Representing Flexible Temporal Behaviors in the Situation Calculus

Alberto Finzi and Fiora Pirri
DIS- Università di Roma “La Sapienza”

Abstract

In this paper we present an approach to flexible behaviors implemented 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. We provide a version of the Golog interpreter suitable for managing flexible plans on multiple timelines.

1 Introduction

A core problem for the executive control of robotic behaviors is to correctly manage tasks alternation. In real-world domains robots have to perform multiple tasks either simultaneously or in rapid succession, by using diverse sensing and actuating tools, like motion, navigation, visual exploration, mapping, and several layers 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], NOVA [21], and RMPL [23]). However, from the standpoint of cognitive robotics it is both important to ensure optimal performance in practical applications, but also to provide a logical framework so as to manage the preconditions of actions, and to guarantee the coherence of actions effects. Coherence requires also that control processes, interacting with basic perceptual-motor process and with cognitive processes, establish priorities among individual processes, in order to allocate resources to them during 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 aspects that reason 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. Indeed, on the one hand, it is possible to introduce a separated timeline for each component of the dynamic system (each entity which is part of any autonomous system, e.g. a robot), so that concurrency and flexibility can be clearly addressed; at the same time 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 \( \text{BAT} = (\Sigma, \mathcal{D}_{\sigma_0}, \mathcal{D}_{\sigma_{sa}}, \mathcal{D}_{\sigma_{ana}}, \mathcal{D}_{\sigma_{ap}}) \). 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 a function \textit{time}. Durative actions are considered as processes \cite{15:19}, represented by fluents, and durationless actions are to start and terminate these processes. For example, \text{going}\,(\text{hill, } s) is started by the action \text{startGo}\,(\text{hill, } t) and it is ended by \text{endGo}\,(\text{hill, } t')

\textbf{Golog.} Golog is a situation calculus-based programming language for denoting complex actions composed of the primitive (simple or concurrent) actions defined in the B\textit{AT}. Golog programs are defined by means of standard (and not so-standard) Algol-like control constructs: i. action sequence: \text{p}_1|\text{p}_2, ii. test: \phi?, iii. nondeterministic action choice \text{p}_1|\text{p}_2, iv. conditionals, while loops, and procedure calls. An example of a Golog program is:

\begin{verbatim}
while \neg at(hill,3)
   if \neg(\exists x going(x)) do \pi(t, (t < 3)?; startGo(hill, t))
\end{verbatim}

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

\subsection{Constraint Based Interval Paradigm}

The constraint based interval framework (CBI) \cite{10:4}, is a well known framework for temporal planning systems combining temporal reasoning and scheduling, including, e.g., RAX \cite{10}, IxTeT \cite{7}, and INOVA\cite{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 \cite{10} where a domain behavior is seen as the continuous interaction of different components, and each component is represented by \textit{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 \textit{timeline}. Figure 1 illustrates two timelines (state variables) repr. the engine and the navigation processes of a mobile robot: Initially, the robot is \text{at}(p_1) and \text{stop}; when it starts \text{go}(p_1,p_2) the engine is \text{running}; the engine is \text{stop} again once the rover arrives \text{at}(p_2). Some temporal constraints among the activities are: \text{at}(x) holds only if \text{stop} holds, \text{go}(x, y) is followed by \text{at}(y).

\textbf{Domain Constraints.} A \textit{CBI model} \cite{10} \(\mathcal{M}_{\text{cbi}}=(X, I, \mathcal{R})\) is defined by: i. a set \(X=\{x_1, \ldots, x_n\}\) of state variables, one for each components (e.g. \(x_{\text{loc}}\) and \(x_{\text{eng}}\) in Fig. 1); ii. a set \(I=\{I_1, \ldots, I_n\}\) s.t. for each \(x_i\) the set \(I_i\) collects the associated temporal fluents \(p_{i,j}(\vec{y})\), e.g. \(\text{at}(x)\) and \(\text{go}(x, y)\) for location state variable in Fig. 1; iii. a set of temporal constraints \(\mathcal{R}=\{r_{i,j}\}\), usually called \textit{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 \text{a la Allen} \cite{11}. E.g. the arrows in Fig. 1 illustrate the compatibility associated to the fluent \text{at}: \text{at}(x) \text{meets} \text{go}(x, y), and \text{at}(x) \text{during} \text{stop}.

