# PLAN2TIGA User Guide

M. Cialdea Mayer[1] and A. Orlandini[2]

[1]Università degli Studi di Roma Tre
[2]Istituto di Scienze e Tecnologie della Cognizione CNR Roma

## 1 Introduction

The program PLAN2TIGA implements the encoding of flexible plans into networks of timed game automata (TGA), according to the syntax accepted by UPPAAL-TIGA, that can therefore be called on the files generated by PLAN2TIGA in order to check whether the corresponding plan is dynamically controllable and, if so, generate a strategy for robust plan execution.

A flexible plan $\Pi$ is a pair $\langle \mathbf{FTL}, \mathcal{R} \rangle$, where **FTL** is a set of flexible timelines (some of which may represent the evolution of *external state variables*, modeling components of the system that are completely outside the control of the executive), and $\mathcal{R}$ is a set of relations between tokens of the timelines in **FTL**. The related concepts are introduced in [1], and extended in order to treat uncertainty. Tokens are in fact annotated by a *controllability tag*: the exact duration of a controllable token can be decided by the controller (obviously respecting the bounds on its end point and duration), while the end time of uncontrollable tokens is under the control of the *environment*. An external timeline is made up of uncontrollable tokens only, while *planned timelines* may contain both controllable and uncontrollable tokens.

The encoding allows also the modeling of "partial plans", where timelines may have undefined temporal slots, i.e. tokens without value and possibly with no associated duration (called *unallocated tokens*). This allows the planner to interact with the UPPAAL-TIGA verifier during plan construction, and possibly abandon routes that would lead to plans that cannot be executed.

The details of the encoding are described in Section 4, where arguments are also given to prove its correctness.

Let $\Pi = \langle \mathbf{FTL}, \mathcal{R} \rangle$ be a flexible plan and $\mathcal{N}$ the network of automata (an UPPAAL-TIGA *system*) modeling $\Pi$. In the encoding generated by PLAN2TIGA, each automaton $\mathcal{A}_x$ (an UPPAAL-TIGA *process*) in $\mathcal{N}$ models a timeline $FTL_x \in \mathbf{FTL}$, and every token of $FTL_x$ is modeled by a location (an UPPAAL-TIGA *state*) in the corresponding automaton $\mathcal{A}_x$. Moreover, $\mathcal{A}_x$ has a *final state*, called *finish*, that is reached at the end of the timeline. The UPPAAL-TIGA winning condition is a *pure reachability* one, expressed by an UPPAAL-TIGA formula of the form

$$\mathsf{A}\Diamond(\mathcal{A}_{x_1}.finish \wedge \cdots \wedge \mathcal{A}_{x_k}.finish)$$

where $\mathcal{N} = \{A_{x_1}, \ldots, \mathcal{A}_{x_k}\}$: whichever the behaviour of uncontrollable components, the automata in $\mathcal{N}$ can reach their final states (at the same time).

## 2 How to call PLAN2TIGA

PLAN2TIGA takes in input a file describing a flexible plan and encodes it into the language of UPPAAL-TIGA. The program is called with the syntax

```
plan2tiga [options] <filename>
```

The syntax of the file given in input is described in Section 3. The program outputs an UPPAAL-TIGA file defining a system and the related query file, named, by default, `<filename>.xta` and `<filename>.q`.

The possible options are:

**-debug** When called with this option, the query file generated by PLAN2TIGA corresponds to the formula:

$$\mathsf{A}\Diamond\mathcal{A}_{x_1}.finish \wedge \cdots \wedge \mathsf{A}\Diamond\mathcal{A}_{x_k}.finish$$

This allows, when the plan results not to be dynamically controllable, to single out the timelines that may not reach their final state.

**-v** (verbose mode): the program prints a list of the primitive relations into which the definde ones are translated (see Section 3).

## 3 Syntax of the input file

Comments are included between `/*` and `*/`, or from `#` to carriage return. Keywords are not case sensitive.

