Relating Time Progress and Deadlines in Hybrid Systems*

Sébastien Bornot and Joseph Sifakis
SPECTRE-VERIMAG**, Sebastien.Bornot@imag.fr and Joseph.Sifakis@imag.fr

Abstract. Time progress conditions in hybrid systems are usually specified in terms of invariants, predicates characterizing states where time can continuously progress or dually, deadline conditions, predicates characterizing states where time progress immediately stops. The aim of this work is the study of relationships between general time progress conditions and these generated by using state predicates. It is shown that using deadline conditions or invariants allows to characterize all practically interesting time progress conditions. The study is performed by using a Galois connection between the corresponding lattices. We provide conditions for the connection to be a homomorphism and apply the results to the compositional description of hybrid systems.

1 Introduction

Hybrid systems are systems that combine discrete and continuous dynamics. Their semantics is usually defined as a transition system on a set of states $Q$ consisting of

- transition relations $\delta \subseteq Q \times Q$ for $a \in A$ where $A$ is a possibly infinite set of action names,
- time progress relations $\triangleright \subseteq Q \times Q$ for $t \in \mathbb{R}_+$ such that

$$\forall q_1, t_1, t_2, \exists q_2, q_3, q_1 \trianglerighteq q_2 \land q_2 \trianglerighteq q_3 \Rightarrow q_1 \triangleright t_1 + t_2 q_3 \ (\text{additivity property}).$$

The behavior of a hybrid system is characterized by the set of the execution sequences of the transition system. Additivity property guarantees that the set of states reached from a state within a given time is independent of the sequence of the time steps performed.

Usually, hybrid systems are modeled as hybrid automata (cf [ACH+95]), automata extended with a set of real valued variables. The variables can be tested and modified at transitions. Continuous state changes are specified by associating with automaton states evolution laws and constraints restricting the domain of variables.

* in proc. HART '97, LNCS 1201 p. 286-300
** VERIMAG is a joint laboratory of CNRS, Institut National Polytechnique de Grenoble, Université J. Fourier and Vérilog SA associated with IMAG. VERIMAG Centre Equitation, 2, av. de Vignate, 38610 Gières, France
Example 1. The following example represents the hybrid automaton for a thermostat. The variable $\theta$ represents the temperature which decreases (resp. increases) at states OFF and ON according to the laws $\theta \vdash_{\text{OFF}} t$ (resp. $\theta \vdash_{\text{ON}} t$). Furthermore, the conditions $m < \theta$ and $\theta < M$ are invariants restricting the values of $\theta$ between minimal and maximal values $m$ and $M$ respectively. Transitions occur when $\theta$ reaches limit values.

![Diagram of the thermostat example]

**Fig. 1.** the thermostat example

The hybrid automaton represents the transition system with $Q = \{\text{ON}, \text{OFF}\} \times \mathbb{R}$ and $A = \{\text{START}, \text{STOP}\}$ such that

1. $q \xrightarrow{\text{START}} q' \iff q = (\text{OFF}, m) \land q' = (\text{ON}, m)$
2. $q \xrightarrow{\text{STOP}} q' \iff q = (\text{ON}, M) \land q' = (\text{OFF}, M)$
3. $(\text{OFF}, \theta) \xrightarrow{t} (\text{OFF}, \theta \vdash_{\text{OFF}} t)$ if $\forall t' \ 0 \leq t' < t \land m < \theta \vdash_{\text{OFF}} t'$
4. $(\text{ON}, \theta) \xrightarrow{t} (\text{ON}, \theta \vdash_{\text{ON}} t)$ if $\forall t' \ 0 \leq t' < t \land \theta \vdash_{\text{ON}} t' < M$

Notice that the invariants $m < \theta$ and $\theta < M$ play an important role in this description as they do not allow temperature to progress beyond limit values, $\theta = m$ and $\theta = M$. Furthermore, when these values are reached time cannot progress by making the execution of the enabled transitions "urgent".

In this paper we consider hybrid systems represented as transition systems whose time progress relations are specified as a pair $(\triangleright, f)$ where $\triangleright$ is an evolution law, total function from $Q \times \mathbb{R}^+$ into $Q$ and $f$ is a time progress function, predicate on $Q \times \mathbb{R}^+$ such that $q \xrightarrow{t} q' \iff q = q \triangleright t \land f(q, t)$.

Such a representation is common in hybrid automata where evolution laws are specified either explicitly or by a system of differential equations. Time progress function describes how from a given state time can progress by some amount. If for a given state it is false for any positive time, time cannot progress from this state. We call such a state *deadline state* because stopping time progress is used in practice to enforce a transition meeting a deadline.
Time progress functions are usually specified in terms of state predicates without mentioning time explicitly. These predicates characterize either the states where time can continuously progress (invariants in [ACH+95]) or du-
ally, the states where time progress immediately stops (deadline conditions in
[SY96]). Given such a predicate and an evolution law $\triangleright$, one can define progress functions: from a given state $q$ time can progress by $t$ if all the states encountered along the $\triangleright$-trajectory satisfy the invariant or dually do not satisfy the
deadline condition. Invariants and deadline conditions are dual notions. In this paper we consider deadline conditions; the results can be adapted to invariants
by dualization.