\textbf{Planning Problem in CBI.} Given the \textit{CBI model} \(\mathcal{M}_{\text{cbi}},\) a \textit{planning problem} is defined by \(\mathcal{P}_{\text{cbi}}=(\mathcal{M}_{\text{cbi}}, P_c)\), where \(P_c\) is a \textit{candidate plan}, representing an incomplete instance of a plan. The \textit{candidate plan} defines both the initial situation and the goals. In particular, \(P_c\) consists of : i. a \textit{planning horizon} \(h\); ii. a set of temporal properties to be satisfied for each state variable \(x_i \in X\) (e.g. \text{at}(p_1) ends before 10 and after 20 in the timeline of Figure 1); iii. the set of precedence constraints among fluents \(p_{k_1} < p_{k_2}\) which are to hold in a timeline, e.g. \(\text{at}(p_1) < \text{at}(p_2)\); 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 \textit{candidate plan} is \textit{fully supported}[10] if all its associated constraints \(r_{i,j}\) are satisfied. E.g. in Figure 1 \text{at}(p_1) and \text{at}(p_2) are fully supported. A candidate plan is said to be a \textit{complete plan} if: i. each temporal property on each timeline is fully supported; ii. all timelines fully cover the planning horizon. Given the \textit{planning problem} \((\mathcal{M}_{\text{cbi}}, P_c)\), the planning task is to provide a \textit{sufficient plan}[10], i.e. a \textit{complete plan} with the maximum flexibility: the planner is to minimally ground the (temporal) variables to allow for on-line binding of the values.

\section{Temporal Flexible Situation Calculus}

In this section we present the Temporal Flexible Situation Calculus framework (IFSC) which integrates the CBI paradigm introduced above.

\textbf{Actions.} We define a partition \(\{a_v; v \in C\}\) of the set of actions according to the different components \(C\) the system has to care of. Typization of actions induces also a partition on the set of situations hence on the set of histories. Histories become streams of situations over timelines. To this end we introduce a type operator \(\nu\). The SC foundational axioms are extended with the following definitions. Let \(H = \bigcup^n_{i=1} H_i\) be a set specifying the types of actions, we assume this set finite. For each name of actions the following holds:

\begin{align*}
H_1(a) \wedge H_3(a') \rightarrow \neg (a = a')\quad \text{for } i \neq j, \quad \text{and } \nu(A(\vec{x}))=\nu(A'(\vec{y})), \quad \equiv \bigvee^n_{i=1} H_1(A(\vec{x})) \wedge H_i(A'(\vec{y})), \quad \text{together with the unique name for actions (here the indices stay only for different names), were } a \text{ denotes an action variable while } A \text{ denotes the name of an action function.}
\end{align*}

\textbf{Situations.} Typization is inherited by situations as follows:

\begin{align} \forall a. \quad & \nu(a) = \nu(\text{do}(a, S_0)). \\ \forall a s. \quad & \nu(a) = \nu(s) \equiv \\ & \forall a s'. s = do(a', s') \rightarrow \nu(a) = \nu(a') \land \nu(a') = \nu(s'). \quad (1) \end{align}
For each component $v \in C$, a possible timeline $T_v$ is represented by a situation $\sigma_v = do(a_1, \ldots, a_i, S_0)$, with $\nu(\sigma_v) = \nu(a_i), i = 1, \ldots, k$. Evolution of the set of timelines requires the following notation for set of situations: $sc = \{ s_j | s_j \in C \}$, called situation class, spanning over different types. In other words, $\exists \exists s \in sc \land Q \equiv \exists s \ldots s_n, \bigwedge^{n-1}_{i=1} \alpha \lor \nu(s_i) \neq \nu(s_j) \lor Q$. Where $\nu(s_i)$ is the type of situation $s_i$ corresponding to some component $i \in C, \alpha = \mid C \mid$. Analogously for $\forall s, s \in sc \rightarrow Q$. A set of possible timelines is modeled by a set of situations $sc$, equipped with the following $\leq_s$ relation:

$$do(a_i, sc) \leq_s s \equiv \exists s' \in s \land s' \rightarrow sc \rightarrow s' \leq s.$$  