Below, `<int>` is a sequence of digits, and `<name>` a string (different from keywords) made up of alphanumeric characters and the underscore, and beginning with an alphabetic character.

```
           <file> ::= "horizon" "=" <int>
                      <plan>
                      <observation>
           <plan> ::= "plan" "{"
                         <timeline list> /* planned timelines */
                         <relation list>
                       "}"
    <observation> ::= "observation" "{"
                         <timeline list> /* external timelines */
                       "}"
  <timeline list> ::= "timelines" "{"
                          <timeline> <timeline>*
                      "}"
       <timeline> ::= <name> /* state variable name */
                        "{"
                          <token> <token>*
                        "}"
          <token> ::= "token" <int> /* token identifier */
                         <tag> /* controllability tag */
                      "{"
                         <name> /* value */
```

```
                        <interval> /* end time */
                        <interval> /* duration */
                     "}"
                   | "unallocated" <int>
                     "{"
                        <interval> /* end time */
                        <interval> /* duration */
                     "}"
                   | "unallocated" <int>
                     "{"
                        <interval> /* end time */
                     "}"

          <tag> ::= "controllable" | "uncontrollable"
     <interval> ::= "[" <int> "," <int> "]"
<relation list> ::= "" /* no constraints */
                  | "relations" "{"
                        <relation> <relation>*
                     "}"
     <relation> ::= <token id> <rel 1> <token id>
                  | <token id> <rel 2> <int>
     <token id> ::= <name> <int>
                      /* name=state variable name */
        <rel 1> ::=   /* primitive relations */
                     "start_before_start" <interval+inf>
                   | "end_before_end" <interval+inf>
                   | "start_before_end" <interval+inf>
                   | "end_before_start" <interval+inf>
                     /* defined relations */
                   | "equals"
                   | "meets"
                   | "met_by"
                   | "before" <interval+inf>
                   | "after" <interval+inf>
                   | "overlaps" <interval+inf> <interval+inf>
                   | "overlapped_by" <interval+inf> <interval+inf>
                   | "contains" <interval+inf> <interval+inf>
                   | "during" <interval+inf> <interval+inf>
                   | "starts" <interval+inf>
                   | "started_by" <interval+inf>
                   | "finishes" <interval+inf>
                   | "finished_by" <interval+inf>
                   | "contains_start" <interval+inf> <interval+inf>
                   | "contains_end" <interval+inf> <interval+inf>
                   | "starts_during" <interval+inf> <interval+inf>
                   | "ends_during" <interval+inf> <interval+inf>
        <rel 2> ::= /* primitive relations */
                     starts_before <interval+inf>
                   | starts_after <interval+inf>
                   | ends_before <interval+inf>
```

```
                    | ends_after <interval+inf>
                      /* defined relations */
                    | "starts_at"
                    | "ends_at"
  <interval+inf> ::= <interval>
                    | "[" <int> "," "infty" "]"
```

The semantics of the relations is given in Section 3.1 and an abstract description of the encoding and its properties is given in Section 4.

The plan has to satisfy the following condition: each integer used as a token identifier must be different from the others in the same timeline.

If the end time of the last token of a timeline is not equal to the plan horizon, PLAN2TIGA adds a last unallocated time slot to the timeline, reaching the horizon.

The description of a sample plan is given in Figure 1.

```
horizon = 350

plan {
   timelines {
       pm { token 1 { earth [10,20] [10,20] }
            unallocated 2 { [110,130] [100,110] }
            token 4 { slewing [140,160] [30,30] }
            token 5 { science [190,200] [40,50] }
            unallocated 6 { [230,250] }
            token 8 uncontrollable
                   { comm [260,300] [30,50] }
            token 9 {earth [350,350] [50,90] } }
### no other planned state variables
    }
    relations { gv 5 contains [0,infty][0,infty] pm 8
                pm 1 starts_at 0
                pm 5 end_before_start [0,50] pm 8 }
     }

observation {
       timelines {
          gv { token 1 { visible [60,100] [60,100] }
               token 2 { not_visible [90,130] [1,100] }
               token 3 { visible [150,190] [60,100] }
               token 4 { not_visible [210,250] [1,100] }
               token 5 { visible [300,320] [60,100] }
               token 6 { not_visible [350,350] [1,100] } }
                 }
           }
```