Formally, given a deadline condition $d(q)$, a time progress function $tp(d)(q,t)$
can be defined: $tp(d)(q,t) = \forall t': 0 < t' \leq t \cdot \neg d(q \triangleright t')$

Conversely, from given a time progress function $f(q,t)$ one can define a
deadline condition $dl(f)(q) : dl(f)(q) = \forall t > 0 \cdot \neg f(q,t)$. This simply means that the
deadline condition corresponding to $f(q,t)$ is satisfied by all the states from
which time cannot progress by any positive quantity.

If deadline conditions or invariants are useful for specification purposes, it
is important to have available in some explicit form progress functions for sim-
ulation or analysis purposes. Explicit knowledge of progress function can help
accelerating simulation by making it driven by deadline events.

The question arises about the nature of the correspondence between time
progress functions and deadline conditions. Is it possible by using deadline condi-
tions or invariants to characterize all time progress functions? The (obvious)
answer is no. However, we show in section 2 that using deadlines allows to char-
acterize some reasonably large class of progress functions. Formally speaking, we
show that the pair of functions $(dl, tp)$ is a Galois connection between the lattice
of time progress functions and the lattice of deadline conditions.

In section 3, we investigate the relationships between the structures of the two
lattices and provide conditions for $tp$ and $dl$ to be homomorphisms. Furthermore,
we illustrate the use of the results for the compositional description of hybrid
systems. We show how for modal formulas describing a global deadline condition
in terms of local deadline conditions; a global progress function can be obtained
in terms of local progress functions.

2 The correspondence between $TP$ and $DL$

We study relations between time progress functions and deadline conditions for
hybrid systems with set of states $Q$ and evolution law $\triangleright : Q \times R^+ \to Q$ that is
additive and assumed fixed through the paper. Both time progress functions and
deadline conditions are considered as predicates, that is, functions into the set
$\{tt, ff\}$. We use standard notation $\lor, \land, \neg$ and $\Rightarrow$ to represent disjunc-
tion, conjunction, negation and implication. We represent by true and false respectively
the functions $\lambda x. tt$ and $\lambda x. ff$. 
2.1 The lattice of time progress functions TP

For a given evolution law $\triangleright$, a time progress function is a function $f$, $f : Q \times \mathbb{R}^+ \rightarrow \{tt, ff\}$ such that:

- $f(q, 0) = tt$
- $\forall t_1, t_2. f(q, t_1 + t_2) = f(q, t_1) \land f(q \triangleright t_1, t_2)$ (additivity)

Example 2. If $q \triangleright t = q + t$ then

$$f_1(q, t) = q + t \leq 2 \lor (t = 0) \text{ and }$$

$$f_2(q, t) = 0 \leq q \land (q + t \leq 2) \lor (t = 0)$$

are time progress functions while

$$f_3(q, t) = 0 \leq q + t \leq 2 \lor (t = 0)$$

is not a time progress function as $f_3(-1, 2) = tt$, and $f_3(-1, t) = ff \forall t \in [0, 1)$.

Let TP be the set of time progress functions. TP is partially ordered by $\triangleright$ with bottom element the function $\lambda q. 0 = 0$ and top element $true$.

We represent by $\sqcap$ and $\sqcup$ respectively, the greatest lower bound and least upper bound operations on TP. Notice that from the above definition, we have that if $f_1, f_2$ are time progress functions then $f_1 \land f_2$ is a time progress function. Consequently, $f_1 \land f_2 = f_1 \land f_2$. However, $f_1 \lor f_2$ is not in general a time progress function. For instance, if $q \triangleright t = q + t$, the function

$$f_4(q, t) = 0 \leq q \land (q + t \leq 2) \lor 2 \leq q \land (q + t \leq 4) \lor (t = 0)$$

is the disjunction of two time progress functions but it is not a time progress function as $f_4(0, 4) = ff$ while $f_4(0, 2) = tt$ and $f_4(2, 2) = tt$. However, one can find

$$f_5(q, t) = 0 \leq q \land (q + t \leq 2) \lor (t = 0) \sqcup 2 \leq q \land (q + t \leq 4) \lor (t = 0)$$

which is equal to $0 \leq q \land (q + t \leq 4) \lor (t = 0)$ and is the least time progress function implied by both $0 \leq q \land (q + t \leq 2) \lor (t = 0)$ and $2 \leq q \land (q + t \leq 4) \lor (t = 0)$.