**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:

$$time(S_0) = t_0, time(do(a, s)) = time(a).$$

$$time(s) \leq time(s') \equiv \exists a \in s' \land s' \rightarrow s = do(a, s') \land s = do(a', s) \land s' \rightarrow s \land time(a) \leq time(a').$$

We can thus define the time of a situation class as:

$$time(sc) = t$$

which is a shortcut for $\exists s' \in s \land s' \rightarrow s \land time(s) = t \land time(s') = t' \Rightarrow t' \geq t$. We shall use $\mathcal{t}$ as a subscript, i.e. $\mathcal{t}_{k}$, to denote that $t$ refers to the timeline of component $k$.

**Consistency of the extension** The above axioms concerning types, time and situation classes, are added to the foundational axioms in $\Sigma$ and are conservative (i.e. obtained by extending the language), therefore the extended set $\Sigma'$ is still 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 $sp$, meaning starts process $P$ and $ep$, meaning ends process $P$. A process is defined by a fluent $P(x, t)$, (here $t$ is its start time). Successor state axioms (SSP) for processes extend the set of $SSA$ for fluents and are defined as follows:

$$P(x, t^-), do(a(s)) \equiv a = sp(x, t^-) \lor P(x, t^-), a \land \forall t, a \neq ep(x, t).$$

For example, moving towards $\theta$ can be defined as:

$$move(\theta, t^-), do(a(s)) \equiv a = move(h, \theta, t^-) \lor move(\theta, t^-), s \land \forall t, a \neq ep(\theta, t).$$

Action preconditions axioms define the conditions under which the start and end actions can 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 $D_{rsp}$ of successor state axioms for processes is *linear* if all processes mentioned in it are linear. Let $D_{SSP}$ be a set of linear successor state axioms for processes, extending the $BAT$ $D = \Sigma \cup D_{ssp} \cup D_{una} \cup D_{ss} \cup D_{AP}$ then:

**Lemma 1** $D \cup D_{SSP}$ is consistent iff $D_{ssp}$ is.

**Proof (sketch).** Just note that since the (SSA) for processes mentions neither fluents nor other processes depending only on the start and end actions, in the same situation $s$ it is not possible that $sp(x, t) < sQ(y, t')$ and $sp(x, t) > sQ(y, t')$; the same argument can be used for end actions.

**Intervals.** Intervals are very well suited for representing *flexible behaviors*, as they mention temporal variables which are always free, and get assigned only according to a specified set of behaviors; this fact imposes a bookkeeping of time variables for consistency, as will be explained better in the next paragraph. In this paragraph we show how embedding intervals in the $SC$ leads to the construction of a temporal network. All the conditions on actions and situations concerning the processes involved with time, can be gathered into the following formula $\Psi$, simply specifying that the process $P_t$ holds on the associated timelines from $t_1$ to $t_1^+$ with arg. $\tau$:

$$\Psi(\tau, t_1^+, t_1^+, sc) \equiv \exists s, s_1, s_1' \in sc \land s_1 \equiv do(P, \tau, t_1^+, s_1') \land (\forall s, s_1 \in sc \land s_1 \equiv do(P, \tau, t_1^+, s_1')).$$

Note that the above formulas $\Psi$ have explicit free variables $\tau, t_1^+, t_1^+, sc$ (and one implicit variable, namely $i$, denoting the type, gathering all the terms with $\nu$, that are not mentioned). We shall not repeat such explicit free variables in the definition below. The macro-definition for interval relations are now straightforward as follows, where $p, m, o, d, f, e$ are as usual *precedes, meets, overlaps, during, starts, finishes, equals* with $op$ ranging over all relations:

$$P_t(\tau, t_1^+, t_1^+) \ op P_t(\tau, t_1^+, t_1^+) \equiv \Psi_{\tau} \land \neg \Psi_{\tau}.$$
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 \( I \) between them (e.g. \( p, e \) etc.). According to the underlying temporal algebra, TCN 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 \( X = \{X_1, \ldots, X_n\} = \{\langle t_1^-, t_1^+ \rangle, \ldots, \langle t_n^-, t_n^+ \rangle\} \) be a set of time variables, denoting intervals (i.e. each variable \( X_i \) denotes an interval \( \langle t_i^-, t_i^+ \rangle \)), and let \( I \) be a set of binary relations; the assignment set \( V = \{s_1, e_1\}, \ldots, \{s_n, e_n\} : s_i, e_i \in \mathbb{R}, s_i < e_i \} \) is called a solution set for \( I \), if \( X_1 = \{s_1, e_1\}, \ldots, X_n = \{s_n, e_n\} \) satisfies all the constraints defining \( I \). 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.

