to account for more rich domains;
2. show that the new extension respects the fundamental constraints required to do inference within the system.

The flexibility of both the successor state axioms $\mathcal{D}_{ss}$ and the action precondition axioms $\mathcal{D}_{ap}$ allows the user to define any domain.

This is the reason why there have been many contributions to extensions of the Situation Calculus such as ([42, 58, 40, 66, 61, 4, 59, 21, 9, 65]). All these contributions, further, have coped with the constraints required by the regression inference, including the specification of the Golog programming language (see Section 7) such as ([38, 11, 64, 30]), requiring axioms to be based on the construction of explicit definition. In particular, macro definitions (see also [65] for a paragraph on "Why Macros?") are explicit definitions of predicates that are not added to the language, therefore they stand also for abbreviations of the formulas defining them (*the definiens*).

We refer the reader to [65, 61] for a complete introduction to the inference mechanisms in the Situation Calculus.

In this paper we extend the Situation Calculus by adding a new set of axioms to the set of its foundational axioms, and by introducing macro definitions. In order to ensure that all the constraints are satisfied we need to go into details that are rather boring, although often straightforward, therefore many details are postponed and described in the Appendix. In particular, all proofs, likewise lemmas, for this section are given in Appendix A and Appendix B.

3.3 Time, types and bag of timelines

The set of foundational axioms of the Situation Calculus, together with the set of new axioms are reported in Table 1. We introduce three kind of extensions. The first extension, concerning time, is essentially the same as the one introduced by Reiter in [65] and [63, 56], but making explicit the definition of *start* [65]. With the second kind of extension we introduce *name types* as objects, i.e. specific elements of sort object, denoted by constants: these are used in the perspective of describing a system with several components that can be each *named* by a specific constant. Note that because a robot system is composed of a finite set of parts we assume that the set of components is bound to be finite, although what a component can do might be described by an infinite set of actions. Each name type is *extensively defined* by a collection of actions that the component can execute.

Name types are used to classify actions according to the actuating agent, for example often this has been implicitly assumed in the presence of different agents, by naming each agent specifically, here we do the same but in a systematic way. The third extension deals with set of situations, with some specific properties which are obtained by types and time. These kind of situations are called *timelines* and a set of timelines is here specified by what we call *bag of timelines*.

Notation: in the following sections $\mathcal{L}_X$ denotes the language specified by the axioms X. More precisely we assume that the signature includes all the symbols (functions and predicates) mentioned in X, equality, the quantifiers and connectives of FOL. Thus, for example, $\mathcal{L}_\Sigma$ is the language of the axioms Σ. Also, we shall use the ordering relation $s \sqsubseteq s'$ between situations s and s', that abbreviates $s \sqsubset s' \vee s = s'$ (see foundational

Foundational Axioms of SC, Σ		
$\neg(s \sqsubset S_0)$	$s \sqsubset do(a, s') \equiv s \sqsubseteq s'$	$do(a, s) = do(a', s') \equiv s = s' \wedge a = a'$
$\forall P.P(S_0) \wedge \forall as.P(s) \rightarrow P(do(a, s)) \rightarrow \forall sP(s)$		

Flexible Time SC extension axioms $\mathcal{A}^+ = \Sigma \cup Ax_0 \cup Ax_1 \cup Ax_2 \cup Ax_3$		
Time axioms, Ax_0 $\Sigma_{time} = \Sigma \cup Ax_0$	Type axioms, Ax_1-Ax_2 $\Sigma_H = \Sigma_{time} \cup Ax_1$, $\Sigma_{=_\nu} = \Sigma_H \cup Ax_2$	Timelines axioms Ax_3 and Bag of timelines axioms, Ax_4
$\Sigma_{=_\nu} = \Sigma_H \cup \Sigma_{time} = \Sigma \cup Ax_0 \cup Ax_1 \cup Ax_2$		$\mathcal{A}^+ = \Sigma_{=_\nu} \cup Ax_3 \cup Ax_4$
T1. $time(S_0) = t_0$	H1 $\forall a. \bigvee_{i=1}^{n} (H(i,a) \wedge \bigwedge_{i=1}^{n} (H(i,a) \rightarrow \bigwedge_{\substack{j=1 \\ j \neq i}}^{n} \neg H(j,a)))$	G1. $\left\{ \forall s,\ s_{j1}, \ldots s_{jk}.\ s \in_{\mathscr{S}} \mathscr{B}(\langle s_{j1}, \ldots, s_{jk} \rangle_j) \equiv \bigvee_{1 \leq p \leq k} s = s_{jp} \wedge (s_{jp} = S_0 \vee \bigvee_i \mathscr{T}(i, s_{jp})) \right\}_{k \in \mathbb{N}}$
T2 $\forall \vec{x},\ t.time(A(\vec{x}, t)) = t \rightarrow t > t_0$	H2 $\forall a, a'.a =_\nu a' \equiv \exists i.\ H(i,a) \wedge H(i,a'))$	G2. $\forall s \forall \mathfrak{s}, \mathfrak{s}'\ .(s \in_{\mathscr{S}} \mathfrak{s} \equiv s \in_{\mathscr{S}} \mathfrak{s}') \equiv (\mathfrak{s} =_{\mathscr{S}} \mathfrak{s}')$
T3. $\forall \vec{x}, t, s.\ time(do(A(\vec{x}, t), s)) = time(A(\vec{x}, t))$	E1. $\forall s, s'(s = s' \rightarrow s =_\nu s') \wedge (s \sqsupset S_0 \rightarrow \neg(s =_\nu S_0 \vee S_0 =_\nu s))$	G3. $\exists \mathfrak{s}.\mathfrak{s} = \mathscr{B}_0$
	E2. $\forall a \neg(a =_\nu S_0)$	G4. $\forall s\ \forall \mathfrak{s} \forall i.s = S_0 \vee \mathscr{T}(i,s) \rightarrow \exists \mathfrak{s}'\ (\mathfrak{s}' =_{\mathscr{S}} \mathfrak{s} \cup_{\mathscr{S}} \mathscr{B}(s))$
	E3. $\forall a, a', s'(a =_\nu do(a', s')) \equiv (a =_\nu a') \wedge (s' \sqsupset S_0 \rightarrow (s' =_\nu a))$	G5. for all sentences ϕ $\varphi(\mathscr{B}_0) \wedge (\forall \mathfrak{s} \forall s.\varphi(\mathfrak{s}) \wedge \varphi(\mathscr{B}(s)) \rightarrow \varphi(\mathfrak{s} \cup_{\mathscr{S}} \mathscr{B}(s))) \rightarrow \forall \mathfrak{t} \varphi(\mathfrak{t})$
	E4. $\forall a', s, s'(s =_\nu do(a', s')) \equiv (s =_\nu a') \wedge (s' \sqsupset S_0 \rightarrow s' =_\nu s)$	W1. $\forall a, s.\ \bigwedge_{i=1}^{n} \mathscr{T}(i, do(a,s)) \equiv (s = S_0 \wedge H(i,a)) \vee (s \sqsupset S_0 \wedge a =_\nu s \wedge \mathscr{T}(i,s))$
	E5. $\forall a, s(s =_\nu a) \equiv (a =_\nu s)$	

Table 1: The foundational axioms Σ, of the basic Situation Calculus, the axioms $\mathcal{A}^+ = \Sigma \cup Ax_0 \cup Ax_1 \cup Ax_2 \cup Ax_3 \cup Ax_4$ of the Flexible time Situation Calculus, extending Σ with *time*, *types* and *bag of timelines*. The extension of the foundational axioms Σ is incremental. Four sets are built: first the set Σ_{time}, extending Σ with time; then the set Σ_H, extending Σ_{time} with types; then the set $\Sigma_{=_\nu}$ extending Σ_H with equivalence relations over actions and situations; finally the set $\mathcal{A}^+$ extending $\Sigma_{=_\nu}$ with *timelines* and *bag of timelines*.

axioms Σ in Table 1). We shall use $\vec{x}$ to denote a tuple of variables, a to denote variables of sort action, $A(\vec{x})$ to denote action functions with arguments $\vec{x}$. When a situation mentions only actions in the form $A(\vec{x})$ then its only variables are variables of sort object, thus we use the symbol α to denote actions which are either ground or in the form $A(\vec{x})$. We use the symbol s to denote variables of sort situations, σ to denote histories of actions, such as $\sigma = do(a_m, \ldots, do(a_1, S_0))$, that is, a sequence of actions of length m, $m >= 1$ and S_0 to denote the initial situation, S_0 is a constant. As we shall extend the signature the new symbols will be contextually introduced.

□

3.4 Representing Time in TFSC

Time has been extensively introduced in the Situation Calculus in [58, 63, 60], where actions are instantaneous, and their time is selected by the function *time*(.). Durative actions are considered as *processes* [58, 65], represented by fluents, and durationless actions start and terminate these processes. For example, $going(hill, s)$ is started by the action $startGo(hill, t)$ and it is ended by $endGo(hill, t')$.

Analogously as in [63, 56], primitive actions are instantaneous and are represented by the term $A(\vec{x}, t)$ where t is a special argument representing the execution time. For example, $moveTo(room4, 0.5)$ means that $moveTo$ was executed at time 0.5.

We use time selection functions to extract the time of both actions and action sequences. In particular, we introduce a function $time : Sit \cup Act \rightarrow \mathbb{R}^+$, mapping both situation and actions into the positive real line, thus we implicitly assume that the reals are axiomatised (see the Appendix page 45). We also introduce the relation $<$ and $\leq$ defined as $< \vee =$ ranging over the reals $\mathbb{R}^+$. This is a common assumption in the Situation Calculus (see [65]), thus we leave it like this.

We denote with Ax_0 the set of axioms $T1$-$T3$ and $\Sigma_{time} = \Sigma \cup Ax_0$, see Table 1. Axiom $T1$ says that the time of the initial situation S_0 is the initial time t_0, axiom $T2$ says that the time of an action is the time of its time argument which has to be a positive real number successive to the initial time. Finally the third axiom $T3$ says that the time of a situation $do(a, s)$ is the time of the action a. The set Σ_{time} is a conservative extension of the axioms Σ, of the basic Situation Calculus (see Lemma 1 in the Appendix). Here by conservative extension we mean that Σ_{time} is obtained by extending the original language $\mathcal{L}_\Sigma$ to the new language $\mathcal{L}_{\Sigma_{time}}$ without changing the initial theory Σ and its deductive closure, when only the original language is considered.

The set Ax_0, however, does not ensure that the ordering $\sqsubseteq$ on situations is coherent with time. In other words $s \sqsubseteq s' \rightarrow time(s) \leq time(s')$ does not hold, in general, in a model $\mathcal{M}$ in which $\Sigma_{time} \cup \mathcal{D}_{una}$ is verified. Nevertheless it is always possible to build a model of $\Sigma_{time} \cup \mathcal{D}_{una}$ in which the above condition is verified (see Lemma 2 in the Appendix). Thus, to add coherence between situations and time we need to add a further axiom

$$T4. \quad s \sqsubseteq s' \rightarrow time(s) \leq time(s') \tag{4}$$

This new axiom will restrict the set of models to time coherent situations. In this models, although $s \sqsubseteq s' \rightarrow time(s) \leq time(s')$ will be verified, the inverse implication $(time(s) \leq time(s')) \rightarrow s \sqsubseteq s'$ in general will not hold.

3.5 Typed Actions and Situations

The second column of Table 1 illustrates the axiomatisations for name types. The distinction between sorts and name types is that sorts induce a partition on the domain, while name types are defined in the language via constant symbols involving only the sort Obj, still inducing a partition on actions hence on situations. In particular, axioms $Ax_1 = \{H1, H2\}$ regulate *name types*, and the set $Ax_2 = \{E1, E2, E3, E4, E5\}$ extend types to situations via the relation $=_\nu$ that is defined by Axiom $(H2)$. Axiom $(H1)$ settles the required specifications for name types to be coherent with respect to actions. The disjunction of components mentioned in the first conjunct of $(H1)$ states that each action is ascribed to some component i. On the other hand the second conjunct of $(H1)$

states that, whenever an action is ascribed to a component with name type i, it cannot be ascribed to any other component. Note that $(H1)$ does not affect the set $\mathcal{D}_{una}$ of inequalities for actions and, clearly, in a basic action theory, with a single component, $(H1)$ is always satisfied.

The axioms $(H1)$ and $(H2)$ can be safely added to Σ_{time}, forming the theory $\Sigma_H = \Sigma_{time} \cup Ax_1$, and the theory Σ_H maintains satisfiability, see Lemma 3, in the Appendix page 46.

The partition of actions, according to name types, is equipped with the relation $=_\nu$, defined by $(H2)$, see Table 1. We show that $=_\nu$ is an equivalence relation on the set of actions in Lemma 4, see the Appendix, page 47.

Axioms $E1$-$E5$ (see Table 1, from row nine, second column) are, thus, needed to extend the relation $=_\nu$, defined by axiom $(H2)$ for actions, to the set of situations. Axiom $(E1)$ states that if two situations are equal then they must be also of the same type, but no situation is of the same type as S_0. Axiom $(E2)$ states that no action can be of the same type of S_0. Axiom $(E3)$ states recursively that an action is of the same type of a situation $do(a', s')$ if it is of the same type of the action a' and it is of the same type as s', whenever s' is not S_0. Finally, axiom $(E4)$ says that two situations are of the same type if they mention actions and situations of the same type, and $(E5)$ states symmetry between actions and situations.

Also these axioms can be safely added to the theory built so far. Lemma 5, in the Appendix B, page 47, shows that adding axioms $Ax_2 = E1$-$E5$ to the theory Σ_H, in so obtaining the new theory $\Sigma_{=_\nu} = \Sigma_H \cup Ax_2$, can be done consistently, also if the axioms set $\mathcal{D}_{una}$ is included. Furthermore we show in Lemma 6 (see the Appendix page 48), that $=_\nu$ is an equivalence relation both on actions and on situations. This fact will be used to form timelines (see Section 3.6)

Theorem 1 *Let Σ be the set of foundational axioms of the Situation Calculus, and let $\mathcal{D}_{una}$ be the set of unique name for actions, then the set of axioms $\Sigma_{=_\nu} = \Sigma \cup Ax_0 \cup Ax_1 \cup Ax_2$ is a sound axiomatisation of the temporal flexible Situation Calculus, that is, the set of axioms and $\mathcal{D}_{una}$ together form a satisfiable theory.*

□

Now, with $H1$-$H2$ a new predicate H is introduced in the language, and it is defined for each component i. Let $i_1, i_2, \ldots, i_n$ be a finite set of constants denoting the components of a system, each i_k is a name type. Each $H(i, a)$ can be introduced by an extensional definition as follows:

$$\forall a.H(i, a) \equiv \phi(i, a) \tag{5}$$

When the action names for a specific component is a finite set then the extensional description of the component might be done as follows:

$$\forall a.H(i, a) \equiv \exists \vec{x}_1, \ldots, \vec{x}_n. \bigvee_{i=1}^{n} A_i(\vec{x}_i) = a \tag{6}$$

Example 2 *If we want to define the actions for the robot component* Pan-tilt *we would introduce the constant* pan-tilt *and define it by its actions as follows:*

$$\forall a.H(\textit{pan-tilt}, a) \equiv \quad \exists\, t\ \theta.a = \textit{pan}(\theta, t) \vee \exists\, t\ \gamma.a = \textit{tilt}(\gamma, t) \vee \exists\, t\ x\ y.\ a = \textit{scan}(x, y, t). \tag{7}$$

□

This set of definitions for H is added to $\mathcal{D}_{S_0}$, as they are all uniform in S_0. The easiest way to ensure consistency is to ascribe each action to a single type; in Lemma 7 (see the Appendix page 50) we show the conditions to ensure consistency of the type definitions with the axiom $H1$. As usual with typed languages there are drawbacks, if we consider a generic action name, such as *run*, that could be ascribed to more than one component for all its arguments, then we need either to specialise it to each type or to create a single component gathering all those subscribing the action *run*.

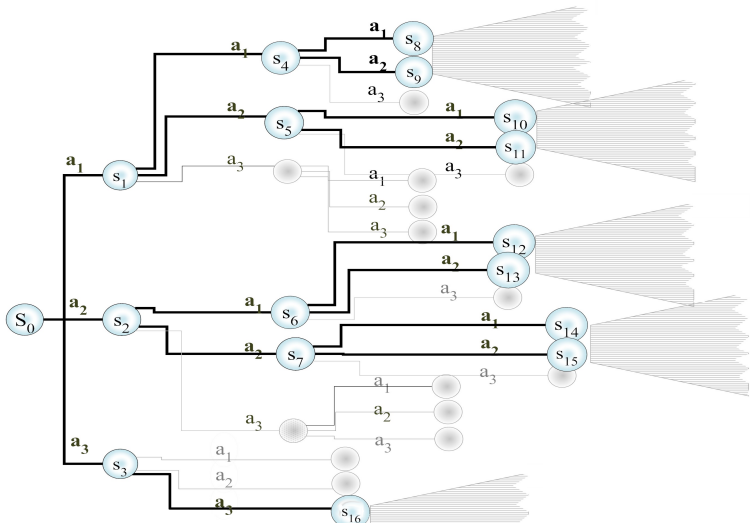

Figure 2: Timelines on a tree of situations. For this representation $H(i, a_1) \wedge H(i, a_2)$ and $H(j, a_3)$ are possible types, $\mathscr{T}(i, do(a_1, S_0)) \wedge \ldots \mathscr{T}(i, do(a_2, S_0))$, and $\mathscr{T}(j, do(a_3, S_0)$ are timelines. By the SSA for timeline, these extend along the situations as indicated by the thick black lines.

3.6 Timelines and bag of timelines

We introduce in this section the concept of a *timeline*. This concept is particularly useful for flexible planning because it makes possible to describe the interaction between processes performed by different components of the system. This concept makes it also possible to deal with the time at which a process starts and ends in a flexible way, according to the way processes interact.

A timeline is denoted by a fluent $\mathscr{T}(i, s)$ and it is defined by an *improper* successor state axiom as follows, see Table 1 axiom $(W1)$:

$$(W1) \quad \bigwedge_{i=1}^{n} \mathscr{T}(i, do(a, s)) \equiv (s \sqsupset S_0 \wedge a =_{\nu} s \wedge \mathscr{T}(i, s)) \vee (s = S_0 \wedge H(i, a)). \tag{8}$$

Note that (8) is not uniform in s[3] as it mentions S_0. Nevertheless the disjunction is obviously exclusive and thus the right hand side never diverges, indeed (8) is regressable as it is shown in Lemma 11, in the Appendix B.

Example 3 *The timeline for the pan-tilt unit can be defined as follows:*

$$\forall a\ s.\mathscr{T}(\textit{pan-tilt}, do(a, s)) \equiv s \sqsupset S_0 \wedge a =_{\nu} s \wedge \mathscr{T}(\textit{pan-tilt}, s) \vee s = S_0 \wedge H(\textit{pan-tilt}, a).$$

Which, by the previous Example 2, are all the histories built up by the actions pan(θ, t), tilt(γ, t) and scan(x, y, t).

Note that, given the extended set $\Sigma_{=_{\nu}}$ the introduction of timelines does not affect the set of successor state axioms and action precondition axioms, thus:

[3]The typical form of a successor state axiom requires the right hand side to mention only the situation s named in the left hand side, being thus uniform in s, to ensure that the *regression* via the right hand side ends in S_0, in so not diverging into two, or more, different situations.

Corollary 1 *Let $\mathcal{D}_{ss} \cup \mathcal{D}_{ap}$ be the set of successor state axioms mentioning also timelines and action precondition axioms. Let $\mathcal{D}_{S_0}$ be the set of formulas, uniform in S_0*[4]*. $\Sigma_{=_\nu} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ is satisfiable iff $\Sigma_{=_\nu} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{ss}$ is satisfiable.*

The characteristics properties of timelines are stated below.

Theorem 2 *A timeline represents the $=_\nu$-equivalence class of situations of the same type.*

The above theorem (see B.4 for the proof) states that all actions in a timeline are of the same type, and whenever a set of actions are of the same type they form a timeline, thus not all situations form timelines.

In Section 4 we show that, under precise conditions, the set of timelines form the set of situations executable by the system components.

Example 4 *Timelines are sequences (histories) of actions indicated in thick black in Figure 2. The histories of actions not belonging to the set of timelines are represented in light gray, that is, histories of actions leading to situations, that do not belong to timelines, are depicted by thin gray lines.*

So a timeline represents the equivalence class of histories of the same type, yet how to ensure a meaningful interaction between timelines that can support switching tasks, is not proven. Suppose that we need to say that while the robot is exploring a given environment to correctly scan the surrounding it should stop or decelerate. The system component controlling the exploration actions should suitably synchronise with the component controlling the pan-tilt and the camera. To treat the interleaving between these two processes we have to ensure that at each time step of the operation loop all timelines are available for choice and switching decisions.

To this end we introduce a new concept that can support set of timelines. This new concept comes with a new sort, we call this new sort the sort of *bag of timelines*.

Intuitively, a bag of timelines is interpreted as a set of lists of actions, where each list of actions is a timeline. We require a bag of timelines to be a finite set and to mention only situations which are timelines, and possibly S_0.

First we have to introduce the sort $\mathscr{S}$ standing for bag of timelines. This is defined as the codomain of a countable set of function symbols $\mathscr{B} : (S^n \mapsto S^n) \mapsto \mathscr{S}$ mapping a permutation of a tuple of situations into an element of the sorted domain, *whose intended interpretation is a bag of timelines*.

Equality on these terms should account for idempotence and commutativity. Therefore we shall extend equality to account for permutations and repetitions of equal arguments.

An $\mathscr{S}$-term $\mathbb{s}$ is defined as $\mathscr{B}(\langle s_{j1}, \ldots, s_{jm}\rangle_j)$. Here, if $m = 0$ we obtain the empty bag of timelines, and we denote the empty bag with the constant $\mathscr{B}_0$. With $\langle s_{j1}, \ldots, s_{jm}\rangle_j$ we denote a permutation of $\{1, \ldots, m\}$.

To formalise these ideas we adapt the finite set axiomatisation, from the system $\mathbb{F}$ of Brown and Wang [79], to add bags of timelines to the language together with the specific symbols $\in_{\mathscr{S}}, =_{\mathscr{S}}$ and empty bag, here identified with the constant term $\mathscr{B}_0$. The axioms, listed in Table 1 are here reported again, where all variables appearing without quantifiers are implicitly universally quantified outside.

$$
\begin{array}{ll}
(G1) & \left\{\forall s.s \in_{\mathscr{S}} \mathscr{B}(\langle s_{j1}, \ldots, s_{jk}\rangle_j) \equiv \bigvee_{1\le p\le k} s = s_{jp} \wedge \left(s_{jp} = S_0 \vee \bigvee_i \mathscr{T}(i, s_{jp})\right)\right\}_{k\in\mathbb{N}} \\
 & \text{Here } i \text{ ranges over the finite set of name types.} \\
(G2) & \forall s.\,(s \in_{\mathscr{S}} \mathbb{s} \equiv s \in_{\mathscr{S}} \mathbb{s}') \equiv (\mathbb{s} =_{\mathscr{S}} \mathbb{s}') \\
(G3) & \exists \mathbb{s}.\mathbb{s} = \mathscr{B}_0 \\
(G4) & \forall s \forall i.\ s = S_0 \vee \mathscr{T}(i, s) \rightarrow \exists \mathbb{s}'.\,(\mathbb{s}' =_{\mathscr{S}} \mathbb{s} \cup_{\mathscr{S}} \mathscr{B}(s)) \\
(G5) & \text{For every sentence } \varphi: \\
 & \varphi(\mathscr{B}_0) \wedge (\forall \mathbb{s} \forall s.\varphi(\mathbb{s}) \wedge \varphi(\mathscr{B}(s)) \rightarrow \varphi(\mathbb{s} \cup_{\mathscr{S}} \mathscr{B}(s))) \rightarrow \forall \mathbb{t}\varphi(\mathbb{t})
\end{array}
\tag{9}
$$

[4] See for the definition of uniform formulas Section 3.1 and [61]. Here $\mathcal{D}_{S_0}$ mentions also all the definitions $H(i, a)$ for each name type i.

The above axiom set ($G1$) defines the symbol $\in_{\mathscr{S}}$. Note that axiom ($G1$) could be bound by a $n \in \mathbb{N}$ and transformed into a single axiom, otherwise there is an axiom for each $k \in \mathbb{N}$. The set of axioms ($G1$) says that a situation s can belong to a bag of timelines, if s is equal to some of the timelines specified in the bag and each situation in the bag of timelines is either S_0 or it is, indeed, a timeline. Note that, following Theorem 2, although S_0 is not a timeline it can belong to a bag of timelines. Axiom ($G2$) is the extensionality axiom limited to bags of timelines. Axiom ($G3$) is the unconditional existence axiom, provided that the empty bag is the constant term $\mathscr{B}_0$. Axiom ($G4$) is the conditional existence for finite bags of timelines, provided that $\mathbb{s}$ and $\mathscr{B}(s)$ are finite bags. Axiom ($G4$) too tells us that a bag of timelines can include S_0. This axiom would allow bags unbound in size. To get bags bound in size whenever axiom ($G1$) requires so for some n then ($G4$) is changed accordingly. Note that, in ($G4$), $\cup_{\mathscr{S}}$ is derivable from ($G1$), see the set operations as obtained in Example 6.

Finally the last axiom ($G5$) is the inductive characterisation of finite sets, it tells that whenever sentences specify terms denoting bags of timelines these terms will denote finite bags.

Example 5 *Let us consider the two timelines $\mathscr{T}(pan\_tilt, do(pan(\theta), do(tilt(\gamma), S_0)))$ and $\mathscr{T}(laser, do(acquire, S_0))$, then:*

a.	$\mathscr{B}(\langle do(pan(\theta),\ do(tilt(\gamma), S_0))_{p1}, do(acquire, S_0)_{p2}\rangle_p)$	is a bag of timelines, by (G1)
b.	$\mathscr{B}(\langle do(pan(\theta), do(tilt(\gamma), S_0))_{p1}, do(acquire, S_0)_{p2}\rangle_p) =_{\mathscr{S}}$ $\mathscr{B}(\langle do(acquire, S_0)_{q1}, do(pan(\theta), do(tilt(\gamma), S_0))_{q2}\rangle_q)$	by (G1,G2)
c.	$\mathscr{B}(\langle do(tilt(\gamma), S_0)_{p1}, do(tilt(\gamma), S_0)_{p2}, S_{0_{p3}}, S_{0_{p,4}}\rangle_p) =_{\mathscr{S}}$ $\mathscr{B}(\langle do(tilt(\gamma), S_0)_{p1}, S_{0_{p3}}, \rangle_p).$	by (G1,G2)

□

The other usual symbols for sets can be extended to bag of timelines, using definitions or, more generally, the induction axiom.

Example 6 *The operators $\subseteq_{\mathscr{S}}$, $\cup_{\mathscr{S}}$ and $\cap_{\mathscr{S}}$ can be defined as follows:*

$$\begin{array}{lll} x. & (\mathbb{s} \subseteq_{\mathscr{S}} \mathbb{t}) & \stackrel{def}{=} \forall s.s \in_{\mathscr{S}} \mathbb{s} \rightarrow s \in_{\mathscr{S}} \mathbb{t} \\ xx. & (\mathbb{s} \cup_{\mathscr{S}} \mathbb{s}' =_{\mathscr{S}} \mathbb{t}) & \stackrel{def}{=} \forall s.s \in_{\mathscr{S}} \mathbb{t} \equiv (s \in_{\mathscr{S}} \mathbb{s} \vee s \in_{\mathscr{S}} \mathbb{s}') \\ xxx. & (\mathbb{s} \cap_{\mathscr{S}} \mathbb{s}' =_{\mathscr{S}} \mathbb{t}) & \stackrel{def}{=} \forall s.s \in_{\mathscr{S}} \mathbb{t} \equiv (s \in_{\mathscr{S}} \mathbb{s} \wedge s \in_{\mathscr{S}} \mathbb{s}') \end{array} \tag{10}$$

On the other hand it is possible to use induction to prove properties of bag of timelines. For example the following simple property:

$$\forall \mathbb{s}, \mathbb{s}', \mathbb{s}''.\mathbb{s} \subseteq_{\mathscr{S}} \mathbb{s}' \wedge \mathbb{s}' \subseteq_{\mathscr{S}} \mathbb{s}'' \rightarrow \mathbb{s} \subseteq_{\mathscr{S}} \mathbb{s}''. \tag{11}$$

can be proved by induction as follows:

$$\begin{array}{lll} & \textit{Let:} & \\ 1. & \varphi(\mathscr{B}_0) & = \forall \mathbb{s}', \mathbb{s}''.\mathscr{B}_0 \subseteq_{\mathscr{S}} \mathbb{s}' \wedge \mathbb{s}' \subseteq_{\mathscr{S}} \mathbb{s}'' \rightarrow \mathscr{B}_0 \subseteq_{\mathscr{S}} \mathbb{s}''. \\ 2. & \varphi(\mathbb{s}) & = \forall \mathbb{s}', \mathbb{s}''.\mathbb{s} \subseteq_{\mathscr{S}} \mathbb{s}' \wedge \mathbb{s}' \subseteq_{\mathscr{S}} \mathbb{s}'' \rightarrow \mathbb{s} \subseteq_{\mathscr{S}} \mathbb{s}''. \\ 3. & \mathbb{s}^{\circ} & = \mathscr{B}(s) \\ 4. & \varphi(\mathbb{s} \cup \mathbb{s}^{\circ}) & = \forall \mathbb{s}', \mathbb{s}''.(\mathbb{s} \cup \mathbb{s}^{\circ}) \subseteq_{\mathscr{S}} \mathbb{s}' \wedge \mathbb{s}' \subseteq_{\mathscr{S}} \mathbb{s}'' \rightarrow (\mathbb{s} \cup \mathbb{s}^{\circ}) \subseteq_{\mathscr{S}} \mathbb{s}''. \end{array} \tag{12}$$

Then :

a.	$\phi(\mathscr{B}_0) \equiv \top$	*By (G1) and (G2)*
b.	$(\mathbb{s} \cup_{\mathscr{S}} \mathbb{s}^\circ) \subseteq_{\mathscr{S}} \mathbb{s}' \wedge \mathbb{s}' \subseteq_{\mathscr{S}} \mathbb{s}'' \rightarrow \mathbb{s} \subseteq_{\mathscr{S}} \mathbb{s}' \wedge \mathbb{s}' \subseteq_{\mathscr{S}} \mathbb{s}''$	*by (G2), (x) and (xx) of Ex. 6 and Taut.*
c.	$(\mathbb{s} \cup_{\mathscr{S}} \mathbb{s}^\circ) \subseteq_{\mathscr{S}} \mathbb{s}' \wedge \mathbb{s}' \subseteq_{\mathscr{S}} \mathbb{s}'' \rightarrow \mathbb{s}^\circ \subseteq_{\mathscr{S}} \mathbb{s}' \wedge \mathbb{s}' \subseteq_{\mathscr{S}} \mathbb{s}''$	*by (G2), (x) and (xx) of Ex. 6 and Taut.*
d.	$(\mathbb{s} \subseteq_{\mathscr{S}} \mathbb{s}') \wedge \mathbb{s}' \subseteq_{\mathscr{S}} \mathbb{s}'' \rightarrow \mathbb{s} \subseteq_{\mathscr{S}} \mathbb{s}''$	*by Ind. Hyp.*
e.	$(\mathbb{s}^\circ \subseteq_{\mathscr{S}} \mathbb{s}') \wedge \mathbb{s}' \subseteq_{\mathscr{S}} \mathbb{s}'' \rightarrow \mathbb{s}^\circ \subseteq_{\mathscr{S}} \mathbb{s}''$	*by Ind. Hyp.* (13)
f.	$(\mathbb{s} \subseteq_{\mathscr{S}} \mathbb{s}') \wedge (\mathbb{s}' \subseteq_{\mathscr{S}} \mathbb{s}'') \wedge (\mathbb{s}^\circ \subseteq_{\mathscr{S}} \mathbb{s}') \rightarrow \mathbb{s}^\circ \subseteq_{\mathscr{S}} \mathbb{s}'' \wedge \mathbb{s} \subseteq_{\mathscr{S}} \mathbb{s}''$	*by d,e and Taut.*
g.	$\mathbb{s} \subseteq_{\mathscr{S}} \mathbb{s}'' \wedge \mathbb{s}^\circ \subseteq_{\mathscr{S}} \mathbb{s}'' \rightarrow (\mathbb{s} \cup_{\mathscr{S}} \mathbb{s}^\circ) \subseteq_{\mathscr{S}} \mathbb{s}''$	*by f, (G2), (x) and (xx) of Ex. 6*
h.	$(\mathbb{s} \cup_{\mathscr{S}} \mathbb{s}^\circ) \subseteq_{\mathscr{S}} \mathbb{s}' \wedge \mathbb{s}' \subseteq_{\mathscr{S}} \mathbb{s}'' \rightarrow \mathbb{s} \cup_{\mathscr{S}} \mathbb{s}^\circ \subseteq_{\mathscr{S}} \mathbb{s}''$	*by b, g and Taut.*
i.	$\varphi(\mathscr{B}_0) \wedge (\forall \mathbb{s}\mathbb{s}' \forall s. \varphi(\mathbb{s}) \wedge \varphi(\mathscr{B}(s)) \rightarrow \varphi(\mathbb{s} \cup_{\mathscr{S}} \mathbb{s}^\circ))$	*by a, d, e, and g*
j.	$\forall \mathbb{t}\, \varphi(\mathbb{t})$	*by i and (G5)*

A precedence relation $\preceq_{\mathscr{S}}$ between two bags of timelines can be defined as follows:

$$\mathbb{s} \preceq_{\mathscr{S}} \mathbb{t} \equiv \forall s\, \exists s'.s \in_{\mathscr{S}} \mathbb{s} \rightarrow s' \in_{\mathscr{S}} \mathbb{t} \wedge s \sqsubseteq s' \wedge \forall s' \exists s.s' \in_{\mathscr{S}} \mathbb{t} \rightarrow s \in_{\mathscr{S}} \mathbb{s} \wedge s' \sqsupseteq s \tag{14}$$

□

The axiomatisation of bags of timelines is sound. Let $Ax_3 = G1\text{-}G5$ and $\mathcal{A}^+ = \Sigma_{=_\nu} \cup Ax_3$ then:

Theorem 3 *$\mathcal{A}^+ \cup \mathcal{D}_{una}$ is satisfiable.*

□

So far we have extended the language of a basic theory of actions in the Situation Calculus to include time, types, a new equality symbol $=_\nu$ and bag of timelines, the final language is thus $\mathcal{L}_{TFSC}$. The extended language in particular includes all the formulas inductively defined using also the following set of atoms which, in turn, can be defined using the symbols $=_{\mathscr{S}}$ and $\in_{\mathscr{S}}$:

Definition 1 *If $\mathbb{s}$ and $\mathbb{t}$ are terms of sort bag of timelines and s is a term of sort situation, then $s \in_{\mathscr{S}} \mathbb{s}$, $\mathbb{s} =_{\mathscr{S}} \mathbb{t}$, $\mathbb{s} =_{\mathscr{S}} \mathbb{t} \cup_{\mathscr{S}} \mathbb{t}'$, $\mathbb{s} =_{\mathscr{S}} \mathbb{t} \cap_{\mathscr{S}} \mathbb{t}'$, $\mathbb{s} =_{\mathscr{S}} \mathbb{t} \setminus_{\mathscr{S}} \mathbb{t}'$, $\mathbb{s} \subseteq_{\mathscr{S}} \mathbb{t}$, $\mathbb{s} \preceq_{\mathscr{S}} \mathbb{t}$ are atoms of the extended language.*

In [65] sets are often implicitly assumed, for example to define sets of actions with concurrent processes. Here the definition of sets of situations through bag of timelines is more involved. Indeed, we shall use them in Section 5 to build macro definitions of temporal compatibilities, from which we shall obtain the temporal network specifying time constraints and temporal relations. Macro definitions will then be reduced to sentences of the TFSC, therefore to prove properties about these sentence we shall often use regression, a central computational mechanism in the Situation Calculus.

We introduce here the theorem ensuring that sentences mentioning bag of timelines are regressable, under analogous restriction conditions given in [61] and we refer the reader to the Appendix, page 54, for the details. Here by a regressable sentence and a k-uniform term we mean, respectively, a sentence and a term that satisfy the conditions specified in [61] and suitably extended to bag of timelines (see the Appendix, Definition 4 and Definition 5, page 55). Let $\mathcal{D}^+$ be a basic action theory with Σ extended to $\mathcal{A}^+$ (see the above Theorem 3), then:

Theorem 4 *Let $\phi(\mathbb{s}_1, \ldots, \mathbb{s}_k)$ be a regressable sentence mentioning terms of sort bag of timelines. There exists a formula $\mathcal{R}(\phi(\mathbb{s}_1, \ldots, \mathbb{s}_k))$ uniform in S_0 such that;*

$$\mathcal{D}^+ \models \mathcal{R}(\phi(\mathbb{s}_1, \ldots, \mathbb{s}_k)) \equiv \phi(\mathbb{s}_1, \ldots, \mathbb{s}_k) \tag{15}$$

4 The system at work: processes in TFSC

For each type H_i, encoding a system component, we assume that there exists a set of processes and a set of fluents describing the behaviours of the component. It follows that also these actions need to be specified for the type $H(i,a)$ of each component i. Processes span the subtree of situations, over a single interval between a start and end action: for each process there are two actions, starting and ending the process, abbreviated by $start_\pi$, meaning *starts process* π and end_π, meaning *ends process* π. To simplify the presentation we shall add to the *start* and *end* actions the type which, in general, given the H and the $=_\nu$ is not needed.