**Proposition 1.** (TP, $\triangleright$, $\sqcap$, $\sqcup$) is a distributive lattice with:

$$f_1 \sqcap f_2 = f_1 \land f_2 \text{ and } f_1 \sqcup f_2 = \bigvee_{i=1}^{\infty} f_1 \lor_i f_2$$

where:

$$f_1 \lor_i f_2 = f_1 \lor f_2$$

$$f_1 \lor_{i+1} f_2(q, t) = \exists t'. 0 \leq t' \leq t. (f_1 \lor_i f_2)(q, t') \land (f_1 \lor f_2)(q \triangleright t', t - t')$$
Proof.

- $f_1 \sqcap f_2 = f_1 \land f_2$ is immediate.
- For $f_1 \sqcup f_2 = \bigvee_{i=1}^{\infty} f_1 \lor f_2$:
  $f_j \Rightarrow f_1 \lor f_2 \Rightarrow f_1 \sqcup f_2$ for $j \in \{1, 2\}$
  On the other hand, if for some arbitrary time progress function $f$, $f_j \Rightarrow f$
  for $j \in \{1, 2\}$, we will show by induction that $\forall i \in \mathbb{N} : f_1 \lor f_2 \Rightarrow f$ and
  therefore $f_1 \sqcup f_2 \Rightarrow f$
  $f_1 \lor f_2 \Rightarrow f$
  If $f_1 \lor_{i-1} f_2 \Rightarrow f$, then for all $(q, t)$ such that $(f_1 \lor f_2)(q, t) = tt$ we have by
  definition:
  $\exists t' \ 0 \leq t' \leq t \cdot (f_1 \lor_{i-1} f_2)(q, t') \land (f_1 \lor f_2)(q \triangleright t', t - t')$
  and then: $\exists t' \ 0 \leq t' \leq t \cdot f(q, t') \land f(q \triangleright t', t - t')$
  by additivity of $f$: $f(q, t) = tt$.

2.2 The lattice of deadlines

Consider the set of state predicates $DL$ whose elements $d$ are unary predicates on $Q$ (functions from $Q$ into $\{tt, ff\}$). We shall interpret the elements of $DL$
 as deadline conditions. $DL$ is a boolean lattice with the standard operations of
 conjunction, disjunction and negation.

We define the pair of functions $(tp, dl)$ relating $DL$ and $TP$:
$tp : DL \rightarrow TP$ such that $tp(d)(q, t) = \forall t' \ 0 \leq t' < t \cdot \neg d(q \triangleright t')$
$dl : TP \rightarrow DL$ such that $dl(f)(q) = \forall t > 0 \cdot \neg f(q, t)$

It is trivial to check that $tp(d)$ is a progress function. We call $tp(d)$ the
 progress function corresponding to $d$ and $dl(f)$ the deadline condition corre-
 sponding to $f$. 
Notice that the definition of $tp$ depends on the evolution law $\rightarrow$ which can be considered as a family of curves parameterized with time in the space of variables. If a curve at a state $q$ is parameterized with $t_0$ then the state $q \rightarrow t$ reached by letting time pass by $t$, is on the curve parameterized by $t_0 + t$.

**Example 3.** For $q \rightarrow t = q + t$ and $d(q) = 2 < q < 3$ we have,
\[
\begin{align*}
\forall t' \ 0 \leq t' < t \ . \ \neg 2 < q + t' < 3 \quad \text{which gives} \\
\forall t' \ 0 \leq t' < t \ . \ \neg 2 < q + t' < 3
\end{align*}
\]
\[
\begin{align*}
tp(d)(q, t) = (t = 0) \lor q + t \leq 2 \lor 3 \leq q.
\end{align*}
\]
If we compute the deadline condition corresponding to the latter time progress function we find: $dl(tp(d))(q) = 2 \leq q < 3$ which differs from $d$ in that it is left-closed. However, we have $tp(2 \leq q < 3) = tp(2 < q < 3)$.

Consider now that $d = \neg(2 < q < 3)$ which means that time can progress only from states $q$ such that $2 < q < 3$. We find
\[
\begin{align*}
tp(d)(q, t) = \forall t' \ 0 \leq t' < t \ . \ 2 < q + t' < 3
\end{align*}
\]
which is equivalent to
\[
\begin{align*}
tp(d)(q, t) = (t = 0) \lor 2 < q \land q + t \leq 3.
\end{align*}
\]
The deadline condition corresponding to the latter is again $d = \neg(2 < q < 3)$.

### 2.3 The Galois connection between TP and DL

**Proposition 2.** For any deadline condition $d$, $d \Rightarrow dl \ tp(d)$

**Proof.**
\[
\begin{align*}
dl \ tp(d)(q) = \forall t > 0 \ . \ \neg tp(d)(q, t) \\
= \forall t > 0 \ . \ \neg \forall t' \ 0 \leq t' < t \ . \ \neg d(q \rightarrow t') \\
= \forall t > 0 \ . \ \exists t' \ 0 \leq t' < t \ . \ d(q \rightarrow t')
\end{align*}
\]
If $d(q) = tt$, by choosing $t' = 0$, we have $dl \ tp(d) = tt$.