Figure 1: A sample plan, with two unallocated tokens

## 3.1 Semantics of the primitive and defined relations

The semantics of the primitive relations is defined as follows. If $A = [b_A, e_A]$ and $B = [b_B, e_B]$ are time intervals, with $b_A, e_A, b_B, e_B \in \mathbb{N}$, the semantics of the relation $A\ \mathsf{r}\ [lb, ub]\ B$, with $\mathsf{r} \in \{$start_before_start, end_before_end,

start_before_end, end_before_start}, $lb \in \mathbb{N}$ and $ub \in \mathbb{N} \cup \{\infty\}$, is given in the following table.

| the relation | holds if |
|---|---|
| $A$ start_before_start $[lb, ub]$ $B$ | $lb \leq b_B - b_A \leq ub$ |
| $A$ end_before_end $[lb, ub]$ $B$ | $lb \leq e_B - e_A \leq ub$ |
| $A$ start_before_end $[lb, ub]$ $B$ | $lb \leq e_B - b_A \leq ub$ |
| $A$ end_before_start $[lb, ub]$ $B$ | $lb \leq b_B - e_A \leq ub$ |

If $A = [b, e]$ is a time interval, with $b, e \in \mathbb{N}$, the semantics of the relation $A$ r $[lb, ub]$ $t$, with r $\in$ {starts_before, starts_after, ends_before, ends_after}, $t, lb \in \mathbb{N}$ and $ub \in \mathbb{N} \cup \{\infty\}$, is given in the following table.

| the relation | holds if |
|---|---|
| $A$ starts_before $[lb, ub]$ $t$ | $lb \leq t - b \leq ub$ |
| $A$ starts_after $[lb, ub]$ $t$ | $lb \leq b - t \leq ub$ |
| $A$ ends_before $[lb, ub]$ $t$ | $lb \leq t - e \leq ub$ |
| $A$ ends_after $[lb, ub]$ $t$ | $lb \leq e - t \leq ub$ |

The other relations are defined in terms of the primitive ones, as shown in Table 1.

# 4 Encoding of flexible plans as UPPAAL-TIGA systems

This section is devoted to describe how a plan $\langle \mathbf{FTL}, \mathcal{R} \rangle$ is encoded as an UPPAAL-TIGA system. The encoding establishes an injective mapping $\mu$ from tokens of **FTL** to states of processes of the corresponding system $\mathcal{N}$, and a correspondence can be consequently estabilished between scheduled timelines and runs, according to the following definition.

**Definition 1.** *A run $\rho$ of $\mathcal{N}$ corresponds to a set of scheduled timelines* **TL**, *and vice-versa, if for every state $\mu(x^i)$ of $\mathcal{N}$, $\rho$ enters $\mu(x^i)$ at time $t = \text{start\_time}(x^i)$ and exits $\mu(x^i)$ at time $t = \text{end\_time}(x^i)$.*

The encoding is correct and complete:

**Theorem 1.** *Let $\Pi = \langle \mathbf{FTL}, \mathcal{R} \rangle$ be a flexible plan, and let $\mathcal{N}$ be the* UPPAAL-TIGA *system modeling $\Pi$. Then every instance of $\Pi$ corresponds to a run of $\mathcal{N}$, and every run of $\mathcal{N}$ corresponds to an instance of $\Pi$.*

*Morever, for every token $x^i$ of $\mathcal{N}$, $x^i$ is uncontrollable if and only if the (only) link exiting from $\mu(x^i)$ is uncontrollable.*

Below, the encoding generated by PLAN2TIGA is defined, and arguments are given step by step showing that Theorem 1 holds.

The encoding allows also the modeling of "partial plans", where timelines may have undefined temporal slots, i.e. are tokens without value and possibly with no associated duration. In what follows, we treat such "holes" in a timeline like tokens, calling them *unallocated* when needing to make the distinction.