Temporal Constraint Network Let \( h : I(\tau) \to TCN \), for any set of constraints \( I(\tau), h(I(\tau)) \) is a temporal network \( \mathbb{N} \), with cardinality \( m \), specified by the nodes labeled by the defined processes \( \{P_1, P_2, P_3, P_4\} \) are mentioned in \( I(\tau) \).

By the above definitions the network \( \mathbb{N} \) is consistent if there exists an assignment \( V \) to the temporal variables, which is a feasible solution, i.e. satisfying all the constraints. For example the set \( V = \{10, 160\}, \{10, 70\}, \{75, 95\}, \{95, 150\} \} \) is a solution set for the network \( X = \{\langle t_1^-, t_1^+ \rangle, \langle t_2^-, t_2^+ \rangle, \langle t_3^-, t_3^+ \rangle, \langle t_4^-, t_4^+ \rangle\} \), of the network depicted in Figure 2, given the relations \( \{m, p, d, s, f\} \).

Note that, given the macro definition (6), while the definiendum \( P_i(\tau, t_i^+, t_i^-) \) or \( P_j(\tau, t_j^+, t_j^-) \) is mapped to a temporal network, the definiens \( \Psi_i \mapsto \Psi_j \land \gamma_k \) is interpreted into a structure of the \( SC \), where the assignments for the free temporal variables are obtained by the TCN. We denote with \( V \) the set of definiens \( \Psi_i \mapsto \Psi_j \land \gamma_k \).

A semantic correspondence can be established as follows. Let \( M_{sc} = \langle D, I, h \rangle \) be a structure for the \( SC \)-language, extending \( M = \langle D, I \rangle \) (where \( M \) is a structure for \( SC \), with \( D \) a sort domain and \( I \) an interpretation), with \( h \) a mapping as defined above, then \( M_{sc} \) is a model for \( D \cup I(\tau) \), if there is a consistent temporal network \( h(I(\tau)) \), under some assignment \( V \), satisfying \( I(\tau) \) and \( M, V \models \bigwedge D \land \bigwedge \Psi \).

Given the above definitions we can state the following:

**Lemma 2** \( D_T \cup I(\tau) \) is consistent if there exists a consistent temporal constraint network \( h(I(\tau)) \), under the assignment \( V \), such that \( M, V \models \bigwedge D \land \bigwedge \Psi \).

**Proof (sketch).** Let \( h(I(\tau)) \) be a consistent temporal network, under the assignment \( V \). Consider the set \( \Psi = \{\Psi_i \to \Psi_j \land \gamma_k\} \), and we let \( M \) be a model for \( D_T \). \( M \) can be extended to a model for all the \( \gamma_i \) according to \( V \). As follows. For each process \( P_i \) build a chain of situations of the kind \( \Gamma_i = \{\langle \sigma_P(t'_i), \sigma'\rangle \leq \sigma \leq \langle \sigma_P(t'_i), \sigma''\rangle \leq \sigma_1\} \), with \( \sigma \) a sequence such that \( \sigma_P \) is not mentioned in \( \sigma \). Extend the structure \( M \) to one for \( D_T \cup \bigcup \Gamma_i \) by choosing from \( V \) a suitable assignment to the free variables in each \( \Gamma_i \), and according to the constraints in the \( \gamma_i \), corresponding to the constraints in \( I(\tau) \), 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 \( (M, V) \) is a model for \( D_T \cup \bigcup \Gamma_i \). Finally it is enough to show that any model for \( D_T \cup \bigcup \Gamma_i \), is a model for \( \Psi = \{\Psi_i \to \Psi_j \land \gamma_k\} \).