A process is denoted by a fluent $\pi(i,\vec{x},t^-,s)$, where i is for the type and t^- for its start time. Successor state axioms for processes ($\mathcal{D}_\pi$) extend the set $\mathcal{D}_{ss}$ of successor state axioms for fluents and are defined as follows:

$$\pi(i,\vec{x},t^-,do(a,s)) \quad \equiv \quad a = start_\pi(i,\vec{x},t^-) \vee \pi(i,\vec{x},t^-,s) \wedge \forall t.a \neq end_\pi(i,\vec{x},t). \tag{16}$$

For example the process for the component *nav* moving towards θ, can be defined as:

$$move(nav,\theta,t^-,do(a,s)) \equiv a{=}start_{move}(nav,\theta,t^-) \vee move(nav,\theta,t^-,s) \wedge \neg\exists t'.a = end_{move}(nav,\theta,t').$$

As usual (see [65]) a situation is defined to be *executable* as follows:

$$executable(s) \stackrel{def}{=} \forall a,s'.s = d(a,s') \sqsubseteq s{\rightarrow}Poss(a,s') \tag{17}$$

On the other hand, given a set of processes related to a timeline $\mathscr{T}(i,s)$, their distribution on the timeline is controlled by the fluent $Idle(i,s)$ telling whether a process, of type i, is being executed at the situation s. The successor state axiom for *Idle* is defined as follows:

$$\begin{array}{ll} Idle(i,do(a,s)) \equiv & (s = S_0 \wedge H(i,a) \vee \mathscr{T}(i,s) \wedge a{=}_\nu s) \wedge \\ & (\bigvee_{\pi\in\Pi} \exists\vec{x}\, t.a = end_\pi(i,\vec{x},t)) \vee (\bigwedge_{\pi\in\Pi} \neg\exists\vec{x}\, t.a = start_\pi(i,\vec{x},t) \wedge Idle(i,s))\,. \end{array} \tag{18}$$

That is, $Idle(i,s)$ lasts up to the process starting and after its end. We can then break down the processes along a timeline using the preconditions axioms ($\mathcal{D}_{ap}$) as follows:

$$\begin{array}{ll} Poss(start_\pi(i,\vec{x},t),s) & \equiv (s = S_0 \vee s{=}_\nu start_\pi(i,\vec{x},t)) \wedge Idle(i,s) \wedge time(s) \leq t \wedge \Phi_{start}(i,\vec{x},s); \\ Poss(end_\pi(i,\vec{x},t),s) & \equiv (s = S_0 \vee s{=}_\nu end_\pi(i,\vec{x},t)) \wedge \exists t^-\pi(i,\vec{x},t^-,s) \wedge time(s) < t \wedge \Phi_{end}(i,\vec{x},s). \end{array} \tag{19}$$

Here $\Phi_{start}(i,\vec{x},s)$ and $\Phi_{end}(i,\vec{x},t,s)$ are the precondition formulas for the execution of $start_\pi$ and end_π respectively. These can possibly refer to other timelines, hence to other components. We do not further investigate here this possibility, instead we follow the approach in [43], where global constraints like e.g. $Poss(a,s) \rightarrow time(s) \leq time(a)$ are specified by action preconditions of the form $Poss(A(\vec{x},t),s) \equiv \Phi(\vec{x},t,s) \wedge time(s) \leq t$. Indeed, here, $time(s) < t$ is required for the $end_\pi(i,\vec{x},t)$, to filter out durationless processes.

If a process of a component i is already active in S_0 all other processes of the same components cannot be active. This proviso is intuitive, each component of the complex system can execute a process at a time and the component is idling only if none of its processes are active. This requirement is expressed by the following property, for all types i:

$$Idle(i,S_0) \vee \left(\exists\vec{x}.\pi(i,\vec{x},t_0,S_0) \rightarrow \neg Idle(i,S_0) \wedge \forall\vec{y} \bigwedge_{\substack{\pi'\in\Pi \\ \pi\neq\pi'}} \neg\pi'(i,\vec{y},t_0,S_0) \right) \qquad (processes\ \ consistency). \tag{20}$$

If $\pi(i,\vec{x},t_0,S_0)$ holds for some $\vec{x}$ in S_0, then this is the only active process of type i, hence $\neg Idle(i,S_0)$ holds too, because the i-component has an active process and so it is not idling. Note that there is no need to have a complete description of the initial situation $\mathcal{D}_{S_0}$. Let us define $\mathcal{D}_\pi$ to be the set of successor state axioms

for processes, the set of action precondition axioms for processes and the successor state axiom for *Idle*, let $\mathcal{D}_{ss} \cup \mathcal{D}_{ap}$ be the set of successor state axioms and action precondition for fluents, and let $\mathcal{D}_{S_0}$ be the set of formulas uniform in S_0, such that the above requirement (20) is satisfied in $\mathcal{D}_{S_0}$. Let $\mathcal{D}_T$ be the theory formed by $\mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{\pi} \cup \mathcal{D}_{S_0} \cup \mathcal{A}^+ \cup \mathcal{D}_{una}$, then

Theorem 5 *$\mathcal{D}_T$ is satisfiable iff $\mathcal{D}_{S_0} \cup \mathcal{A}^+ \cup \mathcal{D}_{una}$ is.*

□

Given the successor state axioms (16) and (18) along with the preconditions (19) the processes consistency property (20) holds for each executable timeline (see 17). We first show that any executable situation is a timeline. For this we may assume that the action preconditions for fluents (not processes) must be of the form:

$$Poss(A(\vec{x}, t), s) \equiv (s = S_0 \vee s =_{\gamma} A(\vec{x}, t)) \wedge \Phi(A(\vec{x}, t), s) \tag{21}$$

with A any action, possibly different from $start_{\pi}$ and end_{π}.

Proposition 1 *Let $\mathcal{D}_T = \mathcal{D}_{ss} \cup \mathcal{D}_{ap} \cup \mathcal{D}_{\pi} \cup \mathcal{D}_{S_0} \cup \mathcal{A}^+ \cup \mathcal{D}_{una}$, then, any executable situation σ is a timeline.*

□

Using the above result we can state:

Proposition 2 *Let $\mathcal{D}_T$ be as in Theorem 5 such that (20) holds in $\mathcal{D}_{S_0}$, and let σ be an executable situation, then for any process π and type i:*

$$\mathcal{D}_T \models Idle(i, \sigma) \vee \left(\exists \vec{x}\, t.\pi(i, \vec{x}, t, \sigma) \rightarrow \neg Idle(i, \sigma) \wedge \forall \vec{y} \bigwedge_{\pi' \in \Pi}^{\pi \neq \pi'} \neg \pi'(i, \vec{y}, t, \sigma)\right). \tag{22}$$

□

The precondition axioms in (19) compel executability only on timelines. This requirement does not impose that preconditions of actions are unaffected by other timelines as, in fact, they might be specified in the Φ_Q, but simply that there exists a component able to execute an action sequence. This notion is useful for the generation of executable flexible plans. On the other hand, hybrid executability both for processes and fluents, would require to introduce a distinction between: *executability within the component* (based on the preconditions (19)) and *executability within the system*. With two notions of executability at hands one could exploit non-timeline situations to reason about the system. For example a situation like $\sigma = do([start_{go}(nav, pos_1, 1), start_{scan}(pan, 2), end_{scan}(pan, 3), end_{go}(nav, pos_1, 5)], S_0)$ could be used as a system log and exploited to infer properties about the overall system behaviour. Here we derive only the first notion of executability and we do not develop the latter.

5 Temporal Intervals and Constraints

So far we have given the basic formalism to model parallel processes that can be executed on timelines specified by different components. The way these processes interact in terms of time can be expressed by time constraints taken from the classical relations [2, 47, 6] between time intervals, see Figure 3.

Notation: In this section we shall denote process and fluent symbols with uppercase letters as $P, Q, \ldots$ to treat them uniformly, while in the example names of processes are all indicated by lowercase letters. All the defined predicates are macros, hence they are not added to the language and all the fluents appearing on the right hand side of the definition, that is, in the *definiens*, are defined by a successor state axiom. This fact ensures that macros cannot be reduced to state constraints. A temporal interval is denoted by $[t^-, t^+]$. Temporal interval relations **before**, **meets**, **overlaps**, **during**, **starts**, **finishes**, **equals** are denoted by **b**, **m**, **o**, **d**, **s**, **f**, **e** respectively.

We represent the free temporal variables using the notation **t** to indicate that a temporal variable t occurs free; when necessary, we use τ to represent either a temporal variable that occurs free or a ground term instantiating a temporal variable. Further, we introduce the notation $\sigma[\omega]$ ($\mathfrak{s}[\omega]$ for bags of timelines) to explicitly denote a tuple $\omega = \langle \mathbf{t}_1, \mathbf{t}_2 \ldots, \mathbf{t}_n \rangle$ of free temporal variables mentioned in a situation σ(bag of timelines $\mathfrak{s}$).

Example 7 Consider the usual temporal interval relations: **b**, **m**, **o**, **d**, **s**, **f**, **e**, as defined in the *temporal intervals* literature, started in [2]. Let *pan-tilt* and *nav* (for navigation) be two components of the system, with the processes *scan* belonging to the *pan-tilt* component and *stop* belonging to the *nav* component. To express that the process *scan* can be performed while *nav* is stopped we would like to say: *scan* **d** *stop*, this constraint should be encoded in a suitable TFSC formula mentioning the fluents $scan(pan{-}tilt, t, s)$ and $stop(nav, t, s)$.

□

We, thus, begin by defining two predicates *Started* and *Ended* taking as arguments the processes/fluents arguments together with the starting time and the ending time, respectively. For each process/fluent P, these predicates are defined as follows:

$$\begin{aligned} Started_P(i, \vec{x}, t^-, a, s) &\stackrel{def}{=} P(i, \vec{x}, do(a, s)) \wedge \neg P(i, \vec{x}, s) \wedge time(a) = t^- \\ Ended_P(i, \vec{x}, t^+, a, s) &\stackrel{def}{=} P(i, \vec{x}, s) \wedge \neg P(i, \vec{x}, do(a, s)) \wedge time(a) = t^+ \end{aligned} \tag{23}$$

The meaning of $Started_P$ is the following: a process P which does not hold in s is started by action a at time t^- in so becoming *active*. On the other hand, the meaning of $Ended_P$ is: a process P which is currently holding in s is ended by action a at time t^+ in so becoming *elapsed*.

We can now define explicitly the temporal characterisation of a process in a time lapse.

$$\begin{aligned} &Active_P(i, \vec{x}, t^-, do(a, s)) \stackrel{def}{=} \\ &\quad \mathscr{T}(i, do(a, s)) \wedge Started_P(i, \vec{x}, t^-, a, s) \vee Active_P(i, \vec{x}, t^-, s) \wedge \neg \exists t^+ \; Ended(i, \vec{x}, t^+, a, s) \\ &Elapsed(i, \vec{x}, t^-, t^+, do(a, s)) \stackrel{def}{=} \\ &\quad \mathscr{T}(i, do(a, s)) \wedge Elapsed(i, \vec{x}, t^-, t^+, s) \vee Ended(i, \vec{x}, t^+, a, s) \wedge Active_P(i, \vec{x}, t^-, s) \end{aligned} \tag{24}$$

The meaning of *Active* and *Elapsed* is intuitive: a process is active if it started at some time t^- before the current time and it is still holding, while it is elapsed if it was active at some time before but is no more active.

We assume that at time t_0, the time of S_0, there is no record of past processes but there might be active processes just started at time t_0. This is expressed in the following definitions:

$$\begin{aligned} Active_P(i, \vec{x}, t^-, S_0) &\stackrel{def}{=} P(i, \vec{x}, S_0) \wedge time(S_0) = t^- \\ Elapsed_P(i, \vec{x}, t^-, t^+, S_0) &\stackrel{def}{=} \bot \end{aligned} \tag{25}$$

Example 8 *For example, the interval during which the fluent predicate $at(nav, o, x, s)$ lasts (nav is for the navigation component) can be described by $Elapsed_{at}(nav, o, x, t^-, t^+, s)$ and $Active_{at}(nav, o, x, t^-, s)$ described as follows.*

$$\begin{aligned} Elapsed_{at}(nav, o, x, t^-, t^+, do(a, s)) &\stackrel{def}{=} \mathscr{T}(i, do(a, s)) \wedge (Elapsed_{at}(nav, o, x, t^-, t^+, s) \vee \\ &\qquad Ended_{at}(nav, t^+, o, x, a, s) \wedge Active_{at}(nav, o, x, t^-, s)) \, ; \\ Active_{at}(nav, o, x, t^-, do(a, s)) &\stackrel{def}{=} \mathscr{T}(i, do(a, s)) \wedge (Started_{at}(nav, t^-, o, x, a, s) \vee \\ &\qquad Active_{at}(nav, o, x, t^-, s) \wedge \neg(\exists t^+ \; Ended_{at}(nav, t^+, o, x, a, s))) \, . \end{aligned}$$

□

With the aid of $Elapsed_P$ and $Active_P$ we can represent the above interval relations between processes, and fluents, specified in $\mathcal{D}_T$ according to a TFSC formula $\mathscr{F}_{\mathbf{op}}$ suitably built up from a combination of $Elapsed_X$, $Active_X$ (where X denotes the fluent or process they refer to). Let **op** denote an interval relation:

$$P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{op}\ Q(j,\vec{y},t_j^-,t_j^+)[s,s'] \stackrel{def}{=} \mathscr{F}_{\mathbf{op}}(i,j,\vec{x},\vec{y},t_i^-,t_i^+,t_j^-,t_j^+,s,s'). \tag{26}$$

In particular, we focus on the interval relations $\mathbf{op} \in \{\mathbf{b},\mathbf{m},\mathbf{o},\mathbf{d},\mathbf{s},\mathbf{f},\mathbf{e}\}$. Here, $P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{op}\ Q(j,\vec{y},t_j^-,t_j^+)$ is a situation-suppressed expression that represents the interval relation between P and Q independently from the situations' instances, while the expression $[s,s']$ restores the situations in the formula.

In the following example, we show how some of these relations can be represented in TFSC using the form (26).

Example 9 *The interval relations* **m**, **f**, **s**, *and* **d** *can be macro-defined as follows.*

i. Relation $P(\vec{x})\ \mathbf{m}\ Q(\vec{y})$*:*

$$\begin{aligned} P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{m}\ Q(j,\vec{y},t_j^-,t_j^+)[s,s'] \stackrel{def}{=}\ & Elapsed_P(i,\vec{x},t_i^-,t_i^+,s) \rightarrow \\ & \Big((Active_Q(j,\vec{y},t_j^-,s') \vee Elapsed_Q(j,\vec{y},t_i^-,t_j^+,s')) \wedge (t_i^+ = t_j^-)\Big). \end{aligned}$$

$P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{m}\ Q(j,\vec{y},t_j^-,t_j^+)$ *holds over the timelines* s *and* s'*, with* $\mathscr{T}(i,s)$ *and* $\mathscr{T}(j,s')$ *if, whenever* P *ends at* t_i^+*,* Q *starts at* t_j^- *with* $t_i^+ = t_j^-$*.*

ii. Relation $P(\vec{x})\ \mathbf{f}\ Q(\vec{y})$*:*

$$\begin{aligned} P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{f}\ Q(j,\vec{y},t_j^-,t_j^+)[s,s'] \stackrel{def}{=}\ & Elapsed_P(i,\vec{x},t_i^-,t_i^+,s) \rightarrow \\ & \Big(Elapsed_Q(j,\vec{y},t_i^+,t_j^+,s') \wedge (t_i^+ = t_j^+)\Big). \end{aligned}$$

$P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{f}\ Q(j,\vec{y},t_j^-,t_j^+)$ *holds over the timelines* $\mathscr{T}(i,s)$ *and* $\mathscr{T}(j,s')$ *if, whenever* P *ends at* t_i^+*,* Q *ends at* t_j^+ *with* $t_i^+ = t_j^+$*.*

iii. Relation $P(\vec{x})\ \mathbf{s}\ Q(\vec{y})$*:*

$$\begin{aligned} P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{s}\ Q(j,\vec{y},t_j^-,t_j^+)[s,s'] \stackrel{def}{=}\ & (Elapsed_P(i,\vec{x},t_i^-,t_i^+,s) \rightarrow \\ & (Active_Q(j,\vec{y},t_j^-,s') \vee Elapsed_Q(j,\vec{y},t_i^-,t_j^+,s')) \wedge (t_i^- = t_j^-)) \\ & \vee \\ & (Active_P(i,\vec{x},t_i^-,s) \rightarrow \\ & (Active_Q(j,\vec{y},t_j^-,s') \vee Elapsed_Q(j,\vec{y},t_i^-,t_j^+,s')) \wedge (t_i^- = t_j^-)). \end{aligned}$$

$P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{s}\ Q(j,\vec{y},t_j^-,t_j^+)$ *holds over the timelines* $\mathscr{T}(i,s)$ *and* $\mathscr{T}(j,s')$ *if, whenever* P *starts at* t_i^- *with argument* $\vec{x}$ *along* s*, then* Q *starts at* $t_j^- = t_i^-$*, with argument* $\vec{y}$*, along* s'*.*

iv. Relation $P(\vec{x})\ \mathbf{d}\ Q(\vec{y})$*:*

$$\begin{aligned} P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{d}\ Q(j,\vec{y},t_j^-,t_j^+)[s,s'] \stackrel{def}{=}\ & (Elapsed_P(i,\vec{x},t_i^-,t_i^+,s) \rightarrow \\ & (Active_Q(j,\vec{y},t_j^-,s') \vee Elapsed_Q(j,\vec{y},t_i^-,t_j^+,s')) \wedge \\ & (t_j^- \leq t_i^- \wedge t_i^+ \leq t_j^+)) \\ & \vee \\ & (Active_P(i,\vec{x},t_i^-,s) \rightarrow \\ & (Active_Q(j,\vec{y},t_j^-,s') \vee Elapsed_Q(j,\vec{y},t_i^-,t_j^+,s')) \wedge (t_j^- \leq t_i^-)). \end{aligned}$$

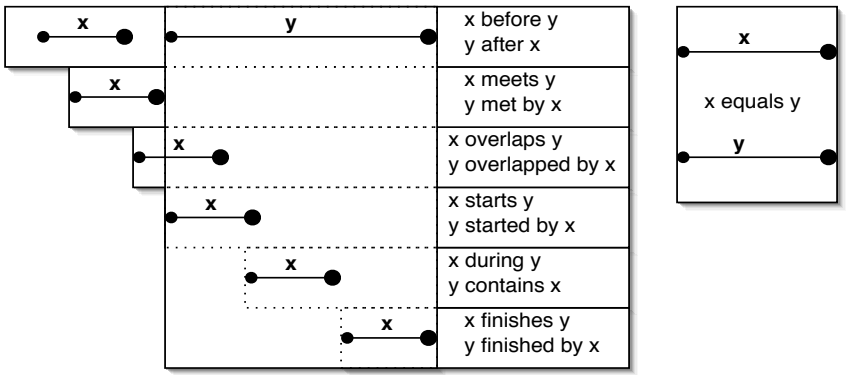

Figure 3: Allen Interval Relations

*$P(i, \vec{x}, t_i^-, t_i^+)$ **d** $Q(j, \vec{y}, t_j^-, t_j^+)$ holds over the timelines $\mathscr{T}(i, s)$ and $\mathscr{T}(j, s')$ if, whenever P starts at t_i^- (and ends at t_i^+) with argument $\vec{x}$ along s, then Q started at $t_j^- \leq t_i^-$ (and ends at $t_i^+ \leq t_i^-$), with argument $\vec{y}$, along s'.*

□

5.1 Temporal Compatibilities: Syntax

Given the abbreviations described in the previous section, as $P(.)$ **op** $Q(.)$, we can construct what we call *compatibilities* that regulate how each process (or fluent) P_i behaves along the timelines. We denote compatibilities by $comp(P_i, LLists)$, where P_i is, indeed, either a process or a fluent symbol, and *LList* is a list of lists, named *List*, of pairs $(\mathbf{op}, P_j)$ composed of an interval relation **op** and a process or fluent symbol P_j. The set of *temporal compatibilities* for a given action theory $\mathcal{D}_T$ in TFSC is denoted T_c, and the syntax for their construction is given below:

$$
\begin{array}{lll}
T_c & ::= & [\,] \mid [comp(P_i, LLists) \mid T_c]; \\
LLists & ::= & [\,] \mid [List \mid LLists]; \\
List & ::= & [\,] \mid [(op, P_j) \mid List].
\end{array}
$$

Example 10 *A set of two compatibilities mentioning the interval relations* $\mathbf{m}, \mathbf{d}, \mathbf{b}, \mathbf{e}$ *binding the interaction between the processes P_1, P_2, P_3 and P_4 is defined as follows:*

$$
\begin{array}{lllll}
T_c = [& comp(P_1, & [[(\mathbf{m}, P_2), & (\mathbf{d}, P_3)], & [(\mathbf{b}, P_4), (\mathbf{e}, P_3)]]), \\
 & comp(P_2, & [[(\mathbf{s}, P_1), & (\mathbf{m}, P_4)], & [(\mathbf{d}, P_3)]]) \quad].
\end{array}
$$

Here the compatibilities state that either (1) the process P_1 meets P_2 and is during P_3, or (2) P_1 is before P_4 and ends P_3; moreover, either (3) P_2 starts P_1 and meets P_4 or (4) it is during P_3.

5.2 Temporal Compatibilities: Semantics

Temporal compatibilities T_c, similarly as in Golog, are not first class citizens of the language, thus their semantics is defined through TFSC macros. The time variables mentioned in the compatibilities (see previous paragraph)

play an important role in their construction because tasks switching might not be defined in advance, that is, constraints might be qualitatively but not metrically fixed. For example, we know that event A has to occur before B, without knowing the precise duration or timing. We show in Section 6 that this flexibility can be managed by temporal variables whose values are constrained by the temporal network associated with the described compatibilities. We recall that we indicate the free temporal variable using the notation $\mathbf{t}$ and the notation $s[\omega]$ to denote the occurrence of free temporal variables $\omega = \langle \mathbf{t}_1, \mathbf{t}_2 \ldots, \mathbf{t}_n \rangle$ in a situation s. For example, the situation $\sigma_1 = do(end_\pi(i, \mathbf{t_1}), do(start_\pi(i, 1.5), S_0))$ represents a process started at time 1.5 with the ending time denoted by the free variable $\mathbf{t_1}$.

The semantics of the above defined temporal compatibilities is specified by a predicate $\mathbb{I}(T_c, \mathbb{s}[\omega])$ denoting the constraints associated with a bag of situations $\mathbb{s}$ given the T_c compatibilities. The predicate $\mathbb{I}(T_c, \mathbb{s}[\omega])$ is obtained by eliciting the time constraints of the variables ω, according to the tail recursive construction illustrated below, using two further predicates $\mathbb{I}_1$ and $\mathbb{I}_2$:

$$\begin{array}{l} \mathbb{I}([\,], \mathbb{s}) \stackrel{def}{=} \top\,. \\ \mathbb{I}([comp(P(i, \vec{x}), LLists) \mid T_c], \mathbb{s}) \stackrel{def}{=} \mathbb{I}_1(comp(P(i, \vec{x}), LLists), \mathbb{s}) \wedge \mathbb{I}(T_c, \mathbb{s})\,. \end{array} \tag{27}$$

Here the induction is defined with respect to $LLists$: if $LLists$ is empty then $\mathbb{I}$ is true; otherwise $\mathbb{I}$ is the conjunction of the predicate $\mathbb{I}_1$ taking as argument, to be expanded, the compatibility $comp(P(i, \vec{x}), LLists)$ and the predicate $\mathbb{I}$ taking as argument the remaining compatibilities T_c. The macro expansion construction proceeds as follows.

$$\begin{array}{l} \mathbb{I}_1(comp(P(i, \vec{x}), [\,]), \mathbb{s}) \stackrel{def}{=} \bot\,. \\ \mathbb{I}_1(comp(P(i, \vec{x}), [List \mid LLists]), \mathbb{s}) \stackrel{def}{=} \mathbb{I}_2(P(i, \vec{x}), List, \mathbb{s}) \vee \mathbb{I}_1(comp(P(i, \vec{x}), LLists), \mathbb{s})\,. \end{array} \tag{28}$$

The $\mathbb{I}_1$ macro denotes the compatibilities of $P(i, \vec{x})$ and it is defined by a disjunction with the $\mathbb{I}_2$ macros described below. Each $\mathbb{I}_2(P(i, \vec{x}), List, \mathbb{s})$ macro collects the set of temporal constraints specified by the compatibility $comp(P(i, \vec{x}), [List])$ over the timelines mentioned in the bag of situations $\mathbb{s}$.

$$\begin{array}{lll} \mathbb{I}_2(P(i, \vec{x}), List, \mathbb{s}) \stackrel{def}{=} \exists s(& s \in \mathbb{s} \wedge \mathscr{T}(i, s) \wedge & \\ & (\bigwedge_{\{(\mathbf{op}, Q(j, \vec{y})) \in List\}} & \exists s'(s' \in \mathbb{s} \wedge \mathscr{T}(j, s') \wedge \\ & & \forall \vec{x}, t_i^-, t_i^+ \exists \vec{y}, t_j^-, t_j^+ (P(i, \vec{x}, t_i^-, t_i^+)\ \mathbf{op}\ Q(j, \vec{y}, t_j^-, t_j^+)[s, s'])))). \end{array} \tag{29}$$

So the predicate $\mathbb{I}_2$ is the bottom of the expansion as it reduces to a conjunction of statements $P(i, \vec{x}, t_i^-, t_i^+)$ **op** $Q(j, \vec{y}, t_j^-, t_j^+)[s, s']$ that we already know how to transform into a formula of TFSC, see Section 5. Note, in particular, that the bag of timelines mentioned in $\mathbb{I}_2$, serves to pick up a pair of situations for each temporal constraint, according to its type i. That is, the expansion of $\mathbb{I}_2(P(i, \vec{x}), List, \mathbb{s})$ says that each element $(\mathbf{op}, Q(j, \vec{y})) \in List$ specifying a temporal relation **op** with the processes $P(i, \vec{x})$, holding over the timeline $s \in \mathbb{s}$, holds over the timeline $s' \in \mathbb{s}$ compatibly with the constraint **op**.

Example 11 *Consider the timelines for the two components* nav*(for navigation) and* eng *(for engine) depicted in Figure 4. The involved compatibilities are represented by the following T_c term*

$$\begin{array}{lll} T_c & = & [\ \ comp(at(nav, x),\ [[(\mathbf{d}\ stop(eng))\ ]\ ]), \\ & & \ \ comp(go(nav, x, x'),\ [[(\mathbf{e}\ run(eng))\ ]\ ])\ \ ], \end{array}$$

*Here T_c states that: $at(nav, x)$ **d** $stop(eng)$, that is, the agent navigation can be at a specified position x only while the engines are stopped. On the other hand the temporal constraint $go(nav, x, x')$ **e** $run(eng)$ tells us that the processes $go(nav, x, x')$ and $running(eng)$ start and stop at the same time.*

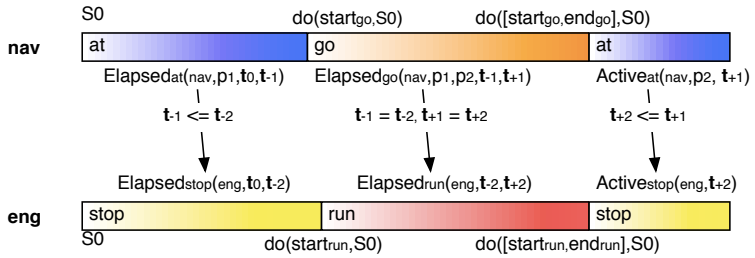

Figure 4: Timelines represented by $\mathbb{s}[\mathbf{t_1^-}, \mathbf{t_1^+}, \mathbf{t_2^-}, \mathbf{t_2^+}] = \mathscr{B}(\{\ do([start_{go}(nav, p_1, p_2, \mathbf{t_1^-}), end_{go}(nav, p_1, p_2, \mathbf{t_1^+})], S_0),$ $do([start_{run}(eng, \mathbf{t_2^-}), end_{run}(eng, \mathbf{t_2^+})], S_0)\})$, and temporal constraints defined by the macro $\mathbb{I}(T_c, \mathbb{s}[\omega])$, with $\omega = \langle \mathbf{t_1^-}, \mathbf{t_1^+}, \mathbf{t_2^-}, \mathbf{t_2^+} \rangle$. Note that each relation *Elapsed* and *Active* labelling the timeline *nav* implies the temporal constraint labelling the arrow.

The timelines in Figure 4 are designated by the following bag of situations:

$$\begin{array}{rcll} \mathbb{s}[\omega] & = & \mathscr{B}(\{ & do([start_{go}(nav, p_1, p_2, \mathbf{t_1^-}), end_{go}(nav, p_1, p_2, \mathbf{t_1^+})], S_0), \\ & & & do([start_{run}(eng, \mathbf{t_2^-}), end_{run}(eng, \mathbf{t_2^+})], S_0)\}), \end{array}$$

where $\omega = \langle \mathbf{t_1^-}, \mathbf{t_1^+}, \mathbf{t_2^-}, \mathbf{t_2^+} \rangle$ are time variables.

To establish the temporal constraints that hold over the timelines in $\mathbb{s}[\omega]$ using the compatibilities T_c (see Figure 5), we can built the predicate $\mathbb{I}(T_c, \mathbb{s}[\omega])$ on $\mathbb{s}$, with time variables ω, according to definition (28), as follows:

$$\begin{array}{rcl} \mathbb{I}(T_c, \mathbb{s}) & \stackrel{def}{=} & \mathbb{I}_1(comp(at(nav, x),\ [[(\mathbf{d}\ stop(eng))\ ]\ ]), \mathbb{s}) \wedge \\ & & \mathbb{I}_1(comp(go(nav, x, x'),\ [[(\mathbf{e}\ run(eng))\ ]\ ]), \mathbb{s}), \end{array}$$

that is, macro expanding $\mathbb{I}_1$ in terms of $\mathbb{I}_2$ (by (28)),

$$\mathbb{I}(T_c, \mathbb{s}) \stackrel{def}{=} \mathbb{I}_2(at(nav, x),\ [(\mathbf{d}\ stop(eng))], \mathbb{s}) \wedge \mathbb{I}_2(go(nav, x, x'), [(\mathbf{e}\ run(eng))], \mathbb{s}),$$

According to the above $\mathbb{I}_2$ expansion, we obtain:

$$\begin{array}{l} \mathbb{I}_2(at(nav, x),\ [(\mathbf{d}\ stop(eng))], \mathbb{s}) \stackrel{def}{=} \exists s(s \in \mathbb{s} \wedge \mathscr{T}(nav, s) \wedge \\ \quad \exists s'(s' \in \mathbb{s} \wedge \mathscr{T}(eng, s') \wedge \forall x, t^1, t^2 \exists t^3, t^4 (at(nav, x, t^1, t^2)\ \mathbf{d}\ stop(eng, t^3, t^4)[s, s'])))\ ; \\ \mathbb{I}_2(go(nav, x, x'), [(\mathbf{e}\ run(nav))], \tau_2, \mathbb{s}) \stackrel{def}{=} \exists s(s \in \mathbb{s} \wedge \mathscr{T}(nav, s) \wedge \\ \quad \exists s'(s' \in \mathbb{s} \wedge \mathscr{T}(nav, s') \wedge \forall x, y, t^1, t^2 \exists t^3, t^4 (go(nav, x, y, t^1, t^2)\ \mathbf{e}\ run(eng, t^3, t^4)[s, s'])). \end{array}$$

Collecting everything together we obtain that $\mathbb{I}(T_c, \mathbb{s})$ reduces to the following TFSC formula denoting the tem-

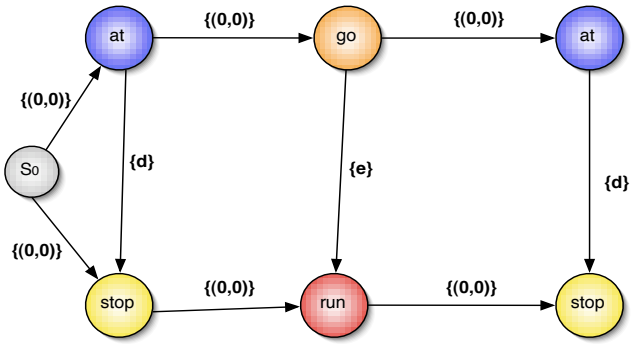

Figure 5: Temporal constraint network associated with the compatibilities $T_c = [comp(at(nav, x), [[(\mathbf{d}\ stop(eng))\ ]\ ]), comp(go(nav, x, x'), [[(\mathbf{e}\ run(eng))]]\,)]$ and the timelines represented by $\mathbb{s}[\mathbf{t_1^-}, \mathbf{t_1^+}, \mathbf{t_2^-}, \mathbf{t_2^+}] = \mathscr{B}(\{\ do([start_{go}(nav, p_1, p_2, \mathbf{t_1^-}), end_{go}(nav, p_1, p_2, \mathbf{t_1^+})], S_0), do([start_{run}(eng, \mathbf{t_2^-}), end_{run}(eng, \mathbf{t_2^+})], S_0)\}).$

poral constraints relative to $\mathbb{s}$ *and* T_c *(see Figure 4).*

$$
\begin{aligned}
\mathbb{I}(T_c, \mathbb{s}) \stackrel{def}{=}\ & \exists s(s \in \mathbb{s} \wedge \mathscr{T}(nav, s) \wedge \\
& \exists s'(s' \in \mathbb{s} \wedge \mathscr{T}(eng, s') \wedge \forall x, t^1, t^2 \exists t^3, t^4(\\
& \quad Elapsed_{at}(nav, x, t^1, t^2, s) \rightarrow \\
& \qquad (Active_{stop}(eng, y, t^3, s') \vee Elapsed_{stop}(eng, y, t^3, t^4, s')) \wedge (t^1 = t^3 \wedge t^2 = t^4) \vee \\
& \quad Active_{at}(nav, x, t^1, s) \rightarrow \\
& \qquad (Active_{stop}(eng, y, t^3, s') \vee Elapsed_{stop}(eng, y, t^3, t^4, s')) \wedge (t^1 = t^3))) \wedge \\
& \exists s'(s' \in \mathbb{s} \wedge \mathscr{T}(eng, s') \wedge \forall x, t^1, t^2 \exists t^3, t^4(\\
& \quad Elapsed_{go}(nav, x, t^1, t^2, s) \rightarrow \\
& \qquad (Active_{run}(eng, y, t^3, s') \vee Elapsed_{run}(eng, y, t^3, t^4, s')) \wedge (t^3 \leq t^1 \wedge t^2 \leq t^4) \vee \\
& \quad Active_{go}(nav, x, t^1, s) \rightarrow \\
& \qquad (Active_{run}(eng, y, t^3, s') \vee Elapsed_{run}(eng, y, t^3, t^4, s')) \wedge (t^3 \leq t^1)))).
\end{aligned}
$$

Discussion In the TFSC framework, parallel timelines are associated with their sets of processes and fluents, therefore, processes and fluents belonging to different timelines influence each other mainly through temporal constraints. This proves that loosely coupled components and temporal constraints are necessary to allow and capture flexible temporal behaviours. Indeed, in the TFSC framework, starting and ending points of the processes are not fixed and events associated with different components are not sequenced, hence only temporal constraints can be forced. This approach allows us to (1) represent temporally flexible behaviours in their generality (2) keep the simple structure of the basic theory of actions for each component. Notice also that, fluents belonging to two separated components can be easily related through temporal compatibilities, e.g. specifying $P(i, \vec{x}, t_1^-, t_1^+)\ \mathbf{e}\ Q(j, \vec{y}, t_2^-, t_2^+)$, implies that whenever $P(\cdot)$ is on timeline i, $Q(\cdot)$ must be on timeline j. Furthermore, since the temporal compatibilities are expressed by a TFSC formula, it is possible to infer properties associated with parallel timelines and their constraints. E.g. it is possible to ask whether $\mathcal{D}_T \models \exists t, p[at(nav, p, t, \sigma_2) \wedge stop(eng, t, \sigma_1) \wedge \sigma_1 \in \mathbb{s} \wedge \sigma_2 \in \mathbb{s}] \wedge \mathbb{I}(T_c, \mathbb{s})$, with $\mathbb{s}$ defined as specified in Example 11. This formula combines parallel processes and temporal constraints. In the next sections, we shall show how it is possible to decouple logical and temporal reasoning in TFSC.

6 Mapping TFSC to Temporal Constraint Networks

In this section, we introduce the construction of a transformation from the compatibility formula $\mathbb{I}(T_c, \mathfrak{s})$, having a macro definition (see (26)) within a background theory $\mathcal{D}_T$ of TFSC, into a *general temporal constraint networks* (TCN) [47]. More specifically, we show that, given a domain theory $\mathcal{D}_T$, a set of temporal compatibilities T_c, and a bag of timelines $\mathfrak{s}[\omega]$, it is possible to build a temporal network as a disjunction of conjunctions of algebraic relations $\mu_{\mathbf{op}}$ over time.

6.1 Temporal Constraint Network

A Temporal Constraint Network (TCN) is a formal structure for handling metric information, the general concept was early introduced by Dechter, Meiri and Pearl in [13] and then further extended in [14] and in [47] to handle both quantitative and qualitative information. Temporal knowledge represented by propositions, can be associated with intervals, and relationships between events timing can be represented by constraints. For example the statement *the robot was close to the door* **before** *it could see it, but it was still there* **after** *it had processed the images*, can be represented as:

$$(a) \quad closeTo(r, d, t_1^-, t_1^+)\ \mathbf{b}\ see(r, d, t_2^-, t_2^+) \quad (b) \quad closeTo(r, d, t_1^-, t_1^+)\ \mathbf{a}\ scan(r, d, t_2^-, t_2^+)$$

A TCN offers a simple representation schema for temporal statements, exploiting a temporal algebra of relations that can be expressed by a *direct constraint graph*, where each node is labelled by a an event associated with a temporal interval, and directed edges between nodes denote the temporal constraints.

Essentially a TCN involves a set of variables $\{t_1^-, t_1^+ \ldots, t_n^-, t_n^+\}$, with time intervals $[t^-, t^+]$ representing the duration of specific events (e.g. $closeTo$), and a set $\mathcal{R}$ of binary constraints coming from the 13 possible relationships that can be stated between any pair of intervals [2], these are illustrated in Figure 3. Note that constraints can be expressed disjunctively, for example if we consider the events $at(r, P_1)$ and $moveTo(r, P_2)$, then in the TCN we could express the statement $at(r, P_1)$ $\{\mathbf{m}, \mathbf{s}\}$ $moveTo(r, P_2)$, saying that the event $at(r, P_1)$ can either start or meet the event $moveTo(r, P_2)$.

According to the underlying temporal algebra, $TCNs$ can express different forms of reasoning; among the most well known are the Point Algebra [78], and the metric point algebra [14], an extensive overview can be found in [6].

