Representing Flexible Temporal Behaviors in the Situation Calculus

Alberto Finzi and **Fiora Pirri**
DIS- Università di Roma “La Sapienza”
Via Salaria 113, I-00198 Rome, Italy
{finzi,pirri}@dis.uniroma1.it

Abstract

In this paper we present an approach to representing and managing temporally-flexible behaviors in the Situation Calculus based on a model of time and concurrent situations. We define a new hybrid framework combining temporal constraint reasoning and reasoning about actions. We show that the Constraint Based Interval Planning approach can be imported into the Situation Calculus by defining a temporal and concurrent extension of the basic action theory. Finally, we provide a version of the Golog interpreter suitable for managing flexible plans on multiple timelines.

1 Introduction

In real-world domains robots have to perform multiple tasks either simultaneously or in rapid alternation, employing diverse sensing and actuating tools, like motion, navigation, visual exploration, mapping, and several modes of perception. To ensure a suitable multiple-task performance, some approaches (e.g. [14; 23]) have recommended that executive control processes supervise the selection, initiation, execution, and termination of actions. From these ideas, a new paradigm has been proposed, called the Constraint Based Interval Planning (CBI), which essentially amalgamates planning, scheduling and resources optimization for reasoning about all the competing activities involved in a flexible concurrent plan (see [10; 4; 7]). The CBI approach, and similar approaches emerged from the planning community, have shown a strong practical impact when it comes to real world applications (see e.g. RAX [10], IxTeT [7], INOVA [21], and RMPL [23]). However, from the standpoint of cognitive robotics it is important to ensure both optimal performance, in practical applications, and to provide a logical framework for ensuring coherence of actions preconditions and effects. The system coherence emerges as a core issue also when control processes negotiate resources allocation with individual perceptual-motor and cognitive processes; indeed, the executive has to establish priorities among individual processes to allot resources for multiple-task performance (see the discussion in [8; 11; 3]). Therefore, different, concurrent, and interleaving behaviors, subject to switching-time criteria and current situation needs, lead to a new integration paradigm. In this paper we suggest that the reactive aspects that have to cope with flexible behaviors and the cognitive capabilities enabling reasoning about these processes, can be combined in the Temporal Flexible Situation Calculus. We present a new approach to flexible behaviors, that exploits the full expressiveness of the Situation Calculus (SC) [19; 12], where computational concerns related to time can be monitored by a temporal network, obtained via a transformation of added constraints. To embed many concepts elaborated in the CBI framework we extend the SC with concurrency and time (extensions of SC with time was already explored in [15; 17; 16]), deploying Allen's interval relations [1], and further constraining the language to represent concurrent timelines. In this framework we can conjugate the advantages of the SC with the expressive power of the CBI paradigm. Our aim here is twofold: on the one hand, it is made possible to introduce a separated timeline for each component of the dynamic system (i.e. each entity, which is part of an autonomous system, such as a robot), so that concurrency and flexibility can be clearly addressed; on the other hand the causal relationships among processes can be dealt with in the SC language, which provides a clear framework for preconditions and postconditions of actions and a simple solution to the frame problem. We show that the CBI perspective, with all its arsenal of specifications in terms of flexible time, alternation constraints, resources optimization, failure recovering, and tasks scheduling, can be imported into the SC ([19; 12]), defining a temporal and concurrent extension of the basic action theory (related approaches are [15; 17; 16; 8; 20; 6]). Finally, we provide a version of the Golog language and interpreter for manipulating flexible plans on multiple timelines.

2 Preliminaries

2.1 Situation Calculus and Golog

The Situation Calculus [12; 19] is a *sorted first order language* for representing dynamic domains by means of *actions*, *situations*, and *fluents*. *Actions* and *situations* are first order terms, and *situation*-terms stand for history of actions, compound with the binary symbol do: $do(a, s)$ is the situation obtained by executing the action a after the sequence s. The dynamic domain is described by a *Basic Action Theory* BAT $= (\Sigma, \mathcal{D}_{S_0}, \mathcal{D}_{ssa}, \mathcal{D}_{una}, \mathcal{D}_{ap})$. We refer the reader to [19] for a complete introduction to the SC. Temporal Concurrent Situation Calculus ($TCSC$) has been earlier introduced in [15; 18; 17]; actions are instantaneous, and their time is selected by the function $time(.)$. Durative actions are considered as *processes* [15; 19], represented by fluents, and durationless actions are to start and terminate these processes. For example, $going(hill, s)$ is started by the action $startGo(hill, t)$ and it is ended by $endGo(hill, t')$.

Golog. Golog is a situation calculus-based programming language for denoting complex actions composed of the primitive (simple or concurrent) actions defined in the BAT. Golog programs are defined by means of standard (and not so-standard) Algol-like control constructs: i. action sequence: $p_1; p_2$, ii. test: $\phi?$, iii. nondeterministic action choice $p_1|p_2$, iv. conditionals, while loops, and procedure calls. An example of a Golog program is:

while $\neg at(hill, 3)$ **do**
  if $\neg(\exists x) going(x)$ **do** $\pi(t, (t < 3)?; startGo(hill, t))$

The semantics of a Golog program δ is a SC formula $Do(\delta, s, s')$ meaning that s' is a possible situation reached by δ once executed from s. For example, $Do(a;p, s, s') \doteq Do(p, do(a, s), s') \wedge Poss(a, s)$ defines the execution of an action a followed by the program p. For more details on the SC and Golog we refer the reader to [19].