**Proposition 3.** For any progress function $f$, $f \Rightarrow tp \ dl(f)$

**Proof.**
\[
\begin{align*}
\forall t' \ 0 \leq t' < t \ . \ \neg dl(f)(q \rightarrow t') \\
= \forall t' \ 0 \leq t' < t \ . \ \neg (\exists t'' > 0 \ . \ \neg f(q \rightarrow t', t'')) \\
= \forall t' \ 0 \leq t' < t \ . \ \exists t'' > 0 \ . \ f(q \rightarrow t', t'')
\end{align*}
\]
If $f(q, t) = tt$, choose $t'' = t - t'$, and by additivity, $tp \ dl(f)(q, t) = tt$. 

A consequence of the above propositions and of the fact that \( tp \) and \( dl \) are anti-monotonic, is that the pair \((tp,dl)\) is a Galois connection (see for example [Ore44, San77]) between \( DL \) and \( TP \) where \( dl \) and \( TP \) are respectively the dual function of \( dl \) \((dl = \lambda f . \neg dl(\neg f))\) and the dual lattice of \( TP \).

The following properties result from the application of well-known results about Galois connections.

**Properties:**
\[
\begin{align*}
tp(d_1 \lor d_2) &= tp(d_1) \land tp(d_2) = tp(d_1) \land tp(d_2) \\
\text{dl}(f_1 \lor f_2) &= \text{dl}(d_1) \land \text{dl}(d_2)
\end{align*}
\]

**Definition 4.** Given a time predicate \( g : R_+ \rightarrow \{tt, ff\} \) we say that \( g \) is left-closed if
\[
\forall t_0 \ . \ g(t_0) \implies \exists \epsilon > 0 \ . \ \forall t \leq \epsilon \ . \ \neg g(t_0 + \epsilon) \ (\text{cf figure 3}).
\]
We say that \( g \) is right-closed if in the above definition \( g(t_0 + \epsilon) \) is replaced by \( g(t_0 - \epsilon) \).

![left-closed sets and a non left-closed set](image)

**Fig. 3.** left-closure

**Proposition 5.**

- The image of \( dl \), \( im(dl) \), contains only left-closed deadline conditions i.e., deadline conditions \( d \) such that for all \( q \ . \ \forall t : \neg d(q,t) \) is left-closed.
- The image of \( tp \), \( im(tp) \), contains only right-closed time progress functions i.e., functions \( f \) such that for all \( q \ . \ f(q,t) \) is right-closed.
- \( im(dl) \) and \( im(tp) \) are isomorphic via \( tp \).

**Proof.**
- For all \( f \) and \( q \),
  \[ dl(f)(q) = \forall t > 0 . \neg f(q, t). \]

If \( dl(f)(q) = ff \) then \( \exists t > 0 . f(q, t) \). By additivity we have
  \[ \exists t > 0 . \forall t' \leq t . f(q \triangleright t', t - t'). \]

Consequently:
  \[ \exists t > 0 . \forall t' \leq t . dl(f)(q \triangleright t') = ff. \]

- For all \( d, q \) and \( t \),
  \[ tp(d)(q, t) = \forall t' < t . \neg d(q \triangleright t'). \]

If \( tp(d)(q, t) = ff \) then \( \exists t' 0 \leq t' < t . d(q \triangleright t'). \)
We can write this \( \exists t' 0 < t' \leq t . d(q \triangleright t' - t'). \)
For all \( t'' \) such that \( 0 \leq t'' < t' \) we have
  \[ \exists t_0 = t - t'' . 0 \leq t_0 < t - t'' \land d(q \triangleright t_0) \]
and then \( tp(d)(q, t - t'') = ff. \)
Finally,
  \[ \exists \epsilon < t' . \forall \epsilon' \leq \epsilon . tp(d)(q, t - \epsilon') = ff. \]

\( \lambda t . tp(d)(q, t) \) is right closed.

- Isomorphism of \( \hat{im}(dl) \) and \( \hat{im}(tp) \) via \( tp \) is a direct result from the fact that
  \( tp, dl \) is a Galois connection.

Notice that a consequence of the above propositions is that if \( d \) is left-closed then
\( d = dl \) \( tp(d) \) and if \( f \) is right-closed then \( f = tp \) \( dl(f) \). This means that left-closed
deadline conditions and right-closed time progress functions are in bijection. This
implies that functions which are not right-closed such as \( f(q, t) = q + t < 2 \) for
\( q \triangleright t = q + t \) cannot be obtained as images of deadline conditions. Such functions
can be considered as non well-defined because time can get arbitrarily close to
a bound without reaching it, enforcing the existence of converging infinite time
sequences. It can be shown that if \( f \) is not right-closed then \( tp \) \( dl(f) \) is the right-
closure of \( f \). Dually, deadline conditions that are not left-closed have the same
image via \( tp \) as their left-closure which means that they do not characterize all
the states from which time cannot progress.