Note that by Lemma 1 \( D_T \cup I(\tau) \) is consistent iff \( D_{sc}, V \models I(\tau) \) is consistent. Logical implication can be now defined as follows, note that time variables are quantified when they are mentioned in the successor state axioms for processes. Let \( M_{sc} = \langle D, I, h \rangle \), and \( \beta(\tau) \) be a formula with all its free temporal variables among \( \tau \):

\[ D_T \cup I(\tau) \models \beta(\tau) \text{ iff for any } h, \text{ for any } V, \quad M_{sc}, V \models D_T \cup I(\tau) \implies M_{sc}, V \models \beta(\tau). \quad (7) \]

**TCN construction.** Two things are now in order: the inference mechanism constructing a set \( I_{sc}(\tau) \) capturing the temporal network topology associated to a situation class \( sc(\tau) \), the mechanism implementing instantiation of situations. We can explain this with an example. Let \( \overline{s(\tau)} \) denote situation class \( sc \) whose only free variables are among \( \tau \), for example \( \overline{s(\tau)} = do(s_P(t_1^+, t_2^-), e_P(t_j^-)) \), of the network depicted in Figure 2, given the relations \( \{m, p, d, s, f\} \).

The three nodes network that would be constructed as a consequence of the compatibilities together with the current situation is

\[ I_{sc}(t_1^+, t_1^-, t_2^-, t_2^+, t_3^-, t_3^+, t_4^-, t_4^+) = \{P_1[t_1^+, t_1^-], P_2[t_2^-, t_2^+], P_3[t_3^-, t_3^+], P_4[t_4^-, t_4^+]\} \]

Therefore a situation class \( sc \), through \( I_{sc}(\tau) \) fully specifies the interval relations needed to build a temporal network. The time range of the solutions to the network—given a situation class \( sc \)—is the flexible situation class, which is a set of pairs of the kind \( \langle \overline{s(\tau)}, i(\tau) \rangle \), where \( i(\tau) \) is the constraint over \( \tau \) variables, e.g. \( i(\tau) = 10 \leq t_1 \land t_2 > 7 \).

**Progression.** Let \( \kappa = \langle s_{sc}, \tau \rangle \) denote a set of ground situations (for example the histories along specified timelines) and a set of (ground) time points along these situations, and let \( \phi \) be a sentence mentioning this set \( \kappa \). We want to determine if, according to the timing and advancements of the current state
of affair we can forget about the past and think of the future. This is the well known progression problem [19] in the SC. We face here a simplified version of the progression problem, and with regressive formulas, which is all we need. Let us state the problem as follows. Let \( \varphi \) be a sentence, order the situation terms in \( \varphi \) according to the relation \( \leq \) introduced in (2). Consider the set of smallest situations (the ordering is partial) with \( \text{min} = \langle \sigma, \tau \rangle \), and the smallest time point \( i.e. \) with respect to some \( t \), mentioned in \( \text{min} \). We now consider the domain theory \( \mathcal{D}_T \) which is equal to \( \mathcal{D}_T \) but with \( \mathcal{D}_T \) replaced by a set \( W^* = \{ \varphi \} | \mathcal{D}_T | = \varphi \) where \( \varphi \) mentions only ground situations \( \sigma \), with \( \{ \sigma_i \} = \text{min} \) and no free variables of sort time \( \tau \). Let \( \Phi(t_1, \ldots, t_n) \) be a SC sentence mentioning only ground situations \( \{ \sigma_i \} \), \( \geq \text{min} \), and possibly free variable of sort time all among \( t_1, \ldots, t_n \). Then

**Lemma 3** \( \mathcal{D}_T \cup \mathcal{I}(\tau) = \bigtriangleup (\mathcal{D}_T \cup \mathcal{I}(\tau) \cup W^*) \).

**Theorem 1** \( \mathcal{D}_T \cup \mathcal{I}(\tau) = \Phi(t_1, \ldots, t_n) \) iff \( \mathcal{D}_T \cup W^* \cup \mathcal{I} = \Phi(t_1, \ldots, t_n) \).