Let $\{t_1^-, t_1^+, \ldots, t_n^-, t_n^+\}, n \in N$ be a set of time variables, where each couple of variables t_i^-, t_i^+ denotes an interval $[t_i^-, t_i^+]$ possibly associated with some event; let TCN involve a set of binary constraints $\mathcal{R} = \{\mathbf{op}_1, \ldots, \mathbf{op}_m\}$, $m \in M$. The temporal constraint network TCN represented with a labelled direct graph can be described using conjunctions and disjunctions of constraints as follows:

$$\bigvee_{z \in Z} \bigwedge_{i,j \in J_z} [t_{i,z}^-, t_{i,z}^+]\ \mathbf{op}_{i,z}\ [t_{j,z}^-, t_{j,z}^+]. \tag{30}$$

The assignment $V = \{\langle v_i^-, v_i^+ \rangle \mid v_i^+(t_i^+) = s_i \text{ and } v_i^-(t_i^-) = e_i \text{ with } s_i, e_i \in \mathbb{R}^+, s_i < e_i\}$ to the variables is called a solution if it satisfies all the constraints in $\mathcal{R}$, defining the TCN. The network is consistent if at least one solution exists (see [14]). A classification of complexity for satisfiability problems (specifically for the Allen's interval algebra), has been given in [35], following previous results of [52].

6.2 Mapping compatibilities to temporal constraints

Consider the macro definition (26) and the definitions of the temporal operators $\{\mathbf{m}, \mathbf{b}, \mathbf{f}, \mathbf{d}, \mathbf{s}, \mathbf{e}\}$ as given in Example 9. We have that:

$$P(i, \vec{x}, t_i^-, t_i^+)\ \mathbf{op}\ Q(j, \vec{y}, t_j^-, t_j^+)[s, s'] \stackrel{def}{=} \mathscr{F}_{\mathbf{op}}(i, j, \vec{x}, \vec{y}, t_i^-, t_i^+, t_j^-, t_j^+, s, s')$$

where $\mathscr{F}_{\mathbf{op}}(\cdot)$ is a formula of $\mathcal{L}_{TFSC}$ (*definiens*) while $P(\cdot)$ **op** $Q(\cdot)$ is a macro of the pseudo language (*definiendum*). Clearly, by $\mathcal{M}, v \models P(\cdot)$ **op** $Q(\cdot)$ we mean $\mathcal{M}, v \models \mathscr{F}_{\mathbf{op}}(\cdot)$. Furthermore, we can note that each **op** can be given an algebraic interpretation $\gamma_{\mathbf{op}}$ of a temporal constraint á la Allen as follows. Let $\mathbf{op} \in \{\mathbf{b}, \mathbf{m}, \mathbf{o}, \mathbf{d}, \mathbf{s}, \mathbf{f}, \mathbf{e}\}$, there is an algebraic relation interpreting **op**, say $\gamma_{\mathbf{op}}$, defined as follows:

$$\begin{array}{ll}
\gamma_{\mathbf{b}}(t_i^-, t_i^+, t_j^-, t_j^+) \stackrel{def}{=} (t_i^+ \leq t_j^-) & \gamma_{\mathbf{m}}(t_i^-, t_i^+, t_j^-, t_j^+) \stackrel{def}{=} (t_i^+ = t_j^-) \\
\gamma_{\mathbf{o}}(t_i^-, t_i^+, t_j^-, t_j^+) \stackrel{def}{=} (t_i^- \leq t_j^- \wedge t_i^+ \leq t_j^+) & \gamma_{\mathbf{d}}(t_i^-, t_i^+, t_j^-, t_j^+) \stackrel{def}{=} (t_j^- \leq t_i^- \wedge t_i^+ \leq t_j^+) \\
\gamma_{\mathbf{s}}(t_i^-, t_i^+, t_j^-, t_j^+) \stackrel{def}{=} (t_i^- = t_j^-) & \gamma_{\mathbf{f}}(t_i^-, t_i^+, t_j^-, t_j^+) \stackrel{def}{=} (t_i^+ = t_j^+) \\
\gamma_{\mathbf{e}}(t_i^-, t_i^+, t_j^-, t_j^+) \stackrel{def}{=} (t_i^- = t_j^- \wedge t_i^+ = t_j^+) &
\end{array} \tag{31}$$

Within the TFSC approach, the *definiens* $\mathscr{F}_{\mathbf{op}}(\cdot)$ is interpreted into structures of the TFSC, letting the assignments to temporal variables ω to freely vary on these structures. However we shall show that the TCN, that we obtain from the predicate $\mathbb{I}(T_c, \mathbb{s}[\omega])$, will make it possible to suitably specify these variables values.

The following theorem states that the predicate $\mathbb{I}(T_c, \mathbb{s}[\omega])$ can be transformed into a normal form, given a suitable indexing of the time variables with respect to the processes and the interval relations **op**.

Theorem 6 *Let $\mathbb{s}[\omega]$ be a bag of timelines mentioning a set of timelines $\{\sigma_1, \ldots, \sigma_n\}$, where each σ_i is a timeline term and where ω collects all the free variables in $\mathbb{s}$.*

Then the predicate $\mathbb{I}(T_c, \mathbb{s}[\omega])$ can be reduced to the following form:

$$\bigvee_{z \in Z} \bigwedge_{\langle q_1, q_2, q_3, q_4 \rangle \in J_z} \forall \vec{x} \exists \vec{y}. P_{z,q_1}(i_{z,q_1}, \vec{x}, \tau^-_{z,q_2}, \tau^+_{z,q_2})\ \mathbf{op}_{z,q_3}\ Q_{z,q_3}(j_{z,q_3}, \vec{y}, \tau^-_{z,q_4}, \tau^+_{z,q_4})[\sigma_{i_{z,q_1}}, \sigma_{j_{z,q_3}}]. \tag{32}$$

Here, Z and J_z are finite sets of indexes and the $\tau_{i,j}$s are either free variables or ground terms mentioned in ω.

□

Theorem 6 allows to eliminate temporal quantifiers in $\mathbb{I}(T_c, \mathbb{s}[\omega])$ obtaining a normal form where temporal terms $\tau_{i,j}$ can be are either temporal variables or ground terms. Here, the index z ranges on the disjunctions, while the other indexes $q_1, \ldots, q_4$ range on the possible conjuncts P_{z,q_1}, on the associated temporal variables τ_{z,q_2}, on the relations $\mathbf{op}_{z,q_3}$ Q_{z,q_3}, and on their variables τ_{z,q_4}.

To put in evidence the interval relations, when i, j, σ_i, σ_j can be extrapolated from the context, we use the abbreviation $\varphi_{\mathbf{op}}(\tau_k^-, \tau_k^+, \tau_g^-, \tau_g^+)$ to denote the interval relation:

$$\forall \vec{x} \exists \vec{y} P(i, \vec{x}, \tau_k^-, \tau_k^+)\ \mathbf{op}\ Q(j, \vec{y}, \tau_g^-, \tau_g^+)[\sigma_i, \sigma_j]. \tag{33}$$

Now, given the disjunction of conjunctions of interval relations as defined in (32), we need to make explicit the underlying algebraic relations. The algebraic interval relations associated with $\varphi_{\mathbf{op}}(\tau_k^-, \tau_k^+, \tau_g^-, \tau_g^+)$ depends also on the associated domain theory $\mathcal{D}_T$. In fact, the left hand side $P(\cdot)$ of the interval relation $P(\cdot)$ **op** $Q(\cdot)$ works as the enabler of the interval constraint: if no process (or fluent) $P(\cdot)$ is either active or elapsed along the timelines, then the associated interval relation **op** is not applicable; otherwise, if $P(\cdot)$ is active or elapsed, the algebraic relation for **op** depends whether $P(\cdot)$ is still active or is elapsed. More precisely, we distinguish the following three cases:

$$\begin{array}{ll}
E_P: & \mathcal{D}_T \models \exists \vec{x}, t^-, t^+ Elapsed_P(i, \vec{x}, t^-, t^+, \sigma_i) \wedge \sigma_i \in \mathbb{s}, \\
A_P: & \mathcal{D}_T \models \exists \vec{x}, t^- Active_P(\vec{x}, t^-, \sigma_i) \wedge \sigma_i \in \mathbb{s}, \\
N_P: & \text{neither } E_P \text{ nor } A_P \text{ hold.}
\end{array} \tag{34}$$

Given these cases, we can introduce the algebric relation $\mu_{\mathbf{op}}(t_k^-, t_k^+, t_g^-, t_g^+)$ associated with $\varphi_{\mathbf{op}}(t_k^-, t_k^+, t_g^-, t_g^+)$ as follows:

$$\begin{array}{ll}
m_1 & \text{If } N_P \text{ holds then, for the given } \mathbf{op}, \text{ no temporal constraint is imposed;} \\
m_2 & \text{if } E_P \text{ holds then, for the given } \mathbf{op}, \mu_{\mathbf{op}} = \gamma_{\mathbf{op}}; \\
m_3 & \text{if } A_P \text{ holds then we have that, for } \mathbf{op} \in \{\mathbf{m}, \mathbf{f}\} \text{ no temporal constraint is imposed, as for the remaining cases:}
\end{array} \tag{35}$$

$$\mu_{\mathbf{b}} = \gamma_b, \quad \mu_{\mathbf{e}} = (t_k^- = t_g^-), \quad \mu_{\mathbf{o}} = (t_k^- \leq t_g^-), \quad \mu_{\mathbf{d}} = (t_g^- \leq t_k^-).$$

The following theorem shows the relation between $\mu_{\mathbf{op}}$ and $\varphi_{\mathbf{op}}$.

Theorem 7 *Let $\mu_{\mathbf{op}}$ be any of the algebraic interval relations defined above, and let $\mathcal{M} = (D, I)$ be a structure of $\mathcal{L}_{TFSC}$ such that $\mathcal{M}$ is a model of $\mathcal{D}_T$ and suppose that for some assignment v to the free temporal variables the following holds:*

(*i*) $\mathcal{M}, v \models \forall\vec{x}\exists\vec{y}.P(i, \vec{x}, \tau_p^-, \tau_p^+)$ **op** $Q(j, \vec{y}, \tau_q^-, \tau_q^+)[\sigma_i, \sigma_j]$,
(*ii*) $\mathcal{M}, v \models \exists\vec{x}\,Elapsed_P(i, \vec{x}, \tau_p^-, \tau_p^+, \sigma_i)$ *or* $\mathcal{M}, v \models \exists\vec{x}\,Active_P(\vec{x}, \tau_p^-, \sigma_i)$.

Here σ_i, σ_j are ground timelines of type i and j, with $\tau_p^{+(I,v)} = d_p^+$, $\tau_p^{-(I,v)} = d_p^-$, $\tau_q^{+(I,v)} = d_q^+$, $\tau_q^{-(I,v)} = d_q^-$ with d_p^+, d_p^-, d_q^-, d_q^+ elements of the domain D canonically interpreted in $\mathbb{R}^+$.
Then, the algebraic relation $\mu_{\mathbf{op}}$ holds on $\langle d_p^-, d_p^+, d_q^-, d_q^+\rangle$.

□

The theorem says that, given the conditions (i) and (ii) for a temporal relation $\forall\vec{x}\exists\vec{y}P(\cdot)$ **op** $Q(\cdot)$, the algebraic relation $\mu_{\mathbf{op}}$ holds on the same time values. Note that, the condition (i) of Theorem 7 states that the temporal relation $\forall\vec{x}\exists\vec{y}P(\cdot)$ **op** $Q(\cdot)$ is consistent with respect to the theory $\mathcal{D}_T$ and the bag of timelines $\mathbb{s}$. Instead, the conditions (ii) play the role of the conditions m_2 and m_3.

6.3 Compatibilities without logical constraints

We now want to introduce a notion of consistency which depends only on the logical structure independently of the temporal constraints. Considering again the macro definition (26) and the definitions of the temporal operators as given in Example 9, by $\{P(i, \vec{x}, \tau_k^-, \tau_k^+)$ **op** $Q(j, \vec{y}, \tau_g^-, \tau_g^+)[\sigma_i, \sigma_j]\}_\Lambda$ we indicate the formula obtained from $P(i, \vec{x}, \tau_k^-, \tau_k^+)$ **op** $Q(j, \vec{y}, \tau_g^-, \tau_g^+)[\sigma_i, \sigma_j]$ excluding its temporal constraints. For example, for **op** = **m**

$$\begin{aligned}\{P(i, \vec{x}, t_i^-, t_i^+)\ \mathbf{m}\ Q(j, \vec{y}, t_j^-, t_j^+)[s, s']\}_\Lambda \quad = \quad & Elapsed_P(i, \vec{x}, t_i^-, t_i^+, s) \rightarrow \\ & ((Active_Q(j, \vec{y}, t_j^-, s') \vee Elapsed_Q(j, \vec{y}, t_i^-, t_j^+, s')).\end{aligned}$$

We can now introduce the notion of *partially consistent* interval relation.

Definition 2 *Given a TFSC theory $\mathcal{D}_T$ and a bag of timelines $\mathbb{s}$, the interval relation $P(i, \vec{x}, \tau_k^-, \tau_k^+)$* **op** *$Q(j, \vec{y}, \tau_g^-, \tau_g^+)[\sigma_i, \sigma_j]$ is* partially consistent *with respect to $\mathcal{D}_T$ if there exists a model $\mathcal{M}$ of $\mathcal{D}_T$ and an assignment v to the free temporal variables such that*

$$\mathcal{M}, v \models \{P(i, \vec{x}, \tau_k^-, \tau_k^+)\ \mathbf{op}\ Q(j, \vec{y}, \tau_g^-, \tau_g^+)[\sigma_i, \sigma_j]\}_\Lambda$$

□

This notion allows us to separate the temporal and logical structure associated to the compatibilities and will be exploited in the construction of the network illustrated in the next subsection. We are now ready, reversing the process and using the results of Theorem 6 and Theorem 7, to show how from the compatibility constraint predicate $\mathbb{I}(T_c, \mathbb{s}[\omega])$ the temporal constraint network can be built.

6.4 Network Construction

The network construction proceeds as follows.

Temporal Network	
Temporal relations	$\mu_{\mathbf{op}}(\tau_p^-, \tau_p^+, \tau_q^-, \tau_q^+)$
Temporal constraint network	$\bigvee \bigwedge \mu_{\mathbf{op}}(\tau_p^-, \tau_p^+, \tau_q^-, \tau_q^+)$
Assignment V solution of the network	$V = \{\langle v_i^-, v_i^+\rangle \mid v_i^+(t_i^+) = s_i, v_i^-(t_i^-) = e_i \text{ with } s_i, e_i \in \mathbb{R}^+, s_i < e_i\}$
Compatibilities and constraints in TFSC	
Temporal variables	$\omega{=}\langle \mathbf{t}_1^-, \mathbf{t}_1^+, \ldots \mathbf{t}_n^-, \mathbf{t}_n^+\rangle, n \in N$
Compatibility term	T_c
Constraint formula for $\mathbb{s}$ with free variables in ω	$\mathbb{I}(T_c, \mathbb{s}[\omega])$
Between TFSC and TCN mapping	
Mapping Temporal Constraints	$\zeta : (\mathcal{D}_T, T_c, \mathbb{s}[\omega]) \rightarrow TCN$
Mapping Ordering Constraints	$Ord : \mathbb{s}[\omega] \rightarrow TCN$

Table 2: Notation for TFSC and TCN mapping.

(a) Transform the predicate $\mathbb{I}(T_c, \mathbb{s}[\omega])$ into a disjunctive normal form, as indicated in Theorem 6, obtaining the form (32):

$$\bigvee_{z\in Z} \bigwedge_{\langle q_1,q_2,q_3,q_4\rangle\in J_z} \forall\vec{x}\exists\vec{y}.P_{z,q_1}(i_{z,q_1}, \vec{x}, \tau_{z,q_2}^-, \tau_{z,q_2}^+)\ \mathbf{op}_{z,q_3}\ Q_{z,q_3}(j_{z,q_3}, \vec{y}, \tau_{z,q_4}^-, \tau_{z,q_4}^+)[\sigma_{i_{z,q_1}}, \sigma_{j_{z,q_3}}].$$

(b) Let $\tau_{1,1}^-, \tau_{1,1}^+, \ldots, \tau_{n,m}^-, \tau_{n,m}^+$ be time variables or instances thereof mentioned in each of the conjuncts $\forall\vec{x}\exists\vec{y}.P(\cdot)\ \mathbf{op}\ Q(\cdot)$ as in (32).

(c) For each conjunct

$$\forall\vec{x}\exists\vec{y}.P_{z,q_1}(i_{z,q_1}, \vec{x}, \tau_{z,q_2}^-, \tau_{z,q_2}^+)\ \mathbf{op}_{z,q_3}\ Q_{z,q_3}(j_{z,q_3}, \vec{y}, \tau_{z,q_4}^-, \tau_{z,q_4}^+)[\sigma_{i_{z,q_1}}, \sigma_{j_{z,q_3}}],$$

if this is *partially consistent* with $\mathcal{D}_T$ then, given the E_P, A_P, and N_P as specified in (34), we can define the corresponding $\mu^{q_1}_{\mathbf{op}_{z,q_3}}$ as specified by the rules (m_1)-(m_3) in (35). Otherwise, if not *partially consistent* in $\mathcal{D}_T$, the associated index z is collected in a set of indexes Z^*, that is, $z \in Z^*$.

(d) From the above disjunctive normal form (32), we can obtain the algebraic relations:

$$\bigvee_{z\in Z'} \bigwedge_{(q_1,q_2,q_3,q_4)\in J_z} \mu^{q_1}_{\mathbf{op}_{z,q_3}}(\tau_{z,q_2}^-, \tau_{z,q_2}^+, \tau_{z,q_4}^-, \tau_{z,q_4}^+).$$

Here, Z' is for $Z \setminus Z^*$ and collects only the indexes not excluded at the step (c) (i.e. Z' collects only consistent disjuncts).

Here, $\mu^{q_1}_{\mathbf{op}_{z,q_3}}$ is the algebraic temporal relation indexed by z, q_3 - as the $\mathbf{op}_{z,q_3}$ in the form (32) - and q_1 because depending on P_{z,q_1} according to the rules (m_1)-(m_3). That is, given the domain theory $\mathcal{D}_T$ and the bag of timelines $\mathbb{s}$, the above algebraic relation can be obtained from the form (32) once we substitute each conjunct $\forall\vec{x}\exists\vec{y}.P(\cdot)\ \mathbf{op}\ Q_{z,j}(\cdot)$ as specified in the step (c). $\bigvee \bigwedge \mu^{q_1}_{\mathbf{op}_{z,q_3}}(\cdot,\cdot,\cdot,\cdot)$ is, indeed, the temporal constraint network implicitly represented by $\mathbb{I}(T_c, \mathbb{s}[\omega])$ given $\mathcal{D}_T$. In other words, this network is a labelled direct graph which can be described using conjunctions and disjunctions of temporal constraints. Therefore, following the notation introduced in Section 6.1, we can denote this temporal network as $\zeta(\mathcal{D}_T, T_c, \mathbb{s}[\omega])$ because it depends on $\mathcal{D}_T$, T_c, and $\mathbb{s}[\omega]$.

Notice that in the temporal network $\zeta(\mathcal{D}_T, T_c, \mathbb{s}[\omega])$ obtained by the (a)-(d) transformation, the time ordering constraints, enforced by the situations mentioned in the bag of timelines $\mathbb{s}[\omega]$, are not made explicit, although

they hold in force of Lemma 2, (see Section 3). Therefore, to obtain the complete set of time constraints implicit in $\mathfrak{s}[\omega]$, given T_c and $\mathcal{D}_T$, we have to consider also the temporal ordering defined by the structure of the bag of timelines $\mathfrak{s}[\omega]$. That is, for any situation $\sigma[\omega'] \in \mathfrak{s}[\omega]$ (where ω' are the variables mentioned in σ), if $\sigma_1[\omega_1] \sqsubseteq \sigma_2[\omega_2] \sqsubseteq \sigma[\omega']$, then $time(\sigma_1[\omega_1]) \leq time(\sigma_2[\omega_2]) \leq time(\sigma[\omega'])$. Given the time variables $\mathbf{t}_1^-$, $\mathbf{t}_1^+, \ldots, \mathbf{t}_m^-$, $\mathbf{t}_m^+$ used for the time variables ω, we call this set of ordering constraints $Ord(\mathfrak{s}[\omega])$. For example, considering again the bag of timelines depicted in Figure 4, $Ord(\mathfrak{s}[\omega]) = (\mathbf{t}_0 \leq \mathbf{t}_1^+ \wedge \mathbf{t}_1^- \leq \mathbf{t}_1^+) \wedge (\mathbf{t}_0 \leq \mathbf{t}_2^- \wedge \mathbf{t}_2^- \leq \mathbf{t}_2^+)$ collecting the ordering constraints associated with the two timelines in $\mathfrak{s}[\omega]$.

Corollary 2 *Let $\zeta(\mathcal{D}_T, T_c, \mathfrak{s}[\omega])$ be a temporal constraint network obtained by $\mathbb{I}(T_c, \mathfrak{s}[\omega])$, according to steps (a-d) of the network construction and let $\mathcal{M}$ be a structure of TSFC which is a model for $\mathcal{D}_T$.*

Let $V = \{\langle v_i^-, v_i^+ \rangle \mid v_i^+(t_i^+) = s_i$ and $v_i^-(t_i^-) = e_i$ with $s_i, e_i \in \mathbb{R}^+, s_i < e_i\}$ be an assignment to the time variables.

Then V is a solution for the network iff for any assignment v to the free temporal variables of $\mathfrak{s}[\omega]$, which is like V, $\mathcal{M}, v \models \mathbb{I}(T_c, \mathfrak{s}[\omega])$.

□

The corollary says that we can use the temporal constraint network as a service for the theory of actions to fix the temporal constraints between processes and fluents. Given the compatibility constraints and the *Ord* ordering constraints introduced above, we can express with $network(\mathcal{D}_T, T_c, \mathfrak{s}[\omega])$ the conjunction of the temporal constraint network and the ordering constraints as follows:

$$network(\mathcal{D}_T, T_c, \mathfrak{s}[\omega]) = \zeta(\mathcal{D}_T, T_c, \mathfrak{s}[\omega]) \wedge Ord(\mathfrak{s}[\omega]). \tag{36}$$

We shall not discuss in this paper methods of simplification of the constraints nor for computing a satisfiable set of time values for a temporal network.

7 Flexible High Level Programming in TFGolog

In this section, we introduce the syntax and the semantics of a TFGolog interpreter that can be used to generate a temporal constraint network and the related flexible temporal plan.

7.1 TFGolog Syntax

Given the extended action theory presented above, the following constructs inductively build Golog programs:

1. *Primitive action:* α.
2. *Nondeterministic choice:* $\alpha|\beta$. Do α or β.
3. *Test action:* $\phi?$. Test if ϕ is true in the current bag of timelines.
4. *Nondeterministic argument choice:* choose $\vec{x}$ for $p(\vec{x})$.
5. *Action sequence:* $p_1; p_2$. Do p_1 followed by p_2.
6. *Partial order action choice:* $p_1 \prec p_2$. Do p_1 before p_2.
7. *Parallel execution:* $p_1 \| p_2$. Do p_1 concurrently with p_2.
8. *Conditionals:* **if** ϕ **then** p_1 **else** p_2.
9. *Nondeterministic iteration:* p^*. Do p n times, with $n \geq 0$.

10. *While loops:* **while** ϕ **do** p_1.

11. *Procedures, including recursion.*

Hence, compared to Golog, here we also have the parallel execution and partial order operator that can be defined over parallel timelines.

Example 12 *Considering again the two components* nav *(for navigation) and* eng *(for engine), depicted in Figure 4, a possible TFGolog program encoding the robot task approaching position pos, within the time interval d, can be written as follows:*

$$\begin{array}{l}\textbf{proc}(approach(d, pos),\ \pi(t_1, \pi(t_2, \pi(t_3,\\ \qquad [\pi(x,\ start_{go}(nav, x, pos, t_1))\ \prec\ (at(nav,\ pos)?) \parallel start_{run}(eng, t_3)\,;\ end_{run}(eng, t_2)\,;\ (t_2 - t_1 < d)?])))\\ ).\end{array}$$

Here, we are stating that, the robot starts to go to pos at time t_1, meanwhile the engine starts to work at time t_3 and it is switched off, at the arrival to pos, at time t_2. Notice that end_{go} is not explicitly specified, but should be inferred by the interpreter because needed to satisfy at(nav, pos).

7.2 TFGolog Semantics

The semantics of a TFGolog program p with respect to $\mathcal{D}_T$ can be defined in the TFSC.

Given an initial bag of timelines $\mathfrak{s}$, an interval (h_s, h_e) specifying the time horizon over which the program is to be instantiated from h_s to h_e, the execution trace $\mathfrak{s}'$ of a program p is recursively defined through the macro $DoTF(p, \mathfrak{s}, \mathfrak{s}', (h_s, h_e))$.

First we shall introduce some further notation that extends Golog abbreviations to bag of timelines:

$$\mathfrak{s}' = ddo(a, s, \mathfrak{s}) \stackrel{def}{=} (s \in \mathfrak{s} \wedge a =_\nu s) \wedge (\mathfrak{s}' = (\mathfrak{s} \setminus_{\mathscr{S}} \mathscr{B}(\{s\})) \cup_{\mathscr{S}} \mathscr{B}(\{do(a, s)\})). \tag{37}$$

Here, the two operations of difference and union are the one already defined for bag of timelines, as shown in Example 6 and Definition 1. Notice that for $a = A(\vec{x}, \mathbf{t})$ with $\mathbf{t}$ free variable, $\mathfrak{s}$ equals to $ddo(A(\vec{x}, \mathbf{t}), \sigma, \mathfrak{s})$ with $\mathfrak{s}$ mentioning the free variable $\mathbf{t}$.

Furthermore, we extend the function *time* to bags of timelines as follows:

$$ttime(\mathfrak{s}) \stackrel{def}{=} max\{time(\sigma) | \sigma \in \mathfrak{s}\} \tag{38}$$

Here $max\{.\}$ is defined by a first order formula, for example if it were defined for two elements it would be as follows:

$$max\{a, b\} = x \stackrel{def}{=} (x = a \wedge (a > b) \vee x = b \wedge (b > a)).$$

Further, we define the executability of a bag of timelines, over a specified horizon (h_s, h_e) as

$$\begin{array}{rcl} exec(\mathfrak{s}, \mathfrak{s}', (h_s, h_e)) & \stackrel{def}{=} & (\mathfrak{s} = \mathfrak{s}' \wedge ttime(\mathfrak{s}) \geq h_s \wedge ttime(\mathfrak{s}) \leq h_e) \vee \exists \mathfrak{s}'', s, a(\mathfrak{s}' = ddo(a, s, \mathfrak{s}'') \wedge \\ & & executable(do(a, s)) \wedge exec(\mathfrak{s}, \mathfrak{s}'', (h_s, h_e)) \wedge time(a) \geq h_s \wedge time(a) \leq h_e), \end{array} \tag{39}$$

$exec(\mathfrak{s}, \mathfrak{s}', (h_s, h_e))$ states that $\mathfrak{s}'$ is an executable extension of $\mathfrak{s}$. The definition is inductive with respect to $\mathfrak{s}'$, where the base case is $\mathfrak{s}' = \mathfrak{s}$ and the inductive step is given for $\mathfrak{s}' = ddo(a, s, \mathfrak{s}'')$, assuming $exec(\mathfrak{s}, \mathfrak{s}', (h_s, h_e))$, with $do(a, s) \in \mathfrak{s}'$ executable timeline such that $s \in \mathfrak{s}''$.

We can now specify the $DoTF(p, \mathfrak{s}, \mathfrak{s}', (h_s, h_e))$ as follows.

1. *Primitive action* with horizon:

$$DoTF(a, \mathbb{s}, \mathbb{s}', (h_s, h_e)) \stackrel{def}{=} \\ \exists s(s \in \mathbb{s} \wedge a =_\nu s \wedge Poss(a, s) \wedge time(s) \geq h_s \wedge time(s) \leq h_e \wedge time(s) \geq time(a) \wedge \\ (time(a) \leq h_e \wedge \mathbb{s}' = ddo(a, s, \mathbb{s}) \vee time(a) > h_e \wedge \mathbb{s} = \mathbb{s}'))).$$

Here, if the primitive action is applicable to $s \in \mathbb{s}$, and a can be scheduled after the horizon then it is neglected along with the rest of the program (i.e. each action, which can start after the horizon could be neglected; this temporal planning strategy is employed in several timeline-based planners, e.g. [51, 33]). Notice that $Poss(a, s)$ require a and s to be of the same type. Notice also that, for $a = A(\vec{x}, \mathbf{t})$ with $\mathbf{t}$ free variable, the free variable $\mathbf{t}$ is mentioned in $\mathbb{s}'$. We recall that $\mathbb{s}[\omega]$ denotes a bag of situations $\mathbb{s}$ with free variables ω.

2. *Program sequence*:

$$DoTF(prog_1 ; prog_2, \mathbb{s}, \mathbb{s}', (h_s, h_e)) \stackrel{def}{=} \exists \mathbb{s}''(DoTF(prog_1, \mathbb{s}, \mathbb{s}'', (h_s, h_e)) \wedge DoTF(prog_2, \mathbb{s}'', \mathbb{s}', (h_s, h_e))$$

Here, the second program $prog_2$ is executed starting from the execution $\mathbb{s}''$ of the first program $prog_1$.

3. *Partial-order action choice*:

$$DoTF(prog_1 \prec prog_2, \mathbb{s}, \mathbb{s}', (h_s, h_e)) \stackrel{def}{=} \exists \mathbb{s}'', \mathbb{s}'''(DoTF(prog_1, \mathbb{s}, \mathbb{s}'', (h_s, h_e)) \wedge \\ DoTF(prog_1, \mathbb{s}''', \mathbb{s}', (h_s, h_e)) \wedge \mathbb{s}'' \preceq_{\mathscr{S}} \mathbb{s}''') \wedge exec(\mathbb{s}'', \mathbb{s}''', (h_s, h_e))$$

Here, given the execution $\mathbb{s}''$ of the first program $prog_1$, the second program $prog_2$ can be executed starting from an executable extension $\mathbb{s}'''$ of $\mathbb{s}''$. If $\mathbb{s}''=\mathbb{s}'''$ then we have the sequence case.

4. *Parallel execution*:

$$DoTF(prog_1 \parallel prog_2, \mathbb{s}, \mathbb{s}', (h_s, h_e)) \stackrel{def}{=} \\ \exists \mathbb{s}'' DoTF(prog_1, \mathbb{s}, \mathbb{s}', (h_s, h_e)) \wedge DoTF(prog_2, \mathbb{s}, \mathbb{s}'', (h_s, h_e)) \wedge (\mathbb{s}' = \mathbb{s}'').$$

The parallel execution of two programs from $\mathbb{s}$, under the horizon (h_s, h_e), can be specified by the conjunction of the execution of the two programs over the timelines $\mathbb{s}'$ and $\mathbb{s}''$. The execution is correct iff the obtained timelines are equal.

5. *Test action*:

$$DoTF(\phi?, \mathbb{s}, \mathbb{s}', (h_s, h_e)) \stackrel{def}{=} \phi[\mathbb{s}] \wedge \mathbb{s} = \mathbb{s}' .$$

Here $\phi[\mathbb{s}]$ stands for a generalisation of the standard $\phi[s]$ (in the TFSC) extended to bag of timelines, e.g. $P_1[\mathbb{s}] \wedge P_2[\mathbb{s}]$ stands for $P_1(s_1) \wedge P_2(s_2)$ with $s_1, s_2 \in \mathbb{s}$, i.e. each fluent is evaluated with respect to its specific timeline.

6. *Nondeterministic action choice:*

$$DoTF(prog_1|prog_2, \mathbb{s}, \mathbb{s}', (h_s, h_e)) \stackrel{def}{=} DoTF(prog_1 , \mathbb{s}, \mathbb{s}', (h_s, h_e)) \vee DoTF(prog_2, \mathbb{s}, \mathbb{s}', (h_s, h_e)).$$

Here, analogously to standard Golog, the execution of the action choice is represented as the disjunction of the two possible executions.

7. *Nondeterministic argument selection:*

$$DoTF(\pi(x, prog(x)), \mathfrak{s}, \mathfrak{s}', (h_s, h_e)) \stackrel{def}{=} \exists x DoTF(prog(x), \mathfrak{s}, \mathfrak{s}', (h_s, h_e)).$$

The execution of the nondeterministic argument selection is represented as in standard Golog.

8. *Conditionals:*

$$DoTF(\textbf{if } \phi \textbf{ then } prog_1 \textbf{ else } prog_2, \mathfrak{s}, \mathfrak{s}', (h_s, h_e)) \stackrel{def}{=}$$
$$\phi[\mathfrak{s}] \wedge DoTF(prog_1, \mathfrak{s}, \mathfrak{s}', (h_s, h_e)) \vee \neg\phi[\mathfrak{s}] \wedge DoTF(prog_2, \mathfrak{s}, \mathfrak{s}', (h_s, h_e)).$$

9. *Nondeterministic iteration:*

$$DoTF(prog^*, \mathfrak{s}, \mathfrak{s}', (h_s, h_e)) \stackrel{def}{=}$$
$$\forall P\{\forall \mathfrak{s}_1 P(\mathfrak{s}_1, \mathfrak{s}_1) \wedge \forall \mathfrak{s}_1, \mathfrak{s}_2, \mathfrak{s}_3[P(\mathfrak{s}_1, \mathfrak{s}_2) \wedge DoTF(prog, \mathfrak{s}_2, \mathfrak{s}_3, (h_s, h_e)) \rightarrow P(\mathfrak{s}_1, \mathfrak{s}_3)]\} \rightarrow P(\mathfrak{s}, \mathfrak{s}').$$

10. The semantics of conditionals, while loops, and procedures is defined in the usual way.

We show, now, that given two fully ground bags of timelines $\mathfrak{s}_{init}$ and $\mathfrak{s}$ such that $DoTF(prog, \mathfrak{s}_{init}, \mathfrak{s}, (h_s, h_e))$ then the timelines in the bag of timelines $\mathfrak{s}$ complete the timelines in $\mathfrak{s}_{init}$. Furthermore, we show that, if the initial bag of timelines $\mathfrak{s}_{init}$ mentions only executable timelines, then $\mathfrak{s}$ mentions only executable timelines too.

Proposition 3 *Let $\mathcal{D}_T \models DoTF(prog, \mathfrak{s}_{init}, \mathfrak{s}, (h_s, h_e))$ with $ttime(\mathfrak{s}) \leq h_e$ and $h_s \leq ttime(\mathfrak{s}_{init})$, then $\mathfrak{s}_{init} \preceq_{\mathscr{S}} \mathfrak{s}$.*

□

Proposition 4 *Let $\mathcal{D}_T \models DoTF(prog, \mathfrak{s}_{init}, \mathfrak{s}, (h_s, h_e))$ with $ttime(\mathfrak{s}) \leq h_e$ and $h_s \leq ttime(\mathfrak{s}_{init})$, if any $\sigma' \in \mathfrak{s}_{init}$ is executable, then any $\sigma \in \mathfrak{s}$ is executable.*

□

7.3 Generating Flexible Plans in TFGolog

In this section, we describe how TFGolog programs characterise temporally flexible execution traces represented by bags of timelines.

The $DoTF(prog, \mathfrak{s}_{init}, \mathfrak{s}, (h_s, h_e))$ macro defined above defines the bag of timelines $\mathfrak{s}$ that are executable extensions of $\mathfrak{s}_{init}$ within the horizon (h_s, h_e), but temporal constraints are not considered. However, a correct extension $\mathfrak{s}$ for $\mathfrak{s}_{init}$ should also satisfy the temporal constraint network induced by the compatibilities T_c (see Section D) and represented by $\mathbb{I}(T_c, \mathfrak{s}[\omega])$ in TFSC (Corollary 2). Therefore, we introduce the following notion of *temporally flexible execution* of a TFGolog program.

Definition 3 *Let $\mathcal{D}_T$ be a domain theory, T_c a set of compatibilities, prog a TFGolog program, (h_s, h_e) a horizon, and $\mathfrak{s}_{init}$ an initial fully ground bag of situations. Let $\mathfrak{s}[\omega]$ be a bag of timelines with free temporal variables ω, $\mathfrak{s}[\omega]$ is a* temporally flexible execution *of prog if the following holds.*

$$\mathcal{D}_T \models \exists t_1, \ldots, t_n . DoTF(prog, \mathfrak{s}_{init}, \mathfrak{s}[t_1, \ldots, t_n], (h_s, h_e)) \wedge \mathbb{I}(T_c, \mathfrak{s}[t_1, \ldots, t_n]). \tag{40}$$

□

The following proposition shows that, given a *temporally flexible execution* of *prog*, the possible assignments of ω can be characterised by the solution assignments of the temporal constraint network $network(\mathcal{D}_T, T_c, \mathfrak{s}[\omega])$.

Proposition 5 *Let $\mathcal{D}_T$, T_c, prog, and $\mathbb{s}[\omega]$ be, respectively, a TFSC domain theory, a set of compatibilities, a TFGolog program, and a bag of timelines with ω free temporal variables and let $\mathcal{M}$ a model of $\mathcal{D}_T$. Furthermore, let $\mathbb{V}$ be a set of assignments which are solutions for network$(\mathcal{D}_T, T_c, \mathbb{s}[\omega])$ and let $\mathbb{A}$ be the set of assignments to the temporal variables for $\mathcal{M}$.*
Given a ground bag of timelines $\mathbb{s}_{init}$:

$$v \in \mathbb{A} \text{ and } \mathcal{M}, v \models DoTF(prog, \mathbb{s}_{init}, \mathbb{s}[\omega], (h_s, h_e)) \wedge \mathbb{I}(T_c, \mathbb{s}[\omega])$$

iff

$$v \in \mathbb{V} \text{ and } \mathcal{M}, v \models DoTF(prog, \mathbb{s}_{init}, \mathbb{s}[\omega], (h_s, h_e)).$$

□

As a consequence of the definition of *temporally flexible execution* and of the above statement, we have the following corollary which directly follows from Proposition 5.

Corollary 3 *Let $\mathcal{D}_T$, T_c, and prog be, respectively, a domain theory, a set of compatibilities, and a TFGolog program. Let (h_s, h_e) be a horizon, $\mathbb{s}_{init}$ an initial fully ground bag of situations, and $\mathbb{s}[\omega]$ a* temporally flexible execution *of* prog.