3 Translating deadline conditions into progress functions

3.1 Well-defined deadline conditions

The results of the previous section establish some strong correspondence between
deadline conditions and time progress functions. However, in practice, deadline
conditions or equivalently invariants of a hybrid system are obtained as a combi-
nation of deadline conditions of its components. In this section we provide results
for the compositional computation of time progress functions. We investigate the
conditions for the functions $tp$ and $dl$ to be lattice homomorphisms. Then, we provide results for translating modal deadline formulas into progress functions.

To have a lattice homomorphism it is necessary that $tp(d_1 \wedge d_2) = tp(d_1) \sqcap tp(d_2)$. First observe that in general this equality does not hold. Consider the deadline conditions $d_1$ and $d_2$ defined by:

$$d_1 = \bigvee_i p_{2i}, \quad d_2 = \bigvee_i p_{2i+1}, \text{ where } p_i(q) = 1 - 2^{-i} \leq q < 1 - 2^{-i-1}.$$  

We have $d_1 \wedge d_2 = \text{false}$ and consequently, $tp(d_1 \wedge d_2) = \text{false}$. However, $\forall t > 1 \cdot tp(d_1) \sqcap tp(d_2)(0, t) = \text{false}$. Thus, in general, $tp(d_1) \sqcap tp(d_2) \Rightarrow tp(d_1 \wedge d_2)$ and the implication is strict. Notice that this is due to the fact there is an accumulation point of the alternations between deadline conditions which does not allow time progress beyond $t = 1$ (included). In fact, $tp(d_1) \sqcup tp(d_2)(0, 1) = \text{false}$.

We call well-defined the deadline conditions $d$ that are left-closed and such that the function $\lambda t. dl(q \triangleright t)$ changes only a finite number of times in any finite interval. Notice that well-defined deadline conditions are closed under disjunction and conjunction and form a sub-lattice of $DL$.

**Proposition 6.** The restriction $tp$ to well-defined deadline conditions is a homomorphism.

*Proof.* We have trivially $tp(d_1 \lor d_2) = tp(d_1) \sqcup tp(d_2)$ and $tp(d_1) \sqcup tp(d_2) \Rightarrow tp(d_1 \lor d_2)$ by definition of $tp$.

Let us prove that $tp(d_1 \land d_2) \Rightarrow tp(d_1) \sqcap tp(d_2)$ if the finite variability condition holds. Suppose $tp(d_1 \land d_2)(q, t) = \text{true}$:

$$tp(d_1 \land d_2)(q, t) = \forall t' < t \cdot \neg (d_1 \land d_2)(q \triangleright t') = \forall t' < t \cdot \neg (d_1(q \triangleright t') \lor \neg d_2(q \triangleright t'))$$

Since $\lambda t d_1(q \triangleright t)$ and $\lambda t d_2(q \triangleright t)$ have a finite set of points of discontinuity in $[0, t]$ we can divide this interval into a finite set of open subintervals $[t_0, t_1], [t_1, t_2], \ldots, [t_{n-1}, t_n]$ with $t_0 = 0$ and $t_n = t$, such that

$$\forall i < n \cdot (\forall t' \in [t_i, t_{i+1}] \cdot \neg d_1(q \triangleright t')) \lor (\forall t' \in [t_i, t_{i+1}] \cdot \neg d_2(q \triangleright t')).$$

We show that one can find $t'_i$'s such that $t'_0 = 0$, $t'_n = t$ and

$$\forall i < n' \cdot (\forall t' \in [t'_i, t'_{i+1}] \cdot \neg d_1(q \triangleright t')) \lor (\forall t' \in [t'_i, t'_{i+1}] \cdot \neg d_2(q \triangleright t')).$$

As $d_1$ and $d_2$ are left-closed, one can find $t'_i$’s such that

$$(\forall t' \in [t_i, t_{i+1}] \cdot \neg d_1(q \triangleright t')) \lor (\forall t' \in [t_i, t_{i+1}] \cdot \neg d_2(q \triangleright t')).$$

Suppose that for a given $i$ we have $\forall t' \in [t_i, t_{i+1}] \cdot \neg d_1(q \triangleright t')$, and $d_1(q \triangleright t_i) = \text{true}$. Then $d_2(q \triangleright t_i) = \text{false}$, since $tp(d_1 \land d_2)(q, t) = \text{true}$ and $\forall t' \in [t_{i-1}, t_i] \cdot \neg d_2(q \triangleright t')$, since $d_2$ is left-closed, it follows that there exists some $\epsilon$ such that $\forall t' \in [t_i - \epsilon, t_i] \cdot \neg d_2(q \triangleright t')$, and $\forall t' \in [t_i + \epsilon, t_{i+1}] \cdot \neg d_1(q \triangleright t')$. So it is sufficient
to take $t' \equiv t + \epsilon$ instead of $t_i$.
Thus we obtain $\forall i < n . tp(d_1)(q \triangleright t_i, t_{i+1} - t_i) \lor tp(d_2)(q \triangleright t_i, t_{i+1} - t_i)$ which is equivalent to $(tp(d_1) \lor tp(d_2))(q,t)$.