The UPPAAL-TIGA system has a set of global clocks: *plan_clock*, whose value corresponds to the time elapsed since the beginning of the execution, and

| the relation | is defined as |
|---|---|
| $A$ equals $B$ | $A$ start_before_start $[0,0]$ $B$ and $A$ end_before_end $[0,0]$ $B$ |
| $A$ meets $B$ | $A$ end_before_start $[0,0]$ $B$ |
| $A$ met_by $B$ | $B$ meets $A$ |
| $A$ before $[lb,ub]$ $B$ | $A$ end_before_start $[lb,ub]$ $B$ |
| $A$ after $[lb,ub]$ $B$ | $B$ before $[lb,ub]$ $A$ |
| $A$ overlaps $[lb_1,ub_1][lb_2,ub_2]$ $B$ | $A$ start_before_start $[lb_1,ub_1]$ $B$ and $A$ end_before_end $[lb_2,ub_2]$ $B$ and $B$ start_before_end $[0,\infty]$ $A$ |
| $A$ overlapped_by $[lb_1,ub_1][lb_2,ub_2]$ $B$ | $B$ overlaps $[lb_1,ub_1][lb_2,ub_2]$ $A$ |
| $A$ contains $[lb_1,ub_1][lb_2,ub_2]$ $B$ | $A$ start_before_start $[lb_1,ub_1]$ $B$ and $B$ end_before_end $[lb_2,ub_2]$ $A$ |
| $A$ during $[lb_1,ub_1][lb_2,ub_2]$ $B$ | $B$ contains $[lb_1,ub_1][lb_2,ub_2]$ $A$ |
| $A$ starts $[lb,ub]$ $B$ | $A$ start_before_start $[0,0]$ $B$ and $A$ end_before_end $[lb,ub]$ $B$ |
| $A$ started_by $[lb,ub]$ $B$ | $B$ starts $[lb,ub]$ $A$ |
| $A$ finishes $[lb,ub]$ $B$ | $A$ start_before_start $[lb,ub]$ $B$ and $A$ end_before_end $[0,0]$ $B$ |
| $A$, finished_by $[lb,ub]$ $B$ | $B$ finishes $[lb,ub]$ $A$ |
| $A$ contains_start $[lb_1,ub_1][lb_2,ub_2]$ $B$ | $A$ start_before_start $[lb_1,ub_1]$ $B$ and $B$ start_before_end $[lb_2,ub_2]$ $A$ |
| $A$ contains_end $[lb_1,ub_1][lb_2,ub_2]$ $B$ | $A$ start_before_end $[lb_1,ub_1]$ $B$ and $B$ end_before_end $[lb_2,ub_2]$ $A$ |
| $A$ starts_during $[lb_1,ub_1][lb_2,ub_2]$ $B$ | $B$ contains_start $[lb_1,ub_1][lb_2,ub_2]$ $A$ |
| $A$ ends_during $[lb_1,ub_1][lb_2,ub_2]$ $B$ | $B$ contains_end $[lb_1,ub_1][lb_2,ub_2]$ $A$ |
| $A$ starts_at $t$ | $A$ starts_before $[0,0]t$ |
| $A$ ends_at $t$ | $A$ ends_before $[0,0]t$ |

Table 1: Semantics of the defined temporal relations

a clock for every relation between two tokens belonging to $\mathcal{R}$. Moreover, a global constant $H$ is used, whose value is strictly greater than the plan horizon.

As already said, the UPPAAL-TIGA system is made up of processes modeling the timelines in **FTL**. Every process has a "final state", $finish$, and, in order to check whether the system models a dynamically controllable plan, and, if so, to generate a controllable execution strategy, the UPPAAL-TIGA winning conditions are *pure reachability* ones, expressed by the UPPAAL-TIGA formula

$$\mathsf{A}\Diamond(\mathcal{A}_{x_1}.finish \wedge \cdots \wedge \mathcal{A}_{x_k}.finish)$$

where $\mathbf{FTL} = \{FTL_{x_1}, \ldots, FTL_{x_k}\}$ and $A_{x_i}$ is the automaton modeling $FTL_{x_i}$: whichever the behaviour of uncontrollable components, the automata in $\mathcal{N}$ can reach their final states.