Given any model $\mathcal{M}$ of $\mathcal{D}_T$ and any v s.t. $\mathcal{M}, v \models DoTF(prog, \mathbb{s}_{init}, \mathbb{s}[\omega], (h_s, h_e)) \wedge \mathbb{I}(T_c, \mathbb{s}[\omega])$, we have that $v \in \mathbb{V}$, hence it is a solution of network$(\mathcal{D}_T, T_c, \mathbb{s}[\omega])$.

□

In other words, this corollary states that, given a *temporally flexible execution* $\mathbb{s}[\omega]$, the possible assignments to ω are the solutions $\mathbb{V}$ of *network*$(\mathcal{D}_T, T_c, \mathbb{s}[\omega])$.

Once we have established a relation between the *temporally flexible execution* and the network, we may want to explicitly represent the solutions in the signature of $\mathcal{L}_{TFSC}$. Suppose we obtain, as a solution of the network, a tuple of real numbers $\mathbf{m} = \langle \mathbf{m_1}, \ldots, \mathbf{m_n} \rangle$ then there are two possibilities. If $\mathbf{m}$ is a tuple of rational numbers, they are representable in $\mathcal{L}_{TFSC}$, hence we can explicitly refer to $\mathbb{s}[\mathbf{m}]$ to represent a ground $\mathbb{s}[\omega]$, e.g., ensuring that $\mathcal{D}_T \models DoTF(prog, \mathbb{s}_{init}, \mathbb{s}[\mathbf{m}], (h_s, h_e)) \wedge \mathbb{I}(T_c, \mathbb{s}[\mathbf{m}]))$ to check (40) for the instance $\mathbf{m}$. Otherwise, $\mathbf{m} = \langle \mathbf{m_1}, \ldots, \mathbf{m_n} \rangle$ might be numbers not in the signature of $\mathcal{L}_{TFSC}$. If we want to represent also these cases, a possible solution is to extend the language to $\mathcal{L}^{\mathbf{m}}_{TFSC}$ adding for each number its corresponding symbol name as a constant. Then, in the extended language $\mathcal{L}^{\mathbf{m}}_{TFSC}$, we can obtain a suitable interpretation for the new symbols, by associating the interpretation of each new symbol $\mathbf{m}_i$ to the correspondent variable assignment, that is, ensuring that $v(\mathbf{t}_i) = \mathbf{m}_i^I$, according to Proposition 5.

Notice that the $\mathbb{s}[\omega]$, analogously to standard Golog, can be obtained from the program *prog* as constructive proof of (40). The main difference with respect to the standard Golog approach relies in the presence of the free variables ω, these are the temporal variables $\langle \mathbf{t}_1, \ldots, \mathbf{t}_m \rangle$ associated with the actions $A_i(\vec{d_i}, \mathbf{t}_i)$ extending the bag of timelines $\mathbb{s}_{init}$ to $\mathbb{s}[\omega]$. The description of an algorithm implementing the interpreter is beyond the scope of this paper, here we just notice that, similarly to the standard Golog approach, the interpreter has to instantiate the nondeterministic choices searching for the possible alternatives. However, in this case, the temporal variables for the actions $A_i(\vec{d_i}, \mathbf{t}_i)$, instead of been instantiated, are left free (because constrained by the temporal network represented by $\mathbb{I}(T_c, \mathbb{s}[\omega])$). An interpreter of this kind can be implemented in the Constraint Logic Programming language (CLP) [32] which combines logic programming and constraint management. For example, in [10] we exploit an implementation of the TFGolog interpreter developed in C++ and the Eclipse 5.7 engine for CLP.

Example 13 *Considering again the two components* nav *(for navigation) and* eng *(for engine), depicted in Figure 4, a possible TFGolog program encoding the robot task approaching position pos, within the time interval d , can be written as follows:*

$$\begin{aligned}&\textbf{proc}(approach(d, pos), \quad \pi(t_1, \pi(t_2, \pi(t_3,\\ &\qquad [\pi(x,\ start_{go}(nav, x, pos, t_1)) \prec at(nav,\ pos)? \parallel start_{run}(eng, t_3);\ end_{run}(eng, t_2);\ (t_2 - t_1 < d)?])))).\end{aligned}$$

Here, we are stating that, the robot starts to go to pos at time t_1, meanwhile the engine starts to work at time t_3 and it is switched off, at the arrival to pos, at time t_2. Given an initial, fully ground, bag of timelines:

$$\mathbb{s}_{init} = \mathscr{B}(\{do(start_{go}(nav, p_1, p_2, 2), S_0), do(start_{run}(eng, 1), S_0)\}),$$

stating that, at the beginning, the agent starts going from p_1 to p_2 at time 2 and starts the engine at time 1, a temporally flexible execution $\sigma[\omega]$ for the program approach(d, pos) is such that

$$\mathcal{D}_T \models \exists t_1, t_2, t_3. DoTF(approach(5, p_2), \mathbb{s}_{init}, \mathbb{s}[t_1, t_2, t_3], (0, 10)) \wedge \mathbb{I}(T_c, \mathbb{s}[t_1, t_2, t_3]),$$

where $d = 5$, $pos = p_2$, and $(h_s, h_e) = (0, 10)$.

7.4 TFGolog and Sequential Temporal Golog

To understand how TFGolog relates to other Golog versions in the literature we now show that TFGolog extends Sequential Temporal Golog. Given the axioms of sequential temporal SC in [64], it is possible to accommodate time in the Golog semantics. The Sequential Temporal Golog [64] can be directly obtained from the classical Golog, the only change needed is the *Do* macro for primitive actions:

$$Do(a, s, s') \stackrel{def}{=} Poss(a, s) \wedge start(s) \leq time(a) \wedge s' = do(a, s),$$

where $start(s)$ represents the activation time for the situation s. Everything else about *Do* remains the same.

It is possible to show that TFGolog extends STGolog. Intuitively, any STGolog program can be expressed as TFGolog program working with a single timeline, grounded time, infinite horizon, without time constraints. More formally, we can state the following proposition.

Proposition 6 *Given a Sequential Temporal SC theory $\mathcal{D}_{STSC}$ it is possible to define a TFSC theory $\mathcal{D}_T$ such that for any STGolog program $prog_{st}$ there exists a TFGolog program $prog_{tf}$ such that*

$$\mathcal{D}_{STSC} \models Do(prog_{st}, \sigma, \sigma') \qquad iff \qquad \mathcal{D}_T \models DoTF(prog_{tf}, \mathbb{s}, \mathbb{s}', (0, \infty)),$$

where $\mathbb{s} = \mathscr{B}(\sigma)$ and $\mathbb{s}' = \mathscr{B}(\sigma')$.

□

Notice that STGolog concurrent temporal processes can be expressed by interleaving start and end actions along a unique timeline represented by a single situation. STGolog, indeed, induces a complete order on activities. Therefore, this model of concurrency cannot represent partially ordered activities, that might have parallel runs, since in this case the sequence of activations has to be decided at the execution time.

Reiter [65] proposes a concurrent version of STGolog, that permits the execution of sets actions, with a set $c = \{a_1, \ldots, a_n\}$ in place of primitive actions. This notwithstanding the order of the activations is already fixed in the generated sequence of actions. For example

$$[\{start_a, start_b\}, end_a, end_b] \tag{41}$$

is a concurrent execution of two processes a and b where a and b start is synchronised, but end of a has to occur before the end of b.

On the other hand, in TFGolog we can express a more general (flexible) execution plan as follows:

$$[start_a, end_a] ||| [start_b, end_b]. \tag{42}$$

Here process a can end either before or after or even during the end of b.

The point here is that strict sequentialization, as illustrated above for concurrent STGolog, is due to the concurrency model based on interleaving actions. This model hampers the possibility to generate flexible sequences in which switching is made possible. These limiting aspects are inherited by all the Golog-family, based on the interleaving model, including ConGolog [11] and IndiGolog [29].

8 Example: Attentive robot exploration of the environment

Consider an autonomous robot exploring an environment and performing observations (e.g. a rescue rover [10]), robot stimuli might guide the robot according to compatible constraints between components and tasks.

Let us model the executive control of the robot via a set of interacting *components* enabling switching between tasks. For example, some typical components of an autonomous mobile robot are: the *Head* controller, that is the PTU or pan-tilt unit, the *Locomotion* controller, the *Navigation* module, including simultaneous localisation and mapping *SLAM*, the *Attention* system providing a saliency map for focusing on regions to be processed by the *Visual Processing* component etc. We refer the reader to [10] for a detailed description of this domain. In particular, in [10] we present a mobile robot whose executive control system is designed deploying TFSC and TFGolog (see Figure 6).

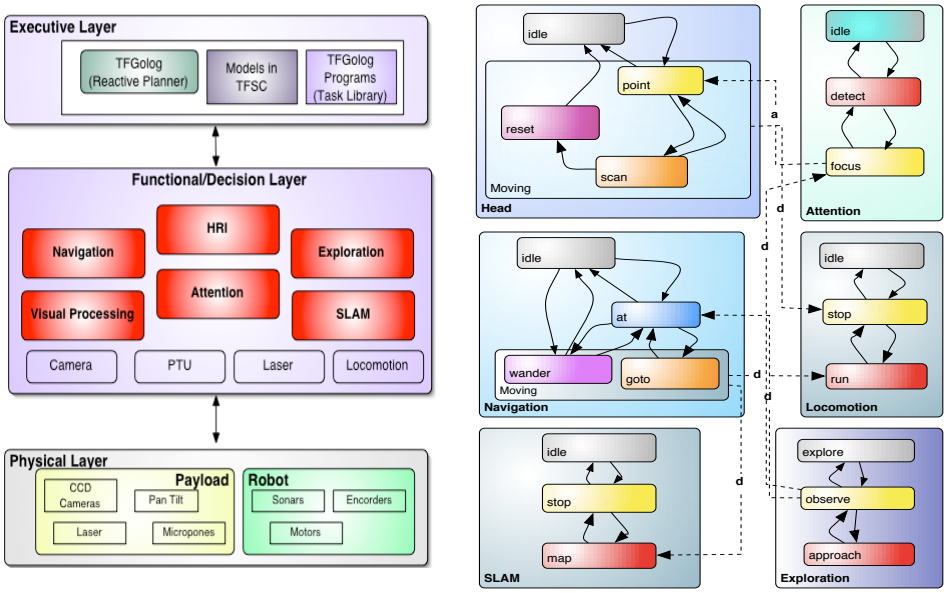

Figure 6: Robot Control Architecture (left) and some control modules (right) with their processes/states (round boxes), transitions (arrows), and temporal relations (dotted arrows)

TFSC Representation. Each component of the control system can be represented by a type in TCSF and it is associated with a set of processes. For example, we can consider as names of types the constants: *ptu*, *slm*, *att*,*exp*, *nav*, *lcm* denoting, respectively, the Head component (*ptu*), the SLAM module (*slm*), the Attention module (*att*), the Explore module (*exp*), the Navigation module (*nav*), and the Locomotion component (*lcm*). Each of these types is associated with a set of primitive actions, e.g. for the Locomotion component we have the primitive actions $start_{stop}(lcm,t)$, $end_{stop}(lcm,t)$, $start_{run}(lcm,t)$, and $end_{run}(lcm,t)$, here the *lcm* type is defined as follows:

$$H(lcm,a) \equiv \exists t\, a = start_{stop}(lcm,t) \vee \exists t\, a = end_{stop}(lcm,t) \vee \exists t\, a = start_{run}(lcm,t) \vee \exists t\, a = end_{run}(lcm,t).$$

As for the related processes, we can introduce specific fluents for each component, for example, given the component depicted in Figure 6, possible fluents in this domain are:

Head: $\{idle(ptu, s), point(ptu, x, t, s), scan(ptu, z, x, t, s), reset(ptu, z, x, t, s)\}$;
SLAM: $\{idle(slm, s)\ map(slm, s, t), stop(slm, s, t)\}$ };
Attention: $\{idle(att, s)\ detect(att, s, t), focus(att, s, t)\}$ };
Explore: $\{idle(exp, s), explore(exp, t, s), observe(exp, s, t), approach(exp, s, t)\}$ };
Navigation: $\{idle(nav, s), goto(nav, x, y, t, s), wander(nav, t, s), at(nav, x, s)\}$;
Locomotion: $\{idle(lcm, s), run(lcm, t, s), stop(lcm, t, s)\}$.

Each process is explicitly represented in the TFSC model as described in Section 3. For example, in our case the process *scan* is modelled by the fluent $scan(ptu, z, x, t, s)$ and the actions $start_{scan}(ptu, z, x, t)$ and $end_{scan}(ptu, z, x, t)$. The preconditions and the effects are encoded in the $\mathcal{D}_T$ as specified in (16). For example, the successor state axiom for $scan(ptu, z, x, t, s)$ is the following:

$$scan(ptu, z, x, t, do(a, s)) \equiv a = start_{scan}(ptu, z, x, t) \vee scan(ptu, z, x, t, s) \wedge \neg\exists t'(a = end_{scan}(ptu, z, x, t')),$$

while the preconditions for $start_{scan}(ptu, z, x, t)$ and $end_{scan}(ptu, z, x, t)$ are:

$$Poss(start_{scan}(ptu, z, x, t), s) \equiv (s = S_0 \vee s =_\nu start_{scan}(ptu, z, x, t)) \wedge idle(ptu, s) \wedge time(s) \leq t\,;$$
$$Poss(end_{scan}(ptu, z, x, t), s) \equiv (s = S_0 \vee s =_\nu end_{scan}(ptu, \vec{x}, t)) \wedge \exists t'\, scan(ptu, z, x, t', s) \wedge time(s) \leq t\,.$$

Temporal Compatibilities Some *temporal compatibilities* T_c among the activities can be defined as follows:

$$\begin{aligned} T_c \quad =[\quad & comp(point(ptu, x), \quad [[(\mathbf{m}\ scan(ptu, x))]]), \\ & comp(focus(att, x), \quad [[(\mathbf{a}\ \ point(ptu, x))]]), \\ & comp(scan(ptu, z, x), \quad [[(\mathbf{d}\ \ stop(exp)), (\mathbf{a}\ map(slm))]]), \\ & comp(goto(nav, x, y), \quad [[(\mathbf{d}\ \ idle(ptu)), (\mathbf{d}\ map(slm))]])]. \end{aligned}$$

These compatibilities state the following temporal constraints. $point(ptu, x)$ **m** $scan(ptu, x)$, i.e. upon the PTU is pointed toward a location x the head is expected to scan the region around that point x, and the temporal relation $focus(att, x)$ **a** $point(ptu, x)$ tells that attention focus is preceded by a PTU pointing towards a region of the environment specified by x. After focusing the head can direct the cameras towards the region. Also, $scan(ptu, z, x)$ **d** $stop(lcm)$ prescribes that while the *Head* is scanning the environment the robot must be in stop mode to avoid invoking stabilisation processes. The constraints $goto(nav, x, y)$ **d** $idle(ptu)$ and $goto(nav, x, y)$ **d** $map(slm)$ state that, while the robot is moving, the pant-tilt unit is to be idle and attention is to be active. Figure 7 illustrates a possible evolution of the timelines up to a planning horizon considering the overall system.

TFGolog programs Once we have defined the TFSC domain, we can introduce a partial specification of the robot behaviours using TFGolog programs. For example, we can say that: at the very beginning, i.e. time 0, the pant-tilt is idling with attention enabled; from time 0 to 3 the robot should remain where it is (e.g. pos_{init}), perform overall scanning with attention on, gathering information from the environment. Furthermore, given a direction θ it should focus attention $focus$ towards it, before 30 and after 20, and move towards it before 50. This partially specified plan of actions can be encoded by the following TFGolog program:

$$\begin{aligned} & \mathbf{proc}(partialPlan(p, p', \theta), \\ & \pi(t_1, \pi(t_2, \pi(t_3, \pi(t_4, \pi(t_6, \pi(t_5, \pi(t_7, \pi(t_8, \pi(t_9, \pi(t_{10}, \pi(t_{11}, \pi(t_{12}, \pi(t_{13}, \pi(t_{14}, \\ & \quad Active_{map}(slm, 0, t_1) \wedge t_1 > 0)? \parallel \\ & \quad Elapsed(att, detect, t_2, t_3) \prec Elapsed_{focus}(att, \theta, t_4, t_5)? \parallel \\ & \quad Elapsed_{idle}(ptu, 0, t_6)? \prec (Elapsed_{scan}(ptu, \theta, t_7, t_8) \wedge 20 \leq t_6 \wedge t_8 \leq 30)? \parallel \\ & \quad Elapsed(lcm, stop, t_9, t_{10})? \parallel \\ & \quad (Elapsed_{at}(nav, p, 0, t_{11}) \wedge t_{12} > 3)? \prec (Elapsed_{at}(nav, p', t_{13}, t_{14}) \wedge t_{15} < 50)? \\ &))))))))))) \\ &). \end{aligned}$$

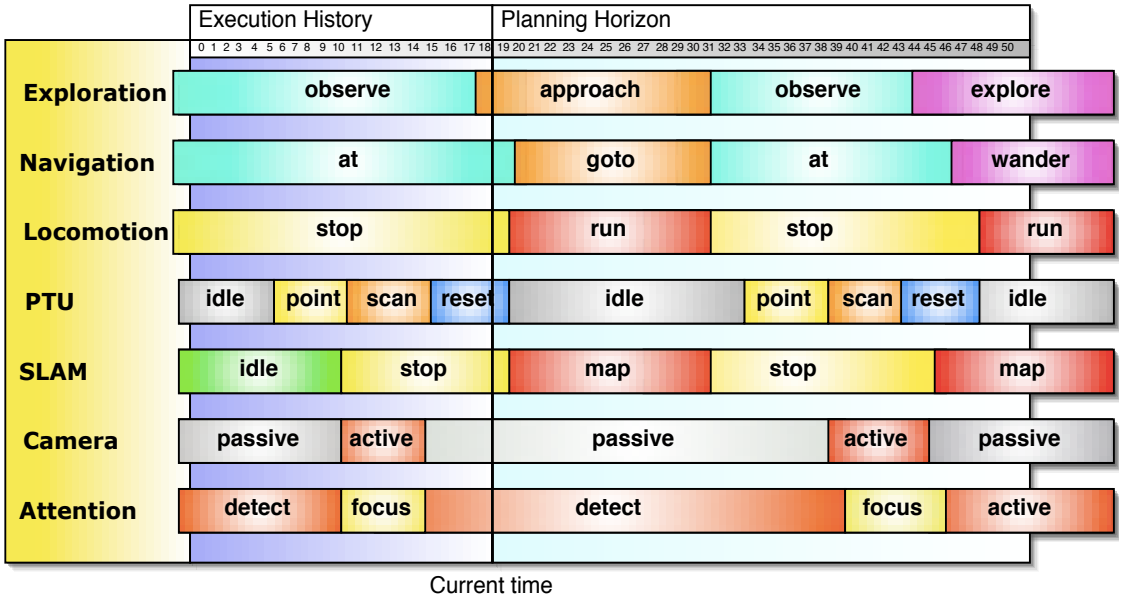

Figure 7: The history of states over a period of time (timelines) illustrating the evolution of several system components up to a planning horizon.

If we fix the horizon equal to $(h_e, h_s) = (0, 50)$ and the initial bag of situations $\mathbb{s}_0 = \mathscr{B}(S_0)$, a complete plan for *partialPlan* can be obtained as a bag of timelines mentioning timelines for each components such that

$$\mathcal{D}_T \models \exists t_1, \ldots, t_n . DoTF(partialPlan(p_1, p_3, \theta), \mathbb{s}_0, \mathbb{s}[t_1, \ldots, t_n], (0, 50)) \wedge \mathbb{I}(T_c, \mathbb{s}[t_1, \ldots, t_n]).$$

For example, let:

$$\begin{array}{lcll}
\sigma_{slm} & = & do([& start_{map}(slm, \mathbf{t}_1)], \quad S_0); \\
\sigma_{att} & = & do([& start_{det}(att, \mathbf{t}_2) \quad end_{det}(att, \mathbf{t}_3), \\
 & & & start_{focus}(att, \mathbf{t}_4) \quad end_{focus}(att, \mathbf{t}_5), \quad start_{det}(att, \mathbf{t}_6)], \quad S_0); \\
\sigma_{exp} & = & do([& start_{exp}(exp, \mathbf{t}_7) \quad end_{exp}(exp, \mathbf{t}_8), \\
 & & & start_{obs}(exp, \mathbf{t}_9), \quad end_{obs}(exp, \mathbf{t}_{10}), \quad start_{exp}(exp, \mathbf{t}_{11})], \quad S_0); \\
\sigma_{ptu} & = & do([& start_{point}(ptu, \theta, \mathbf{t}_{12}), \quad end_{point}(ptu, \theta, \mathbf{t}_{13}), \quad start_{scan}(\mathbf{t}_{14}), \\
 & & & end_{scan}(ptu, \mathbf{t}_{15}), \quad start_{reset}(ptu, \mathbf{t}_{16}), \quad end_{reset}(ptu, \mathbf{t}_{17})], \quad S_0); \\
\sigma_{nav} & = & do([& start_{go}(nav, p_3, \mathbf{t}_{18}), \quad end_{go}(nav, p_3, \mathbf{t}_{19})], \quad S_0); \\
\sigma_{lcm} & = & do([& start_{start}(lcm, \mathbf{t}_{20}), \quad end_{start}(lcm, \mathbf{t}_{21})], \quad S_0).
\end{array}$$

and $\mathbb{s}[\mathbf{t}_1, \ldots, \mathbf{t}_{21}] = \mathscr{B}(\langle \sigma_{slm}[\mathbf{t}_1], \sigma_{att}[\mathbf{t}_2, \mathbf{t}_3, \mathbf{t}_4], \sigma_{exp}[\mathbf{t}_7, \ldots, \mathbf{t}_{11}], \sigma_{ptu}[\mathbf{t}_{12}, \ldots, \mathbf{t}_{17}], \sigma_{nav}[\mathbf{t}_{18}, \mathbf{t}_{19}], \sigma_{lcm}[\mathbf{t}_{20}, \mathbf{t}_{21}] \rangle)$, the bag of situations $\mathbb{s}[\omega]$ defined above is obtained by the macro $\mathbb{I}(T_c, \mathbb{s}[\omega])$ which represents a temporal network, denoted by $network(\mathcal{D}_T, T_c, \mathbb{s}[\omega])$, with the following temporal constraints:

$$\begin{array}{l}
\{0 \leq \mathbf{t}_1 \leq 50;\ 0 \leq \mathbf{t}_2 \leq \mathbf{t}_3 = \mathbf{t}_4 \leq \mathbf{t}_5 = \mathbf{t}_6 \leq 50; \\
0 \leq \mathbf{t}_7 \leq \mathbf{t}_6 = \mathbf{t}_7 \leq \mathbf{t}_8 = \mathbf{t}_9 \leq \mathbf{t}_{10} = \mathbf{t}_{11} \leq 50; \\
0 \leq \mathbf{t}_{12} \leq \mathbf{t}_{12} = \mathbf{t}_{13} \leq \mathbf{t}_{14} = \mathbf{t}_{15} \leq \mathbf{t}_{16} = \mathbf{t}_{17} \leq 50; \\
0 \leq \mathbf{t}_{18} \leq \mathbf{t}_{19} \leq 50; 0 \leq \mathbf{t}_{20} \leq \mathbf{t}_{21} \leq 50; \\
\mathbf{t}_1 \leq \mathbf{t}_{18}; \mathbf{t}_1 \leq \mathbf{t}_{20}; \mathbf{t}_{13} \leq \mathbf{t}_4; \mathbf{t}_9 \leq \mathbf{t}_4; \\
\mathbf{t}_5 \leq \mathbf{t}_{10}; 0 \leq \mathbf{t}_4 \leq \mathbf{t}_5 \leq \mathbf{t}_{18}; \mathbf{t}_{21} \leq \mathbf{t}_{12}\}.
\end{array}$$

Beyond partial plans, TFGolog can encode more general and intuitive behaviour fragments for tasks that can be selected and compiled if they are compatible with the execution context. For example, the following behaviour fragment can induce the interpreter to produce a plan to find a location within the deadline d and reach it:

$$\begin{aligned}&\textbf{proc}(findLocation(d),\\ &\quad \pi(t_1,\pi(t_2,\pi(t_3,\pi(t_4,\pi(t_5,\\ &\qquad start_{exp}(exp,t_1) \prec end_{appr}(exp,t_2) \,\|\\ &\qquad start_{wand}(nav,t_3)\,;\, end_{wand}(nav,t_4)\,;\, \pi(x, start_{goto}(nav,x,t_5))\,;\, (t_4 - t_3{<}d)?))))))\\ &).\end{aligned}$$

This TFGolog script starts both the exploration and wandering activities; the wandering phase has a timeout d, after which the robot has to go somewhere. The timeout d is provided by the calling process that can be either another TFGolog procedure or a decision taken by the operator. Note that the procedure here is partially specified, indeed, we only mention processes belonging to the *Exploration* and *Navigation* timelines, but all the other timelines are to be managed by the TFGolog interpreter.
Another example is the following script that manages the switch to the *explore* mode during an *approach* phase in the case of a *stop*:

$$\begin{aligned}&\textbf{proc}(switchApproachExplore(x,d),\\ &\pi(t_1,\pi(t_2,\pi(t_3,\\ &\quad ((((Active_{appr}(exp,x,t_1) \wedge Active_{stop}(lcm,t_1) \wedge idle(nav,t_1))? \prec\\ &\qquad (start_{map}(slm,t_2) \prec start_{exp}(exp,t_3)))\,|\\ &\quad (Active_{appr}(exp,x,t_1) \wedge Active_{stop}(lcm,t_1) \wedge \exists y(at(nav,y,t_1) \wedge y \neq x))?\,;\\ &\qquad ((Active_{map}(slm,t_1))? \prec start_{run}(loc,t_3)\,|\\ &\qquad \neg Active_{map}(slm,t_1)? \prec start_{map}(slm,t_2) \prec start_{exp}(exp,t_3))) \wedge (t_3 - t_1) \leq d)))\\ &).\end{aligned}$$

Here, the script manages a switch between the approach and explore tasks caused by a *stop* during an approach to a target location x. The overall switch should occur within a deadline d. It considers two cases: (1) the stop occurred while the navigation is in *idle*, hence the robot is not localised; (2) the stop occurred while navigation is *at* position y, therefore the robot is localised. In the first case, the system should switch to the exploration mode (i.e. $start_{exp}$) and restart the SLAM mapping (i.e. $start_{map}$) to re-localise the robot. Notice that the generation of the restart sequence is left to the interpreter because it depends on the context. For instance, if *map* is running, the interpreter is to switch off *map* and restart it. In the second case, the *stop* occurred while the robot is localised *at* position y (that is not the target position), the system can just restart the engine to continue to approach the target, otherwise, since the system behaviour is not the expected one, to keep the robot safety, the activities are to be reconfigured in the exploration mode restarting the *map*.

9 Related Works

A very first temporal extension of the Situation Calculus is proposed by Pinto and Reiter in [57, 56, 65]. In these works, the authors provide an explicit representation of time and event occurrences assimilating a single timeline to a timed situation. They specify durative actions by instantaneous starting and ending actions actuating processes. Concurrent executions of instantaneous actions are also enabled as reported in [65]. Pinto and Reiter in [57] show that a modal logic for concurrency can be embedded in a suitable Situation Calculus extension. These topics are addressed also by Miller and Shannahan in [49], where they propose a method to represent incomplete narratives in the Situation Calculus. In this case, differently from our approach, the problem of an unknown ordering amongst events is enabled by non monotonic reasoning on temporal events.

Indeed, while Pinto and Reiter in [57, 56, 65] propose a situation-based timeline representation, where time is scanned by actions, from which is recovered, Miller and Shannahan suggest in [49] a non monotonic time-based framework where each time point is connected with a situation and the frame problem is addressed via minimisation.

These approaches are substantially different from our; in fact, as we already stated in Section 4, our framework assumes the durative actions representation proposed in [65, 56], but considering multiple timelines and flexible intervals. Our approach, furthermore, contributes substantially to the formalisation of tasks switching and components interaction that has been treated and faced with methodologies, distinct from ours, such as the flexible temporal planning framework.

Pinto in [57, 56] has considered the interaction within processes too, but under the perspective of exogenous actions as natural processes.

In [66], Reiter and Yuhua exploit the temporal extension of the Situation Calculus already presented in [57, 56, 65] for modelling complex scheduling tasks by axiomatising a deadline driven scheduler. In this case, the tasks are to be scheduled for a single CPU, and a schedule of length n is a sequence of n ground actions represented by a single grounded situation term, therefore constraints and flexible plans are not taken into account.

Temporal properties in the Situation Calculus are also investigated by Gabaldon in [24] and by Bienvenu, Fritz and McIlraith in [7, 23], mainly focusing on search control in forward planning. Gabaldon, in fact, in [24] proposes to formalising control knowledge for a forward chaining planner using Linear Temporal Logic (LTL) expressions, represented in the Situation Calculus, and shows how a progression algorithm can be deployed in this setting. In the context of preference based planning [5], Bienvenu et al. [7] propose a logical language for specifying qualitative temporal preferences in the Situation Calculus. In this framework, temporal preferences can be expressed in LTL and the temporal connectives are interpreted in the Situation Calculus following the approach proposed by Gabaldon in [24].

Kvarnström and Doherty in [36] present a forward-chaining planner based on domain-dependent search control knowledge represented as formulas in the Temporal Action Logic (TAL); a narrative based linear metric time logic is used for reasoning about action and change. The authors disregard temporal constraint networks and flexible planning although in [44], following an approach similar to the one taken in [20, 19], the authors propose a first step towards the integration of constraint-based representations within the TAL framework.

A procedural approach to model-based executive control through temporally flexible programs is provided by the model-based programming paradigm of Williams and colleagues [80]. In this approach, the reactive system's controller is specified by programs and models of the system components. In particular, the authors develop the Reactive Model-based programming (RMPL) language and its executive (Titan). Titan control executes RMPL programs using extensive component-based declarative models of the embedded system to track states, analyse anomalous situations, and generate control sequences. RMPL programs are complete procedural specification of the system behaviour. In contrast, we deploy the TFGolog framework where partially specified programs can be encoded. The system we propose in this paper copes with high-level agent programming and can be seen as a trade off between the model-based programming approach (e.g. RMPL-Titan) and the model-based reactive planning (e.g. IDEA [50, 18]), but based on a logical framework and inspired by neuroscience principle on task switching. Indeed, similarly to RMPL, we use high-level programs to design the controller, but the constructs are defined in FOL; further, to enable run-time switching, our programs are partially specified scripts to be completed on-line by the program interpreter that works as a planner.

In the literature, we can find several works investigating the combination of logic-based framework and temporal constraint reasoning. For example, Dechter and Schwalb in [69] present a logic-based framework combining qualitative and quantitative temporal constraints. This framework integrates reasoning in a propositional and narrative-based representation of a dynamic domain - in the style of the Event Calculus - with inference techniques proper of the temporal constraint networks formalism of Dechter, Meiri and Pearl [14]. The integration is based on the notion of conditional temporal network (CTN) which allows decoupling propositional and temporal constraints and treating them in isolation. Analogously to our approach, the logical machinery determines a

temporal network that can be solved with constraint propagation techniques.

The combination of logic-based and constraint-based temporal reasoning is also investigated within the Constraint Logic Programming (CLP) paradigm. For example, the TCLP framework proposed by Schwalb and Vilain [70] augments logic programs with temporal constraints. Indeed Schwalb and Vilain investigate a decidable fragment called Simple TCLP accommodating intervals of event occurrences and temporal constraints between them. Lamma and Milano in [37] extend the Constraint Logic Programming framework to temporal reasoning, elaborating on the extensions of Vilain and Kautzs Point Algebra, on Allen's Interval Algebra and on the STP framework proposed by Dechter, Meiri and Pearl. Lamma and Milano show how it is possible to cope with disjunctive constraints even in an interval based framework.

10 Summary and Outlook

Cognitive control has to deal with several components, with flexible behaviours that can be adapted to different contexts and with the ability to switch between tasks, on stimuli requests.

In this paper, we have presented a methodology to incorporate these attitudes in the Situation Calculus. We have introduced the Temporally Flexible Situation Calculus (TFSC) that combines temporal constraint reasoning and reasoning about actions. In this framework, we have shown how to incorporate multiple parallel timelines and temporal constraints among the activities. For this purpose, we have introduced sets of concurrent, temporal, situations describing a constructive method to associate a temporal constraint network to each set of concurrent timelines represented by a collection of situations. In this way, causal logic-based reasoning and temporal constraint propagation methods can be integrated. We have described an approach for modelling complex dynamic domains in TFSC illustrating how temporally flexible behaviours can be represented. We have shown how this framework can be exploited to design and develop a model-based control system for an autonomous mobile robot capable of balancing high-level deliberative activities and reactive behaviours, more details on the application can be found in [10].

References

[1] Natasha Alechina, Mehdi Dastani, Brian Logan, and John-Jules Ch. Meyer. A logic of agent programs. In *AAAI*, pages 795–800, 2007.

[2] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.

[3] A. R. Aron. The neural basis of inhibition in cognitive control. *The Neuroscientist*, 13:214 – 228, 2007.

[4] Fahiem Bacchus, Joseph Y. Halpern, and Hector J. Levesque. Reasoning about noisy sensors and effectors in the situation calculus. *Artif. Intell.*, 111(1-2):171–208, 1999.

[5] J. Baier, F. Bacchus, and S. McIlraith. A heuristic search approach to planning with temporally extended preferences. In *Proceedings of IJCAI-2007)*, pages 1808–1815, 2007.

[6] Federico Barber. Reasoning on interval and point-based disjunctive metric constraints in temporal contexts. *J. Artif. Intell. Res. (JAIR)*, 12:35–86, 2000.

[7] M. Bienvenu, C. Fritz, and S. McIlraith. Planning with qualitative temporal preferences. In *Proceedings of KR-06*, pages 134–144, 2006.

[8] Stephen A. Block, Andreas F. Wehowsky, and Brian C. Williams. Robust execution on contingent, temporally flexible plans. In *AAAI*, 2006.

[9] Craig Boutilier, Raymond Reiter, Mikhail Soutchanski, and Sebastian Thrun. Decision-theoretic, high-level agent programming in the situation calculus. In *Proceedings of AAAI-2000*, pages 355–362, 2000.

[10] A. Carbone, A. Finzi, A. Orlandini, and F. Pirri. Model-based control architecture for attentive robots in rescue scenarios. *Autonomous Robots*, 24(1):87–120, 2008.

[11] G. de Giacomo, Y. Lespérance, and H. J. Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artif. Intell.*, 121(1-2):109–169, 2000.

[12] A.K. Jonsson D.E. Smith, J. Frank. Bridging the gap between planning and scheduling. *Knowledge Engineering Review*, 15(1), 2000.

[13] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. In *KR*, pages 83–93, 1989.

[14] Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artif. Intell.*, 49(1-3):61–95, 1991.

[15] Yiannis Demiris and Bassam Khadhouri. Hierarchical attentive multiple models for execution and recognition of actions. *Robotics and Autonomous Systems*, 54(5):361–369, 2006.

[16] J. Duncan. Disorganization of behaviour after frontal-lobe damage. *Cognitive Neuropsychology*, 3:271–290, 1986.

[17] Matthias Fichtner, Axel Großmann, and Michael Thielscher. Intelligent execution monitoring in dynamic environments. *Fundamenta Informaticae*, 57(2–4):371–392, 2003.

[18] A. Finzi, F. Ingrand, and N. Muscettola. Model-based executive control through reactive planning for autonomous rovers. In *Proceedings of IROS-2004*, pages 879–884, 2004.

[19] A. Finzi and F. Pirri. Flexible interval planning in concurrent temporal golog. In *Proceedings of Cognitive Robotics 2004*, 2004.

[20] A. Finzi and F. Pirri. Representing flexible temporal behaviors in the situation calculus. In *Proceedings of IJCAI-2005*, pages 436–441, 2005.

[21] A. Finzi, F. Pirri, and R. Reiter. Open world planning in the situation calculus. In *Proceedings of AAAI/IAAI-2000*, pages 754–760, 2000.

[22] Jeremy Forth and Murray Shanahan. Indirect and conditional sensing in the event calculus. In *ECAI*, pages 900–904, 2004.

[23] C. Fritz and S. McIlraith. Decision-theoretic golog with qualitative preferences. In *Proceedings of the 10th International Conference on Principles of Knowledge Representation and Reasoning (KR06)*, pages 153–163, Lake District, UK, June 2006.

[24] A. Gabaldon. Precondition control and the progression algorithm. In Shlomo Zilberstein, Jana Koehler, and Sven Koenig, editors, *ICAPS-2004*, pages 23–32. AAAI, 2004.

[25] Sandra Clara Gadanho. Learning behavior-selection by emotions and cognition in a multi-goal robot task. *J. Mach. Learn. Res.*, 4:385–412, 2003.

[26] Michael Gelfond and Vladimir Lifschitz. Action languages. *Electron. Trans. Artif. Intell.*, 2:193–210, 1998.

[27] M. Ghallab and H. Laruelle. Representation and control in ixtet, a temporal planner. In *Proceedings of AIPS-1994*, pages 61–67, 1994.

[28] G. De Giacomo, Y. Lespérance, and H. Levesque. ConGolog, a concurrent programming language based on the situation calculus. *Artif. Intell.*, 121(1–2):109–169, 2000.

[29] G. De Giacomo, Y. Lespérance, and H. J. Levesque. Reasoning about concurrent execution, prioritized interrupts, and exogenous actions in the situation calculus. In *IJCAI-1997*, pages 1221–1226, 1997.

[30] H. Grosskreutz and G. Lakemeyer. ccgolog – a logical language dealing with continuous change. *Logic Journal of the IGPL*, 11(2):179–221, 2003.

[31] H. Grosskreutz and G. Lakemeyer. Probabilistic complex actions in golog. *Fundam. Inf.*, 57(2-4):167–192, 2003.

[32] J. Jaffar and M.J. Maher. Constraint logic programming: A survey. *Journal of Logic Programming*, 19/20:503–581, 1994.