**Proof.** (sketch) One direction follows by cut. For the other direction, suppose \( \mathcal{D}_T \cup \mathcal{I}(\tau) = \Phi(t_1, \ldots, t_n) \), let us regress \( \Phi(t_1, \ldots, t_n) \) along the different timelines, this can be done by separating \( \Phi \) into a DNF, and considering each conjunct separately, we regress it with respect to \( \text{min} \), and let \( G \) be the regressed formula – note that there need not be change in regression to account for regressed sentences as time variable will be kept as they are, without turning to quantified variables through successor state axioms. Then by definition of \( W^* \), \( G \in W^* \) 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:** \( \alpha \).
2. **Nondeterministic choice:** \( \alpha | \beta \). Do \( \alpha \) or \( \beta \).
3. **Test action:** \( ? \). Test if \( \phi \) is true in the current sit. class.
4. **Nondet. arg. choice:** \( \exists \) for \( \text{prog} \).
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:** \( p_1 \parallel p_2 \). Do \( p_1 \) concurrently with \( p_2 \).
8. **Conditionals:** if \( \phi \) then \( p_1 \) else \( p_2 \).
9. **While loops:** while \( \phi \) do \( p_1 \).
10. **Procedures, including recursion.**

**Golog Semantics.** The macro \( \text{Do}(p, s, s', h) \) gives the semantics for the above constructs; where \( p \) is a program, \( s \) and \( s' \) are situation classes, and \( h \) specifies the finite horizon.

- **Null program:**
  \( \text{Do}(\text{Nil}, s, s', h) \triangleq \text{time}(s) \leq h \land [s \in s \equiv s \in s'] \)

- **Primitive first program action with horizon:**
  \[ \text{Do}(\alpha; \text{prog}, s, s', h) \triangleq s_i \in s \land \text{Poss}(\alpha, s_i) \land \text{time}(s_i) \leq h \land [\text{time}(\alpha) \leq h \land \text{Do}(\text{prog}, \alpha, s, s', h) \lor \text{time}(\alpha) > h \land \text{Do}(\text{prog}, s, s', h)] \]

Here, if the first program action is applicable to \( s_i \in s \), and \( \alpha_i \) 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:**
  \( \text{Do}(\text{prog}_1 \prec \text{prog}_2, s, s', h) \triangleq \text{Do}(\text{prog}_1; \text{prog}_2, s, s', h) \lor \exists a. \text{select}(a, s) \land \text{Do}(\text{prog}_1, a \prec \text{prog}_2, s, s', h) \)

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 \( \text{select}(a, s) \) representing the selection criterion (set to true if no selection holds).

- **Parallel execution:**
  \[ \text{Do}(\text{prog}_1 \parallel \text{prog}_2, s, s', h) \triangleq \text{Do}(\text{prog}_1, s, s', h) \land \text{Do}(\text{prog}_2, s, s', h) \]

- **Test action:**
  \[ \text{Do}(\phi; \text{prog}, s, s', h) \triangleq \phi[s] \land \text{Do}(\text{prog}, s, s', h) \]

Here \( \phi[s] \) stands for generalization of the standard \( \phi[s] \) in the SC extended to situation classes, e.g. \( P_1[s] \equiv P_2[s] \) is for \( P_1(s_1) \land P_2(s_2) \) with \( s_1, s_2 \in s \).

- 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 \((\mathcal{M}_c, \mathcal{P}_c)\) introduced in Section 2.2 can be easily coded and solved in the TSCC framework. Given a \( D_T \) representing the timelines and processes in \( \mathcal{M}_c \), a candidate plan \( P_c \) can be encoded by a Golog program \( \text{prog}_c \). This is possible once we introduce, for each interval constraint for \( p_{i,j} \) in \( P_c \), a Golog procedure of the kind:

\[ \text{proc}(\pi_{i,j}, \langle \psi_{i,j}(\tau), e, t^+ \rangle \land \gamma((t^-, t^+))) \]

where \( \psi_{i,j} \) is the macro (5) associated with the process \( P_{i,j} \), and \( \gamma \) any temporal constraint. For example, \( \text{go}(d, e) \) ends in \([6, 10] \) can be represented as

\[ \text{proc}(\pi_1, \langle \psi_{go}(d, e, t^-, t^+) \rangle \land 6 \leq t^+ \leq 10 \rangle) \]

Given the \( \pi_{i,j} \) procedures, a partial specification of a single timeline \( T_i \) can be easily defined using the \( \prec \) operator:

\[ \text{proc}(\text{plan} \cdot T_j, \pi_{0,j} \prec \pi_{1,j} \prec \pi_{2,j} \prec \cdots \prec \pi_{k,j}) \]