2.2 Constraint Based Interval Paradigm

The constraint based interval framework (CBI) [10; 4], is a well known framework for temporal planning systems combining temporal reasoning and scheduling, including, e.g., RAX [10], IxTeT [7], and INOVA[21]. The CBI paradigm accounts for concurrency and time, and action instances and states are described in terms of temporal intervals linked by constraints. We refer to the timeline-based version of the CBI [10] where a domain behavior is seen as the continuous interaction of different components, and each component is represented by *state variables*; a single state variable is a relevant feature of the components and represents a concurrent thread. Both states and activities are uniformly treated as temporal intervals. The history of states for a state variable over a period of time is called a *timeline*. Figure 1 illustrates two timelines (state variables) repr. the engine and the navigation processes of a mobile robot: initially, the robot is $at(p_1)$ and *stop*; when it starts $go(p_1, p_2)$ the engine is $running$; the engine is $stop$ again once the rover arrives $at(p_2)$. Some temporal constraints among the activities are: $at(x)$ holds only if $stop$ holds, $go(x, y)$ is followed by $at(y)$.

Domain Constraints. A *CBI model* [10] $\mathcal{M}_{cbi}=(X, J, \mathcal{R})$ is defined by: i. a set $X=\{x_1, \ldots, x_n\}$ of state variables, one for each component (e.g. x_{loc} and x_{eng} in Fig. 1); ii. a set $J = \{J_1, \ldots, J_n\}$ s.t. for each x_i the set J_i collects the associated temporal fluents $p_{i,j}(\vec{y})$, e.g. $at(x)$ and $go(x, y)$ for $location$ state variable in Fig. 1; iii. a set of temporal constraints $\mathcal{R}=\{r_{i,j}\}$, usually called *compatibilities*: for each temporal fluent property $p_{i,j}$ there is a *compatibility* relation $r_{i,j}$ representing all its possible legal relations with the other temporal fluents, i.e. which temporal property must proceed, follow, be co-temporal, etc. to $p_{i,j}$ in a legal plan. The latter are specified in terms of metric version of temporal relations *a la Allen* [1]. E.g. the arrows in Fig. 1 illustrate the compatibility associated to the fluent at: $at(x)\ meets\ go(x, y)$, and $at(x)\ during\ stop$.

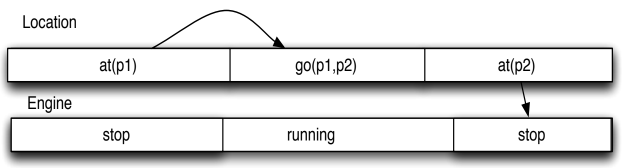

Figure 1: Timelines and constraints

Planning Problem in CBI. Given the *CBI model* $\mathcal{M}_{cbi}$, a *planning problem* is defined by $\mathcal{P}_{cbi} = (\mathcal{M}_{cbi}, P_c)$, where P_c is a *candidate plan*, representing an incomplete instance of a plan. The *candidate plan* defines both the initial situation and the goals. In particular, P_c consists of : i. a *planning horizon* h; ii. a set of temporal properties to be satisfied for each state variable $x_i \in X$ (e.g. $at(p_1)$ ends before 10 and after 20 in the timeline of Figure 1); iii. the set of precedence constraints among fluents $p_{k,1} \prec p_{k,2}$, which are to hold on a timeline, e.g. $at(p_1) \prec at(p_2)$; iv. the set of constraints $\mathcal{R}=\{r_{i,j}\}$ associated with the temporal properties $p_{i,j}$ mentioned in the timelines. A fluent $p_{i,j}(\vec{x})$ mentioned in a *candidate plan* is *fully supported* [10] if all its associated constraints $r_{i,j}$ are satisfied. E.g. in Figure 1 $at(p_1)$ and $at(p_2)$ are fully supported. A candidate plan is said to be a *complete plan* if: i. each temporal property on each timeline is fully supported; ii. all timelines fully cover the planning horizon. Given the *planning problem* $(\mathcal{M}_{cbi}, P_c)$, the planning task is to provide a *sufficient plan* [10], i.e. a *complete plan* with the maximum flexibility: the planner is to minimally ground the (temporal) variables to allow for on-line binding of the values.

3 Temporal Flexible Situation Calculus

In this section we present the Temporal Flexible Situation Calculus framework (*TFSC*), which integrates the CBI paradigm, introduced above, into the language of SC.

Actions. We define a partition $\{\alpha_v; v \in \mathcal{C}\}$ of the set of actions according to the different components $\mathcal{C}$ the system has to care of. For example, the component *Pan-Tilt-Unit* manages actions like start-scan $s_{pts}(.)$, or end-scan $e_{pts}(.)$. To induce the partition we introduce a type operator ν, and extend the SC foundational axioms with the following definitions. Let $\mathcal{H} = \bigcup_{i=1}^{n} H_i$ be a set specifying the types of actions, we assume this set finite. For each name of action the following holds:
$H_i(a) \wedge H_j(a') \rightarrow \neg(a = a')$ for $i \neq j$, and $\nu(A(\vec{x}))=\nu(A'(\vec{y})) \equiv \bigvee_{i=1}^{n} H_i(A(\vec{x})) \wedge H_i(A'(\vec{y}))$, together with the unique name for actions, here the indices stay only for different components names, a denotes an action variable, while A denotes the name of an action function.