### 4.0.1 Representation of timelines

Each timeline $FTL_x$ corresponds to an UPPAAL-TIGA process $\mathcal{A}_x$, and the mapping $\mu$ maps every token of $FTL_x$ to a state of $\mathcal{A}_x$. The main properties of $\mathcal{A}_x$ are the following:

1. the states of $\mathcal{A}_x$ are $\{start, finish\} \cup \{\mu(x^i) \mid x^i \text{ is a token of } FTL_x\}$;
2. the initial state of $\mathcal{A}_x$ is *start*;
3. every state of $\mathcal{A}_x$ has exactly one incoming edge, except for *start*, that has none;
4. every state of $\mathcal{A}_x$ has exactly one outgoing edge, except for $finish$, that has none;
5. there is an edge in $\mathcal{A}_x$ from *start* to $\mu(x^1)$, where $x^1$ is the first token of $FTL$ and there is an edge in $\mathcal{A}_x$ from $\mu(x^k)$ to $finish$, where $x^k$ is the last token of $FTL$;
6. for every pair of consecutive tokens $x^i$ and $x^{i+1}$ in $FTL$, there is an edge from $\mu(x^i)$ to $\mu(x^{i+1})$ in $\mathcal{A}_x$. Moreover, if $x^i$ is uncontrollable (which holds, in particular, if $x$ is an external variable), then the transition from $\mu(x^i)$ to $\mu(x^{i+1})$ is uncontrollable, too.

The process $\mathcal{A}_x$ has a local clock $clock_x$, that is reset every time a state is entered and checked before exiting, in order to control the state duration.

The role of *start* and $finish$ is to make every other state have an incoming and an outgoing edge. They are associated no invariant. For any token $x^i$ of the state variable $x$, with $\text{end\_time}(x^i) = (e, e')$ and $\text{duration}(x^i) = (d, d')$, the corresponding state $\mu(x^i)$ has the invariants $plan\_clock \leq e'$ and, if $d' \neq \infty$, $clock_x \leq d'$. If $x^i$ is an unallocated token with no duration constraints, the invariant $clock_x \leq d'$ is obviously omitted.

As described below, the local clock $clock_x$ is reset on the edge entering each state, therefore its value when exiting $\mu(x^i)$ represents the duration of the token $x^i$. The invariants thus ensure that:

- system runs do not exit $\mu(x^i)$ before (global) time $e'$;
- system runs do not exit $\mu(x^i)$ before $d'$ time units have elapsed since the run has entered $\mu(x^i)$.

The core of the transitions of the process $\mathcal{A}_x$ is described next (other guards and assigments can be added to transitions, in order to model relations, as specified in Section 4.0.2). The transitions from *start* and to $finish$ are considered separately.

While showing which guards and clock resettings are associated with the chain of transitions $start \to x^1 \to \ldots x^k \to finish$, we also inductively show that every schedule of $FTL_x$ corresponds to a run of $\mathcal{A}_x$ and vice versa: if the begin and and time of a scheduled token $x^i$ are $t$ and $t'$, respectively, then the guards on the transition allow a run to enter the state $\mu(x^i)$ at time $t$ and exit it at time $t'$. Conversely, if a run of $\mathcal{A}_x$ enters the state $\mu(x^i)$ at time $t$ and exits it at time $t'$, then there is a schedule of $FTL_x$ where the begin and end time of $x^i$ are $t$ and $t'$, respectively.

1. Let $x^1$ be the first token of the timeline. The edge $start \rightarrow \mu(x^1)$ has the guard $plan\_clock = 0$, therefore any run $\rho$ of $\mathcal{A}_x$ enters $\mu(x^1)$ at time $0 = \text{start\_time}(x^1)$; correspondingly, the start time of the first token in any schedule $TL_x$ of $FTL_x$ is 0.

2. Let $x^i$ and $x^{i+1}$ be two consecutive tokens of $FTL_x$, with $\text{end\_time}(x^i) = (e, e')$ and $\text{duration}(x^i) = (d, d')$. Then the edge $\mu(x^i) \rightarrow \mu(x^{i+1})$ has the guards

$$plan\_clock \geq e \text{ and } clock_x \geq d$$

   Moreover, the local clock is reset:

$$clock_x := 0$$

   If $x^i$ is uncontrollable (which holds, in particular, if $x$ is an external variable), then the edge $\mu(x^i) \rightarrow \mu(x^{i+1})$ is uncontrollable, too. If $x^i$ is an unallocated token with no duration constraints, then the guard on $clock_x$ is omitted.

   Since $clock_x$ is reset when a run enters $\mu(x^i)$, its value when the run exits $\mu(x^i)$ is equal to the time the run stays in $\mu(x^i)$. I.e. it represents the duration of the corresponding scheduled token.