Given a set of timelines \( \{ T_i \} \), a candidate plan \( P_c \) can be represented as a parallel execution of the \( \text{plan} \cdot T_i \):

\[ \text{proc}(\text{prog}_c, \text{plan} \cdot T_i : \text{Nil} \parallel \cdots \parallel \text{plan} \cdot T_k : \text{Nil}) \]

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

**Proposition 1** Given a CBI planning problem \((\mathcal{M}_c, \mathcal{P}_c)\), where \( P_c \) is a candidate plan \([10]\), for any complete plan \( P \) of \((\mathcal{M}_c, \mathcal{P}_c) \), maximally flexible in the time variables \( \tau \), there exists a \( D_T \), a Golog program \( \text{prog}_c \), where

\[ D_T \cup \mathcal{I}(\tau) \models \text{Do}(\text{prog}_c, \text{ini}, \text{sc}(\tau), h) \]

with \( \langle \text{sc}, i(\tau) \rangle \) a flexible situation, and such that:

\[ g(P) = \langle \text{sc}, i(\tau) \rangle \]

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: $I_{pt} = \{ptIdle, ptScan(x, z)\}$; $I_{nav} = \{at(x), go(x, y)\}$. Each of these is to be encoded in the $D_T$ as in (4), e.g. $ptScan(t, x, x, do(a, s)) \equiv a = p_{pt}(x, x, t) \lor \forall ptScan(t, x, x, s) \land \exists t'.a = p_{pt}(x, x, t')$. The temporal compatibilities $T_C$ among the activities are defined by a set of temporal constraints patterns, defined by macros (6), e.g.: 

$$
go(x, y, t_1, t_2) \quad m \quad at(x, t_1, t_2); \quad at(x, t_1, t_2) \quad m \quad go(x, y, t_1, t_2); \quad go(x, y, t_1, t_2) \quad d \quad ptScan(x, x, t_1, t_2); \quad ptScan(x, x, t_1, t_2) \quad d \quad at(x, t_1, t_2).$$

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 $\theta$ before 30 and after 20, and be back at $p_1$ before 50. This candidate plan can be encoded in the following Golog program:

$$\text{proc}(\text{mission},) \quad \psi_{at}(p_1, 0, 3) \land (\psi_{pt}(p_1, 0, 3) \land \psi_{at}(p_3, 0, 3, \theta) \land \psi_{pt}(p_3, 0, 3, \theta) \land \psi_{at}(p_1, 0, 30, \theta) \land \psi_{pt}(p_1, 0, 30, \theta) \land \psi_{at}(p_1, 0, 0, \theta) \land \psi_{pt}(p_1, 0, 0, \theta) \land \psi_{at}(p_1, 0, 50, \theta) \land \psi_{pt}(p_1, 0, 50, \theta) ) \land Nil).$$

Given the horizon $h = 50$, and an initial situation class $ini$, a possible complete plan is a flexible situation $(\sigma\mu[\tau 1], i(\tau))$, with $\sigma\mu[\tau 1] = \{\sigma_{nav}[\tau 1], \sigma_{pt}[\tau 2]\}$, s.t. $D_T \cup I_{\sigma\mu}(\tau) = Do(\text{mission}, ini, \sigma\mu[\tau 1], 50)$. e.g. 

$$\sigma_{nav} = do[\psi_{at}(p_1, 0, 3), \psi_{pt}(p_1, 3, 3), \psi_{at}(p_3, t_1, t_2), \psi_{pt}(p_3, t_1, t_2), \psi_{at}(p_1, t_3, t_4), \psi_{pt}(p_1, t_3, t_4), \psi_{at}(p_1, t_5, t_6), \psi_{pt}(p_1, t_5, t_6)]$$

and $i(\tau)$ is the minimal set of constraints solving the $h(\sigma\mu[\tau 1])$ temporal network, i.e. $t_1 = t_2 < t_3 < t_4 < t_5 < t_6 < 50$; $t_7 = t_8 < t_9 < t_10 < t_11 \quad \{t_1 = t_2 < t_3 < t_4 < t_5 < t_6 < 50, t_7 = t_8 < t_9 < t_10 < t_11 \}$. 

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/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


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