Situations. Typization of actions induces also a partition on the set of situations whence on the set of histories. Histories become streams of situations over timelines. Typization is inherited by situations as follows:

$$\begin{aligned} &\forall a. && \nu(a) = \nu(do(a, S_0)). \\ &\forall a\ s. && \nu(a) = \nu(s) \equiv \\ & && \forall a' s'. s = do(a', s') \rightarrow \nu(a) = \nu(a') \wedge \nu(a') = \nu(s'). \end{aligned} \tag{1}$$

For each component $v \in \mathcal{C}$, a timeline is specified by a history of actions concerning v. The evolution of the set of timelines is a set of situations $sc = \{s_v | v \in \mathcal{C}\}$, called *situation class*, spanning different types. In other words, $\exists s.s \in sc$ abbreviates: $\exists s_1 \dots s_n\ \exists a_1 \dots a_n.\ \bigvee_{i=1}^{n} [s = do(a_i, s_i) \wedge \nu(a_i) = \nu(s_i) \wedge \bigwedge_{j=1, i \neq j}^{n} \nu(a_i) \neq \nu(s_j)]$. Where $\nu(s_i)$ is the type of situation s_i corresponding to some component $i \in \mathcal{C}$, $n{=}|\mathcal{C}|$. The sc class is equipped with the following $\leq_c$ relation:

$$sc' \leq_c sc \equiv \forall s'\ s.s \in sc \wedge s' \in sc' \rightarrow s' \leq s. \tag{2}$$

$do(a_v, sc)$ denotes the sequence resulting from adding the action a_v to a type-compatible situation mentioned in sc.

Time. Time is part of the sorted domain of $TFSC$, as noted in Section 2; we extend the time selection function from actions to situations (see [16] for a different notation), and situation classes as follows:

$$\begin{array}{l} time(S_0) = t_0.\ \ time(do(a, s)) = time(a). \\ time(s) \leq time(s') \equiv \exists a\ a'\ s''.s = do(a, s') \wedge \\ \quad s' = do(a', s'') \wedge time(a) \leq time(a'). \end{array} \tag{3}$$

The time of a situation class sc, depends on the time of the situations in sc, so $time(sc) = t$ abbreviates:

$$\exists s \forall s'.s \in sc \wedge time(s) = t \wedge s' \in sc \wedge time(s') = t' \rightarrow t \geq t'. \tag{4}$$

Consistency of the extension. The above axioms concerning types, time and situation classes, are added to the foundational axioms in Σ and are conservative (i.e. obtained by extending the language), therefore the extended set (which we still denote with Σ) must be consistent.

Processes. Processes span the subtree of situations, over a single interval specified by a start and end action. They are implicitly typed by the actions process: for each process there are two actions, starting and ending the process, abbreviated by s_P, meaning *starts process P* and e_P, meaning *ends process P*. A process is denoted by a fluent $P(\vec{x}, t^-, s)$, (here t^- is its start time). Successor state axioms (SSP) for processes extend the set of SSA for fluents and are defined as follows:

$$\begin{array}{l} P(\vec{x}, t^-, do(a, s)) \equiv a = s_P(\vec{x}, t^-) \vee \\ \qquad\qquad P(\vec{x}, t^-, s) \wedge \forall t.a \neq e_P(\vec{x}, t). \end{array} \tag{5}$$

For example, moving towards θ, can be defined as:

$$\begin{array}{l} move(\theta, t^-, do(a, s)) \equiv a{=}s_{move}(\theta, t^-) \vee \\ \qquad move(\theta, t^-, s) \wedge \forall t'.a \neq e_{move}(\theta, t'). \end{array}$$

Action preconditions axioms define the conditions for the $start$ and end actions to be executed. Let P be a process, we say that it is *linear* if it does not mention any other process in the right-hand side of the definition. The set $\mathcal{D}_{ssp}$ of successor state axioms for processes is *linear* if all processes mentioned in it are linear. Let $\mathcal{D}_{ssp}$ be a set of linear successor state axioms for processes, let the BAT be $\mathcal{D} \cup \mathcal{D}_{ssp} = \Sigma \cup \mathcal{D}_{S_0} \cup \mathcal{D}_{una} \cup \mathcal{D}_{ss} \cup \mathcal{D}_{AP} \cup \mathcal{D}_{ssp}$ then:

Lemma 1 *$\mathcal{D} \cup \mathcal{D}_{ssp}$ is consistent iff $\mathcal{D}_{S_0}$ is.*

Proof (sketch). The proof relies on the *relative consistency* of the SC. Furthermore, linearity of $\mathcal{D}_{ssp}$ ensures that there are no loops w.r.t. time, hence for each start and end action, as defined in the processes, it is not possible to introduce dependencies that could lead, for instance, to $s_P(\vec{x}, t) < s_Q(\vec{y}, t')$ and $s_P(\vec{x}, t) > s_Q(\vec{y}, t')$.