3.2 Translating modal deadline formulas - Application to compositional specification

In this section we present results for the compositional computation of time progress functions when deadline conditions are expressed as modal formulas.

In [SY96] is proposed a variant of timed automata where transitions are labeled with two kinds of conditions: guards (enabling conditions) that characterize states from which transitions can be executed and deadline conditions that characterize states from which transition execution is enforced by stopping time progress. In general, a deadline condition $d$ depends on the corresponding guard $g$. To avoid time deadlocks it is necessary that $d \Rightarrow g$; when $d = g$ the transition is eager and when $d = false$ there is no constraint on time progress.
Timed automata with deadline conditions have been used to show that extending compositionally an untimed (discrete) description into a timed one requires in general the use of modal formulas to express the guards of the composed system in terms of the guards of the components.

Example 4. To illustrate this thesis, consider a discrete (untimed) producer-consumer system with a one-space buffer (figure 4). It is composed of two processes, a producer and a consumer, whose parallel composition is a four state automaton. Suppose that the actions produce, put, get and consume are submitted to timing constraints expressed respectively with guards $g_1 = 2 \leq x \leq 5$, $g_2 = 1 \leq x \leq 2$, $g_3 = 2 \leq y \leq 4$, $g_4 = 1 \leq y \leq 4$, where $x$ and $y$ are clocks used to measure sojourn times at states of each process (reset at transitions of the associated process). There are at least two different practically interesting choices for the guard of the transition 23.

- For $g_{23} \equiv 1 \leq x \leq 2 \land 2 \leq y \leq 4 = g_2 \land g_3$ the actions put and get terminate synchronously by respecting the lower and upper bounds of the guards of the components. It is easy to see that this kind of strong synchronization may be the cause of Zeno behavior [HNSY94] in the composed system even though the components are nonZeno.
- For $g_{23} \equiv (1 \leq x \leq 2 \land 2 \leq y) \lor (2 \leq y \leq 4 \land 1 \leq x)$ a process may wait for his partner. Both lower bounds are respected but only one upper bound. This kind of synchronization with waiting is implicit in timed Petri nets [Sif77, SDLSS94] and can be defined so as to preserve nonZeno by parallel composition. It is easy to see that expressing $g_{23}$ in terms of $g_2$ and $g_3$ requires the use of modal operators: $g_{23}$ is true if one of the two guards has been true and the other is currently true.
We assume that the language of the deadline conditions is defined by the syntax:

\[ d ::= g \mid d \mid false \text{ where,} \]

\[ g ::= true \mid false \mid c \in C \mid g \land g \mid g \lor g \mid \square g \mid \diamond g \mid \Box g \mid \lozenge g. \]

\( C \) is a set of conditions representing atomic guards.

The following definitions express the semantics of this language as a function \( \cdot \) associating with a formula \( d \) a predicate \( \cdot \) on \( Q \) in terms of the meaning of the constants \( false \equiv false, true \equiv true \) and by taking the meaning \( \cdot \) of \( c \) to be well-formed and closed predicates on \( Q \).

\[
\begin{align*}
| g_1 \land g_2 | &= | g_1 | \land | g_2 | \\
| g_1 \lor g_2 | &= | g_1 | \lor | g_2 | \\
\diamond g \mid (q) &= \exists t \geq 0. | g \lim (q \uparrow t) \\
\Box g \mid (q) &= \forall t \geq 0. | g \lim (q \uparrow t) \\
\lozenge g \mid (q) &= \exists t \geq 0. \exists q'. q = q' \Rightarrow t \wedge | g(q') | \\
\Box g \mid (q) &= \forall t \geq 0. \forall q'. q = q' \Rightarrow t \Rightarrow | g(q') | \\
| g \downarrow (q) | &= g(q) \land \exists t \geq 0. \forall t' \leq t. | g \lim (q \uparrow t') |
\end{align*}
\]
Notice that □, ◦, ◊, ◊ correspond to well-known modalities of temporal logic [MP91] meaning respectively always, eventually, always in the past, and once in the past. The operator ↓ is a falling edge operator.