[33] Ari K. Jonsson, Paul H. Morris, Nicola Muscettola, Kanna Rajan, and Benjamin D. Smith. Planning in interplanetary space: Theory and practice. In *Artificial Intelligence Planning Systems*, pages 177–186, 2000.

[34] Kazuhiko Kawamura, Tamara E. Rogers, and Xinyu Ao. Development of a cognitive model of humans in a multi-agent framework for human-robot interaction. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 1379–1386, New York, NY, USA, 2002. ACM.

[35] Andrei Krokhin, Peter Jeavons, and Peter Jonsson. Reasoning about temporal relations: The tractable subalgebras of allen's interval algebra. *J. ACM*, 50(5):591–640, 2003.

[36] J. Kvarnström and P. Doherty. Talplanner: A temporal logic based forward chaining planner. *Annals of Mathematics and Artificial Intelligence*, 30(1-4):119–169, 2000.

[37] E. Lamma, M. Milano, and P. Mello. Extending constraint logic programming for temporal reasoning. *Annals of Mathematics and Artificial Intelligence*, 22(1-2):139–158, 1998.

[38] H. J. Levesque, R. Reiter, Y. Lesperance, F. Lin, and R. B. Scherl. GOLOG: A logic programming langauge for dynamic domains. *Journal of Logic Programming*, 31:59–84, 1997.

[39] Hector Levesque and Gerhard Lakemeyer. *Handbook of Knowledge Representation*, chapter Cognitive Robotics. Elsevier, 2007.

[40] Hector J. Levesque. What is planning in the presence of sensing? In *AAAI/IAAI, Vol. 2*, pages 1139–1146, 1996.

[41] Hector J. Levesque, Fiora Pirri, and Raymond Reiter. Foundations for the situation calculus. *Electron. Trans. Artif. Intell.*, 2:159–178, 1998.

[42] H.J. Levesque. Knowledge, action, and ability in the situation calculus. In *Proceedings of TARK-94*, pages 1–4. Morgan Kaufmann, 1994.

[43] F. Lin and R. Reiter. State constraints revisited. *Journal of Logic and Computation*, 5(4):655–677, 1994.

[44] M. Magnusson and P. Doherty. Deductive planning with temporal constraints. In *Proceedings of Commonsense-2007*, 2007.

[45] U. Mayr and SW. Keele. Changing internal constraints on action: the role of backward inhibition. *Journal of Experimental Psychology*, 129(1):4–26, 2000.

[46] J. McCarthy. Situations, actions and causal laws. Technical report, Stanford University, 1963. Reprinted in Semantic Information Processing (M. Minsky ed.), MIT Press, Cambridge, Mass., 1968, pp. 410-417.

[47] Itay Meiri. Combining qualitative and quantitative constraints in temporal reasoning. *Artif. Intell.*, 87(1-2):343–385, 1996.

[48] E.K. Miller and J.D. Cohen. An integrative theory of prefrontal cortex function. *Annual Rev. Neuroscience*, 24:167 – 202, 2007.

[49] R. Miller and M. Shanahan. Narratives in the situation calculus. *Journal of Logic and Computation*, 4(5):513–530, 1994.

[50] N. Muscettola, G. A. Dorais, C. Fry, R. Levinson, and C. Plaunt. Idea: Planning at the core of autonomous reactive agents. In *Proc. of NASA Workshop on Planning and Scheduling for Space*, 2002.

[51] Nicola Muscettola. Hsts: Integrating planning and scheduling. *Intelligent Scheduling*, pages 451–461, 1994.

[52] Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations: a maximal tractable subclass of allen's interval algebra. *J. ACM*, 42(1):43–66, 1995.

[53] A. Newell. *Unified theories of cognition*. Harvard University Press, 1990.

[54] D. A. Norman and T. Shallice. *Consciousness and Self-Regulation: Advances in Research and Theory*, volume 4, chapter Attention to action: Willed and automatic control of behaviour. Plenum Press, 1986.

[55] Andrea Philipp and Iring Koch. Task inhibition and task repetition in task switching. *The European Journal of Cognitive Psychology*, 18(4):624–639, 2006.

[56] J. Pinto. Occurrences and narratives as constraints in the branching structure of the situation calculus. *Journal of Logic and Computation*, 8(6):777–808, 1998.

[57] J. Pinto and R. Reiter. Reasoning about time in the situation calculus. *Annals of Mathematics and Artificial Intelligence*, 14:2510–268, 1995.

[58] J.A. Pinto and R. Reiter. Reasoning about time in the situation calculus. *Annals of Mathematics and Artificial Intelligence*, 14(2-4):251–268, September 1995.

[59] F. Pirri and A. Finzi. An approach to perception in theory of actions: Part i. *Electron. Trans. Artif. Intell.*, 3(C):19–61, 1999.

[60] F. Pirri and R. Reiter. Planning with natural actions in the situation calculus. *Logic-based artificial intelligence*, pages 213–231, 2000.

[61] Fiora Pirri and Ray Reiter. Some contributions to the metatheory of the situation calculus. *Journal of ACM*, 46(3):325–361, 1999.

[62] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380. Academic Press, San Diego, CA, 1991.

[63] R. Reiter. Natural actions, concurrency and continuous time in the situation calculus. In *Proceedings of KR'96*, pages 2–13, 1996.

[64] R. Reiter. Sequential, temporal GOLOG. In *Proceedings of KR'98*, pages 547–556, 1998.

[65] R. Reiter. *Knowledge in action : logical foundations for specifying and implementing dynamical systems*. MIT Press, 2001.

[66] R. Reiter and Z. Yuhua. Scheduling in the situation calculus: A case study. *Annals of Mathematics and Artificial Intelligence*, 21(2-4):397–421, 1997.

[67] J.S. Rubinstein, E.D. Meyer, and J. E. Evans. Executive control of cognitive processes in task switching. *Journal of Experimental Psychology: Human Perception and Performance*, 27(4):763–797, 2001.

[68] Erik Sandewall. *Features and fluents (vol. 1): the representation of knowledge about dynamical systems*. Oxford University Press, Inc., 1994.

[69] E. Schwalb, K. Kask, and R. Dechter. Temporal reasoning with constraints on fluents and events. In *Proceedings of AAAI-1994*, pages 1067–1072, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence.

[70] E. Schwalb and L. Vila. Logic programming with temporal constraints. In *TIME '96: Proceedings of the 3rd Workshop on Temporal Representation and Reasoning (TIME'96)*, Washington, DC, USA, 1996. IEEE Computer Society.

[71] M.P. Shanahan. A cognitive architecture that combines internal simulation with a global workspace. *Consciousness and Cognition*, 15:433–449, 2006.

[72] Murray Shanahan. *Solving the frame problem: a mathematical investigation of the common sense law of inertia*. MIT Press, 1997.

[73] Murray Shanahan. The event calculus explained. In *Artificial Intelligence Today*, pages 409–430. 1999.

[74] A. Tate. *"I-N-OVA" and "I-N-CA", Representing Plans and other Synthesised Artifacts as a Set of Constraints*, pages 300–304. 2000.

[75] Michael Thielscher. FLUX: A logic programming method for reasoning agents. *Theory and Practice of Logic Programming*, 5(4–5):533–565, 2005.

[76] Michael Thielscher and Thomas Witkowski. The features-and-fluents semantics for the fluent calculus. In *KR*, pages 362–370, 2006.

[77] S. P. Tipper. Does negative priming reflect inhibitory mechanisms? a review and integration of conflicting views. *Quarterly Journal of Experimental Psychology*, 54:321 – 343, 2001.

[78] Marc B. Vilain and Henry A. Kautz. Constraint propagation algorithms for temporal reasoning. In *AAAI*, pages 377–382, 1986.

[79] H. Wang and K. J. Brown. Finite set theory, number theory and axioms of limitation. *Mathematische Annalen*, 164:26–29, 1966.

[80] B. Williams, M. Ingham, S. Chung, P. Elliott, M. Hofbaur, and G. Sullivan. Model-based programming of fault-aware systems. *AI Magazine*, Winter 2003.

Appendix A

A Notational conventions and preliminaries

We recall that the set of axioms Σ (see also Table 1) is:

$$\begin{array}{ll} 1. & \neg(s \sqsubset S_0), \\ 2. & s \sqsubset do(a, s') \equiv s \sqsubseteq s', \\ 3. & do(a, s) = do(a', s') \equiv a = a' \wedge s = s', \\ 4. & \forall P.P(S_0) \wedge \forall as.P(s) \rightarrow P(do(a, s)) \rightarrow \forall s P(s). \end{array} \tag{43}$$

$\mathcal{D}_{una}$ is the set of axioms of the form $A_i(\cdot) \neq A_j(\cdot)$ with A_i and A_j names of actions, and the set of axioms specifying that identical action terms must have the same arguments (see Section 3.1). The set $\Sigma \cup \mathcal{D}_{una}$ is satisfiable in some model $\mathcal{M}_0 = (D, I)$ in which the real line is interpreted as usual. In fact in order to introduce time (see [58]) we may assume that the signature of initial language $\mathcal{L}_{sitcalc}$ includes:

(i) All rational constants p/q, and the special symbols 0 and 1;

(ii) usual operators, such as $+$, $-$ and $\cdot$;

(iii) the relations $<$ and, being $=$ in the language, the defined relation $\leq =< \vee =$ and the defined relation $>$ standing for $(\neg \leq)$.

Thus Σ can be suitably extended to include all the axioms for the theory of reals (additive, multiplicative, order, least upper bound) and the axiom $\forall t.0 \leq t$ with t ranging over the reals. We also may assume that there always exists a structure $\mathcal{M}$ for the classical basic Situation Calculus, in which the real numbers have an intended interpretation.

For the next theorems we stipulate the following. Let $\mathcal{S}$ be the signature of standard Situation Calculus with equality, including symbols for actions, situations, $\sqsubseteq$, indexed symbols t_i for the rational numbers, and the symbols indicated in the above items (i-iii), then:

$$\begin{array}{ll} 1. & \mathcal{L}_1 \text{ is } \mathcal{L}_{sitcalc}, \text{ the language defined on } \mathcal{S}. \\ 2. & \mathcal{L}_2 \text{ is } \mathcal{L}_1 \text{ extended to the signature including the symbols for the terms mentioning } \textit{time}. \\ 3. & \mathcal{L}_3 \text{ is like } \mathcal{L}_2 \text{ extended to the signature including the symbols } H, \text{ a finite amount of new constants,} \\ & \text{of sort object, for types, and } =_\nu. \\ 4. & \mathcal{L}_4 \text{ is like } \mathcal{L}_3 \text{ extended to the signature including the symbols for the terms mentioning timelines and} \\ & \text{the terms of sort bag of timelines, i.e. the symbols } \mathscr{T}, \in_{\mathscr{S}}, =_{\mathscr{S}}, \cup_{\mathscr{S}}, \cap_{\mathscr{S}}, \subseteq_{\mathscr{S}} \text{ and the symbol } \mathscr{B}. \\ & \mathcal{L}_4 \text{ is } \mathcal{L}_{TFSC}, \text{ the language of the } \textit{Temporal Flexible Situation Calculus}. \end{array} \tag{44}$$

A formula φ of a language $\mathcal{L}$ is said to be restricted to the language $\mathcal{L}_Q$, over the signature Q, and denoted $\varphi_{\setminus \mathcal{L}_Q}$ if φ mentions only the symbols of Q.

Finally we recall two theorems of [61] that will be used in the next proofs.

Theorem 8 (Relative Satisfiability) *A basic action theory $\mathcal{D}$ is satisfiable iff $\mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ is.*

Theorem 9 (Regression) *Suppose W is a regressable formula of $\mathcal{L}_{sitcalc}$ and $\mathcal{D}$ is a basic theory of actions. Then $\mathcal{R}[W]$ is a formula uniform in S_0. Moreover,*

$$\mathcal{D} \models (\forall)(W \equiv \mathcal{R}[W]),$$

where $(\forall)\phi$ denotes the universal closure of the formula ϕ with respect to its free variables.

For the definition of the regression operator $\mathcal{R}$ we refer the reader for details to [61], see also equation (85).

Appendix B

B Proofs of Section 3

In the following we assume that the reals are axiomatised, as noted in A above, that the language includes countable many terms taking values in the reals and countable many constant symbols denoting the rational numbers, plus 0 and 1. We also assume that actions form their arguments in the domain Obj and $\mathbb{R}^+$.

B.1 Lemma 1-7

Lemma 1 *Σ_{time} is a conservative extension of Σ and any model of $\Sigma \cup \mathcal{D}_{una}$ can be extended to a model of $\Sigma_{time} \cup \mathcal{D}_{una}$.*

Proof of Lemma 1.

Let $\Sigma_{time} = \Sigma \cup Ax_0$, that is, the set formed by the foundational axioms of the Situation Calculus, given above, and the axioms $T1$-$T3$ (see Table1 page 9). We have to prove that $\Sigma_{time} \cup \mathcal{D}_{una}$ is a conservative extension of $\Sigma \cup \mathcal{D}_{una}$, that is, for any formula φ in the language $\mathcal{L}_1$ of $\Sigma \cup \mathcal{D}_{una}$:

$$\Sigma_{time} \cup \mathcal{D}_{una} \models \varphi \text{ iff } \Sigma \cup \mathcal{D}_{una} \models \varphi$$

Let $\mathcal{M}_0 = \langle D, I \rangle$ be a model of $\Sigma \cup \mathcal{D}_{una}$ with D including the positive real line (see paragraph A above). We define a structure $\mathcal{M}_1 = (D, I')$ for $\mathcal{L}_2$ having the same domain as $\mathcal{M}_0$ and with I' interpreting all the symbols of $\mathcal{L}_1$ like I, thus, in particular, we might assume that there is a specific term t_0 indicating the 0, and such that for all positive t, $(t_0 \leq t)^{(I,v)}$, and thus this is replicated in $\mathcal{M}_1, v$.

Now for the terms mentioning *time* the interpretation I' is specified as follows, for any assignment v to the free variables:

(1) $(time(S_0))^{(I',v)}$ is mapped to $t_0^{(I',v)}$.
(2) For each action $A(\vec{x}, t)$ we set $(time(A(\vec{x}, t)))^{(I',v)} = t^{(I',v)}$ iff $t > 0$.
(3) For all situation s and actions A:
$(time(do(A(\vec{x}, t), s)))^{(I',v)} = time(A(\vec{x}, t))^{(I',v)}$.

It follows that $T1$-$T3$ are satisfied in $\mathcal{M}_1$. This concludes the interpretation of *time*. We have shown that any structure for $\mathcal{L}_1$ which is a model of $\Sigma \cup \mathcal{D}_{una}$ can be suitably extended to the language $\mathcal{L}_2$ in so being a model of $\Sigma_{time} \cup \mathcal{D}_{una}$. Now, by monotonicity, if $\Sigma \cup \mathcal{D}_{una} \models \varphi_{\setminus \mathcal{L}_1}$ then $\Sigma_{time} \cup \mathcal{D}_{una} \models \varphi_{\setminus \mathcal{L}_1}$. For the other direction, suppose that $\Sigma_{time} \cup \mathcal{D}_{una} \models \varphi_{\setminus \mathcal{L}_1}$ and $\Sigma \cup \mathcal{D}_{una} \not\models \varphi_{\setminus \mathcal{L}_1}$, then there is a model $\mathcal{M}$ of $\Sigma \cup \mathcal{D}_{una}$ not satisfying $\varphi_{\setminus \mathcal{L}_1}$. Now, $\mathcal{M}$ can be extended to satisfy $\Sigma_{time} \cup \mathcal{D}_{una}$, hence we have a contradiction.

Lemma 2 *There exists a model $\mathcal{M}$ of $\Sigma_{time} \cup \mathcal{D}_{una}$ such that, for all s and s', $\mathcal{M}$ models:*

$$s \sqsubseteq s' \rightarrow time(s) \leq time(s') \tag{45}$$

Proof of Lemma 2. Let $\mathcal{M}_0 = (D, I)$ be a model of $\Sigma_{time} \cup \mathcal{D}_{una}$, using Lowenheim-Skolem theorem let $\mathcal{M}_1$ be a model elementary equivalent to $\mathcal{M}_0$ but with a countable domain. We build a new model $\mathcal{M}_2 = (D', I')$, having the same domain for Act and Obj as $\mathcal{M}_1$, and interpreting in these domains everything, like I. However, the domain of situations in D' is $D_{S_0} = \{S_0^{I,v}\} = \{[]\}$, that is, the domain of situations includes only the interpretation of the constant S_0, which is the usual one and it is like in I:

$$\begin{array}{lll} (1) & \mathcal{M}_1, v \models time(S_0) = t_0 & \text{iff } \mathcal{M}_2, v \models time(S_0) = t_0 \\ (2) & \mathcal{M}_1, v \models A(x,t) = t & \text{iff } \mathcal{M}_2, v \models A(x,t) = t \\ (3) & \mathcal{M}_1, v \models A(x,t) = A'(x,t) & \text{iff } \mathcal{M}_2, v \models A(x,t) = A'(x,t) \end{array} \tag{46}$$

The set of terms for the sort *Act* is countable, thus we can enumerate the terms of sort action, order them according to time $a_1, a_2, \ldots$, and consider the interpretation in $\mathcal{M}_2$, according to v, with respect to time along a chain as follows:

$$C = t_0^{(\mathcal{M}_2,v)} <^{(\mathcal{M}_2)} time(a_1)^{(\mathcal{M}_2,v)} \leq^{(\mathcal{M}_2)} time(a_2)^{(\mathcal{M}_2,v)} \leq^{(\mathcal{M}_2)} \cdots \leq^{(\mathcal{M}_2)} time(a_m)^{(\mathcal{M}_2,v)} \leq^{(\mathcal{M}_2)} \cdots \tag{47}$$

Here $\leq$ has the usual interpretation. Since C is countable because there are countable many terms in the language of sort *Act*, we can assume that each term $time(a_i)$ is suitably interpreted. Furthermore, being both $\mathcal{M}_1$ and $\mathcal{M}_2$ models of Σ_{time}, by axiom (T2), for all actions A, $t_0 < time(A(x,t))$. Now, given that the domain of sort situation is $D_{S_0} = \{S_0^{(\mathcal{M}_2)}\} = \{[]\}$, we shall build the following two sets:

$$\begin{array}{l} T_{t_i} = \{a^{(\mathcal{M}_2,v)} \in Act \mid time(a)^{(\mathcal{M}_2,v)} = t_i^{(\mathcal{M}_2,v)}, t_i^{(\mathcal{M}_2,v)} \text{ the } i\text{-th element in } C\} \\ D_{s_i} = \{[a_1^{\mathcal{M}_2,v}, \ldots, a_i^{\mathcal{M}_2,v}] \mid a_1^{(\mathcal{M}_2,v)}, \ldots, a_i^{(\mathcal{M}_2,v)} \in Act, a_i^{(\mathcal{M}_2,v)} \in T_{t_i} \text{ and } [a_1^{\mathcal{M}_2,v}, \ldots, a_{i-1}^{\mathcal{M}_2,v}] \in D_{s_{i-1}}\} \end{array} \tag{48}$$

Each D_{s_i} is countable. Taking the union of the D_{s_i}:

$$Sit = \bigcup_{i=0}^{\infty} D_{s_i} \tag{49}$$

We still get a countable set such that $D_{s_i} \subseteq Sit$ for all D_{s_i}. The interpretation of *do* can now be defined as usual on the sequences in Sit, the interpretation of $\sqsubset$ can be set also to be the usual one, given that the interpretation of each element in Sit is a finite sequence of elements of the domain *Act*. Thus we extend the interpretation I' of $\mathcal{M}_2$ to J accordingly. Indeed, let $\mathcal{M} = (D' \cup Sit, J)$:

$$\begin{array}{l} [a_1^{(J,v)}, \ldots, a_i^{(J,v)}] = do^{(J,v)}(a_i^{(J,v)}, [a_1^{(J,v)}, \ldots, a_{i-1}^{(J,v)}]) \\ s^{(J,v)} \sqsubset^J s'^{(J,v)} \text{ iff the sequence } s^{(J,v)} \text{ is a proper initial subsequence of } s'^{(J,v)} \end{array} \tag{50}$$

It follows, by the definition of the D_{s_i}, that if $[a_1^{(J,v)}, \ldots, a_p^{(J,v)}] = s^{(J,v)} \in D_{s_p}$ and $[a_1^{(J,v)}, \ldots, a_q^{(J,v)}] = s'^{(J,v)} \in D_{s_q}$, with $p < q$ then $s^{(J,v)} \sqsubset^J s'^{(J,v)}$, hence $time^I(a_p^{(J,v)}) < time^I(a_q^{(J,v)})$ since $a_P^{(J,v)} \in T_{t_p}$ and $a_q^{(J,v)} \in T_{t_q}$. Now, in the new structure $\mathcal{M}$ we can define, for each $[a_1^{(\mathcal{M}_2,v)}, \ldots, a_i^{(\mathcal{M}_2,v)}] \in D_{s_i} \subseteq Sit$:

$$time([a_1, \ldots, a_i])^{(\mathcal{M},v)} = time(do(a_i, s))^{(\mathcal{M},v)} = time(a_i)^{(\mathcal{M}_2,v)} \tag{51}$$

Hence $time(s) < time(s')$ and, clearly, $\mathcal{M} = (D' \cup Sit, J)$ is a model of $\Sigma_{time} \cup \mathcal{D}_{una}$.

Thus the claim holds. □

Lemma 3 *Let Ax_1 denote the axioms H1-H2 and $\Sigma_H = \Sigma_{time} \cup Ax_1$: Σ_H is a conservative extension of Σ_{time} and any model of $\Sigma_{time} \cup \mathcal{D}_{una}$ can be extended to a model of $\Sigma_H \cup \mathcal{D}_{una}$.*

Proof of Lemma 3. Let $\Sigma_H = \Sigma_{time} \cup Ax_1$ (see the axioms *H*1-*H*2 Table 1, page 9). We have to prove that $\Sigma_H \cup \mathcal{D}_{una}$ is a conservative extension of $\Sigma_{time} \cup \mathcal{D}_{una}$, that is, for any formula φ in the language $\mathcal{L}_2$ of $\Sigma_{time} \cup \mathcal{D}_{una}$:

$$\Sigma_H \cup \mathcal{D}_{una} \models \varphi \text{ iff } \Sigma_{time} \cup \mathcal{D}_{una} \models \varphi$$

Let $\mathcal{M}_0 = \langle D, I \rangle$ be a model of $\Sigma_{time} \cup \mathcal{D}_{una}$.

We extend I to I' to interpret the type predicates $H(i,a)$ and the relation $=_\nu$, between actions. Let $\mathcal{M}_1 = (D, I')$ be a structure having the same domain as $\mathcal{M}_0$ and I' will interpret all predicate symbols, and function symbols and constant of $\mathcal{L}_2$ as I, and for the extended language $\mathcal{L}_3$ we shall proceed with the following interpretation.

- We first consider the interpretation of the predicate $H(i,a)$. Here we shall only provide a partition of name types, as follows. We order the constants denoting types, namely $i_1, i_2, \ldots, i_m$ and the action names

$A_1, A_2, \ldots, A_n, \ldots$ and we define a mapping $f : A_p \mapsto (mod(p-1, m)+1)$, with $p \geq 1$ and m the number of constants denoting types, so that each action name is assigned precisely to a single type.

Now, for any assignment v:

$$\mathcal{M}_0, v \models a = A_p(\vec{x}) \text{ iff } \mathcal{M}_1, v \models a = A_p(\vec{x})$$

We thus set

$$\langle i_k, A_p(\vec{x})\rangle^{(I',v)} \in H^{I'} \text{ iff } f(A_p) = k = (mod(p-1, m)+1)$$

It follows that

$$\text{if } \mathcal{M}_0, v \models A_p(\vec{y}) = A_p(\vec{x}) \text{ then } \mathcal{M}_1, v \models H(i_k, A_p(\vec{x})) \wedge H(i_k, A_p(\vec{y})) \text{ for } f(A_p) = k = (mod(p-1, m)+1)$$

- Next we consider the interpretation of $=_\nu$, for any assignment v, as follows:

$$\langle A_p(\vec{y}), A_q(\vec{x})\rangle^{(I',v)} \in {=_\nu}^{I'} \text{ iff } \mathcal{M}_1, v \models H(i_k, A_p(\vec{y})) \wedge H(i_k, A_q(\vec{x}))$$

By the construction it follows that:

1. if $\mathcal{M}_0, v \models A(\vec{x}) = A(\vec{y})$ then $\mathcal{M}_1, v \models A(\vec{x}) = A(\vec{y})$ and $\mathcal{M}_1, v \models A(\vec{x}) =_\nu A(\vec{y})$
2. $\mathcal{M}_0, v \models A(\vec{x}) \neq B(\vec{y})$ iff $\mathcal{M}_1, v \models A(\vec{x}) \neq B(\vec{y})$
3. $H^{I'}(i^{I'}, A_p^{I'}(\vec{x}^{\nu})) \cap H^{I'}(j^{I'}, A_q^{I'}(\vec{x}^{\nu})) = \emptyset$ iff $i^{I'} \neq j^{I'}$.

Hence $H1$-$H2 \cup \mathcal{D}_{una}$ are verified in $\mathcal{M}_1$.

Now, by an analogous argument as in Lemma 1 we obtain that $\Sigma_H \cup \mathcal{D}_{una}$ is a conservative extension of $\Sigma_{time} \cup \mathcal{D}_{una}$.

□

Lemma 4 *The relation $=_\nu$ is an equivalence relation on the set of actions.*

Proof of Lemma 4. That the relation $=_\nu$ is reflexive and symmetric follows from $(H2)$ and the property of $\wedge$. And the same for transitivity:

1.	$a=_\nu a' \wedge a'=_\nu a'' \rightarrow H(i,a) \wedge H(i,a') \wedge H(i,a'')$	(by (H2))	
2.	$H(i,a) \wedge H(i,a'') \rightarrow a=_\nu a''$	(by (H2))	(52)
3.	$a=_\nu a' \wedge a'=_\nu a'' \rightarrow a=_\nu a''$	(by 1, 2 and Taut.)	

Hence $=_\nu$ is a reflexive, transitive and symmetric relation on the set of actions partitioned by $(H1)$, i.e. it is an equivalence relation. □

Lemma 5 *Let Ax_2 denote the set of four axioms $E1$-$E4$ and $\Sigma_{=_\nu} = \Sigma_H \cup Ax_2$. Any model of $\Sigma_H \cup \mathcal{D}_{una}$ can be extended to a model of $\Sigma_{=_\nu} \cup \mathcal{D}_{una}$.*

Proof of Lemma 5. Let $\Sigma_{=_\nu} = \Sigma_H \cup Ax_2$, we have to prove that $\Sigma_{=_\nu} \cup \mathcal{D}_{una}$ is satisfiable iff $\Sigma_H \cup \mathcal{D}_{una}$ is.

Let $\mathcal{M}_1 = (D, I)$ be a model of $\Sigma_H \cup \mathcal{D}_{una}$, $\mathcal{M}_1$ exists according to Lemma 3. Let $\mathcal{M} = (D, I')$, where I' interprets all symbols like I and $=_\nu$, on all actions, like I. We thus manage to extend I' to interpret $=_\nu$ also on situations as follows:

(a) if $\mathcal{M}_1, v \models s = s'$ then $\langle s, s'\rangle^{(I',v)} \in =_\nu{}^{I'}$

(b) if $\mathcal{M}_1, v \models \neg(s = S_0)$ then $(\langle s, S_0\rangle^{(I',v)}, \langle S_0, s\rangle^{(I',v)}) \notin =_\nu{}^{I'}$

(c) if $\mathcal{M}_1, v \models (a{=_\nu}a')$ then both $\langle a, do(a', S_0)\rangle^{(I',v)} \in =_\nu{}^{I'}$ and $\langle a', do(a, S_0)\rangle^{(I',v)} \in =_\nu{}^{I'}$

(d) if $\mathcal{M}_1, v \models \neg(a{=_\nu}a')$ then both $\langle a, do(a', S_0)\rangle^{(I',v)} \notin =_\nu{}^{I'}$ and $\langle a', do(a, S_0)\rangle^{(I',v)} \notin =_\nu{}^{I'}$

The interpretation I' can thus be extended for any assignment v to all situations as follows:

(e) if $\mathcal{M}_1, v \models (a{=_\nu}a')$ and $\langle s, a'\rangle^{(I',v)} \in =_\nu{}^{I'}$ then $\langle a, do(a', s)\rangle^{(I',v)} \in =_\nu{}^{I'}$

(f) if $\mathcal{M}_1, v \models \neg(a{=_\nu}a')$ or $\langle s, a'\rangle^{(I',v)} \notin =_\nu{}^{I'}$ then $\langle a, do(a', s)\rangle^{(I',v)} \notin =_\nu{}^{I'}$

(g) if $\mathcal{M}_1, v \models (s{=_\nu}s')$ and $\langle s, a'\rangle^{(I',v)} \in =_\nu{}^{I'}$ then $\langle s, do(a', s')\rangle^{(I',v)} \in =_\nu{}^{I'}$

(h) if $\mathcal{M}_1, v \models \neg(s{=_\nu}s')$ or $\langle a', s\rangle^{(I',v)} \notin =_\nu{}^{I'}$ then $\langle s, do(a', s')\rangle^{(I',v)} \notin =_\nu{}^{I'}$

(i) if $(a{=_\nu}s)\rangle^{(I',v)} \in =_\nu{}^{I'}$ then $\langle s, a\rangle^{(I',v)} \in =_\nu{}^{I'}$

This concludes the extension of I to I'. The construction implies that $\mathcal{M} = (D, I')$ is a model of ($H1$-$H2$, $E1$-$E5$). □

Lemma 6 *Let $=_\nu$ be a relation on the terms of sort actions and situations. Then $=_\nu$ is an equivalence relation both on situations and on actions and situations.*

Proof of Lemma 6.

First note that the relation $=_\nu$ is reflexive both on situations and on actions. By axiom E5 it is symmetric on action and situations. We show that it is symmetric and transitive on situations, likewise that it is transitive over actions and situations:

$$\begin{array}{lll} \text{Basic case:} & \text{symmetry } s & \\ & s{=_\nu}S_0 \equiv S_0{=_\nu}s & \text{(By E1.)} \end{array} \tag{53}$$

Let, now, $s, s' \sqsupset S_0$, we shall first show:

$$(a).\quad do(a, s){=_\nu}do(a', s') \quad\equiv\quad a{=_\nu}a' \wedge s{=_\nu}s' \wedge s'{=_\nu}a \wedge a'{=_\nu}s \tag{54}$$

Indeed:

$$\begin{array}{llll} 3.1 & do(a, s){=_\nu}do(a', s') & \equiv & do(a, s){=_\nu}a' \wedge s'{=_\nu}do(a, s) \quad \text{(By E4)} \\ 3.2 & do(a, s){=_\nu}a' & \equiv & a'{=_\nu}do(a, s) \quad \text{(By E5)} \\ 3.3 & a'{=_\nu}do(a, s) & \equiv & a{=_\nu}a' \wedge (s{=_\nu}a') \quad \text{(By E4)} \\ 3.4 & s'{=_\nu}do(a, s) & \equiv & s'{=_\nu}a \wedge s'{=_\nu}s \quad \text{(By E4)} \\ 3.5 & do(a, s){=_\nu}do(a', s') & \equiv & (a{=_\nu}a') \wedge (s{=_\nu}s') \wedge (s'{=_\nu}a) \wedge (a'{=_\nu}s) \quad \text{(By 3.2, 3.3, 3.4, E5 and Ind. Hyp.)} \end{array} \tag{55}$$

We can thus show symmetry for situations:

$$\begin{array}{lllll} \text{symmetry-s :} & do(a, s){=_\nu}do(a', s') & \equiv & do(a', s'){=_\nu}do(a, s) & \\ & do(a, s){=_\nu}do(a', s') & \equiv & (a{=_\nu}a') \wedge (s{=_\nu}s') \wedge (s'{=_\nu}a) \wedge (a'{=_\nu}s) & \text{(By 3)} \\ & & \equiv & (a'{=_\nu}a) \wedge (s'{=_\nu}s) \wedge (s{=_\nu}a') \wedge (a{=_\nu}s') & \text{(By Ind. Hyp.)} \\ & & \equiv & do(a', s'){=_\nu}do(a, s) & \end{array} \tag{56}$$

We shall, now, show transitivity for action and situations, here (symm) shall refer to both (E5) and symmetry-s:

$$T1.\quad a{=_\nu}s \wedge s{=_\nu}s' \rightarrow a{=_\nu}s'. \tag{57}$$

For either $s = S_0$ or $s' = S_0$ or both, it is trivially true, by (E2). Let $s, s' \sqsupset S_0$

1.	$a=_\nu s \wedge s=_\nu s' \wedge a = a \rightarrow a=_\nu s' \wedge s'=_\nu s \wedge a=_\nu a$	(By (E1) and (E5))	
2.	$a=_\nu s \wedge s'=_\nu s \wedge a=_\nu a \rightarrow do(a, s')=_\nu do(a, s)$	(by (a))	
3.	$do(a, s')=_\nu do(a, s) \rightarrow a = do(a, s') \wedge s=_\nu do(a, s')$	(by (E4))	(58)
4.	$a = do(a, s') \rightarrow a=_\nu a \wedge a=_\nu s'$	(by (E3) and (symm.))	
5.	$a=_\nu s \wedge s=_\nu s' \rightarrow a=_\nu s'$	(by 1,4 and Taut)	

$T2.$ $a'=_\nu s \wedge s=_\nu s' \wedge s'=_\nu a \rightarrow a'=_\nu a.$ (59)

1.	$a'=_\nu s \wedge s'=_\nu s \rightarrow s=_\nu do(a', s')$	(By (E4) and (symm))	
2.	$a=_\nu s' \wedge s'=_\nu s \rightarrow a=_\nu s$	(by T1. and (symm.))	
3.	$a=_\nu s \wedge s=_\nu do(a', s') \rightarrow a=_\nu do(a', s')$	(by 1, 2 and T1.)	(60)
4.	$a=_\nu do(a', s') \rightarrow a=_\nu a' \wedge a=_\nu s'$	(by (E4))	
5.	$a'=_\nu s \wedge s=_\nu s' \wedge s'=_\nu a \rightarrow a'=_\nu a$	(by 1,2, 4, (symm.) and Taut)	

Similarly, from (a), (E3), (E4) and (E5) it is possible to prove that

$T3.$ $a=_\nu a' \wedge a'=_\nu s \rightarrow a=_\nu s.$ (61)

Finally transitivity for situations can be shown by induction on s. For $s = S_0$ and $s'' = S_0$:

1.	$do(a, S_0)=_\nu do(a', s') \wedge do(a', s')=_\nu do(a'', S_0) \rightarrow a=_\nu a''$	(By (a) above and Lemma 4)	
2.	$a=_\nu a'' \rightarrow a''=_\nu do(a, S_0)$	(by (E3))	
3.	$a''=_\nu do(a, S_0) \rightarrow do(a, S_0)=_\nu do(a'', S_0)$	(by (E4))	(62)
4.	$do(a, S_0)=_\nu do(a', s') \wedge do(a', s')=_\nu do(a'', S_0) \rightarrow do(a, S_0)=_\nu do(a'', S_0)$	(by 1, 3 and Taut.)	

For $s \sqsupset S_0$:

1.	$do(a, s)=_\nu do(a', s') \wedge do(a', s')=_\nu do(a'', s'') \rightarrow a=_\nu s' \wedge a'=_\nu s \wedge a'=_\nu s'' \wedge a''=_\nu s'$	(By (a))	
2.	$s=_\nu s' \wedge s'=_\nu s'' \rightarrow s=_\nu s''$	(by Ind. Hyp.)	
3.	$a=_\nu s' \wedge s'=_\nu s'' \rightarrow a=_\nu s''$	(by (a))	(63)
4.	$a=_\nu a'' \wedge s=_\nu s'' \wedge a=_\nu s'' \rightarrow do(a, s)=_\nu do(a'', s'')$	(by (d))	
5.	$do(a, s)=_\nu do(a', s') \wedge do(a', s')=_\nu do(a'', s'') \rightarrow do(a, s)=_\nu do(a'', s'')$	(by 5 and Taut.)	

We have thus shown that $=_\nu$ is an equivalence relation on the set of actions and situations.

□

B.2 Proof of Theorem 1

We have to show that $\Sigma \cup \mathcal{D}_{una}$ together with the set of axioms $Ax_0 - Ax_2$, that is, ($T1$-$T3$, $H1$-$H2$, $E1$-$E5$) forms a satisfiable set. We have shown, incrementally that $\Sigma_{time} = \Sigma \cup Ax_0$ is a conservative extension, Lemma 1, that $\Sigma_H = \Sigma_{time} \cup Ax_1$ conservatively extends Σ_{time}, Lemma 3, and that $\Sigma_{=_\nu} = \Sigma_H \cup Ax_2$ is a conservative extension of Σ_H, Lemma 5. And, in particular, that all are conservative extensions of $\Sigma \cup \mathcal{D}_{una}$. Hence any model of $\Sigma \cup \mathcal{D}_{una}$ can be extended to a model of $\Sigma \cup Ax_0 \cup Ax_1 \cup Ax_2$.

□

B.3 Proof of Corollary 1

By Theorem 1 we know that $\Sigma_{=_\nu} \cup \mathcal{D}_{una}$ is satisfiable in some model $\mathcal{M}$ of $\mathcal{L}_3$. On the other hand the satisfiability of $\mathcal{D}_{S_0}$ and hence of $\Sigma_{=_\nu} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ depends on the design of $\mathcal{D}_{S_0}$ and, in particular, on the definition of $H(i,a)$, for each component i, which are in $\mathcal{D}_{S_0}$. If $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \cup \Sigma_{=_\nu}$ is satisfiable, then following the same arguments of the relative satisfiability theorem (see Theorem 8) a model $\mathcal{M}$ of $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \cup \Sigma_{=_\nu}$ can be easily extended to a model of $\Sigma_{=_\nu} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap}$. The other direction follows from the fact that a model of $\Sigma_{=_\nu} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap}$ is also a model of $\Sigma_{=_\nu} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$.

□

The only concern is given by the specifications of the $H(i,a)$ for each type i and each action A, mentioned in $\mathcal{D}_{una}$, in $\mathcal{D}_{S_0}$, whether there exists a model for $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \cup Ax_1$. If this model exists then using the previous theorem and lemmas this can be extended to a model of $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \cup \Sigma_{=_\nu}$.