Intervals. Intervals mention temporal variables, which are always free, and get assigned only according to a specified set of behaviors, in so being very well suited for representing *flexible behaviors*. In this paragraph we show how embedding intervals in the SC leads to the construction of a temporal network. The following formula Ψ, specifies the conditions for a process P_i to hold on its induced timeline, from t_i^- to t_i^+ with arg. $\overline{x}$:

$$\begin{array}{l} \mathbf{\Psi_i}(\overline{x}, t_i^-, t_i^+, sc) = \exists s_i s_i' s_i''.s_i \in sc \wedge \\ \quad s_i \geq do(e_{P_i}(\overline{x}, t_i^+), s_i') > do(s_{P_i}(\overline{x}, t_i^-), s_i'') \wedge \\ \qquad \neg \exists t\ s.s_i' \geq do(e_{P_i}(\overline{x}, t), s) \geq s_i'' \end{array} \tag{6}$$

Let $\mathbf{p}, \mathbf{m}, \mathbf{o}, \mathbf{d}, \mathbf{s}, \mathbf{f}, \mathbf{e}$ abbreviate the usual *precedes, meets, overlaps, during, starts, finishes, equals*, as defined in the *temporal intervals* literature, started in [1], with *op* ranging over all relations, and let $\gamma_{\mathbf{op}}$ denote:

$$\begin{array}{l} \gamma_{\mathbf{p}} = (t_i^+ < t_j^-);\ \ \gamma_{\mathbf{m}} = (t_i^+ = t_j^-);\ \ \gamma_{\mathbf{s}} = (t_i^- = t_j^-); \\ \gamma_{\mathbf{f}} = (t_i^+ = t_j^+);\ \ \gamma_{\mathbf{e}} = (t_i^+ = t_j^+ \wedge t_i^- = t_j^-); \\ \gamma_{\mathbf{o}} = (t_i^- < t_j^- \wedge t_j^- < t_i^+ \wedge t_i^+ < t_j^+); \\ \gamma_{\mathbf{d}} = (t_i^- > t_j^- \wedge t_j^+ \geq t_i^+ \vee t_i^- \leq t_j^- \wedge t_j^+ < t_i^+). \end{array}$$

We represent the interval relations between processes, given in situation-suppressed form (i.e not mentioning situations), and taking the "end-time" t^+ as argument, according to the following macro-definition, where the free variables mentioned in Ψ_i (see (6) above) are just hidden:

$$P_i(\overline{x}, t_i^-, t_i^+)\ \mathbf{op}\ P_j(\overline{x}, t_j^-, t_j^+) \overset{\Delta}{=} \Psi_i \rightarrow [\Psi_j \wedge \gamma_{\mathbf{op}}]. \tag{7}$$

Note that the metrical version (like in [10]) including durative relations can be analogously defined. Letting each operator in the macro being not commutative, inverse can be defined as $P_i\ \mathbf{op}^{-1}\ P_j = P_j\ \mathbf{op}\ P_1$.
We assume that the domain theory $\mathcal{D}_T$ is associated with temporal relations, as those given in (7), specifying the temporal interactions between processes as constraint patterns. For example, wanting to say that *going* has always to be preceded by *beingAt* we would associate to $\mathcal{D}_T$ the set $\{going([t_g^-, t_g^+])\ \mathbf{p}\ beingAt([t_b^-, t_b^+])$. We call this set the *temporal compatibilities* abbreviated by T_c.

We denote $\mathbb{I}(\tau, sc)$ the set of the (7) interval relations associated with the situation class sc, where τ is the set of time variables, varying over the intervals $[t^+, t^-]$, related to the set of timelines sc. The $\mathbb{I}(\tau, sc)$ is obtained by abducing the time constraints on τ, according to a construction illustrated in the paragraphs below. For instance, for a suitable sc and $\tau = \{[t_1^-, t_1^+], [t_2^-, t_2^+], [t_3^-, t_3^+], [t_4^-, t_4^+]\}$ we can get the set:

$$\begin{array}{l} \mathbb{I}(\tau, sc) = \ \ \{P_1(t_1^-, t_1^+)\ \mathbf{s}\ P_2(t_2^-, t_2^+), P_1(t_1^-, t_1^+)\ \mathbf{o}\ P_2(t_2^-, t_2^+), \\ \quad P_1(t_1^-, t_1^+)\ \mathbf{d}\ P_2(t_2^-, t_2^+), P_3(t_3^-, t_3^+)\ \mathbf{d}\ P_2(t_2^-, t_2^+), \\ \quad P_3(t_3^-, t_3^+)\ \mathbf{s}\ P_4(t_4^-, t_4^+), P_4(t_4^-, t_4^+)\ \mathbf{d}\ P_2(t_2^-, t_2^+)\}, \end{array}$$

whose constraints are depicted in the network of Fig. 2.

Temporal Constraint Network. The satisfiability problem we are concerned with is the following. Let $\mathcal{D}_T = \mathcal{D} \cup \mathcal{D}_{ssp} \cup \mathbb{I}(\tau, sc)$ be a basic theory of actions, extended with time specifications (see eq. (3) above), successor state axioms for processes, time interval constraints, in which time variables are free, and with an associated set of compatibilities T_c. We seek an assignment set for time variables, which is a feasible solution for the constraints, and s.t. a substitution of the