We did not consider negation in order to preserve the property of closeness. However, we use in the sequel negated formulas with the usual meaning. This implies the following relations: 

\[ ¬\text{true} = \text{false}, \quad ¬\text{false} = \text{true}, \]

\[ ¬(g_1 \land g_2) = ¬(g_1) \lor ¬(g_2), \quad ¬(g_1 \lor g_2) = ¬(g_1) \land ¬(g_2), \quad ¬□g = ◦¬g, \]

\[ ¬◊g = □¬g, \quad ¬◊ = ◊¬g \text{ and } ¬◊ = ◊¬g. \]

**Proposition 7.** Any deadline can be expressed as a formula of the following language:

\[ X ::= g \mid (g \lor g) \downarrow (◊g) \downarrow (□g) \downarrow (◊g) \downarrow c \downarrow X \land X \mid X \lor X \]

In order to prove this we will need the following lemma:

**Lemma 8.** For all guards, g, g_1, and g_2, the following relations hold:

\[ \text{true} \downarrow = \text{false} \downarrow = \text{false} \downarrow \]

\[ (g_1 \land g_2) \downarrow = (g_1 \downarrow \land g_2) \lor (g_1 \land g_2 \downarrow) \]

\[ (g_1 \lor g_2) \downarrow = (g_1 \downarrow \land (¬g_2 \lor g_2 \downarrow)) \lor ((¬g_1 \lor g_1 \downarrow) \land g_2 \downarrow) \]

\[ (□g) \downarrow = \text{false} = (◊g) \downarrow. \]

**Proof.** We have \( g \downarrow (q) = g(q) \land \exists t > 0 \land \forall t' 0 < t' \leq t \cdot ¬g(q > t'). \) So it is clear that \( \text{true} \downarrow = \text{false} \downarrow = (□g) \downarrow = (◊g) \downarrow. \)

For the other cases we have:

- \( (g_1 \land g_2) \downarrow \): the falling edges of \( g_1 \land g_2 \) are the falling edges of one of the guards while the other is true. \( (g_1 \land g_2) \downarrow = (g_1 \downarrow \land g_2) \lor (g_1 \land g_2 \downarrow) \).

- \( (g_1 \lor g_2) \downarrow \): the falling edges of \( g_1 \lor g_2 \) are the falling edges common to \( g_1 \) and \( g_2 \) and the falling edges of one guard when the other is false.

\( (g_1 \lor g_2) \downarrow = (g_1 \downarrow \land (¬g_2 \lor g_2 \downarrow)) \lor ((¬g_1 \lor g_1 \downarrow) \land g_2 \downarrow) \).

**Proof of proposition 7:** trivial if the deadline is not of the form \( g \downarrow \); otherwise by induction on the structure of \( g \).

**Theorem 9.** For any deadline formula \( d \), \( tp(d) \) can be expressed as a formula of the following language:

\[ Y ::= ¬g(q) \lor (t = 0) \mid g(q) \lor (t = 0) \mid ¬g(q > t) \lor (t = 0) \mid tp(c) \mid tp(¬c) \mid Y \land Y \mid Y \lor Y \]

**Sketch of proof:** By induction on the structure of \( d \):

- \( tp(d_1 \land d_2) = tp(d_1) \land tp(d_2) \)
- \( tp(d_1 \lor d_2) = tp(d_1) \lor tp(d_2) \)
- \( tp(g) \): by induction on the structure of \( g \):
  - \( tp(\text{true}), tp(\text{false}), tp(c), tp(g_1 \land g_2), tp(g_1 \lor g_2) \) are easily reduced.
• If ◻ g or □ g are false at a state q, they remain false forever. Thus, it is sufficient to test their value at the current state q to know if time can pass: \( tp(◻ g) = \neg ◻ g(q) \lor (t = 0) \) and \( tp(□ g) = \neg □ g(q) \lor (t = 0) \).

• If □ g or ◦ g are true at a state q they remain true forever. Following a similar reasoning as before one can prove:

\[
\begin{align*}
tp(□ g) &= (\neg □ g(q \triangleright t) \lor t = 0) \cup tp(g) \\
&\quad \lor (tp(g) \land \neg ◦ g).
\end{align*}
\]

- \( tp(g_1 \lor g_2) \downarrow \) : from the previous lemma we know that

\[
(g_1 \lor g_2) \downarrow = (g_1 \downarrow \land (\neg g_2 \lor g_2) \downarrow) \lor ((\neg g_1 \lor g_1) \downarrow \lor g_2 \downarrow).
\]

It is easy to check that \( g \lor g \downarrow \) is well-formed if \( g \) is well-formed and closed. Then we can reduce \( tp((g_1 \lor g_2) \downarrow) \) to

\[
[tp(g_1 \downarrow) \lor (tp(g_2) \land tp(g_2) \downarrow)] \lor [tp(g_2 \downarrow) \lor (tp(g_2) \land tp(g_2) \downarrow)].
\]