So we may make some assumption on the definition of the $H(i,a)$ to show that, under these conditions, a model of $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \cup Ax_1$ exists.

Lemma 7 *Let $\mathcal{D}^-_{S_0} \cup \mathcal{D}_{una}$ be satisfiable in some model $\mathcal{M}$ of $\mathcal{L}_2$, let it be uniform in S_0 and not mentioning the predicate $H(\cdot)$. Then the definitions of $H(\cdot)$, for each type i, and action A referred to, in $\mathcal{D}_{una}$, can be safely added to $\mathcal{D}^-_{S_0} \cup \mathcal{D}_{una}$ in the form:*

$$\forall a.H(i,a) \equiv \varphi(i,a), \text{with } \varphi \text{ not mentioning } H(\cdot) \tag{64}$$

If there are formulas $\varphi(i,a)$ specifying actions and components, such that, for each i.

$$\begin{array}{ll} i. & \mathcal{D}^-_{S_0} \not\models \forall a.\varphi(i,a), \\ ii. & \mathcal{D}^-_{S_0} \models \exists a.\varphi_i(i,a) \wedge \bigwedge^{n}_{\substack{i\neq j \\ j=1}} \neg\varphi_j(j,a). \end{array} \tag{65}$$

then the extended $\mathcal{D}_{S_0}$ will satisfy, for all types i:

$$\begin{array}{ll} a. & H(i,a), \text{ in } \mathcal{D}_{S_0} \text{ occurs only in formulas of the form } \forall a.H(i,a) \equiv \phi(i,a), \text{with } \phi \text{ not mentioning } H(\cdot), \\ b. & \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \cup \{\forall a.\bigvee^n_{i=1} H(i,a)\} \cup \{\forall a.H(i,a) \rightarrow \bigwedge^{n}_{\substack{i\neq j \\ j=1}} \neg H(j,a)\} \\ & \quad \text{is satisfiable in some model} \mathcal{M}' \text{ of } \mathcal{L}_3 \end{array} \tag{66}$$

Proof. Assume that $\mathcal{D}^-_{S_0}$ is satisfiable in some structure $\mathcal{M}$ of $\mathcal{L}_2$ and, thus, it does not mention $H(\cdot)$. Then we extend the theory $\mathcal{D}_{S_0}$ to $\mathcal{L}_3$ according to the following construction.

First note that here we mention i as an element of the domain object, no axioms for types are assumed so far, although we can assume that there are n elements of the domain object specifying components (despite we use natural numbers to denote them). Define, similarly as in Lemma 3, an indexing function $i = mod(j-1,n)+1$ for action names A_j so that actions are grouped in such a way that $\mathcal{M} \not\models \forall a\varphi(i,a)$, with $\varphi(i,a)$ a suitable formula mentioning the A_j specified in $\mathcal{D}_{una}$, and satisfying the conditions of the Lemma. For example, if there is a finite set of action names ascribed to a component i, then $\varphi_i(i,a)$ is $\exists \vec{x_1}, \ldots, \vec{x_k} \bigvee^k_{j=1} a = A_j(\vec{x_j})$, as in (6), with the A_j suitably chosen with $i = mod(j-1,n)+1$, and it satisfies all the conditions of the lemma. Given $\mathcal{M}$, by the Lowenheim-Skolem theorem, there is a model $\mathcal{M}^{\sim}$ of $\mathcal{D}_{S_0} \cup \mathcal{D}_{una}$ which is elementary equivalent to $\mathcal{M}$ and has a countable domain. As usual we define a new structure $\mathcal{M}_1 = (D, I)$ for $\mathcal{L}_3$, with the same domain and the same interpretation as $\mathcal{M}^{\sim}$ on all symbols of $\mathcal{L}_2$. Furthermore $\mathcal{M}_1$ interprets all the new constants symbols that are added in the construction, as illustrated below in (67).

The construction with new constants is given according to the above specified enumeration of formulas $\varphi(i,a)$,

and the above specified conditions as follows:

$$\begin{array}{lcl}
\Delta_0 & = & \{\psi \mid \mathcal{M}^{\sim} \models \psi\} \\
\ldots & & \\
\Delta_i & = & \{H(i,a) \equiv \varphi(i,a) \mid \mathcal{M}^{\sim}, v \models \varphi(i,a) \text{ iff } \mathcal{M}_1, v \models H(i,a) \text{ and } \exists a.\varphi(i,a) \text{ not used in } \Delta_j, 0 < j < i\} \bigcup \\
 & & \{H(i,c) \mid \mathcal{M}_1, v \models H(i,a) \wedge \varphi(i,a), a^v = \overline{d} = c^I,\ c \text{ a fresh constant symbol}\} \bigcup \\
 & & \{\neg H(i,c) \mid \mathcal{M}_1, v \models \neg\varphi(i,a) \wedge \neg H(i,a), a^v = \overline{d} = c^I,\ c \text{ a fresh constant symbol}\} \bigcup \\
 & & \{H(i,c) \rightarrow \bigwedge_{\substack{j=1 \\ j\neq i}}^{n} \neg H(j,c) \mid \mathcal{M}_1, v \models \bigwedge_{\substack{j=1 \\ j\neq i}}^{n} \neg\varphi_j(j,a) \wedge \varphi_i(i,a), , a^v = \overline{d} = c^I,\ c \text{ a fresh constant symbol}\}
\end{array} \tag{67}$$

Each Δ_i, $0 \leq i \leq n$ is satisfiable in $\mathcal{M}_1$, by construction, furthermore

$$\mathbb{D} = \bigcup_{i=0}^{n} \Delta_i \tag{68}$$

is satisfiable in $\mathcal{M}_1$ and H is complete in $\mathbb{D}$, which is the diagram of H, in $\mathcal{M}_1$, that is, $H(i,c) \in \mathbb{D}$ iff $\mathcal{M}_1, v \models H(i,a)$ and $\neg H(i,c) \in \mathbb{D}$ iff $\mathcal{M}_1, v \models \neg H(i,a)$, with $a^v = \overline{d} = c^I$. By the constraints on each $\varphi(i,a)$ in the enumeration, $\mathcal{M}^{\sim} \not\models \forall a.\varphi(i,a)$, and $\mathcal{M}^{\sim}, v \models \varphi_i(i,a) \wedge \bigwedge_{\substack{j=1 \\ j\neq i}}^{n} \neg\varphi_j(j,a)$ hence $H(i,c) \rightarrow \bigwedge_{\substack{j=1 \\ j\neq i}}^{n} \neg H(j,c) \in \Delta_i$. It remains to show that $\bigwedge_{i=1}^{n} \neg H(i,c) \notin \Delta_i$. But that $\bigwedge_{i=1}^{n} \neg H(i,c) \in \Delta_i$ is impossible, since for each Δ_i all the added constants are fresh hence if $\neg H(i,c) \in \Delta_i$, then $\neg H(j,c) \notin \mathbb{D}$, $i \neq j$. Also because if $\bigwedge_{\substack{j=1 \\ j\neq i}}^{n} \neg H(j,c) \in \Delta_i$, then by the condition on the subset, it must be that $H(i,c)$ hence $\bigwedge_{i=1}^{n} \neg H(i,c) \notin \Delta_i$.

It follows that $\mathcal{M}_1 \not\models \exists a. \bigwedge_{i=1}^{n} \neg H(i,a)$ and since $\mathcal{M}_1$ is also a model of $\mathcal{D}_{S_0}$ it follows that $\mathcal{D}_{S_0} \not\models \exists a. \bigwedge_{i=1}^{n} \neg H(i,a)$. On the other hand $H(i,c) \rightarrow \bigwedge_{\substack{j=1 \\ j\neq i}}^{n} \neg H(j,c) \in \Delta_i$, for each i, hence $\mathbb{D} \models H(i,c) \rightarrow \bigwedge_{\substack{j=1 \\ j\neq i}}^{n} \neg H(j,c)$ for each i with c new constants, hence $\mathcal{M}_1 \models \forall a.H(i,a) \rightarrow \bigwedge_{\substack{j=1 \\ j\neq i}}^{n} \neg H(j,a)$. Thus $\mathcal{D}_{S_0} \bigcup \mathcal{D}_{una} \bigcup \{\forall a.H(i,a) \rightarrow \bigwedge_{\substack{j=1 \\ j\neq i}}^{n} \neg H(j,a)\} \bigcup \{\forall a. \bigvee_{i=1}^{n} H(i,a)\}$ is satisfiable in $\mathcal{M}_1$.

Therefore, under the conditions (65), $\mathcal{M}_1 \models H1$. Following again Lemma 3 the construction can lead to a model for $\mathcal{D}_{S_0} \cup \mathcal{D}_{una} \cup Ax_1$.

□

B.4 Proof of Theorem 2

Theorem 2 *A timeline represents the $=_\nu$-equivalence class of situations of the same type.*

Proof of the theorem. Recall that a timeline is defined by an (*improper*) successor state axiom as follows:

$$\mathcal{T}(i, do(a,s)) \equiv (s \sqsupset S_0 \wedge a =_\nu s \wedge \mathcal{T}(i,s)) \vee (s = S_0 \wedge H(i,a)). \tag{69}$$

We show that a timeline corresponds to an $=_\nu$-equivalence class. Define:

$$[\ do(a,s)\ ] = \{do(a',s') \mid do(a,s) =_\nu do(a',s')\} \tag{70}$$

$[\ do(a,s)]$ is an $=_\nu$-equivalence class because $=_\nu$ is an equivalence relation on actions and situations (see Lemma 6). We show, by induction on s' that:

$$\begin{array}{l}
do(a',s') \in [do(a,s)] \text{ iff } \exists i.\mathcal{T}(i, do(a',s')) \wedge \mathcal{T}(i, do(a,s)). \\
\text{By definition of the equivalence class, it implies that we show} \\
do(a,s) =_\nu do(a',s') \equiv \exists i.\mathcal{T}(i, do(a',s')) \wedge \mathcal{T}(i, do(a,s)).
\end{array} \tag{71}$$

Basic case $s' = S_0$

$\Rightarrow$ A. $s' = S_0$ and $s = S_0$

1. $do(a', S_0) =_\nu do(a, S_0) \rightarrow a =_\nu a'$ (By (a), Lemma 6)
2. $a =_\nu a' \rightarrow \exists i.H(i, a) \wedge H(i, a')$ (By (H2))
3. $H(i, a) \wedge H(i, a') \wedge s' = S_0 \wedge s = S_0 \rightarrow$ $\exists i.\mathscr{T}(i, do(a', S_0)) \wedge \mathscr{T}(i, do(a, S_0))$ (By (W1)) (72)
4. $do(a', S_0) =_\nu do(a, S_0) \rightarrow \exists i.\mathscr{T}(i, do(a', S_0)) \wedge \mathscr{T}(i, do(a, S_0))$ (By A1, A3 and Taut.)

B. $s' = S_0$ and $s \sqsupset S_0$ and the Ind. Hyp. is $s =_\nu do(a', S_0) \rightarrow \exists i.\mathscr{T}(i, do(a', S_0)) \wedge \mathscr{T}(i, s)$, for $s \in [do(a, s)]$.

1. $do(a', S_0) =_\nu do(a, s) \rightarrow a =_\nu do(a', S_0) \wedge s =_\nu do(a', S_0)$ (By (E4))
2. $s =_\nu do(a', S_0) \rightarrow \exists i.\mathscr{T}(i, do(a', S_0)) \wedge \mathscr{T}(i, s)$ (By Ind. Hyp.)
3. $a =_\nu do(a', S_0) \rightarrow a =_\nu a'$ (By (E3).)
4. $s =_\nu do(a', S_0) \rightarrow a' =_\nu s$ (By (E4)) (73)
5. $a' =_\nu a' \wedge a' =_\nu s \rightarrow a =_\nu s$ (By (T3) Lemma 6)
6. $\exists i.\mathscr{T}(i, do(a', S_0)) \wedge \mathscr{T}(i, s) \wedge a =_\nu s \rightarrow$ $\exists i.\mathscr{T}(i, do(a', S_0)) \wedge \mathscr{T}(i, do(a, s))$ (By 2 and (W1))
7. $do(a', S_0) =_\nu do(a, s) \rightarrow \exists i.\mathscr{T}(i, do(a', S_0)) \wedge \mathscr{T}(i, do(a, s))$ (By 1, 6 and Taut.)

$\Leftarrow$ C. $s' = S_0$ and $s = S_0$

1. $\exists i.\mathscr{T}(i, do(a', S_0)) \wedge \mathscr{T}(i, do(a, S_0)) \rightarrow \exists i.H(i, a') \wedge H(i, a)$ (By (W1))
2. $\exists i.H(i, a) \wedge H(i, a') \rightarrow a =_\nu a'$ (By (H2))
3. $a' =_\nu a \wedge s = S_0 \rightarrow a' =_\nu do(a, S_0)$ (By (E3)) (74)
4. $a' =_\nu do(a, S_0) \rightarrow do(a, S_0) =_\nu do(a', S_0)$ (By (E4))
5. $\exists i.\mathscr{T}(i, do(a', S_0)) \wedge \mathscr{T}(i, do(a, S_0)) \rightarrow do(a, S_0) =_\nu do(a', S_0)$ (By 1,4 and Taut.)

D. $s' = S_0$ and $s \sqsupset S_0$, the Ind. Hyp. is $\exists i.\mathscr{T}(i, do(a', S_0)) \wedge \mathscr{T}(i, s) \rightarrow s =_\nu do(a', S_0)$

1. $\exists i.\mathscr{T}(i, do(a', S_0)) \wedge \mathscr{T}(i, do(a, s)) \rightarrow$ $\exists i.\mathscr{T}(i, do(a', S_0)) \wedge a =_\nu s \wedge \mathscr{T}(i, s)$ (By (W1))
2. $\exists i.\mathscr{T}(i, do(a', S_0)) \wedge a =_\nu s \wedge \mathscr{T}(i, s) \rightarrow s =_\nu do(a', S_0)$ (By Ind. Hyp.)
3. $a =_\nu s \wedge s =_\nu do(a', S_0) \rightarrow a =_\nu do(a', S_0)$ (By (E3) and symm.) (75)
4. $a =_\nu do(a', S_0) \wedge s =_\nu do(a', S_0) \rightarrow do(a, s) =_\nu do(a', S_0)$ (By (E4))
5. $\exists i.\mathscr{T}(i, do(a', S_0)) \wedge \mathscr{T}(i, do(a, s)) \rightarrow do(a', S_0) =_\nu do(a, s)$ (By 1,4 and Taut.)

Induction $s' \sqsupset S_0$ (we may also assume w.l.g $s \sqsupset S_0$, otherwise it can be reduced to the above basic cases.)

$\Rightarrow$ The Ind. Hyp. is $s =_\nu s' \rightarrow \exists i.\mathscr{T}(i, s) \wedge \mathscr{T}(i, s')$ with $s' \in [s]$.

1. $do(a', s') =_\nu do(a, s) \rightarrow a =_\nu a' \wedge s =_\nu s' \wedge a =_\nu s' \wedge a' =_\nu s$ (By (b,c), Lemma 6)
2. $a' =_\nu s \wedge s =_\nu s' \rightarrow a' =_\nu s'$ (By (T1), Lemma 6)
3. $a =_\nu s' \wedge s' =_\nu s \rightarrow a =_\nu s$ (By (T1), Lemma 6)
4. $s =_\nu s' \rightarrow \exists i.\mathscr{T}(i, s') \wedge \mathscr{T}(i, s)$ (By Ind. Hyp. and 1) (76)
5. $\exists i.\mathscr{T}(i, s') \wedge \mathscr{T}(i, s) \wedge a =_\nu s \wedge a' =_\nu s' \rightarrow$ $\exists i.\mathscr{T}(i, do(a, s)) \wedge \mathscr{T}(i, do(a', s'))$ (By (W1))
6. $do(a', s') =_\nu do(a, s) \rightarrow \exists i.\mathscr{T}(i, do(a, s)) \wedge \mathscr{T}(i, do(a', s'))$ (By 1, 5 and Taut.)

$\Leftarrow$ The induction hypothesis is $\exists i.\mathscr{T}(i, s) \wedge \mathscr{T}(i, s') \rightarrow s =_\nu s'$ with $s' \in [s]$.

1. $\exists i.\mathscr{T}(i, do(a', s')) \wedge \mathscr{T}(i, do(a', s')) \rightarrow$ $a' =_\nu s' \wedge a =_\nu s \wedge \mathscr{T}(i, s) \wedge \mathscr{T}(i, s')$ (By (W1))
2. $\exists i.\mathscr{T}(i, s) \wedge \mathscr{T}(i, s') \rightarrow s =_\nu s'$ (By Ind. Hyp.)
3. $a' =_\nu s' \wedge s' =_\nu s \wedge a =_\nu s \rightarrow a' =_\nu s \wedge a =_\nu s' \wedge a =_\nu a'$ (By (T1,T2), Lemma 6) (77)
4. $a =_\nu s' \wedge s =_\nu s' \wedge a =_\nu a' \rightarrow \rightarrow do(a, s) =_\nu do(a', s')$ (By (d), Lemma 6)
5. $\exists i.\mathscr{T}(i, do(a', s')) \wedge \mathscr{T}(i, do(a', s')) \rightarrow do(a, s) =_\nu do(a', s')$ (By 1,4 and Taut.)

We have, thus, shown that a timeline represents an equivalence class indexed by a type (that is by a component of the system). Furthermore it follows from the definition that S_0 does not belong to any equivalence class hence to no timeline. Thus any induction on timelines needs that the basic case is $do(a, S_0)$. □

B.5 Proof of Theorem 3

We have to show that the axioms $G1$-$G5$ have a model which is also a model of $\Sigma_{=_\nu} \cup \mathcal{D}_{una}$.

A structure for $\mathcal{L}_3$ which is a model for $\Sigma_{=_\nu} \cup \mathcal{D}_{una}$ has been provided in Theorem 1, furthermore, if we consider a satisfiable initial database $\mathcal{D}_{S_0}$ as shown in Corollary 1, there exists a model of $\Sigma_{=_\nu} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0}$ that can be extended to a model of $\mathcal{D} = \Sigma_{=_\nu} \cup \mathcal{D}_{una} \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{ap}$, with successor state axioms mentioning also timelines. Therefore let $\mathcal{M}_1 = (D, I_0)$ be such a model. We shall show that this model $\mathcal{M}_1$ can be extended to a model $\mathcal{M}_2$ in which we give an appropriate interpretation for the sort $\mathscr{S}$ of bags of timelines, and such that the axioms $G1$-$G5$ are satisfied. Let $\mathcal{M}_2 = (D \cup \{\mathscr{S}\}, I)$ i.e. $\mathcal{M}_2$ has the same domain as $\mathcal{M}_1$ for sorts object-including reals (time)- actions and situations, I is like I_0 for all symbols of the language $\mathcal{L}_3$ and it is extended to interpret the above mentioned elements in $\mathscr{S}$ according to the following steps.

Let $n \in \mathbb{N}$ and $\langle s_1, \ldots, s_n \rangle$ a n tuple of situations in Sit^n, $Sit \subset D$ and ν any valuation on the free variables.

$$\begin{array}{ll} 1. & \text{If } n = 0 \quad \mathscr{B}_0^I \in \mathscr{S}. \\ 2. & \text{If } n > 0 \quad \langle s_1^{(I,\nu)}, \ldots, s_n^{(I,\nu)} \rangle \in \mathscr{S} \text{ iff for each } s_k, k = 1, .., n, \mathcal{M}_1, \nu \models (s_k = S_0 \vee \exists i \mathscr{T}(i, s_k)) \end{array} \tag{78}$$

Now, each term $\mathfrak{s}^{(I,\nu)} \in \mathscr{S}$ is invariant to both permutations of the order of the tuple and compaction of repeated situations. That is, whenever p is a permutation of $\{1, \ldots, n\}$ and $\langle s_1, \ldots, s_n \rangle$ is a tuple of situations in Sit^n, $Sit \subset D$, then

$$\langle s_1^{(I,\nu)}, \ldots, s_n^{(I,\nu)} \rangle = \langle \kappa(s_{p1}^{(I,\nu)}, \ldots, s_{pi}^{(I,\nu)}, s_{pi}^{(I,\nu)}, \ldots, s_{pn}^{(I,\nu)}) \rangle_p^{(I,\nu)} \tag{79}$$

Here $\kappa : \mathscr{S} \mapsto \mathscr{S}$ indicates a compaction function on a tuple of the repeated k elements of a $n + k$ tuple and $\langle \cdot \rangle_p : \mathscr{S} \mapsto \mathscr{S}$ the permutation function on $\{1, \ldots, n\}$. More precisely, $\langle s_{p1}^{(I,\nu)}, \ldots s_{pn}^{(I,\nu)} \rangle_p$ has been obtained from $\kappa(\langle s_1^{(I,\nu)}, \ldots, s_{pi}^{(I,\nu)}, s_{pi}^{(I,\nu)}, \ldots, s_{pn}^{(I,\nu)} \rangle_p^{(I,\nu)})$ by compacting to a single representative the repeated arguments s_{pi}, and $\langle s_1^{(I,\nu)}, \ldots, s_n^{(I,\nu)} \rangle$ has been obtained from $\langle s_{p1}^{(I,\nu)}, \ldots, s_{pi}^{(I,\nu)}, \ldots s_{pn}^{(I,\nu)} \rangle_p^{(I,\nu)}$ by a permutation p of $\{1, \ldots, n\}$. In the language the permutation $\langle \cdot \rangle_p$ incorporates also the compaction κ.

[G1] Given the construction of terms of sort bag of timelines we can state the following, for $n \in \mathbb{N}$ and ν any assignment to the free variables:

$$\mathcal{M}_2, \nu \models s \in_{\mathscr{S}} \mathscr{B}(\langle s_{j1}, \ldots, s_{jn} \rangle_j) \quad \text{iff} \quad \mathcal{M}_1, \nu \models \bigvee_{1 \le k \le n} \left((s = s_{jk} \wedge \exists i. \mathscr{T}(i, s_{jk})) \vee S_0 = s_{jk} \right) \tag{80}$$

And since $\mathcal{M}_2$ is like $\mathcal{M}_1$ for $\mathcal{L}_3$ it follows that (G1) is satisfied in $\mathcal{M}_2$.

We can now generalise the membership relation over bags of situations as follows:

$$\begin{array}{lll} & \text{For all } s^{I,\nu} \in Sit \subset D, & \\ 1. & If\ \mathfrak{s}^{(I,\nu)} = \mathscr{B}_0^I & \text{then } \mathcal{M}_2, \nu \models s \notin_{\mathscr{S}} \mathfrak{s} \\ 2 & \text{Otherwise} & \\ & \mathcal{M}_2, \nu \models s \in_{\mathscr{S}} \mathfrak{s} & \text{iff there is a } n \in \mathbb{N} \text{ and a } n \text{ tuple } \langle s_1^{(I,\nu)}, \ldots s_n^{(I,\nu)} \rangle, \text{of elements of } Sit \subset D \text{ s.t.} \\ & & \mathfrak{s}^{(I,\nu)} = \langle \kappa(s_{p1}^{(I,\nu)}, \ldots s_{pn}^{(I,\nu)}) \rangle_p^{(I,\nu)}, \text{ for some } k = 1, \ldots, n, s^{(I,\nu)} = s_{pk}^{(I,\nu)}, \text{ and} \\ & & \mathcal{M}_2, \nu \models \exists i. \mathscr{T}(i, s) \text{ iff } s^{I,\nu} \neq S_0^I \end{array} \tag{81}$$

Now we have given an interpretation in $\mathcal{M}_2$ to $\in_{\mathscr{S}}$ and built the terms of sort bag of timelines, thus we can define the interpretation for $=_{\mathscr{S}}$:

$$\mathcal{M}_2, \nu \models \mathfrak{s} =_{\mathscr{S}} \mathfrak{s}' \text{ iff for any } s^{I',\nu} \in Sit\ (\mathcal{M}_2, \nu \models s \in_{\mathscr{S}} \mathfrak{s} \text{ iff } \mathcal{M}_2, \nu \models s \in_{\mathscr{S}} \mathfrak{s}') \tag{82}$$

[G2] Follows from (82) above.

[G3] This is a consequence of item (1) of (78), defining the interpretation of the constant $\mathscr{B}_0$.

[G4] By definition of timeline and of bag of timelines, if $\mathcal{M}_2, v \models (s = S_0 \vee \exists i \mathscr{T}(i, s))$ then $\mathscr{B}(s)$ is a bag of timelines. Consider, now, the definition of $\cup_{\mathscr{S}}$ given in Example 10 then

$$\mathcal{M}_2, v \models \mathfrak{s} =_{\mathscr{S}} \mathfrak{s}' \cup \mathscr{B}(s) \text{ iff } \mathcal{M}_2, v \models \forall s.s \in \mathfrak{s} \equiv s \in \mathfrak{s}' \cup \mathscr{B}(s)$$

Hence by simple induction on the structure of $\mathfrak{s}'$, $G4$ is satisfied in $\mathcal{M}_2$.

[G5] Let $\mathscr{S}$ be the terms of sort bag of timelines, and consider the definition of $\subseteq_{\mathscr{S}}$ given in Example 10: $\subseteq_{\mathscr{S}}$ is an ordering relation on $\mathscr{S}$, then every subset of $\mathscr{S}$ has a set of minimal elements and, in particular, $\mathscr{B}_0$ is a minimum. Now suppose that

$$\mathcal{M}_2, v \models \varphi(\mathscr{B}_0) \wedge (\forall s\, \mathfrak{s}.\varphi(\mathfrak{s}) \wedge \varphi(\mathscr{B}(s)) \rightarrow \varphi(\mathfrak{s} \cup_{\mathscr{S}} \mathscr{B}(s))) \tag{83}$$

and for some $\mathfrak{t}$

$$\mathcal{M}_2, v \not\models \varphi(\mathfrak{t})$$

Hence

$$\mathcal{M}_2, v \models \neg\varphi(\mathfrak{t})$$

Let $W = \{\mathfrak{t}' \mid \mathcal{M}_2, v \models \neg\varphi(\mathfrak{t}')\}$ then W has a set of minimal elements $Min = \{\mathfrak{s} \in W \mid \neg\exists \mathfrak{t} \in W, \mathfrak{t} \subset_{\mathscr{S}} \mathfrak{s}\}$. Let $x \in Min$ then, by hypothesis $\mathcal{M}_2, v \models \neg\varphi(x)$, with $x \neq \mathscr{B}_0$. Now, being x minimal in W, there must exist a $\mathfrak{s}'$, $\mathfrak{s}' \in \mathscr{S}$, with $\mathfrak{s}' \subset x$ and $\mathfrak{s}' \notin W$. Then $\mathcal{M}_2, v \models \varphi(\mathfrak{s}')$, since $\mathfrak{s}' \notin W$. We can find an s such that $x = \mathfrak{s}' \cup_{\mathscr{S}} \mathscr{B}(s)$, by the conditional existence. By equation (83)), since by hypothesis $\mathcal{M}_2, v \models \varphi(\mathscr{B}_0)$, and from $\mathcal{M}_2, v \models \varphi(\mathfrak{s}')$ and $\varphi(\mathscr{B}(s))$ it follows that $\mathcal{M}_2, v \models \varphi(\mathfrak{s}' \cup \mathscr{B}(s))$. Hence $\mathcal{M}_2, v \models \varphi(x)$, a contradiction.

We have shown that $\mathcal{M}_2$ is a model of $\mathcal{D}_{una} \cup \Sigma_{=_\nu} \cup (G1\text{-}G5)$.

□

B.6 Proof of Theorem 4

To prove the theorem we shall first extend the definition of uniform terms and regressable sentence (see [61]) to include terms and formulas mentioning terms of sort bag of situations. Let $\mathcal{L}_{TFSC}$ be the language of SC extended to include $\mathcal{A}^+$ and let $\mathcal{D}^+$ be a basic action theory extended with $\mathcal{A}^+$.

Let σ denote a term of sort situation mentioning only S_0 and terms of sort actions $\alpha_i = A(t_1, \ldots, t_m), m \geq 0$, with t_i, $1 \leq i \leq m$, not mentioning terms of sort situation, that is, appealing to the notational convention used in [61] $\sigma = do([\alpha_1, \ldots, \alpha_n], S_0)$, for some $n \geq 0$, and for terms $\alpha_1, \ldots, \alpha_n$ of sort *action*.

Definition 4 *The set of terms of the language $\mathcal{L}_{TFSC}$ uniform in $\sigma_1, \ldots, \sigma_k$, $k \geq 1$, is the smallest set defined as follows:*

1. *Any term not mentioning term of sort* situation *is uniform in $\sigma_1, \ldots, \sigma_k$, $k \geq 1$.*
2. *σ_i is uniform in $\sigma_1, \ldots, \sigma_k$, $i = 1, \ldots, k$.*
3. *If g is an n-ary function symbol other than do and $\mathscr{B}$, and $t_1, \ldots, t_n$ are terms uniform in $\sigma_1, \ldots, \sigma_k$ whose sorts are appropriate for g, then $g(t_1, \ldots, t_n)$ is a term uniform in $\sigma_1, \ldots, \sigma_k$.*
4. *$\mathscr{B}(\sigma_i)$ is a term uniform in $\sigma_1, \ldots, \sigma_k$, $i = 1, \ldots, k$.*
5. *$\mathscr{B}(\langle\sigma_{j1}, \ldots, \sigma_{jk}\rangle_j)$ is a term uniform in $\sigma_1, \ldots, \sigma_k$, for any permutation j of $\{1, \ldots, k\}$.*

Finally:

Items (4) and (5), of Definition 4 above, are correct because bags of timelines are flat sets, that is, they are formed only by situations, which are their individual elements. This follows from the first axiom ($G1$) defining membership only for elements of sort situation. To see this we prove the following lemmas.

Lemma 8 *For all bags of timelines* $\mathbb{t}$ *and* $\mathbb{s}$*:* $\mathbb{t} \notin_{\mathscr{S}} \mathbb{s}$.

B.6.1 Proof of Lemma 8

We prove the claim by induction on $\mathbb{s}$, using (G5). First note that by (G1) $\mathbb{t} \notin \mathscr{B}_0$ and by (G2) $\mathbb{t} \notin \mathscr{B}(s)$, because $\mathbb{t} \neq_{\mathscr{S}} s$. Let: $\phi(\mathbb{s}) = \forall \mathbb{t}.\mathbb{t} \notin \mathbb{s}$, then we have:

$$\begin{array}{lcll} \phi(\mathbb{s}) & = & \forall \mathbb{t}.\mathbb{t} \notin \mathbb{s} & \text{(Ind. Hyp)} \\ \phi(\mathscr{B}_0) & = & \forall \mathbb{t}.\mathbb{t} \notin \mathscr{B}_0 & \text{(by G1)} \\ \phi(\mathscr{B}(s)) & = & \forall \mathbb{t}.\mathbb{t} \notin \mathscr{B}(s) & \text{(by G2)} \\ & & \text{Then} & \\ & & \forall \mathbb{t}.\mathbb{t} \notin \mathbb{s} \wedge \mathbb{t} \notin \mathscr{B}(s) \rightarrow \mathbb{t} \notin \mathbb{s} \cup_{\mathscr{S}} \mathscr{B}(s) & \text{(by Def. of } \cup_{\mathscr{S}} \text{ Ex. 10)} \\ & & \forall \mathbb{t}.\mathbb{t} \notin \mathbb{s} \text{(by Ind. Hyp. and G5)} & \end{array} \tag{84}$$

Hence the claim.

□

The above lemma implies, in particular, that bags of timelines are not ordinal numbers.

Lemma 9 *Let* $\mathscr{P}(\mathbb{s})$ *be the power set of the bag* $\mathbb{s}$*. Then for any bag term* $\mathbb{t}$*,* $\mathbb{t} \cap \mathscr{P}(\mathbb{s}) = \emptyset$.

B.6.2 Proof of Lemma 9

Let $\mathscr{P}(\mathbb{s})$ be the power set of $\mathbb{s}$, that is $\forall x.x \subseteq_{\mathscr{S}} \mathbb{s} \rightarrow x \in_{\mathscr{S}} \mathscr{P}(\mathbb{s})$. We have to show that for all bags of timelines $\mathbb{t}$, $\mathbb{t} \cap_{\mathscr{S}} \mathscr{P}(\mathbb{s}) = \emptyset$. Suppose that there is some x, such that $x \in_{\mathscr{S}} \mathbb{t} \cap_{\mathscr{S}} \mathscr{P}(\mathbb{s})$, then $x \in_{\mathscr{S}} \mathbb{t}$ and $x \in_{\mathscr{S}} \mathscr{P}(\mathbb{s})$. Since $x \in_{\mathscr{S}} \mathscr{P}(\mathbb{s})$ then x is a bag term $\mathbb{s}'$ hence $\mathbb{s}' \in_{\mathscr{S}} \mathbb{t}$ contradicting the previous Lemma 8.

□

Lemma 10 *If* $\mathbb{S}$ *is a set of bag terms then* $\mathbb{S} \neq_{\mathscr{S}} \mathbb{s}$ *for all bag terms* $\mathbb{s}$.

B.6.3 Proof of Lemma 10

Follows from the previous Lemma 9.

□

We define now the set of regressable formulas extending the definition of [61] to include formulas mentioning bag of timelines.

Definition 5 *A formula* W *of* $\mathcal{L}_{TFSC}$ *is* regressable *iff*

1. W *is first order.*
2. W *does not mention variables of sort situation nor of sort bag of timelines.*
3. *Every term of sort* situation *mentioned by* W *is uniform in* $\sigma_1, \ldots, \sigma_n$*,* $n \geq 1$.
4. *For every atom of the form* $Poss(\alpha, \sigma)$ *mentioned by* W*,* α *has the form* $A(t_1, \ldots, t_n)$ *for some n-ary action function symbol* A *of* $\mathcal{L}_{TFSC}$.

5. *Every term* $\mathfrak{s}$ *of sort bag of timelines appearing in* W *is uniform in* $\sigma_1, \ldots, \sigma_k$*, for some* $k \geq 0$.

6. W *does not quantify over situations nor bag of situations.*

First note that, by definition of the regression operator,

$$\begin{array}{ll} \text{if} & \mathcal{D} \models W \equiv W' \\ \text{then} & \mathcal{D} \models \mathcal{R}(W) \equiv \mathcal{R}(W') \end{array} \tag{85}$$

Let W be a *regressable* formula mentioning terms $\mathfrak{s}$ of sort bag of timelines we show that

$$\mathcal{D}^+ \models (\forall) W \equiv \mathcal{R}(W)$$

with $\mathcal{R}(W)$ a formula uniform in S_0.

We show the claim by induction on the structure of the regressable formula W, mentioning terms of sort bag of timelines. We first show, however, that $\mathscr{T}(i, \sigma)$ is regressable if σ is a uniform term.

Lemma 11 *Consider a uniform term of sort situation, of the form* $\sigma^{m+1} = do(A_{m+1}(\vec{x}_{m+1}), do(\ldots, do(A_1(\vec{x}_1), S_0)\ldots))$*, for some* $m \in \mathbb{N}$ *and actions* $A_1, \ldots A_{m+1}$*, that is a timeline. Then:*

$$\mathcal{R}(\mathscr{T}(i, do(A_{m+1}, \sigma^m))) \equiv \bigwedge_{j=1}^{m+1} H(i, A_j(\vec{x}_j)) \tag{86}$$

Proof of the Lemma First note that given σ^{m+1}, as specified in the Lemma, the following holds, by m applications of Axioms (E1) and (E3) and of theorem (T1) of Lemma 6:

$$\begin{array}{lcl} A_{m+1} =_\nu \sigma^m \wedge \sigma^m = do(A_m(\vec{x}_m), do(\ldots, do(A_1(\vec{x}_1), S_0)\ldots)) & \equiv & A_{m+1}(\vec{x}_{m+1}) =_\nu A_m(\vec{x}_m) \wedge \ldots \wedge A_2(\vec{x}_2) =_\nu A_1(\vec{x}_1) \\ & \equiv & H(i, A_{m+1}(\vec{x}_{m+1})) \wedge \ldots \wedge H(i, A_1(\vec{x}_1)) \end{array} \tag{87}$$

Further, by induction on the number of actions mentioned in the uniform situation term σ^{m+1}, we can see that:

$$\mathscr{T}(i, do(A_{m+1}(\vec{x}_{m+1}), \sigma^m)) \equiv \bigwedge_{j=1}^{m+1} \left[A_j(\vec{x}_j) =_\nu \sigma^{j-1} \vee \sigma^{j-1} = S_0 \wedge H(i, A(\vec{x}_j))\right] \tag{88}$$

Indeed, the basic case, for $\sigma^m = S_0$ follows from the definition of $\mathscr{T}(i, do(a, s))$. For the induction, we have:

$$\begin{array}{lcll} \mathscr{T}(i, do(A_{m+1}(\vec{x}_{m+1}), \sigma^m)) & \equiv & \mathscr{T}(i, \sigma^m) \wedge A_{m+1}(\vec{x}_{m=1}) =_\nu \sigma^m \vee & \\ & & \sigma^m = S_0 \wedge H(i, A_{m+1}(\vec{x}_{m=1})) & \\ & \equiv & \bigwedge_{j=1}^{m+1} \left[A_j(\vec{x}_j) =_\nu \sigma^{j-1} \vee \sigma^{j-1} = S_0 \wedge H(i, A(\vec{x}_j))\right] & \\ & & \wedge A_{m+1}(\vec{x}_{m=1}) =_\nu \sigma^m \vee \sigma^m = S_0 \wedge H(i, A_{m+1}(\vec{x}_{m=1})) & \text{(By Ind. Hyp.)} \\ & \equiv & \bigwedge_{j=1}^{m+1} H(i, A_j(\vec{x}_j)) & \text{(By (87), above and Taut.)} \end{array} \tag{89}$$

Thus $\bigwedge_{j=1}^{m+1} H(i, A_j(\vec{x}_j)) \equiv \mathcal{R}(\mathscr{T}(i, do(A_{m+1}, \sigma^m)))$, a formula uniform in S_0.