   Now, if $TL_x$ is a schedule of $FLT_x$, with $\text{start\_time}(x^i) = t$ and $\text{end\_time}(x^i) = t'$, then $e \leq t' \leq e'$ and $d \leq t' - t \leq d'$. The guards on $\mu(x^i) \rightarrow \mu(x^{i+1})$ and the invariants of $\mu(x^i)$ ($plan\_clock \leq e'$ and, possibly, $clock_x \leq d'$) allow a run of $A_x$ to exit $\mu(x^i)$ (and enter $\mu(x^{i+1})$) when the value of the plan clock is $t'$ and the value of $clock_x$ is $t' - t$.

   Conversely, if a run of $A_x$ enters $\mu(x^i)$ when the value of the plan clock is $t$ and exits $\mu(x^i)$ when $plan\_clock = t'$ and $clock_x = d^*$, then forcely $e \leq t' \leq e'$ and $d \leq d^* \leq d'$. Since $d^* = t' - t$, the values $t$ and $t'$ respect the conditions $e \leq t' \leq e'$ and $d \leq t' - t \leq d'$, required for a schedule of $FTL_x$.

3. Let $x^k$ be the last token of the timeline, with $\text{end\_time}(x^k) = (e, e')$ and $\text{duration}(x^k) = (d, d')$. If $x$ is an external variable, then the edge $\mu(x^k) \rightarrow finish$ has only the guard $plan\_clock \geq e$. If $x$ is a planned variable, then the edge $\mu(x^k) \rightarrow finish$ has the guards $plan\_clock \geq e$ and $clock_x \geq d$.

   It is worth noticing that the last transition of the system is controllable, since otherwise the UPPAAL-TIGA game could not be won.

   If $x$ is a planned variable, the correspondence between schedules and runs can be proved like in case 2. Otherwise, the constraints on the end time $t'$ of a schedule of $x^k$ whose start time is $t$ are only $e \leq t' \leq e'$ and $t' - t \leq d'$. The absence of the guard $clock_x \geq d$ allows a run of $\mathcal{A}_x$ to exit the last token of the (external) timeline even when $clock_x$ (i.e. $t' - t$) is less than $d$. And conversely, if a run exits $\mu(x^k)$ when the value of $clock_x$ is smaller than $d$, such a value is still a legal duration for $x^k$.

#### 4.0.2 Encoding of the relations

Relations between a token and a time point are quite simple to be encoded by use of the plan clock.

- The relation $x^i$ starts_before$_{[lb,ub]}$ $t$ is encoded by adding the guards $plan\_clock \leq t - lb$ and $plan\_clock \geq t - ub$ to the edge *entering* $\mu(x^i)$.

  In every instance of the plan, the start time $b$ of the schedule of $x^i$ lies in the interval $[t - lb, t - ub]$. Therefore the guard doesn't prevent a run of $A_x$ to enter $\mu(x^i)$ at time $b$. Conversely, if a run of $A_x$ enters $\mu(x^i)$ at a given time $b$, then $b \in [t - lb, t - ub]$, therefore there is an instance of $x^i$ starting at $b$.

- The relation $x^i$ ends_before$_{[lb,ub]}$ $t$ is encoded by adding the guards $plan\_clock \leq t - lb$ and $plan\_clock \geq t - ub$ to the edge *exiting* $\mu(x^i)$.

  The reasoning showing the correspondence between runs and instances of the plan is similar to the previous case, but for the fact that the end time/exit time are considered.

- The relation $x^i$ starts_after$_{[lb,ub]}$ $t$ is encoded by adding the guards $plan\_clock \geq t + lb$ and $plan\_clock \leq t + ub$ to the edge *entering* $\mu(x^i)$.

  In every instance of the plan, the start time $b$ of the schedule of $x^i$ lies in the interval $[t + lb, t + ub]$. Therefore the guard doesn't prevent a run of $A_x$ to enter $\mu(x^i)$ at time $b$. Conversely, if a run of $A_x$ enters $\mu(x^i)$ at a given time $b$, then $b \in [t + lb, t + ub]$, therefore there is an instance of $x^i$ starting at $b$.