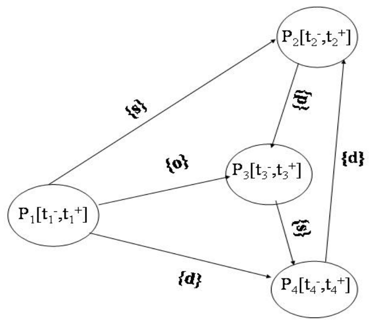

Figure 2: A TCN representing a set of interval constraints on processes P_1, P_2, P_3, P_4

free variables for this set induces an interpretation, which is a model of $\mathcal{D}_T \cup \mathbb{I}(\tau, sc)$. To find such an assignment we appeal to the concept of *general temporal constraint networks* (TCN) introduced in Meiri [13] following the one developed for discrete constraint networks [5]. A TCN involves variables denoting both intervals and points, together with unary and binary constraints. A TCN is associated with a *direct constraint graph* where each node is labeled by a temporal variable and labeled-directed edges represent the binary relations between them (e.g. $\mathbf{p}, \mathbf{e}$ etc.). According to the underlying temporal algebra, $TCNs$ express different forms of reasoning (Allen's interval algebra [1], the Point Algebra [22], metric point algebra [5], and so on) see in particular [2]. Let $\tau = \{X_1, \dots, X_n\} = \{[t_1^-, t_1^+], \dots [t_n^-, t_n^+]\}$ be a set of time variables, denoting intervals (i.e. each variable X_i denotes an interval $[t_1^-, t_1^+]$), and let R be a set of binary relations; the assignment set $V = \{[s_1, e_1], \dots, [s_n, e_n] : s_i, e_i \in \mathbb{R}, s_i < e_i\}$ is called a solution set for R, if $\{X_1 = [s_1, e_1], \dots, X_n = [s_n, e_n]\}$ satisfies all the constraints defining R. The network is consistent if at least one solution set exists. In the sequel we shall identify a TCN with its labeled direct graph and denote with V a solution set. We also say that V satisfies the constraints.

Mapping SC to TCN. Let $h : \mathbb{I}(\tau, sc) \mapsto TCN$, $h(\mathbb{I}(\tau, sc))$ is a temporal network Net, with cardinality m, specified by the nodes labeled by the defined processes $\{P_i, P_j | P_i \mathbf{op} P_j\}$, mentioned in $\mathbb{I}(\tau, sc)$.

By the above definitions the network Net is consistent if there exists an assignment V to the temporal variables τ, which is a feasible solution for $\mathbb{I}(\tau, sc)$. For example the set $V = \{[10, 160], [10, 70], [75, 95], [95, 150]\}$ is a solution set for the variables $\tau = \{[t_1^-, t_1^+], [t_2^-, t_2^+], [t_3^-, t_3^+], [t_4^-, t_4^+]\}$, of the network depicted in Figure 2, given the relations $\{\mathbf{m}, \mathbf{p}, \mathbf{d}, \mathbf{s}, \mathbf{f}\}$. Note that, in the macro definition (7), there are two parts, the left-hand component is the *definiendum* $P_i(\overline{x}, t_i^-, t_i^+)$ $\mathbf{op}$ $P_j(\overline{x}, t_j^-, t_j^+)$, which is mapped into a temporal network, while the right-hand component is the *definiens* $\Psi_i \rightarrow (\Psi_j \wedge \gamma_{op})[\tau]$, which is interpreted into a structure of the SC, where the assignments for the free temporal variables τ are obtained by the TCN. A semantic correspondence, between the network and the formulae of SC, can be established as follows. Let $\mathcal{M} = \langle D, I \rangle$ be a structure of SC (where D is a sorted domain and I an interpretation) we extend $\mathcal{M}$ with the above defined mapping h, i.e. $\mathcal{M}_h = \langle D, I, h \rangle$. Then $\mathcal{M}_h$ is a model for $\mathcal{D} \cup \mathbb{I}(\tau, sc)$, iff there is a consistent temporal network $h(\mathbb{I}(\tau, sc))$, under some assignment V to the free time-variables, satisfying $\mathbb{I}(\tau, sc)$. Given the above definitions we can state the following:

Lemma 2 *Let $\mathcal{D}_T$ be a consistent action theory. $\mathcal{D}_T \cup \mathbb{I}(\tau, sc)$ is consistent iff there exists a consistent temporal constraint network $h(\mathbb{I}(\tau, sc))$, under the assignment V, such that $\mathcal{M}_h, V \models \bigwedge \mathcal{D} \wedge \Psi[\tau]$.*

Proof(sketch). Let $h(\mathbb{I}(\tau, sc))$ be a consistent temporal network, under the assignment V. Consider the formula $\Psi[\tau] =$, $\bigwedge_{i,j \leq m} \Psi_i \rightarrow (\Psi_j \wedge \gamma_{op_{ij}})$, with τ the set of free time-variables in Ψ, and we let $\mathcal{M}$ be a model for $\mathcal{D}_T$. $\mathcal{M}$ can be extended to a model for all the $\gamma_{op_{ij}}$ according to V, as follows. For each process P_i build a chain of situations of the kind $\Gamma_i = \{do(s_{P_i}(t_i'), \sigma') \leq \sigma \leq do(e_{P_i}(t_i''), \sigma'') \leq \sigma_i\}$, with σ a sequence such that e_{P_i} is not mentioned in σ, and the free-temporal variables are from τ according to the P_i timeline in sc. Extend the structure $\mathcal{M}$ to one for $\mathcal{D}_T \cup \bigcup_i \Gamma_i$ by choosing from V a suitable assignment to the free variables τ, appearing in each Γ_i, and according to the constraints in the $\gamma_{op_{ij}}$, corresponding to those in $\mathbb{I}(\tau, sc)$, and such that each sequence is satisfied. This is always possible, because each process is made true of a situation by a start action, and false by an end action, by the sequence construction, on a timeline. Then V is an assignment to τ s.t. $(\mathcal{M}_h, V)$ is a model for $\mathcal{D}_T \cup \bigcup_i \Gamma_i[\tau]$. Finally it is enough to show that any model for $\mathcal{D}_T \cup \bigcup_i \Gamma_i[\tau]$, is a model for $\Psi[\tau]$. $\square$