□

Theorem continue.. For the basic step we consider the following atoms (see Definition 1):

1. if W has the form $\sigma \in \mathscr{B}(\langle \sigma_{j1}, \ldots, \sigma_{jk} \rangle)$, with each $\sigma_{ji} = do([\alpha_1, \ldots, \alpha_n], S_0)$ for some $n \geq 0$, then by axiom (G1),

$$\mathcal{D}^+ \models W \equiv \bigvee_{1 \leq p \leq k} \sigma = \sigma_{jp} \wedge \left(\sigma_{jp} = S_0 \vee \bigvee_i \mathscr{T}(i, \sigma_{jp})\right)$$

So let W' be the RHS of the equivalence in the above formula, by Lemmas 8,9 and (G1) W' does not mention terms of sort bag of timelines and moreover is a regressable formula (Lemma 11) hence the regression theorem, as stated in [61] applies. Therefore $\mathcal{R}(W')$ is a formula uniform in S_0 and, by (85) above:

$$\mathcal{R}(W) \equiv \mathcal{R}(W') \tag{90}$$

and the claim is verified by monotonicity since $\mathcal{D} \subseteq \mathcal{D}^+$.

2. If W has the form $\mathfrak{s} =_{\mathscr{S}} \mathfrak{t}$ then, because W is a regressable sentence, it has the following form:

$$\mathscr{B}(\langle \sigma_{j1}, \ldots, \sigma_{jk} \rangle_j) =_{\mathscr{S}} \mathfrak{t}\mathscr{B}(\langle \sigma'_{p1}, \ldots, \sigma'_{pm} \rangle_p)$$

then by (G2), W is equivalent to the formula W':

$$\forall s. \left(s \in_{\mathscr{S}} \mathscr{B}(\langle \sigma_{j1}, \ldots, \sigma_{jk} \rangle_j) \right) \equiv \left(s \in_{\mathscr{S}} \mathscr{B}(\langle \sigma_{p1}, \ldots, \sigma_{pm} \rangle_p) \right) \tag{91}$$

And, by (G1), W' is equivalent to the following formula W'':

$$\forall s. \left(\bigvee_{1 \leq h \leq k} s = \sigma_{jh} \wedge (\sigma_{jh} = S_0 \vee \bigvee_i \mathscr{T}(i, \sigma_{jh})) \equiv \bigvee_{1 \leq q \leq m} s = \sigma_{pq} \wedge (\sigma_{pq} = S_0 \vee \bigvee_i \mathscr{T}(i, \sigma_{pq})) \right) \tag{92}$$

Finally W'', by first order tautologies and equality, is equivalent to the following sentence W''':

$$\left(\bigwedge_{1 \leq h \leq k} \bigvee_{1 \leq q \leq m} (\sigma_{jh} = \sigma_{pq} \wedge (\sigma_{jh} = S_0 \vee \bigvee_i \mathscr{T}(i, \sigma_{jh}))) \right) \tag{93}$$

Now, by Lemma 11, W''' is a regressable sentence not mentioning terms of sort bag of timelines and hence the regression theorem can be applied and the claim holds.

3. If W has the form:

$$\mathfrak{s} =_{\mathscr{S}} \mathscr{B}_0$$

Then W reduces to either $\top$ or $\bot$, which are regressable sentences in $\mathcal{L}$, and $\mathcal{R}(\top)$ (as $\mathcal{R}(\bot)$) are uniform in S_0, hence the claim holds.

4. If W has the form:

$$\mathscr{B}(\langle \sigma_{r1}, \ldots, \sigma_{rm} \rangle_r) =_{\mathscr{S}} \mathscr{B}(\langle \sigma'_{j1}, \ldots, \sigma'_{jk} \rangle_j) \cup_{\mathscr{S}} \mathscr{B}(\langle \sigma''_{p1}, \ldots, \sigma''_{pn} \rangle_p)$$

then we can use the definition of $\cup_{\mathscr{S}}$ given in Example 10:

$$(\mathfrak{s} \cup_{\mathscr{S}} \mathfrak{s}' =_{\mathscr{S}} \mathfrak{t} \equiv \forall s. s \in_{\mathscr{S}} \mathfrak{t} \equiv (s \in_{\mathscr{S}} \mathfrak{s} \vee s \in_{\mathscr{S}} \mathfrak{s}'))$$

Then W is equivalent to the following formula W':

$$\forall s. s \in_{\mathscr{S}} \mathscr{B}(\langle \sigma_{r1}, \ldots, \sigma_{rm} \rangle_r) \equiv s \in_{\mathscr{S}} \mathscr{B}(\langle \sigma'_{j1}, \ldots, \sigma'_{jk} \rangle_j) \vee s \in_{\mathscr{S}} \mathscr{B}(\langle \sigma''_{p1}, \ldots, \sigma''_{pn} \rangle_p) \tag{94}$$

Again using (G1), we get W'':

$$\begin{aligned} &\forall s. \left(\bigvee_{1 \leq u \leq m} s = \sigma_u \wedge (\sigma_u = S_0 \vee \bigvee_i \mathscr{T}(i, \sigma_u)) \equiv \right. \\ &\left. \bigvee_{1 \leq h \leq k} s = \sigma'_h \wedge (\sigma'_h = S_0 \vee \bigvee_i \mathscr{T}(i, \sigma'_h)) \vee \bigvee_{1 \leq q \leq n} s = \sigma''_q \wedge (\sigma''_q = S_0 \vee \bigvee_i \mathscr{T}(i, \sigma''_q)) \right) \end{aligned} \tag{95}$$

By first order tautologies and equality, W'' reduces to the equivalent formula W''':

$$\bigwedge_{1\leq u\leq m}\Big(\bigvee_{1\leq h\leq k}(\sigma_u=\sigma'_h)\vee\bigvee_{1\leq q\leq n}(\sigma_u=\sigma''_q)\wedge(\sigma_u=S_0\vee\mathscr{T}(i,\sigma_u))\Big) \tag{96}$$

W''' is a regressable sentence of $\mathcal{L}_3$ and does not mention terms of sort bags of timelines, therefore the regression theorem can be applied and also in this case the claim holds.

5. Any other regressable atom (see Definition 1) mentioning the classical set-operators, that can be defined using (G1-G5), can be easily reduced to the previous cases.

Now, by induction hypothesis, regression can be extended to any regressable sentence mentioning terms of sort bag of timelines simply using the inductive definition of $\mathcal{R}$:

$$\begin{array}{lcl}\mathcal{R}[\neg W] & = & \neg\mathcal{R}[W],\\ \mathcal{R}[W_1\wedge W_2] & = & \mathcal{R}[W_1]\wedge\mathcal{R}[W_2],\\ \mathcal{R}[(\exists v)W] & = & (\exists v)\mathcal{R}[W]\end{array} \tag{97}$$

This concludes the proof that regression can be extended to regressable formulas mentioning terms of sort bag of timelines

□

Appendix C

C Proofs for Section 4

C.1 Proof of Theorem 5

Let $\mathcal{D}_T$ be the theory formed by $\mathcal{D}_{ss}\cup\mathcal{D}_{ap}\cup\mathcal{D}_{\pi}\cup\mathcal{D}_{S_0}\cup\mathcal{A}^+\cup\mathcal{D}_{una}$, we have to prove that $\mathcal{D}_T$ is satisfiable iff $\mathcal{D}_{S_0}\cup\mathcal{A}^+\cup\mathcal{D}_{una}$ is. If $\mathcal{D}_T$ is satisfiable, despite the second order axiom, using compactness it follows that also $\mathcal{D}_{S_0}\cup\mathcal{A}^+\cup\mathcal{D}_{una}$ is. For the other direction assume that $\mathcal{D}_{S_0}\cup\mathcal{A}^+\cup\mathcal{D}_{una}$ is satisfiable. Let $\mathcal{M}_1$ be any of such structures. We know from the proof of Theorem 3 that $\mathcal{M}_1$, a structure of $\mathcal{L}_4$, is also a model of $\mathcal{D}_{S_0}\cup\mathcal{D}_{ap}\cup\mathcal{D}_{ssa}\cup\mathcal{D}_{una}\cup\mathcal{A}^+$ where $\mathcal{D}_{ssa}\cup\mathcal{D}_{una}$ mentions successor state axioms for timelines, but no process is yet defined. We also assume that axiom (4), (see also (45)) is verified in $\mathcal{M}_1$.

Here we have to show that $\mathcal{M}_1$ can be transformed into a model of the condition (20) and of the successor state axioms and action precondition axioms for processes. To this end we define a new structure $\mathcal{M}_2=(D,I')$ having the same domain as $\mathcal{M}_1$, but the interpretation I' extends I, fixing an interpretation also for processes and the fluent $Idle$, as specified in the sequel.

Let Π be the set of processes in the language $\mathcal{L}_4$, we enumerate Π

$$\pi_1(i_1,\vec{x},t_0,S_0),\pi_2(i_1,\vec{x},t_0,S_0),\ldots,\pi_m(i_k,\vec{x},t_0,S_0),\pi_{m+1}(i_k,\vec{x},t_0,S_0),\ldots$$

and define subsets of this ordering as follows:

$$\Pi_{i_m}=\{\pi_j(i_m,\vec{x},t_0,S_0)\mid i_m\text{ is the corresponding name type of the process }\pi_j,\text{ for which }H(i_m,a)\text{ is defined}\}$$

In other words, following Corollary B.3 all the actions $start_\pi$ and end_{pi} might have been already suitable assigned to the $H(i,a)$.

We can now order the Π_{i_k} according to the name type and choose one process for each set and state:

$\mathcal{M}_2\models\exists\vec{x}\pi_j(i_m,\vec{x},t_0,S_0)$ iff for all $\pi_k(i_m,\vec{x},t_0,S_0)\in\Pi_{i_m},(k\neq j),\mathcal{M}_2\models\forall\vec{x}\neg\pi_k(i_m,\vec{x},t_0,S_0)$

Further we set for all name types $\neg Idle(i,S_0)$.

This construction implies that $\mathcal{M}_1$ is a model of the condition (20).
Thus we have:

$$\begin{array}{ll}
(1) & \mathcal{M}_2, v \models \neg Idle(i, S_0) \quad \text{for all name types } i \\
(2) & \mathcal{M}_2, v \models \neg Poss(start_\pi(i, \vec{x}, t), S_0) \text{ for all name types } i, \text{ because of (1) above and definition (19)} \\
(3) & \mathcal{M}_2, v \models Poss(end_\pi(i, \vec{x}, t), S_0) \text{ iff } M_2, v \models \Psi_{end}(i, \vec{x}, S_0) \text{ and} \\
 & \quad \mathcal{M}_2, v \models \pi(i, \vec{x}, t_0, S_0) \wedge time(end_\pi) > t_0 \\
(4) & \mathcal{M}_2, v \models time(S_0) = t_0 \quad \text{by the construction of } \mathcal{M}_1
\end{array} \tag{98}$$

Having fixed the definition of processes, *Idle* and *Poss* in $\mathcal{D}_{S_0}$ we can proceed inductively on all situation s similarly as in the relative satisfiability theorem in [61]. In fact, the inductive step of the proof relies on the fact that the right hand side of the successor state axioms $\Psi(\vec{x}, y, a, t, s)$ is uniform in s and hence has already been assigned a truth value in s by $\mathcal{M}_2$, and being fixed for $\mathcal{D}_{S_0}$ the induction is straightforward. □

C.2 Proof of Proposition 1

We proceed by induction on σ, using the definition of *executable* given in (17) and the definitions of the action preconditions. Consider $do(\alpha, S_0)$, in this case, if $H(i, \alpha)$ then $\mathcal{D}_T \models \mathcal{T}(i, do(\alpha, S_0))$. For the inductive step, consider $do(\alpha, \sigma)$. By the hypothesis α is executable in σ, hence $\mathcal{D}_T \models Poss(\alpha, \sigma)$, therefore, by (19) and (21) , $\alpha =_\nu \sigma$. By the induction hypothesis $\mathcal{D}_T \models \mathcal{T}(i, \sigma)$ hence $\mathcal{D}_T \models \mathcal{T}(i, do(\alpha, \sigma))$. □

C.3 Proof of Proposition 2

We have to show that, for σ an executable situation, then

$$\mathcal{D}_T \models Idle(i, \sigma) \vee \left(\exists \vec{x}\, t. \pi(i, \vec{x}, t, \sigma) \rightarrow \neg Idle(i, \sigma) \wedge \forall \vec{y} \bigwedge_{\pi' \in \Pi}^{\pi \neq \pi'} \neg \pi'(i, \vec{y}, t, \sigma) \right) \tag{99}$$

First note that for $\sigma = S_0$ the statement holds because of (20). For any other σ, by the previous Proposition 1 (C.2) it follows that σ must also be a timeline. Suppose that for the given timeline $\mathcal{T}(i, \sigma)$ the statement holds for any $\sigma'' \sqsubseteq \sigma$, and let $do(a, \sigma')$ be the first situation in the equivalence class of the timeline for which the statement does not hold, for some a. We show that this leads to a contradiction. Assume, therefore, there is some model $\mathcal{M}$ of $\mathcal{D}_T$ and some processes $\pi'(i, \vec{y}, t', do(a, \sigma'))$ and $\pi''(i, \vec{x}, t, do(a, \sigma'))$, with $\pi' \neq \pi''$, which do not satisfy (99). Then:

$$\begin{array}{ll}
\mathcal{M} \models & \neg Idle(i, do(a, \sigma)) \wedge (\exists \vec{x},\ t. \pi'(i, \vec{x}, t, do(a, \sigma')) \wedge \exists \vec{z}, t'. \pi''(i, \vec{z}, t', do(a, \sigma')) \vee Idle(i, do(a, \sigma))) \\
\mathcal{M} \models & \neg Idle(i, do(a, \sigma)) \wedge (\exists \vec{x}\, t. \pi'(i, \vec{x}, t, do(a, \sigma')) \wedge \exists z, t'. \pi''(i, \vec{z}, t', do(a, \sigma')))
\end{array} \tag{100}$$

Then:

1. Since $\mathcal{M} \models \exists \vec{x}, t. \pi'(i, \vec{x}, t, do(a, \sigma'))$, then by the successor state axiom for processes (16):

$$\begin{array}{ll}
\mathcal{M} \models & \exists \vec{x}, t. a = start_{\pi'}(i, \vec{x}, t) \vee \pi'(i, \vec{x}, t, \sigma') \wedge \forall t. a \neq end_{\pi'}(i, \vec{x}, t) \\
\mathcal{M} \models & \exists \vec{x}, t. a = start_{\pi'}(i, \vec{x}, t) \vee \exists \vec{z}, t'. \pi'(i, \vec{z}, t', \sigma') \wedge \forall t''. a \neq end_{\pi'}(i, \vec{z}, t'')
\end{array} \tag{101}$$

2. Since $\mathcal{M} \models \exists \vec{x},\ t. \pi''(i, \vec{x}, t, do(a, \sigma'))$, then:

$$\mathcal{M} \models \exists \vec{x}, t. a = start_{\pi''}(i, \vec{x}, t) \vee \exists \vec{z}, t'. \pi''(i, \vec{z}, t', \sigma') \wedge \forall t'. a \neq end_{\pi''}(i, \vec{z}, t') \tag{102}$$

Now, since $do(a, \sigma')$ is the first situation in which the statement fails, it follows that it must be true in σ hence it cannot be that for some $\vec{d}$ and some $\vec{d'}$ in the domain of $\mathcal{M}$, both $\pi'(i, \vec{d}, t', \sigma')$ and $\pi''(i, \vec{d'}, t', \sigma')$ hold. W.l.o.g we may assume any of the two and establish that $\mathcal{M} \models \pi'(i, \vec{d}, t', \sigma')$, for some $\vec{d} \in D$, the domain of $\mathcal{M}$, and $\mathcal{M} \models \forall z, t' \neg\pi''(i, \vec{z}, t', \sigma')$. But now, since $\mathcal{M}$ satisfies π'' with argument $do(a, \sigma)$, it follows, from the successor state axiom for processes, that it must be that $\mathcal{M} \models \exists \vec{y}, t'.a = start_{\pi''}(i, \vec{y}, t')$.

By the statement assumptions $do(a, \sigma)$ must be executable, hence $\mathcal{M} \models \exists \vec{y}, t'.a = start_{\pi''}(i, \vec{y}, t') \wedge Poss(start_{\pi''}(i, \vec{y}, t'), \sigma')$.

This fact, in turn, implies, by the definition of $Poss$ for the action $start_{\pi''}$ (see 19) that $\mathcal{M} \models Idle(i, \sigma')$, hence it cannot be true that $\mathcal{M} \models \pi'(i, \vec{d}, t', \sigma')$, for some $\vec{d} \in D$, given that (99) holds with σ', therefore also for π' it must be that $\mathcal{M} \models \forall z, t' \neg\pi'(i, \vec{d}, t', \sigma')$. We are thus left with

$$\mathcal{M} \models \quad \exists \vec{x}, t.a{=}start_{\pi'}(i, \vec{x}, t) \wedge Idle(i, \sigma') \wedge \exists \vec{y}, t'.a{=}start_{\pi''}(i, \vec{y}, t) \wedge Idle(i, \sigma') \tag{103}$$

But this is not possible for both, by the equality, hence it follows that

$$\mathcal{M} \models \forall \vec{x},\ t.\neg\pi'(i, \vec{x}, t, do(a, \sigma')),$$

and we have a contradiction. □

Appendix D

D Proofs for Section 6

D.1 Proof of Theorem 6

We first introduce three lemmas, then we prove the theorem.

Lemma 12 *Let K and G be finite sets of indexes, the following are tautologies:*

$$\begin{array}{lll} i. & \{\bigvee_{i\in K} D_i\} \rightarrow \{\bigvee_{g\in G} W_g\} \equiv & \bigwedge_{i\in K}\{D_i \rightarrow \{\bigvee_{g\in G} W_g\}\}, \\ ii. & \forall\vec{w}\exists\vec{z}\bigwedge_{i\in K}\{D_i(\vec{w}) \rightarrow \{\bigvee_{g\in G} W_g(\vec{w},\vec{z})\}\} \equiv & \bigwedge_{i\in K}\forall\vec{w}\{D_i(\vec{w}) \rightarrow \{\bigvee_{g\in G} \exists\vec{z}\, W_g(\vec{w},\vec{z})\}\}. \end{array} \tag{104}$$

Proof. By FOL □

Lemma 13 *Given the predicates $Elapsed_X(i, \vec{x}, t^-, t^+, \sigma)$ and $Active_X(i, \vec{x}, t^-, \sigma)$, as defined in (24,25), with σ a ground situation of type i, for any $\mathcal{M}$ of TFSC and assignment v, the following holds:*

$$\begin{array}{lll} \mathcal{M}, v \models Elapsed_X(i, \vec{x}, t^-, t^+, \sigma) & iff & \mathcal{M}, v \models \bigvee_{i\in K}[M_{k,X}(\vec{x}, t^-, t^+, S_0) \wedge (t^- = \tau^-_{k,X} \wedge t^+ = \tau^+_{k,X})]\,; \\ \mathcal{M}, v \models Active_X(i, \vec{x}, t^-, \sigma) & iff & \mathcal{M}, v \models \bigvee_{i\in K}[N_{k,X}(\vec{x}, t^-, S_0) \wedge (t^- = \tau^-_{k,X})]. \end{array} \tag{105}$$

Here $\tau^{\pm}_{k,X}$ is a time variable (or instance) mentioned in σ, $M_{k,X}(\vec{x}, t^-, t^+, S_0)$ and $N_{k,X}(\vec{x}, t^-, S_0)$ are TFSC formulas in S_0 with k in K finite set of indexes.

Proof. We proceed by induction on σ.
Basic case: in this case we state $\sigma = S_0$, hence, by (24) we have that

$$Active_X(i, \vec{x}, t^-, S_0) = X(i, \vec{x}, S_0) \wedge time(S_0) = t^-$$
$$Elapsed_X(i, \vec{x}, t^-, t^+, S_0) = \bot,$$

and we obtain (105) once we state, for instance, $K = \{1\}$, $M_{1,X}(i, \vec{x}, t^-, t^+, S_0) = \bot$, $N_{1,X}(i, \vec{x}, t^-, S_0) = X(i, \vec{x}, S_0)$ and $\tau^-_{1,X} = t_0$, where t_0 is for $time(S_0)$.
Inductive step: now we assume that (105) holds for σ and we prove that it holds for $do(A, \sigma)$.

By (24) we have that:

$$\begin{array}{l} Active_X(i,\vec{x},t^-,do(A,\sigma)) = \mathscr{T}(i,do(A,\sigma)) \wedge Started_X(i,\vec{x},t^-,A,\sigma) \vee \\ \quad Active_X(i,\vec{x},t^-,\sigma) \wedge \neg\exists t^+\ Ended_X(i,\vec{x},t^+,A,\sigma) \\ Elapsed_X(i,\vec{x},t^-,t^+,do(A,\sigma)) = \mathscr{T}(i,do(A,s)) \wedge Elapsed_X(i,\vec{x},t^-,t^+,\sigma) \vee \\ \quad Ended_X(i,\vec{x},t^+,A,\sigma) \wedge Active_X(i,\vec{x},t^-,\sigma). \end{array}$$

Applying the definition of $Started_X$ and $Ended_X$ the previous one can be rewritten as follows:

$$\begin{array}{l} Active_X(i,\vec{x},t^-,do(A,\sigma)) = \mathscr{T}(i,do(A,\sigma)) \wedge X(\vec{x},do(A,\sigma)) \wedge \neg X(\vec{x},\sigma) \wedge time(A) = t^- \vee \\ \quad Active_X(i,\vec{x},t^-,\sigma) \wedge \neg\exists t^+\ (X(\vec{x},\sigma) \wedge \neg X(\vec{x},do(A,\sigma)) \wedge time(A) = t^+); \\ Elapsed_X(i,\vec{x},t^-,t^+,do(A,\sigma)) = \mathscr{T}(i,do(A,s)) \wedge Elapsed_X(i,\vec{x},t^-,t^+,\sigma) \vee \\ \quad X(\vec{x},\sigma) \wedge \neg X(\vec{x},do(A,\sigma)) \wedge time(A) = t^+ \wedge Active_X(i,\vec{x},t^-,\sigma). \end{array} \tag{106}$$

Consider the regression of the following formulas:

$$\begin{array}{l} \mathcal{R}(\mathscr{T}(i,do(A,\sigma)) = R^0_i(S_0); \\ \mathcal{R}(X(\vec{x},\sigma) \wedge \neg X(\vec{x},do(A,\sigma))) = R^1_X(\vec{x},S_0); \\ \mathcal{R}(\mathscr{T}(i,do(A,\sigma)) \wedge X(\vec{x},do(A,\sigma)) \wedge \neg X(\vec{x},\sigma)) = R^2_X(\vec{x},S_0); \\ \mathcal{R}(\neg\exists t^+\ (X(\vec{x},\sigma) \wedge \neg X(\vec{x},do(A,\sigma)) \wedge time(A) = t^+)) = R^3_X(\vec{x},S_0). \end{array}$$

Than we can rewrite the previous equivalence (106) by substituting the regressed formulas

$$\begin{array}{ll} Active_X(i,\vec{x},t^-,do(A,\sigma)) = & R^2_X(\vec{x},S_0) \wedge time(A) = t^- \vee \\ & Active_X(i,\vec{x},t^-,\sigma) \wedge R^3_X(\vec{x},S_0); \\ Elapsed_X(i,\vec{x},t^-,t^+,do(A,\sigma)) = & R^0_i(S_0) \wedge Elapsed_X(i,\vec{x},t^-,t^+,\sigma) \vee \\ & R^1_X(\vec{x},S_0) \wedge Active_X(i,\vec{x},t^-,\sigma) \wedge time(A) = t^+. \end{array}$$

If we now apply the inductive hypothesis we get:

$$\begin{array}{lll} \mathcal{M},v \models & Elapsed_X(i,\vec{x},t^-,t^+,do(A,\sigma)) & \text{iff} \\ \mathcal{M},v \models & \bigvee_{k\in K}[M_{k,X}(\vec{x},t^-,t^+,S_0) \wedge R^0_i(S_0) \wedge (t^- = time(a^-_{k,X}) \wedge t^+ = time(a^+_{k,X}))] \vee & \\ & \bigvee_{k\in K}[N_{k,X}(\vec{x},t^-,S_0) \wedge R^1_X(\vec{x},S_0) \wedge time(A) = t^+; & \\ \mathcal{M},v \models & Active_X(i,\vec{x},t^-,do(A,\sigma)) & \text{iff} \\ \mathcal{M},v \models & \bigvee_{k\in K}[N_{k,X}(\vec{x},t^-,S_0) \wedge R^3_X(\vec{x},S_0) \wedge (t^- = time(a^-_{k,X}))] \vee & \\ & R^2_X(\vec{x},S_0) \wedge time(A) = t^- & \end{array} \tag{107}$$

where $a^{\pm}_{k,X}$ is an action mentioned in the ground situation σ. Since $time(a_{k,X})$ equals to a time variable (or instance) $\tau^{\pm}_{k,X}$ mentioned in σ, the property (23) holds for $do(A,\sigma)$. This concludes the prove. □

Lemma 14 *Given a bag of timelines $\mathfrak{s}[\omega]$ mentioning the set of timelines $\{\sigma_1,\ldots,\sigma_n\}$, where ω is a tuple of the time variables $\mathbf{t}_{i,j}$, the predicate $\mathbb{I}(T_c,\mathfrak{s}[\omega])$ can be transformed into the following form:*

$$\bigwedge_{d\in D}\bigvee_{w\in W_d}\bigwedge_{r\in R_{d,w}}\bigwedge_{k\in K_{d,w,r}}\bigvee_{g\in G_{d,w,r}} \forall\vec{x}\exists\vec{y}(P_d(i_d,\vec{x},\tau^-_k,\tau^+_k)\ \mathbf{op}_{d,r}\ Q_r(j_r,\vec{y},\tau^-_g,\tau^+_g)[\sigma_{i_d},\sigma_{j_r}]). \tag{108}$$

Here D, W_d, $R_{d,w}$ $K_{d,w,r}$, and $G_{d,w,r}$ are finite set of indexes, where τ^-_k, τ^+_k (τ^-_g, τ^+_g) are either temporal variables or ground temporal instances mentioned in the ground situation σ_{i_d} (respectively, σ_{j_r}) with name types i_d (j_r) .

□

Proof. From (27) we have that $\mathbb{I}(T_c,\mathfrak{s})$ is a TFSC formula of the form

$$\begin{array}{ll} \bigwedge_{(comp(P_i(\vec{x}),LL)\in T_c)}\bigvee_{(L\in LL)} & \exists s(s\in\mathfrak{s} \wedge \mathscr{T}(i,s) \wedge (\bigwedge_{\{(\mathbf{op},Q(j,\vec{y}))\in L\}} \exists s'(s'\in\mathfrak{s} \wedge \mathscr{T}(j,s') \wedge \\ & \forall\vec{x},t^-_i,t^+_i\exists\vec{y},t^-_j,t^+_j(P(i,\vec{x},t^-_i,t^+_i)\ \mathbf{op}\ Q(j,\vec{y},t^-_j,t^+_j)[s,s'])))). \end{array}$$

We want to prove that, for each model $\mathcal{M}$ of TFSC and assignment v

$$\begin{array}{ll}\mathcal{M},v\models & \exists s(s\in\mathbb{s}\wedge\mathscr{T}(i,s)\wedge\exists s'(s'\in\mathbb{s}\wedge\mathscr{T}(j,s')\wedge\\ & \forall\vec{x},t_i^-,t_i^+\exists\vec{y},t_j^-,t_j^+(P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{op}\ Q(j,\vec{y},t_j^-,t_j^+)[s,s'])))\\ & \text{iff there exist two ground situation } \sigma_i \text{ and } \sigma_j \text{ in } \mathbb{s} \text{ of type } i \text{ and } j \text{ respectively, such that:}\\ \mathcal{M},v\models & \bigwedge_{k\in K}\bigvee_{g\in G}\forall\vec{x}\exists\vec{y}(P(i,\vec{x},\tau_k^-,\tau_k^+)\ \mathbf{op}\ Q(j,\vec{y},\tau_g^-,\tau_g^+)[\sigma_i,\sigma_j]),\\ & \text{with } K \text{ and } G \text{ finite sets of indexes and } \tau_k^-,\tau_k^+(\tau_g^-,\tau_g^+) \text{ time variables or instances in } \sigma_i\ (\sigma_j).\end{array}\tag{109}$$

First of all, observe that, since $\mathcal{M}$ is a TFSC model, the following holds:

$$\begin{array}{ll}\mathcal{M},v\models & \exists s(s\in\mathbb{s}\wedge\mathscr{T}(i,s)\wedge\exists s'(s'\in\mathbb{s}\wedge\mathscr{T}(j,s')\wedge\\ & \forall\vec{x},t_i^-,t_i^+\exists\vec{y},t_j^-,t_j^+(P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{op}\ Q(j,\vec{y},t_j^-,t_j^+)[s,s'])))\\ & \text{iff there exists two ground situations, } \sigma_i \text{ and } \sigma_j \text{ in } \mathbb{s}, \text{ of type } i \text{ and } j \text{ respectively, such that:}\\ \mathcal{M},v\models & \forall\vec{x},t_i^-,t_i^+\exists\vec{y},t_j^-,t_j^+(P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{op}\ Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j]).\end{array}\tag{110}$$

Therefore, the theorem is proved once we show that

$$\begin{array}{ll}\mathcal{M},v\models\forall\vec{x},t_i^-,t_i^+\exists\vec{y},t_j^-,t_j^+(P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{op}\ Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j]), & \text{with } \sigma_i,\sigma_j\in\mathbb{s}, \text{ iff}\\ \mathcal{M},v\models\bigwedge_{k\in K}\bigvee_{g\in G}\forall\vec{x}\exists\vec{y}(P(i,\vec{x},\tau_k^-,\tau_k^+)\ \mathbf{op}\ Q(j,\vec{y},\tau_g^-,\tau_g^+)[\sigma_i,\sigma_j]), & \end{array}\tag{111}$$

with τ_k^-, τ_k^+, (τ_g^-, τ_g^+) time variables or instances thereof mentioned in σ_i (σ_j). Indeed, from (111) and (27) we obtain that $\mathcal{M},v\models\mathbb{I}(T_c,\mathbb{s})$ iff

$$\mathcal{M},v\models\bigwedge_{d\in D}\bigvee_{w\in W_d}\bigwedge_{r\in R_{d,w}}\bigwedge_{k\in K_{d,w,r}}\bigvee_{g\in G_{d,w,r}}\forall\vec{x}\exists\vec{y}(P_d(i_d,\vec{x},\tau_k^-,\tau_k^+)\,\mathbf{op}_r\,Q_r(j_r,\vec{y},\tau_g^-,\tau_g^+)[\sigma_{i_d},\sigma_{j_r}]),$$

where $d\in D$ is for $(comp(P(\vec{x}),LL)\ in\ T_c)$, $w\in W_h$ is for L *in* LL, and $r\in R_{d,w}$ is for $(\mathbf{op},Q(j,\vec{y}))\in L$. Once we represent $\bigwedge_{r\in R_{d,w}}\bigwedge_{k\in K_{d,w,r}}$ directly as $\bigwedge_{\langle r,k\rangle\in RK_{d,w}}$, we obtain the equation (108) and the Lemma is proved.

Now, it remains to show that (27) holds. In order to prove this, we proceed with a proof by cases, restricting our attention to $\{\mathbf{m},\mathbf{b},\mathbf{s},\mathbf{f},\mathbf{d}\}$.

Case meets: We consider the following form:

$$\begin{array}{lll}P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{m}\ Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j] & \stackrel{def}{=} & Elapsed_P(i,\vec{x},t_i^-,t_i^+,\sigma_i)\rightarrow\\ & & \Big((Active_Q(j,\vec{y},t_j^-,\sigma_j)\vee Elapsed_Q(j,\vec{y},t_i^-,t_j^+,\sigma_j))\wedge(t_i^+=t_j^-)\Big);\end{array}$$

Given the (107), by FOL $\mathcal{M},v\models(Active_Q(j,\vec{y},t_j^-,\sigma_j)\vee Elapsed_Q(j,\vec{y},t_i^-,t_j^+,\sigma_j))$ iff $\mathcal{M},v\models\bigvee_{g\in G}[W_{g,Q}(\vec{y},t_j^-,t_j^+S_0)\wedge(t_j^-=\tau_{g,Q}^-\wedge t_j^+=\tau_{g,Q}^+)]$, with $\tau_{g,Q}$ time variables (instances) mentioned in σ'. Therefore, we have that:

$$\begin{array}{ll}\mathcal{M},v\models P(i,\vec{x},t_i^-,t_i^+)\,\mathbf{m}\,Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j] & \text{iff}\\ \mathcal{M},v\models\{\bigvee_{i\in K}[M_{i,P}(\vec{x},t_i^-,t_i^+,S_0)\wedge(t_i^-=\tau_{k,P}^-\wedge t_i^+=\tau_{k,P}^+)]\}\rightarrow & \\ \quad\{\bigvee_{g\in G}[W_{g,Q}(\vec{y},t_j^-,t_j^+,S_0)\wedge(t_j^-=\tau_{g,Q}^-\wedge t_j^+=\tau_{g,Q}^+)\wedge(t_i^+=t_j^-)]\}, & \end{array}\tag{112}$$

We can consider the formula $\forall\vec{x},t_i^-,t_i^+\ \exists\vec{y},t_j^-,t_j^+\ P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{m}\ Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j]$. We can see that:

$$\begin{array}{ll}\mathcal{M},v\models\forall\vec{x},t_i^-,t_i^+\exists\vec{y},t_j^-,t_j^+P(i,\vec{x},t_i^-,t_i^+)\,\mathbf{m}\,Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j], & \text{(by (112)) iff}\\ \mathcal{M},v\models\forall\vec{x},t_i^-,t_i^+\exists\vec{y},t_j^-,t_j^+\{\bigvee_{i\in K}[M_{i,P}(\vec{x},t_i^-,t_i^+,S_0)\wedge(t_i^-=\tau_{k,P}^-\wedge t_i^+=\tau_{k,P}^+)]\}\rightarrow & \\ \quad\{\bigvee_{g\in G}[W_{g,Q}(\vec{y},t_j^-,t_j^+,S_0)\wedge(t_j^-=\tau_{g,Q}^-\wedge t_j^+=\tau_{g,Q}^+)\wedge(t_i^+=t_j^-)]\}, & \text{(by (104) i.) iff}\\ \mathcal{M},v\models\bigwedge_{i\in K}\{\forall\vec{x},t_i^-,t_i^+[M_{i,P}(\vec{x},t_i^-,t_i^+,S_0)\wedge(t_i^-=\tau_{k,P}^-\wedge t_i^+=\tau_{k,P}^+)]\rightarrow & \\ \quad\{\bigvee_{g\in G}\exists\vec{y},t_j^-,t_j^+[W_{g,Q}(\vec{y},t_j^-,t_j^+,S_0)\wedge(t_j^-=\tau_{g,Q}^-\wedge t_j^+=\tau_{g,Q}^+)\wedge(t_i^+=t_j^-)]\}\}, & \text{(by (104) ii.) iff}\\ \mathcal{M},v\models\bigwedge_{k\in K}\{\forall\vec{x}[M_{k,P}(\vec{x},\tau_{k,P}^-,\tau_{g,P}^+,S_0)\rightarrow\{\bigvee_{g\in G}\exists\vec{y}[W_{g,Q}(\vec{y},\tau_{g,Q}^-,\tau_{g,Q}^+,S_0)\wedge(\tau_{k,P}^+=\tau_{g,Q}^-)]\}\} & \text{iff}\\ \mathcal{M},v\models\bigwedge_{k\in K}\bigvee_{g\in G}\forall\vec{x},\exists\vec{y}P(i,\vec{x},\tau_{k,P}^-,\tau_{g,P}^+)\,\mathbf{m}\,Q(j,\vec{y},\tau_{g,Q}^-,\tau_{g,Q}^+)[\sigma_i,\sigma_j] & \end{array}$$

The last one mentioning only time variables in σ_i and σ_j. This concludes the prove for **m**.

Case before: We consider the following form:

$$P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{b}\ Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j] \stackrel{def}{=} \begin{array}{l} Elapsed_P(i,\vec{x},t_i^-,t_i^+,\sigma_i) \rightarrow \\ \Big((Active_Q(j,\vec{y},t_j^-,\sigma_j) \vee Elapsed_Q(j,\vec{y},t_i^-,t_j^+,\sigma_j)) \wedge (t_i^+ > t_j^-)\Big). \end{array}$$

Analogously to the previous case, $\forall \vec{x},t_i^-,t_i^+\ \exists \vec{y},t_j^-,t_j^+\ P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{b}\ Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j]$ can be transformed into

$$\bigwedge_{k\in K}\{\forall \vec{x} M_{k,P}(\vec{x},\tau_{k,P}^-,\tau_{k,P}^+,S_0) \rightarrow \{\bigvee_{g\in G}\exists \vec{y}[W_{g,Q}(\vec{y},\tau_{g,Q}^-,\tau_{g,Q}^+,S_0) \wedge (\tau_{k,P}^+ < \tau_{g,Q}^-)]\}\}$$

mentioning only time variables in σ and σ'. This concludes the prove for **b**.

Case finishes: We consider the following form:

$$P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{f}\ Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j] \stackrel{def}{=} \begin{array}{l} Elapsed_P(i,\vec{x},t_i^-,t_i^+,\sigma_i) \rightarrow \\ \Big(Elapsed_Q(j,\vec{y},t_i^+,t_j^+,\sigma_j) \wedge (t_i^+ = t_j^+)\Big). \end{array}$$

Given the equation (107), by regression, we have that $\mathcal{M},v \models P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{f}\ Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j]$ iff

$$\begin{array}{l} \mathcal{M},v \models \{\bigvee_{k\in K}[M_{k,P}(\vec{x},t_i^-,t_i^+,S_0) \wedge (t_i^- = \tau_{k,P}^- \wedge t_i^+ = \tau_{k,P}^+)]\} \rightarrow \\ \qquad \{\bigvee_{g\in G}[M_{g,Q}(\vec{y},t_j^-,t_j^+,S_0) \wedge (t_j^- = \tau_{g,Q}^- \wedge t_j^+ = \tau_{g,Q}^+) \wedge (t_i^+ = t_j^+)]\}, \end{array}$$

Analogously to the previous cases, $\mathcal{M},v \models \forall \vec{x},t_i^-,t_i^+\ \exists \vec{y},t_j^-,t_j^+\ P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{f}\ Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j]$ iff

$$\mathcal{M},v \models \bigwedge_{k\in K}\{\forall \vec{x} M_{k,P}(\vec{x},\tau_{k,P}^-,\tau_{k,P}^+,S_0) \rightarrow \{\bigvee_{g\in G}\exists \vec{y}[M_{g,Q}(\vec{y},\tau_{g,Q}^-,\tau_{g,Q}^+,S_0) \wedge (\tau_{k,P}^+ = \tau_{g,Q}^+)]\}\}$$

mentioning only time variables in σ_i and σ_j. This concludes the prove for **f**.