- The relation $x^i$ ends_after$_{[lb,ub]}$ $t$ is encoded by adding the guards $plan\_clock \geq t + lb$ and $plan\_clock \leq t + ub$ to the edge *exiting* $\mu(x^i)$.

  The reasoning showing the correspondence between runs and instances of the plan is similar to the previous case, but for the fact that the end time/exit time are considered.

In order to model relations between two tokens, a global clock is defined for each of them, and a constant $H$ greater than the plan horizon is used to initialize them. A relation $R$ of the form $x^n$ $\mathsf{r}_{[lb,ub]}$ $y^k$ is enforced by resetting the clock $clock_R$ associated to $R$ on the edge entering/exiting $\mu(x^n)$ and checked by the guard on the edge entering/exiting $\mu(y^k)$. Whether the incoming or outgoing edges are concerned depends on the particular relation $\mathsf{r}$.

The primitive relations have the form $x^n$ start/end_before_start/end$_{[lb,ub]}$ $y^k$, i.e. the token at the left of the relation starts or ends before the rightmost one starts or ends. The clock $clock_R$ corresponding to a relation of the above form is reset on the link entering (if the relation has the form $x^n$ start_before_start/end$_{[lb,ub]}$ $y^k$) or exiting (if it has the form $x^n$ end_before_start/end$_{[lb,ub]}$ $y^k$) the state $\mu(x^n)$. A guard checking the value of $clock_R$ is placed on the link entering (if $R$ has the form $x^n$ start/end_before_start$_{[lb,ub]}$ $y^k$) or exiting (if it has the form $x^n$ start/end_before_end$_{[lb,ub]}$ $y^k$) $\mu(y^k)$.

When $clock_R$ is reset, its value is set equal to $H$. It must in fact be guaranteed that guards concerning a relation clock $clock_R$ are not satisfied when the relation is not. So, the value $H$ greater than the plan horizon is used to reset relation clocks (without explicit assignment no clock could reach $H$ during any run).

In detail, when the clock $clock_R$ is reset, it is assigned the value $H$: $clock_R := H$. If the bound of the relation $R$ is the interval $[lb, ub]$, then the guards asso-

ciated with the relation are $H + lb \leq clock_R$ and (if $ub \neq \infty$) $clock_R \leq H + ub$. The different cases are:

- The relation $R$ is $x^n$ start_before_start$_{[lb,ub]}$ $y^k$. Then $clock_R$ is assigned the value $H$ on the edge entering $\mu(x^n)$ and the guards associated to $R$ are placed on the edge entering $\mu(y^k)$.

  We show the correspondence between runs and instances of the plan in this case, the others being similar. In every instance of the plan, if the start time of the schedule of $x^n$ is $t_n$ and the start time of the schedule of $y^k$ is $t_k$, then it must be $lb \leq t_k - t_n \leq ub$. If a run enters $\mu(x^n)$ at time $t_n$, the value of $clock_R$ at time $t_k$ is $H + (t_k - t_n)$. From $lb \leq t_k - t_n \leq ub$ it follows that $H + lb \leq H + (t_k - t_n) \leq H + ub$. therefore the guard allows the run to actually enter $\mu(y^k)$ at time $t_n$.

  Conversely, if a run enters $\mu(x^n)$ at time $t_n$ and enters $y^k$ at time $t_k$, the value of $clock_R$ when it enters $\mu(y^k)$ is $H + (t_k - t_n)$. Therefore, since the guard is satisfied, $H + lb \leq H + (t_k - t_n) \leq H + ub$, which implies $lb \leq t_k - t_n \leq ub$. As a consequence, a schedule of **FTL** where the start time of the schedule of $x^n$ is $t_n$ and the start time of the schedule of $y^k$ is $t_k$ satisfies the relation $R$.

- The relation $R$ is $x^n$ start_before_end$_{[lb,ub]}$ $y^k$. Then $clock_R$ is assigned the value $H$ on the edge entering $\mu(x^n)$ and the guards are placed on the edge exiting $\mu(y^k)$.

- The relation $R$ is $x^n$ end_before_start$_{[lb,ub]}$ $y^k$. Then $clock_R$ is assigned the value $H$ on the edge exiting $\mu(x^n)$ and the guards are placed on the edge entering $\mu(y^k)$.