By Lemma 1 $\mathcal{D}_T \cup \mathbb{I}(\tau, sc)$ is consistent iff $\mathcal{D}_{S_0} \cup \mathbb{I}(\tau, sc)$ is consistent. Logical implication can be now defined as follows. Let $\mathcal{M}_h = \langle D, I, h \rangle$, and $\beta(\tau)$ be a formula with all its free temporal variables among τ:

$$\begin{aligned} &\mathcal{D}_T \cup \mathbb{I}(\tau, sc) \models \beta(\tau) \text{ iff for any } h, \text{ for any V}, \\ &\quad \mathcal{M}_h, V \models \mathcal{D}_T \cup \mathbb{I}(\tau, sc) \text{ implies } \mathcal{M}_h, V \models \beta(\tau). \end{aligned} \tag{8}$$

TCN construction. The TCN construction involves the following inference mechanisms: i. abduce $\mathbb{I}(\tau, sc)$ to obtain the temporal network topology $h(\mathbb{I}(\tau, sc))$; ii. implement the temporal constraints into situations, according to the network. We show the inference process through an example. Let $T_c = \{P_1 \ \mathbf{m} \ P_2\}$, be the compatibilities specified in the domain theory. Let $sc = do([s_{P_1}(t_1^-), e_{P_1}(t_1^+), s_{P_2}(t_2^-), e_{P_2}(t_2^+), s_{P_1}(t_1'^-), e_{P_1}(t_1'^+)], S_0)$, then the abduced constraints, according to T_c, sc and τ, are:

$$\mathbb{I}(\tau, sc) = \{P_1[t_1^-, t_1^+] \ \mathbf{m} \ P_2[t_2{}^-, t_2{}^+], P_1[t_1'^-, t_1'^+] \ \mathbf{m} \ P_2[t_2'^-, t_2'^+]\} \tag{9}$$

From the above constraints a three nodes network $h(\mathbb{I}(\tau, sc))$ would be constructed. The time range of the solutions to the network – given the free temporal variables and the situation class sc – is a *flexible situation class*, which is a triple of the kind $\langle sc, \tau, i \rangle$, where $i(\tau)$ is the constraint over the free-variables, e.g. $i(\tau) = 7 \leq t_1^- \wedge t_2^+ > 10$.

Progression. We want to determine if, according to the timing and advancements of the current situation class, we can forget about past scheduled processes and think of future resources allocation. This is the well known progression problem [19] in the SC. We face here a simplified version of the progression problem, and with regressable formulas, which is all we need. Let us state the problem as follows. Given a sentence φ, one needs first to determine the set of situations in sc to which it can be progressed, given the timelines, and a candidate interval $t = [t^-, t^+]$. To this end one has to order the situation terms in φ according to the relation $\leq_{sc}$ introduced in (2), and get the smallest situations set, w.r.t. $\leq_{sc}$, mentioning t; let it be min. We now consider the domain theory $\mathcal{D}_T^{min}$, which is equal to $\mathcal{D}_T$ but with $\mathcal{D}_{S_0}$ replaced by a set $W^{min} = \{\varphi | \mathcal{D}_T \models \varphi\}$, where φ mentions only ground situations σ, with $\{\sigma_i\} =_{sc} min$, but no free variables of sort time. Let $\Phi(t_1, \ldots t_n)$ be a SC formula mentioning situations $\{\sigma_i\} \geq_{sc} min$, which are ground but for possible free variable of sort time all among $t_1, \ldots, t_n$. Then

Lemma 3 $\mathcal{D}_T \cup \mathbb{I}(\tau, sc) \models \bigwedge(\mathcal{D}_T^{min} \cup \mathbb{I}(\tau, sc) \cup W^{min})$.

Theorem 1 $\mathcal{D}_T \cup \mathbb{I}(\tau, sc) \models \Phi(t_1, \ldots, t_n)$ *iff* $\mathcal{D}_T^{min} \cup W^{min} \cup \mathbb{I}(\tau, sc) \models \Phi(t_1, \ldots, t_n)$

Proof. (sketch) One direction follows by cut. For the other direction, suppose $\mathcal{D}_T \cup \mathbb{I}(\tau, sc) \models \Phi(t_1, \ldots, t_n)$, let us regress $\Phi(t_1, \ldots, t_n)$ along the different timelines, this can be done by separating Φ into a DNF, and considering each conjunct separately, we regress it with respect to min, and let G be the regressed formula – note that no change is needed in regression to account for regressed sentences as time variables will be kept as they are, without turning to quantified variables through successor state axioms (as they are universally quantified in linear $\mathcal{D}_{ssp}$). Then by definition of W^{min}, $G \in W^{min}$ hence, by the previous lemma, the claim follows.

4 Flexible High Level Programming in Golog

Golog 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 sit. class.
4. *Nondet. arg. choice:* choose $\vec{x}$ for $prog(\vec{x})$.
5. *Action sequence:* $p_1; p_2$. Do p_1 followed by p_2.
6. *Partial order act. 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:* $\mathbf{if}\ \phi\ \mathbf{then}\ p_1\ \mathbf{else}\ p_2$.
9. *While loops:* $\mathbf{while}\ \phi\ \mathbf{do}\ p_1$.
10. *Procedures, including recursion.*

Golog Semantics. The macro $Do(p, sc, sc', h)$ gives the semantics for the above constructs; where p is a program, sc and sc' are situation classes, and h specifies the finite horizon.