Case starts: We consider the following form:

$$P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{s}\ Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j] \stackrel{def}{=} \begin{array}{l} (Elapsed_P(i,\vec{x},t_i^-,t_i^+,\sigma_i) \rightarrow \\ (Active_Q(j,\vec{y},t_j^-,\sigma_j) \vee Elapsed_Q(j,\vec{y},t_i^-,t_j^+,\sigma_j)) \wedge (t_i^- = t_j^-)) \\ \wedge \\ (Active_P(i,\vec{x},t_i^-,\sigma_i) \rightarrow \\ (Active_Q(j,\vec{y},t_j^-,\sigma_j) \vee Elapsed_Q(j,\vec{y},t_i^-,t_j^+,\sigma_j)) \wedge (t_i^- = t_j^-)). \end{array}$$

Given the equation (107), we have that $\mathcal{M},v \models P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{s}\ Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j]$ iff

$$\begin{array}{ll} \mathcal{M},v \models & \{\bigvee_{k\in K}[M_{k,P}(\vec{x},t_i^-,t_i^+,S_0) \wedge (t_i^- = \tau_{k,P}^- \wedge t_i^+ = \tau_{k,P}^+)]\} \rightarrow \\ & \qquad \{\bigvee_{g\in G}[W_{g,Q}(\vec{y},t_j^-,t_j^+,S_0) \wedge (t_j^- = \tau_{g,Q}^- \wedge t_j^+ = \tau_{g,Q}^+) \wedge (t_i^- = t_j^-)]\} \wedge \\ & \{\bigvee_{k\in K}[N_{k,P}(\vec{x},t_i^-,t_i^+,S_0) \wedge (t_i^- = \tau_{k,P}^-)]\} \rightarrow \\ & \qquad \{\bigvee_{g\in G}[W_{g,Q}(\vec{y},t_j^-,t_j^+,S_0) \wedge (t_j^- = \tau_{g,Q}^- \wedge t_j^+ = \tau_{g,Q}^+) \wedge (t_i^- = t_j^-)]\}, \end{array}$$

Given this form, we have that $\mathcal{M},v \models \forall \vec{x},t_i^-,t_i^+\ \exists \vec{y},t_j^-,t_j^+\ P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{s}\ Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j]$ iff

$$\begin{array}{ll} \mathcal{M},v \models & \wedge_{k\in K}\{\forall \vec{x} M_{k,P}(\vec{x},\tau_{k,P}^-,\tau_{k,P}^+,S_0) \rightarrow \{\bigvee_{g\in G}\exists \vec{y}[W_{g,Q}(\vec{y},\tau_{g,Q}^-,\tau_{g,Q}^+,S_0) \wedge (\tau_{k,P}^- = \tau_{g,Q}^-)]\}\} \wedge \\ & \wedge_{k\in K}\{\forall \vec{x} N_{k,P}(\vec{x},\tau_{k,P}^-,\tau_{k,P}^+,S_0) \rightarrow \{\bigvee_{g\in G}\exists \vec{y}[W_{g,Q}(\vec{y},\tau_{g,Q}^-,\tau_{g,Q}^+,S_0) \wedge (\tau_{k,P}^- = \tau_{g,Q}^-)]\}\} \end{array}$$

mentioning only time instances or variables in σ_i and σ_j. This concludes the prove for **s**.

Case during: We consider the following form:

$$\begin{aligned} P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{d}\ Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j] \quad \stackrel{def}{=} \quad & (Elapsed_P(i,\vec{x},t_i^-,t_i^+,\sigma_i) \rightarrow \\ & (Active_Q(j,\vec{y},t_j^-,\sigma_j) \vee Elapsed_Q(j,\vec{y},t_i^-,t_j^+,\sigma_j)) \wedge \\ & (t_j^- \leq t_i^- \wedge t_i^+ \leq t_j^+)) \wedge \\ & (Active_P(i,\vec{x},t_i^-,\sigma_i) \rightarrow \\ & (Active_Q(j,\vec{y},t_j^-,\sigma_j) \vee Elapsed_Q(j,\vec{y},t_i^-,t_j^+,\sigma_j)) \wedge (t_j^- \leq t_i^-)). \end{aligned}$$

Analogously to the previous case, $\mathcal{M} \models \forall \vec{x}, t_i^-, t_i^+\ \exists \vec{y}, t_j^-, t_j^+\ P(i,\vec{x},t_i^-,t_i^+)\ \mathbf{d}\ Q(j,\vec{y},t_j^-,t_j^+)[\sigma_i,\sigma_j]$ iff

$$\begin{aligned} \mathcal{M}, v \models \quad & \bigwedge_{k\in K}\{\forall \vec{x} M_{k,P}(\vec{x},\tau_{k,P}^-,\tau_{k,P}^+,S_0) \rightarrow \{\bigvee_{g\in G} \exists \vec{y}[W_{g,Q}(\vec{y},\tau_{g,Q}^-,\tau_{g,Q}^+,S_0) \wedge (\tau_{g,Q}^- \leq \tau_{i,P}^- \wedge \tau_{i,P}^+ \leq \tau_{g,Q}^+)]\}\} \wedge \\ & \bigwedge_{k\in K}\{\forall \vec{x} N_{k,P}(\vec{x},\tau_{k,P}^-,\tau_{k,P}^+,S_0) \rightarrow \{\bigvee_{g\in G} \exists \vec{y}[W_{g,Q}(\vec{y},\tau_{g,Q}^-,\tau_{g,Q}^+,S_0) \wedge (\tau_{g,Q}^- \leq \tau_{i,P}^-)]\}\} \end{aligned}$$

mentioning only time variables or instances in σ_i and σ_j. This concludes the prove for **d**. □

Concluding the proof of Theorem 6:

To conclude the proof we consider the trivial transformation from CNF to DNF formulas, i.e. the CNF form $\bigwedge_{d\in D} \bigvee_{w\in W_d} B_{d,w}$, from the Lemma, can be expressed as an equivalent DNF form $\bigvee_{n\in N} \bigwedge_{m\in M_n} B_{n,m}$ for a suitable set of indexes D, N, W_i, and M_i.

We consider now the form (108), i.e.

$$\bigwedge_{d\in D} \bigvee_{w\in W_d} \bigwedge_{r\in R_{d,w}} \bigwedge_{k\in K_{d,w,r}} \bigvee_{g\in G_{d,w,r}} \forall \vec{x} \exists \vec{y} (P_d(i_d,\vec{x},\tau_k^-,\tau_k^+)\ \mathbf{op}_r\ Q_r(j_k,\vec{y},\tau_g^-,\tau_g^+)[\sigma_{i_h},\sigma_{j_r}]),$$

where D, W, R, K, and G are finite set of indexes, τ_k^-, τ_k^+ (τ_g^-, τ_g^+) are the temporal variables mentioned in the ground situation σ_{i_d}. We consider this formula in the following form:

$$\bigwedge_{d\in D} \bigvee_{w\in W_d} \bigwedge_{\langle r,k\rangle \in RK_{d,w}} \bigvee_{g\in G_{d,w,r}} B_{k,g}^{d,w,r}, \tag{113}$$

with $B_{k,g}^{d,w,r}$ representing $\forall \vec{x}\ \exists \vec{y}\ P_d(i_d,\vec{x},\tau_k^-,\tau_k^+)\ \mathbf{op}_r\ Q_r(j_k,\vec{y},\tau_g^-,\tau_g^+)[\sigma_{i_d},\sigma_{j_r}]$. By applying the CNF to the DNF transformation we can pass through the following equivalent forms.

From (1) $\bigwedge_{d\in D} \bigvee_{\langle w,r\rangle\in WR_d} \bigwedge_{k\in K_{d,w,r}} \bigvee_{g\in G_{d,w,r}} B_{k,g}^{d,w,r}$ we get (2) $\bigwedge_{d\in D} \bigvee_{\langle w,r\rangle\in WR_d} \bigvee_{n=\langle n_1,n_2\rangle\in N_{d,w}} \bigwedge_{m=\langle m_1,m_2\rangle\in M_{d,w,n}} B_{n,m}^{d,w}$, with $B_{n,m}^{d,w}$ representing

$$\forall \vec{x} \exists \vec{y} P_d(i_d,\vec{x},\tau_{n_1,m_1}^-,\tau_{n_1,m_1}^+) \mathbf{op}_{n_1,m_1} Q_{n_1,m_1}(j_{n_1,m_1},\vec{y},\tau_{n_2,m_2}^-,\tau_{n_2,m_2}^+)[\sigma_{i_d},\sigma_{j_{n_1,m_1}}]$$

which is equivalent to (3) $\bigwedge_{d\in D} \bigvee_{\langle w,r\rangle\in WR_d, n\in N_{d,w}} \bigwedge_{m\in M_{d,w,n}} B_{n,m}^{d,w,r}$, and the previous form can be expressed as (4) $\bigwedge_{d\in D} \bigvee_{\langle w,r,n\rangle\in WRN_d} \bigwedge_{m\in M_{d,w,n}} B_{n,m}^{d,w,r}$, where $\langle w,n\rangle \in WRN_d$ abbreviates $\langle w,r\rangle \in WR_d$, $n \in N_{d,w}$.

By applying again the CNF to DNF transformation we get (5) $\bigvee_{z\in Z} \bigwedge_{s=\langle s_1,s_2,s_3\rangle\in S_z} \bigwedge_{m\in M_{z,s,n}} B_m^{z,s}$,

$$\forall \vec{x} \exists \vec{y} P_{z,s_1}(i_{z,s_1},\vec{x},\tau_{s_2,m_1}^-,\tau_{s_2,m_1}^+) \mathbf{op}_{s_3,m_1} Q_{s_3,m_1}(j_{s_3,m_1},\vec{y},\tau_{s_2,m_2}^-,\tau_{s_2,m_2}^+)[\sigma_{i_{z,s_1}},\sigma_{j_{s_1,m_1}}]$$

hence, if we call $\bigwedge_{s\in S_z} \bigwedge_{m\in M_{z,s,n}}$ as $\bigwedge_{q=\langle q_1,q_2,q_3,q_4\rangle\in Q_z}$, we get (6) $\bigvee_{z\in Z} \bigwedge_{q\in Q_z} B_q^z$.

Now, from (6), with B_q^z representing

$$\forall \vec{x} \exists \vec{y} P_{z,q_1}(i_{z,q_1},\vec{x},\tau_{z,q_2}^-,\tau_{z,q_2}^+) \mathbf{op}_{z,q_3} Q_{z,q_3}(j_{z,q_3},\vec{y},\tau_{z,q_4}^-,\tau_{z,q_4}^+)[\sigma_{i_{z,q_1}},\sigma_{j_{z,q_3}}].$$

We obtain the required form (32). This concludes the proof of the theorem. □

D.2 Proof of Theorem 7

First note that each symbol **op** is associated to the temporal relation $\gamma_{\mathbf{op}}(t_1^-, t_1^+, t_2^-, t_2^+)$ obtained as a combination of relations = and < as specified in equation (31). Let $\mathcal{M}$ be a structure of $\mathcal{L}_{TFSC}$ such that $\mathcal{M}$ is a model of $\mathcal{D}_T$ and suppose that for some assignment v the followings hold:

(*i*) $\mathcal{M}, v \models \forall\vec{x}\exists\vec{y}\, P(i, \vec{x}, \tau_p^-, \tau_p^+)\ \mathbf{op}\ Q(j, \vec{y}, \tau_q^-, \tau_q^+)[\sigma_i, \sigma_j]$,
(*ii*) $\mathcal{M}, v \models \exists\vec{x}\, Elapsed_P(i, \vec{x}, \tau_p^-, \tau_p^+, \sigma_i)$ or $\mathcal{M}, v \models \exists\vec{x}\, Active_P(\vec{x}, \tau_p^-, \sigma_i)$.

We can prove the theorem by cases for each **op** in {**m**, **s**, **f**, **b**, **d**}.

First of all we consider the case of **m**. Since

$$P(i, \vec{x}, t_i^-, t_i^+)\ \mathbf{m}\ Q(j, \vec{y}, t_j^-, t_j^+)[\sigma_i, \sigma_j] \stackrel{def}{=} Elapsed_P(i, \vec{x}, t_i^-, t_i^+, \sigma_i) \rightarrow \Big((Active_Q(j, \vec{y}, t_j^-, \sigma_j) \vee Elapsed_Q(j, \vec{y}, t_i^-, t_j^+, \sigma_j)) \wedge (t_i^+ = t_j^-)\Big),$$

we have that for any model $\mathcal{M} \models \mathcal{D}_T$ and assignment v such that the items (i) and (ii) hold,

$$\mathcal{M}, v \models Elapsed_P(i, \vec{x}, \tau_p^-, \tau_p^+, \sigma_i) \rightarrow \Big((Active_Q(j, \vec{y}, \tau_q^-, \sigma_j) \vee Elapsed_Q(j, \vec{y}, \tau_q^-, \tau_q^+, \sigma_j)) \wedge (\tau_p^+ = \tau_q^-)\Big),$$

and,

$$\mathcal{M}, v \models \Big((Active_Q(j, \vec{y}, \tau_q^-, \sigma_j) \vee Elapsed_Q(j, \vec{y}, \tau_q^-, \tau_q^+, \sigma_j)) \wedge (\tau_p^+ = \tau_q^-)\Big). \tag{114}$$

Thus, by (i) and (ii) we get $\mathcal{M}, v \models (\tau_p^+ = \tau_q^-)$,

Let $[d_p^-, d_p^+\, d_q^-, d_q^+]$ be an assignment to the variables according to v (or an interpretation of the ground terms) then, since $\mathcal{M}, v \models (\tau_p^+ = \tau_q^-)$, by the above equation 114 it follows that the algebraic relation $\gamma_m(d_p^-, d_p^+, d_q^-, d_q^+)$ holds too in $\mathcal{M}$.

The proof for **b** and **f** is analogous.

For **s**, we have the following macro expansion:

$$\begin{aligned} P(i, \vec{x}, t_i^-, t_i^+)\ \mathbf{s}\ Q(j, \vec{y}, t_j^-, t_j^+)[\sigma_i, \sigma_j] \stackrel{def}{=}\ & (Elapsed_P(i, \vec{x}, t_i^-, t_i^+, \sigma_i) \rightarrow \\ & (Active_Q(j, \vec{y}, t_j^-, \sigma_j) \vee Elapsed_Q(j, \vec{y}, t_i^-, t_j^+, \sigma_j)) \wedge (t_i^- = t_j^-)) \\ & \wedge \\ & (Active_P(i, \vec{x}, t_i^-, \sigma_i) \rightarrow \\ & (Active_Q(j, \vec{y}, t_j^-, \sigma_j) \vee Elapsed_Q(j, \vec{y}, t_i^-, t_j^+, \sigma_j)) \wedge (t_i^- = t_j^-)), \end{aligned}$$

we have that for any model $\mathcal{M} \models \mathcal{D}_T$ and assignment v and such that (i) holds

$$\mathcal{M}, v \models Active_P(i, \vec{x}, \tau_p^-, \sigma_i) \rightarrow \Big((Active_Q(j, \vec{y}, \tau_q^-, \sigma_j) \vee Elapsed_Q(j, \vec{y}, \tau_q^-, \tau_q^+, \sigma_j)) \wedge (\tau_p^- = \tau_q^-)\Big),$$

and

$$\mathcal{M}, v \models Elapsed_P(i, \vec{x}, \tau_p^-, \tau_p^+, \sigma_i) \rightarrow \Big((Active_Q(j, \vec{y}, \tau_q^-, \sigma_j) \vee Elapsed_Q(j, \vec{y}, \tau_q^-, \tau_q^+, \sigma_j)) \wedge (\tau_p^- = \tau_q^-)\Big).$$

By (ii), either $\mathcal{M}, v \models Active_P(i, \vec{x}, \tau_p^-, \sigma_i)$ or $\mathcal{M}, v \models Elapsed_P(i, \vec{x}, \tau_p^-, \tau_p^+, \sigma_i)$, in both cases $\mathcal{M}, v \models (\tau_p^- = \tau_q^-)$. Hence given the assignment $[d_p^-, d_p^+\, d_q^-, d_q^+]$ to the time variables, the relation $\gamma_{\mathbf{s}}(d_p^-, d_p^+, d_q^-, d_q^+)$ holds in $\mathcal{M}$.

For **d**, we are given the following form:

$$\begin{aligned} P(i, \vec{x}, t_i^-, t_i^+)\ \mathbf{d}\ Q(j, \vec{y}, t_j^-, t_j^+)[\sigma_i, \sigma_j] \stackrel{def}{=}\ & (Elapsed_P(i, \vec{x}, t_i^-, t_i^+, \sigma_i) \rightarrow \\ & (Active_Q(j, \vec{y}, t_j^-, \sigma_j) \vee Elapsed_Q(j, \vec{y}, t_i^-, t_j^+, \sigma_j)) \wedge \\ & (t_j^- \leq t_i^- \wedge t_i^+ \leq t_j^+)) \wedge \\ & (Active_P(i, \vec{x}, t_i^-, \sigma_i) \rightarrow \\ & (Active_Q(j, \vec{y}, t_j^-, \sigma_j) \vee Elapsed_Q(j, \vec{y}, t_i^-, t_j^+, \sigma_j)) \wedge (t_j^- \leq t_i^-)), \end{aligned}$$

we have that, by (i) and (ii), either $\mathcal{M}, v \models Elapsed_P(i, \vec{x}, \tau_p^-, \tau_p^+, \sigma_i)$ then $\mathcal{M}, v \models (\tau_q^- \leq \tau_p^- \wedge \tau_p^+ \leq \tau_q^+)$ or $\mathcal{M}, v \models Active_P(i, \vec{x}, \tau_p^-, \sigma_i)$ then $\mathcal{M}, v \models (\tau_q^- \leq \tau_q^-)$. Also in this case, given the assignment v, mapping the time variables to $[d_p^+\ d_q^-, d_q^+, d_q^-]$ $d_q^- \leq d_p^-$ or $d_q^- \leq d_p^- \wedge d_p^+ \leq d_q^+$. The case for ground terms is analogous. Thus, either the relation $\gamma_{\mathbf{d}}(d_p^-, d_p^+, d_q^-, d_q^+)$ holds or its relaxed version $\gamma'_{\mathbf{d}}(d_p^-, d_p^+)$ holds. □

D.3 Proof of Corollary 2

It is consequence of Theorem 6, Theorem 7, and of the network construction. Indeed, by Theorem 6, we have that $\mathbb{I}(T_c, \mathbb{s}[\omega])$ can be reduced to the form (32):

$$\bigvee_{z \in Z} \bigwedge_{\langle q_1, q_2, q_3, q_4 \rangle \in J_z} \forall \vec{x} \exists \vec{y}. P_{z,q_1}(i_{z,q_1}, \vec{x}, \tau_{z,q_2}^-, \tau_{z,q_2}^+)\ \mathbf{op}_{z,q_3}\ Q_{z,q_3}(j_{z,q_3}, \vec{y}, \tau_{z,q_4}^-, \tau_{z,q_4}^+)[\sigma_{i_{z,q_1}}, \sigma_{j_{z,q_3}}].$$

Form this, following the network construction steps (a)-(d), we can build the temporal constraint network ζ:

$$\bigvee_{z \in Z} \bigwedge_{(q_1, q_2, q_3, q_4) \in J_z} \mu^{q_1}_{\mathbf{op}_{z,q_3}}(\tau_{z,q_2}^-, \tau_{z,q_2}^+, \tau_{z,q_4}^-, \tau_{z,q_4}^+).$$

We have to show that, given a model $\mathcal{M}$ for $\mathcal{D}_T$: (1) if $\mathcal{M}, v \models \mathbb{I}(T_c, \mathbb{s}[\omega])$, then the assignment v represents a solution for $\zeta(\mathcal{D}_T, T_c, \mathbb{s}[\omega])$; (2) given an assignment v which is a solution for $\zeta(\mathcal{D}_T, T_c, \mathbb{s}[\omega])$, then $\mathcal{M}, v \models \mathbb{I}(T_c, \mathbb{s}[\omega])$.

(1) If $\mathcal{M}, v \models \mathbb{I}(T_c, \mathbb{s}[\omega])$, then, given the (32) form, there exists at least one $z \in Z$ such that $\mathcal{M}, v \models \bigwedge_{\langle q_1, q_2, q_3, q_4 \rangle \in J_z} \vec{x}\exists \vec{y}. P_{z,q_1}(\cdot)\ \mathbf{op}_{z,q_2}\ Q_{z,q_3}(\cdot)$. Therefore, for each associated conjunct, indexed by $\langle q_1, q_2, q_3, q_4 \rangle \in J_z$, $\mathcal{M}, v \models \forall \vec{x} \exists \vec{y}. P_{z,q_1}(\cdot)\ \mathbf{op}_{z,q_2}\ Q_{z,q_3}(\cdot)$.

If (i) and (ii) of Theorem 7 are satisfied, then we can apply Theorem 7 that ensures that $\mu^{q_1}_{op_{z,q_3}}$ (obtained from m_2 if $E_{P_{z,q_1}}$ holds or from m_3 if $A_{P_{z,q_1}}$ holds) is satisfied by the assignment v. In the remaining case, since (i) and (ii) are not satisfied then N_P holds, thus $\mu^{p_1}_{op_{z,p_3}}$ is trivially satisfied because it does not impose any constraint on the associated variables. This concludes the proof for (1).

(2) As for the other direction, we have to show that, given a model $\mathcal{M}$ of $\mathcal{D}_T$, given an assignment solution V for the temporal network $\zeta(\mathcal{D}_T, T_c, \mathbb{s}[\omega])$, then the assignment v that is like V w.r.t. the time variables ω is such that $\mathcal{M}, v \models \mathbb{I}(T_c, \mathbb{s}[\omega])$. Analogously to the previous case, since ζ is a disjunction of conjunctions, if ζ is satisfiable, then there exists at least one disjunct $\bigwedge_{\langle q_1, q_2, q_3, q_4 \rangle \in J_z} \mu^{q_1}_{z,q_3}(\cdot)$, indexed by $z \in Z$, such that V is an assignment solution.

Now, given one of the conjuncts $\mu^{q_1}_{z,q_3}(\cdot)$, by the step (c) of the ζ construction, there exists an associated conjunct $\forall \vec{x} \exists \vec{y}. P_{z,q_1}(\cdot)\ \mathbf{op}_{z,q_2}\ Q_{z,q_3}(\cdot)$ in (32) that is also *partially consistent*. We show that, given the assignment v restricted to the time variables ω that is like V on these, $\mathcal{M}, v \models \forall \vec{x} \exists \vec{y}. P_{z,q_1}(\cdot)\ \mathbf{op}_{z,q_2}\ Q_{z,q_3}(\cdot)$ for each $\mathbf{op}_{z,q_2} \in \{\mathbf{m}, \mathbf{f}, \mathbf{s}, \mathbf{d}, \dots\}$.

For each of these cases, $P_{z,q_1}(\cdot)\ \mathbf{op}_{z,q_2}\ Q_{z,q_3}(\cdot)$ can be reduced to the following form (A) $\bigvee(E_P(\vec{x}, \vec{\tau}, \cdot) \rightarrow (Q_{E_P}(\vec{y}, \vec{\tau}, \cdot) \wedge \mu_{\mathbf{op}}(\vec{\tau})))$, where the $E_P(\vec{x}, \vec{\tau}, \cdot)$ are mutually exclusive in the disjunction (i.e. given an assignment for $\vec{x}$,$\vec{\tau}$,and $\cdot$ at least one $\mu_{\mathbf{op}}$ is enabled). We may assume, by contradiction, that v is such that $\mathcal{M}, v \not\models \forall \vec{x} \exists \vec{y}. P_{z,q_1}(\vec{x}, \vec{\tau})\ \mathbf{op}_{z,q_2}\ Q_{z,q_3}(\vec{y}, \vec{\tau})$, hence $\mathcal{M}, v \not\models \forall \vec{x} \exists \vec{y}. \bigvee(E_P(\vec{x}, \vec{\tau}, \cdot) \rightarrow (Q_{E_P}(\vec{y}, \vec{\tau}, \cdot) \wedge \mu_{\mathbf{op}}(\vec{\tau})))$. However, by FOL, from this we get that $\mathcal{M}, v \not\models \forall \vec{x}. \bigvee(E_P(\vec{x}, \vec{\tau}, \cdot) \rightarrow (\exists \vec{y} Q_{E_P}(\vec{y}, \vec{\tau}, \cdot) \wedge \mu_{\mathbf{op}}(\vec{\tau})))$. From this, it follows that there exists v^* that extends v with an assignment for $\vec{x}$, such that $\mathcal{M}, v^* \not\models \bigvee(E_P(\vec{x}, \vec{\tau}, \cdot) \rightarrow (\exists \vec{y} Q_{E_P}(\vec{y}, \vec{\tau}, \cdot) \wedge \mu_{\mathbf{op}}(\vec{\tau})))$, hence, for each disjunct, $\mathcal{M}, v^* \not\models E_P(\vec{x}, \vec{\tau}, \cdot) \rightarrow (\exists \vec{y} Q_{E_P}(\vec{y}, \vec{\tau}, \cdot) \wedge \mu_{\mathbf{op}}(\vec{\tau}))$. At this point, for each disjoint, we have two possible cases: (B) $\mathcal{M}, v^* \not\models E_P(\vec{x}, \vec{\tau}, \cdot) \rightarrow \exists \vec{y} Q_{E_P}(\vec{y}, \vec{\tau}, \cdot)$ or (C) $\mathcal{M}, v^* \not\models E_P(\vec{x}, \vec{\tau}, \cdot) \rightarrow \mu_{\mathbf{op}}(\vec{\tau})$. However, by the partial consistency assumption, (B) is contradicted in at least one disjoint. On the other hand, (C) requires that $\mathcal{M}, v^* \models E_P(\vec{x}, \vec{\tau}, \cdot)$ and (D) $\mathcal{M}, v^* \not\models \mu_{\mathbf{op}}(\vec{\tau})$. But (D) contradicts the assumption, in fact, by assumption, v^* restricted to the temporal variables is like V that solves the algebraic relation represented by $\mu_{\mathbf{op}}(\vec{\tau})$.

This concludes the proof for (2). □

E Proofs for Section 7

E.1 Proof of Proposition 3

We have to show that given a model $\mathcal{M}$ of $\mathcal{D}_T$ if

$$\mathcal{M} \models DoTF(prog, \mathbb{s}_{init}, \mathbb{s}, (h_s, h_e)),$$

and assuming that $ttime(\mathbb{s}) \leq h_e$ and $h_s \leq ttime(\mathbb{s}_{init})$, then

$$\mathcal{M} \models \mathbb{s}_{init} \preceq_{\mathscr{S}} \mathbb{s}.$$

We shall show the statement by induction on the structure of $prog$.
The basic case is given for the primitive action $prog = a$. In this case, we have to show that if

$$\mathcal{M} \models DoTF(a, \mathbb{s}_{init}, \mathbb{s}, (h_s, h_e)),$$

then $\mathbb{s}_{init} \preceq_{\mathscr{S}} \mathbb{s}$. However, by definition, either $\mathcal{M} \models \mathbb{s} = ddo(a, s, \mathbb{s}_{init})$ or $\mathcal{M} \models \mathbb{s}_{init} = \mathbb{s}$, in both cases the statement holds for the basic case.

Now assume, by induction, that the statement holds for $DoTF(prog, \mathbb{s}_{init}, \mathbb{s}, (h_s, h_e))$, we show the following constructs.

1. Consider the *program sequence*: $DoTF(prog_1 ; prog_2, \mathbb{s}_{init}, \mathbb{s}, (h_s, h_e))$, by definition and inductive hypothesis we have that there exists $\mathbb{s}''$ such that $\mathbb{s}_{init} \preceq_{\mathscr{S}} \mathbb{s}''$ and $\mathbb{s}'' \preceq_{\mathscr{S}} \mathbb{s}$, hence, by transitivity we have $\mathbb{s}_{init} \preceq_{\mathscr{S}} \mathbb{s}$.

2. Consider the *Partial-order action choice*: $DoTF(prog_1 \prec prog_2, \mathbb{s}_{init}, \mathbb{s}, (h_s, h_e))$. By definition and the assumptions

$$\mathcal{M} \models \exists \mathbb{s}'', \mathbb{s}'''(DoTF(prog_1, \mathbb{s}_{init}, \mathbb{s}'', (h_s, h_e)) \wedge DoTF(prog_2, \mathbb{s}''', \mathbb{s}, (h_s, h_e)) \wedge \mathbb{s}'' \preceq_{\mathscr{S}} \mathbb{s}''') \quad (115)$$

Thus, there are two bags of timelines $\mathbb{s}'', \mathbb{s}'''$ such that $\mathbb{s}_{init} \preceq_{\mathscr{S}} \mathbb{s}''$ and $\mathbb{s}''' \preceq_{\mathscr{S}} \mathbb{s}$ by inductive hypothesis, and $\mathbb{s}'' \preceq_{\mathscr{S}} \mathbb{s}'''$ by definition, therefore by transitivity of $\preceq_{\mathscr{S}}$ we obtain that $\mathbb{s}_{init} \preceq_{\mathscr{S}} \mathbb{s}$

3. Consider *Nondeterministic action choice*: $\mathcal{M} \models DoTF(prog_1 | prog_2, \mathbb{s}_{init}, \mathbb{s}, (h_s, h_e))$. By definition,

$$\mathcal{M} \models DoTF(prog_1 , \mathbb{s}_{init}, \mathbb{s}, (h_s, h_e)) \vee DoTF(prog_2, \mathbb{s}_{init}, \mathbb{s}, (h_s, h_e)).$$

hence, by inductive hypothesis, $\mathbb{s}_{init} \preceq_{\mathscr{S}} \mathbb{s}$.

4. Consider the *Nondeterministic iteration*: $\mathcal{M} \models DoTF(prog^*, \mathbb{s}_{init}, \mathbb{s}, (h_s, h_e))$. By inductive hypothesis we have that, if $\mathcal{M} \models DoTF(prog, \mathbb{s}', \mathbb{s}'', (h_s, h_e))$, then $\mathbb{s}' \preceq_{\mathscr{S}} \mathbb{s}''$, hence, by transitive closure, $\mathbb{s}_{init} \preceq_{\mathscr{S}} \mathbb{s}$.

The other cases follow by analogous reasoning.

□

E.2 Proof of Proposition 4

Analogously to the previous proof, we show the statement by induction on the structure of $prog$. The base of the induction uses the *primitive action* cases and the other statements are proved using the inductive hypothesis.

1. *Primitive action*. If $prog = a$, the executability is a direct consequence of the executability of a in $\mathbb{s}_{init}$. Indeed, in any model $\mathcal{M}$ of $\mathcal{D}_T$, if $DoTF(a, \mathbb{s}_{init}, \mathbb{s}', (h_s, h_e))$ and $ttime(\mathbb{s}) \geq h_s$ and $ttime(\mathbb{s}') \leq h_e$ hold, by definition of $DoTF$ and FOL, we have that $\exists s(s \in \mathbb{s} \wedge a =_v s \wedge Poss(a, s) \wedge time(s) \geq h_s \wedge time(s) \leq h_e \wedge time(s) \geq time(a) \wedge (time(a) \leq h_e \wedge \mathbb{s}' = ddo(a, s, \mathbb{s}))$. Hence, if $\sigma \in \mathbb{s}'$ holds, either $\sigma \in \mathbb{s}_{init}$ or $\sigma = do(a, \sigma')$ with $\sigma' \in \mathbb{s}_{init}$. In the two cases, by assumption and definition of $DoTF$, $executable(\sigma)$ obtains.

2. *Program sequence*. If $DoTF(prog_1 ; prog_2, \mathbb{s}_{init}, \mathbb{s}, (h_s, h_e))$ holds, we can assume by induction that, there exists $\mathbb{s}''$ such that the property holds for: (1) $DoTF(prog_1, \mathbb{s}_{init}, \mathbb{s}'', (h_s, h_e))$; (2) $DoTF(prog_1, \mathbb{s}'', \mathbb{s}', (h_s, h_e))$. Therefore, by assumption and (1) we can conclude that any $\sigma \in \mathbb{s}''$ is executable. Now, since any $\sigma \in \mathbb{s}''$ is executable, we can apply the inductive hypothesis to (2) and conclude that if $\sigma \in \mathbb{s}$ then σ is executable.

3. *Partial-order action choice*. Analogously to the previous case, if $DoTF(prog_1 \prec prog_2, \mathbb{s}_{init}, \mathbb{s}, (h_s, h_e))$ holds, there exists $\mathbb{s}''$ and $\mathbb{s}'$ such that (1) $DoTF(prog_1, \mathbb{s}_{init}, \mathbb{s}'', (h_s, h_e))$ and (2) $DoTF(prog_2, \mathbb{s}', \mathbb{s}, (h_s, h_e))$ with (3) $exec(\mathbb{s}'', \mathbb{s}', (h_s, h_e))$. Now, from (1) and (3), by the inductive hypothesis and definition of $exec$, any $\sigma \in \mathbb{s}'$ is executable. Therefore, by (3) and the inductive hypothesis, we can also conclude that if $\sigma \in \mathbb{s}$ then σ is executable.

4. *Nondeterministic iteration*. By the inductive hypothesis, if $DoTF(prog, \mathbb{s}', \mathbb{s}'', (h_s, h_e))$ holds and $\sigma \in \mathbb{s}'$, only if it is executable, then $\sigma' \in \mathbb{s}''$ only if it is executable. Analogously to Proof E.1, the statement can be proved by transitive closure.

5. The cases of *test*, *nondet. choice of actions*, *nondet. choice of argument* are straightforward.

□

E.3 Proof of Proposition 5

The proof is a direct consequence of Corollary 2.

By FOL, we have that, for any model $\mathcal{M}$ of $\mathcal{D}_T$, and for any assignment v to ω, $\mathcal{M}, v \models DoTF(prog, \mathbb{s}_{init}, \mathbb{s}[\omega], (h_s, h_e)) \wedge \mathbb{I}(T_c, \mathbb{s}[\omega])$ iff $\mathcal{M}, v \models DoTF(prog, \mathbb{s}_{init}, \mathbb{s}[\omega], (h_s, h_e))$ and $\mathcal{M}, v \models \mathbb{I}(T_c, \mathbb{s}[\omega])$.

However, by Corollary 2 we have that, given a model $\mathcal{M}$ of $\mathcal{D}_T$, for any assignment v to the free variables ω of $\mathbb{s}[\omega]$, $\mathcal{M}, v \models \mathbb{I}(T_c, \mathbb{s}[\omega])$ iff v is an assignment solution for the $network(\mathcal{D}_T, T_c, \mathbb{s}[\omega])$.

Hence, by FOL and Corollary 2, we have that for any model $\mathcal{M}$ of $\mathcal{D}_T$, for any assignment v to ω, $\mathcal{M}, v \models DoTF(prog, \mathbb{s}_{init}, \mathbb{s}[\omega], (h_s, h_e)) \wedge \mathbb{I}(T_c, \mathbb{s}[\omega])$ iff $\mathcal{M}, v \models DoTF(prog, \mathbb{s}_{init}, \mathbb{s}[\omega], (h_s, h_e))$ and v is a solution for $network(\mathcal{D}_T, T_c, \mathbb{s}[\omega])$. □

E.4 Proof of Proposition 6

Given the $\mathcal{D}_{STSC}$, we can build an associated $\mathcal{D}_T$ denoting a single component. We introduce a constant v representing a unique state variable. Each action $a^{st}(\cdot)$ and fluent $P^{st}(\cdot)$ used in $\mathcal{D}_{STSC}$ is also introduced in $\mathcal{D}_T$. For each $start^{st}_p(\cdot)$ and $end^{st}_p(\cdot)$ in $\mathcal{D}_{STSC}$ we introduce an action $start^{st}(v, \cdot)$ and $end^{st}(v, \cdot)$ in $\mathcal{D}_T$. The action preconditions associated with $start^{st}(v, \cdot)$, $end^{st}_p(v, \cdot)$ are the one in $\mathcal{D}_{STSC}$. In this case, the processes are not needed, preconditions $\mathcal{D}_{ap}$ and successor state axioms $\mathcal{D}_{ss}$ coincide with $\mathcal{D}^{st}_{ap}$ and $\mathcal{D}^{st}_{ss}$ respectively. As for $\mathcal{D}_{S_0}$, this coincides with $\mathcal{D}^{st}_{S_0}$. At this point, the $\mathcal{D}_T$ is defined by $\mathcal{D}^{st}_{ss} \cup \mathcal{D}^{st\ +}_{ap} \cup \mathcal{D}^{st}_{S_0} \cup \mathcal{A}^{+} \cup \mathcal{D}_{una}$. Note that, since the durative actions in $\mathcal{D}_{STSC}$ are not the processes in $\mathcal{D}_T$, hence $\mathcal{D}_\pi$ is left empty.

We shall show the statement by induction on the structure of $prog_{st}$.

For the base step we consider the primitive action. We can show that (1) $\mathcal{D}_{STSC} \models Do(a, s, s')$ iff (2) $\mathcal{D}_T \models DoTF(a, \mathbb{s}, \mathbb{s}', (0, \infty))$. If we consider the horizon $(0, \infty)$, (1) can be reduced to

$$\mathcal{D}_T \models \exists s(s \in \mathbb{s} \wedge a =_v s \wedge Poss(a, s) \wedge time(s) \geq time(a) \wedge (\mathbb{s}' = ddo(a, s, \mathbb{s}))).$$

On the other hand, the (1) holds iff

$$\mathcal{D}_{STSC} \models Poss(a, s) \wedge start(s) \leq time(a) \wedge s' = do(a, s).$$

Since $\mathcal{D}_T$ extends $\mathcal{D}_{STSC}$, it is easy to see that (2) entails (1). As for the other direction, $\mathfrak{s}$ is composed of a single situation, therefore, for any situation σ that satisfies (2) it is possible to build a bag of situation $\mathfrak{s}$ that satisfies (1).

By induction on the structure of the program, it is easy to show that the property holds for the other standard Golog constructs. □