- The relation $R$ is $x^n$ end_before_end$_{[lb,ub]}$ $y^k$. Then $clock_R$ is assigned the value $H$ on the edge exiting $\mu(x^n)$ and the guards are placed on the edge exiting $\mu(y^k)$.

# 5 Encoding example

This section shows how the sample plan given in Figure 1 is encoded.

### 5.0.3 System file

```
clock plan_clock, R1_clock, R2_clock, R4_clock;
const int H = 700;

process pm ()
{
clock pm_clock;
state
        start,
        pm1 { plan_clock <= 20 and pm_clock <= 20 },
        pm2 { plan_clock <= 130 and pm_clock <= 110 },
        pm4 { plan_clock <= 160 and pm_clock <= 30 },
        pm5 { plan_clock <= 200 and pm_clock <= 50 },
```

```
        pm6 { plan_clock <= 250 },
        pm8 { plan_clock <= 300 and pm_clock <= 50 },
        pm9 { plan_clock <= 350 and pm_clock <= 90 },
        finish;
init start;
trans
        start -> pm1 { guard plan_clock==0; },
        pm1 -> pm2  { guard plan_clock >= 10 and pm_clock >= 10;
                        assign pm_clock := 0; },
        pm2 -> pm4  { guard plan_clock >= 110 and pm_clock >= 100;
                        assign pm_clock := 0; },
        pm4 -> pm5  { guard plan_clock >= 140 and pm_clock >= 30;
                        assign pm_clock := 0; },
        pm5 -> pm6  { guard plan_clock >= 190 and pm_clock >= 40;
                        assign pm_clock := 0,
                         R4_clock  := H ;  },
        pm6 -> pm8  { guard plan_clock >= 230 and pm_clock >= 0
                        and  R4_clock  >= H + 0 and  R4_clock  <= H + 50
                        and  R1_clock  >= H + 0;
                    assign pm_clock := 0; },
        pm8 -u-> pm9  { guard plan_clock >= 260 and pm_clock >= 30;
                        assign pm_clock := 0,
                         R2_clock  := H ;  },
        pm9 -> finish { guard plan_clock >= 350 and pm_clock >= 50; }
        ;
}

process gv ()
{
clock gv_clock;
state
        start,
        gv1 { plan_clock <= 100 and gv_clock <= 100 },
        gv2 { plan_clock <= 130 and gv_clock <= 100 },
        gv3 { plan_clock <= 190 and gv_clock <= 100 },
        gv4 { plan_clock <= 250 and gv_clock <= 100 },
        gv5 { plan_clock <= 320 and gv_clock <= 100 },
        gv6 { plan_clock <= 350 and gv_clock <= 100 },
        finish;
init start;
trans
        start -> gv1 { guard plan_clock==0;
                       assign plan_clock := 0, gv_clock := 0 ;},
        gv1 -u-> gv2  { guard plan_clock >= 60 and gv_clock >= 60;
                        assign gv_clock := 0; },
        gv2 -u-> gv3  { guard plan_clock >= 90 and gv_clock >= 1;
                        assign gv_clock := 0; },
        gv3 -u-> gv4  { guard plan_clock >= 150 and gv_clock >= 60;
                        assign gv_clock := 0; },
        gv4 -u-> gv5  { guard plan_clock >= 210 and gv_clock >= 1;
```

```
                        assign gv_clock := 0,
                         R1_clock  := H ;  },
        gv5 -u-> gv6  { guard plan_clock >= 300 and gv_clock >= 60
                        and  R2_clock  >= H + 0;
                        assign gv_clock := 0; },
        gv6 -> finish { guard plan_clock >= 350; }
        ;
}

system gv, pm;
```

#### 5.0.4 Query file

```
control: A<> (pm.finish && gv.finish)
```

#### 5.0.5 Query file with the -debug option

```
control: A<>pm.finish
control: A<>gv.finish
```

## References


[1] Marta Cialdea Mayer, Andrea Orlandini, and Alessandro Umbrico. A formal account of planning with flexible timelines. In *Temporal Representation and Reasoning (TIME), 2014 21st International Symposium on*, pages 37–46, 2014.