- Null program:

$$Do(Nil, sc, sc', h) \triangleq time(sc) \leq h \wedge \forall s.[s \in sc \equiv s \in sc']$$

- Primitive first program action with horizon:

$$\begin{aligned}&Do(a_i; prog, sc, sc', h) \triangleq \exists s_i . s_i \in sc \wedge Poss(a_i, s_i) \wedge \\ &\quad time(s_i) \leq h \wedge [time(a_i) \leq h \wedge Do(prog, do(a_i, sc), sc', h) \vee \\ &\quad time(a_i) > h \wedge Do(prog, sc, sc', h)]\end{aligned}$$

Here, if the first program action is applicable to $s_i \in sc$, and a_i can be scheduled after the horizon then it is neglected (i.e. each action, which can be started after the horizon can be postponed).

- Partial order action choice:

$$\begin{aligned}&Do(prog_1 \prec prog_2, sc, sc', h) \triangleq Do(prog_1 : prog_2, sc, sc', h) \vee \\ &\quad \exists a. select(a, sc) \wedge Do(prog_1 \prec a \prec prog_2, sc, sc', h)\end{aligned}$$

Here, either the second program can be directly executed, or it is possible to insert a primitive action a, selected by a suitable fluent predicate $select(a, s)$ representing the selection criterion (set to true if no selection holds).

- Parallel execution:

$$\begin{aligned}&Do(prog_1 \parallel prog_2, sc, sc', h) \triangleq \\ &\quad Do(prog_1, sc, sc', h) \wedge Do(prog_2, sc, sc', h)\end{aligned}$$

- Test action:

$$Do(\phi?; prog, sc, sc', h) \triangleq \phi[sc] \wedge Do(prog, sc, sc', h)$$

Here $\phi[sc]$ stands for generalization of the standard $\phi[s]$ in the SC extended to situation classes, e.g. $P_1[sc] \wedge P_2[sc]$ is for $P_1(s_1) \wedge P_2(s_2)$ with $s_1, s_2 \in sc$, i.e. each fluent is evaluated w.r.t. its specific timeline.

- The semantics of nondet. action choice, nondet. argument selection, conditionals, while loops, and procedures is defined in the usual way.

Flexible Temporal Behaviors in Golog. The CBI planning problem $(\mathcal{M}_{cbi}, P_c)$ introduced in Section 2.2 can be easily coded and solved in the $TFSC$ framework. Given a $\mathcal{D}_T$ representing the timelines and processes in $\mathcal{M}_{cbi}$, a candidate plan P_c can be encoded by a Golog program $prog_c$. This is possible once we introduce, for each interval constraint for $p_{i,j}(\vec{r})$ in P_c, a Golog procedure of the kind:

$$\mathbf{proc}(\pi_{i,j},\ (\psi_{\mathbf{i}}^{\mathbf{j}}(\vec{r}, t^-, t^+) \wedge \gamma(t^-, t^+))?)\,,$$

where $\psi_{\mathbf{i}}^{\mathbf{j}}$ is the macro (6) associated with the process P_i^j (i.e. i-th process in the j-th timeline), and γ any temporal constraint. For example, $go(d, e)$ ends in $[6, 10]$ can be represented as

$$\mathbf{proc}(\pi_{1,1},\ (\psi_{\mathbf{go}}^{\mathbf{nav}}(d, e, t^-, t^+) \wedge 6 \leq t^+ \leq 10)?).$$

Given the $\pi_{i,j}$ procedures, a partial specification of a single timeline $\mathcal{T}_j$ can be easily defined using the $\prec$ operator:

$$\mathbf{proc}(plan\_\mathcal{T}_j, \pi_{0,j} \prec \pi_{1,j} \prec \pi_{2,j} \prec \ldots \prec \pi_{k,j}),$$

Given a set of timelines $\{\mathcal{T}_i\}$, a candidate plan P_c can be represented as a parallel execution of the $plan\_\mathcal{T}_i$:

$$\mathbf{proc}(prog_c,\ plan\_\mathcal{T}_1\ : Nil \parallel \ldots \parallel plan\_\mathcal{T}_k\ :\ Nil\ ).$$

Since a CBI complete plan P is associated with a fully supported set of timelines $\{\mathcal{T}_i\}$ and a set of constraints $i(\tau)$ (see Section 2.2), we can introduce a mapping g transforming a CBI plan P into a flexible situation $\langle sc, \tau, i \rangle$. The following holds.

Proposition 1 *Given a CBI planning problem $(\mathcal{M}_{cbi}, P_c)$, where P_c is a candidate plan [10], for any complete plan P of $(\mathcal{M}_{cbi}, P_c)$, maximally flexible in the time variables τ, there exists a $\mathcal{D}_T$, a Golog program $prog_c$, where*

$$\mathcal{D}_T \cup \mathbb{I}(\tau, sc) \models Do(prog_c, ini, sc[\tau], h).$$

with $\langle sc, \tau, i \rangle$ a flexible situation, and such that:

$$g(P) = \langle sc, \tau, i \rangle$$

The proof is omitted.

Example. We assume an autonomous rover, which has to explore the environment and perform observations. While the robot is moving the pant-tilt unit must be pointing ahead $ptIdle$. To point to a location x the rover must be stopped there, $at(x)$, while the pan-tilt scans in direction z: $ptScan(x, z)$. Hence, we consider two components: *pant-tilt*, *navigation*. Each component has a set of processes: J_{pt}=$\{ptIdle,\ ptScan(z, x)\}$; J_{nav}=$\{at(x), go(x, y)\}$. Each of these is to be encoded in the $\mathcal{D}_T$ as in (5), e.g. $ptScan(t, z, x, do(a, s)) \equiv a = s_{pts}(z, x, t) \vee ptScan(t, z, x, s) \wedge \neg\exists t'.a = e_{pts}(z, x, t')$. The *temporal compatibilities* T_c among the activities are defined by a set of temporal constraints patterns, defined by macros (7), e.g.:

$$\begin{array}{l} go(x, y, t_1, t_2)\ \mathbf{m}\ at(y, t_3, t_4);\ at(x, t_1, t_2)\ \mathbf{m}\ go(x, y, t_3, t_4); \\ go(x, y, t_1, t_2)\ \mathbf{d}\ ptScan(z, x, t_3, t_4); \\ ptScan(z, x, t_1, t_2)\ \mathbf{d}\ at(x, t_3, t_4). \end{array}$$

Consider a partial specification for the rover behavior: from time 0 to 3 it must remain $at(p_1)$, and at time 0 the pant-tilt is $ptIdle$; the rover mission is to observe p_3, in direction θ before 30 and after 20, and be back at p_1 before 50. This *candidate plan*, can be encoded in the following Golog program:

$$\begin{array}{l} \mathbf{proc}(mission, \\ \quad \psi^{nav}_{at}(p_1, 0, 3)? \prec (\psi^{nav}_{at}(p_1, t^1_1, t^1_2) \wedge t^1_2 \leq 50)? : Nil \| \\ \quad \psi^{pt}_{pti}(0, t^2_1)? \prec (\psi^{pt}_{pts}(\theta, p_3, t^2_2, t^3_3) \wedge [t^2_2, t^2_3] \subseteq [20, 30])? \prec Nil). \end{array}$$

Given the horizon $h = 50$, and an initial situation class ini, a possible complete plan is a flexible situation $\langle sc, \tau, i\rangle$, with $sc[\tau]$=$\{\sigma_{nav}[\tau_1], \sigma_{pt}[\tau_2]\}$, s.t. $D_T \cup \mathbb{I}(\tau, sc) \models Do(mission, ini, sc[\tau], 50)$, e.g.

$$\begin{array}{l} \sigma_{nav}{=}do([s_{at}(p_1, 0), e_{at}(p_1, 3), s_{go}(p_3, t^1_1), e_{go}(p_3, t^1_2), \\ \qquad\qquad s_{at}(p_3, t^1_3), e_{at}(p_3, t^1_4), s_{go}(p_1, t^1_5), e_{go}(p_1, t^1_6)], S_0); \\ \sigma_{pt}{=}do([s_{pti}(0), e_{pti}(t^2_1), s_{pts}(t^2_2), e_{pts}(t^2_3), s_{pti}(t^2_4), e_{pti}(t^2_5), s]S_0), \end{array}$$

and $i(\tau)$ is the minimal set of constraints solving the $h(\mathbb{I}(sc, \tau))$ temporal network, i.e. $\{t^1_2 = t^1_3 < t^1_4 = t^1_5 < t^1_6 < 50; t^2_1 = t^2_2 < t^2_3 = t^2_4 < t^2_5; t^2_2 > t^1_3, t^2_3 < t^1_4\}$.

Implementation. We provided a constraint logic programming (CLP) [9] implementation of the Golog interpreter. Since the Golog interpreter is to generate flexible temporal plans, it must be endowed with a constraint problem solver. Analogous to [17] we rely on a logic programming language with a built-in solver for linear constraints over the reals (CLP($\mathcal{R}$)). We appeal to the ECRC Common Logic Programming System ECLIPSE 5.7. Currently, we are deploying the Golog interpreter as the control engine of a monitoring systems for robotics applications.

References

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

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

[3] Giuseppe de Giacomo, Yves Lesperance, and Hector J. Levesque. Congolog, a concurrent programming language based on the situation calculus. *Artif. Intell.*, 121(1-2):109–169, 2000.

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

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

[6] Alfredo Gabaldon. Programming hierarchical task networks in the situation calculus. In *AIPS'02*, 2002.

[7] Malik Ghallab and Herv Laruelle. Representation and control in ixtet, a temporal planner. In *AIPS 1994*, pages 61–67.

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

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

[10] 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.

[11] J. Kvarnstrom, P. Doherty, and P. Haslum. Extending talplanner with concurrency and resources. In *Proc. ECAI-00*, pages 501–505, 2000.

[12] 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.

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

[14] Nicola Muscettola, P. Pandurang Nayak, Barney Pell, and Brian C. Williams. Remote agent: To boldly go where no AI system has gone before. *Artificial Intelligence*, 103(1-2):5–47, 1998.

[15] 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.

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

[17] Fiora Pirri and Raymond Reiter. Planning with natural actions in the situation calculus. pages 213–231, 2000.

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

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

[20] Tran Cao Son, Chitta Baral, and Le-Chi Tuan. Adding time and intervals to procedural and hierarchical control specifications. In *AAAI 2004*, pages 92–97, 2004.

[21] Austin Tate. I-n-ova and i-n-ca - representing plans and other synthesised artifacts as a set of constraints.

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

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