As \( tp(\neg g) \Rightarrow tp(g \downarrow) \) we obtain:

\[
\begin{align*}
& tp(g_1 \lor g_2) \downarrow = (tp(g_1 \downarrow) \lor tp(g_2) \lor tp(g_2) \downarrow) \land (tp(g_2) \downarrow \lor tp(g_2) \downarrow) \land (tp(g_2) \downarrow \lor tp(g_2) \downarrow).
\end{align*}
\]

- \( tp(\neg g) \) : the following reduction rules can be proven:

\[
\begin{align*}
& tp(\neg(g_1 \lor g_2)) = tp(\neg g_1) \lor tp(\neg g_2), \\
& tp(\neg □ g) = (□ g(q) \lor t = 0, \\
& tp(\neg ◻ g) = ( ◻ g(q \triangleright t) \lor t = 0, \\
& tp(\neg ◦ g) = ( ◦ g(q \triangleright t) \lor t = 0 \land and \\
& tp(\neg ◦ g) = ( ◦ g(q) \lor t = 0.
\end{align*}
\]

- \( tp(c \downarrow) = tp(c) \lor tp(\neg c). \) This equivalence is illustrated for an example in figure 5. Consider \( q, t_1, t_2 \) as in figure 5. We have \( tp(c)(q, t_1) \) and \( tp(\neg c)(q \triangleright t_1, t_2 - t_1). \) Thus, \( tp(c) \lor tp(\neg c)(q, t_2) \) is true as is \( tp(c \downarrow)(q, t_2). \) But for any \( t > 0 \) we have \( tp(c)(q \triangleright t_2, t) \) and \( tp(\neg c)(q \triangleright t_2, t). \) Thus \( tp(c) \lor tp(\neg c)(q, t_2 + t) \) is false as is \( tp(c \downarrow)(q, t_2 + t). \)

- \( tp(( ◻ g) \downarrow) = tp( ◻ g) \lor tp(¬ ◻ g) = (¬ ◻ g(q) \lor t = 0) \lor ( ◻ g(q \triangleright t) \lor t = 0). \)

- \( tp(( □ g) \downarrow) = tp( □ g) \lor tp(¬ □ g) = (¬ □ g(q) \lor t = 0) \lor (□ g(q \triangleright t) \lor t = 0) \)

\[\begin{matrix}
\text{Fig 5. } tp(c \downarrow)
\end{matrix}\]
Example 4 (continued) Consider the consumer-producer example.

- If $g_{23} = g_2 \land g_3$ then one can take the corresponding deadline condition $d_{23}$:
  - either $d_{23} = g_{23}$ (eager transition), in which case,
    $\text{tp}(d_{23}) = \text{tp}(g_2) \lor \text{tp}(g_3)$,
  - or $d_{23} = g_{23} \implies (g_2 \implies g_3) \lor (g_2 \land g_3 \implies)$ (delayable transition) which means that the time progress function is:
    \[
    \text{tp}(d_{23}) = \text{tp}(g_2 \land g_3) \lor (g_2 \land g_3 \implies) \\
    = \text{tp}(g_2 \land g_3) \lor \text{tp}(g_2 \land g_3 \implies) \\
    = (\text{tp}(g_2) \lor \text{tp}(g_3)) \land (\text{tp}(g_2) \lor \text{tp}(g_3)) \\
    = (\text{tp}(g_2) \lor \text{tp}(-g_2) \lor \text{tp}(g_3) \lor (\text{tp}(g_2) \lor \text{tp}(-g_2) \lor \text{tp}(g_3)) \\
    = \text{tp}(g_2) \lor \text{tp}(g_3) \lor (\text{tp}(-g_2) \lor \text{tp}(-g_3))
    \]

- If $g_{23} = (1 \leq x \leq 2 \land 2 \leq y) \lor (2 \leq y \leq 4 \land 1 \leq x) = (g_2 \land g_3) \lor (\Diamond g_2 \land g_3)$ we have the case of synchronization with mutual waiting. One can take as deadline condition $d_{23}$:
  - either $d_{23} = g_{23}$ (eager transition) in which case
    \[
    \text{tp}(d_{23}) = \text{tp}(g_2 \land g_3) \lor (\Diamond g_2 \land g_3) \\
    = \text{tp}(g_2 \land g_3) \lor \text{tp}(\Diamond g_2 \land g_3) \\
    = (\text{tp}(g_2) \lor \text{tp}(\Diamond g_3)) \land (\text{tp}(\Diamond g_2) \lor \text{tp}(g_3)) \\
    = (\text{tp}(g_2) \lor \text{tp}(\Diamond g_2)) \lor (\text{tp}(g_2) \land \text{tp}(g_3)) \\
    = \text{tp}(\Diamond g_2) \lor (\text{tp}(g_2) \land \text{tp}(g_3)) \lor (\text{tp}(\Diamond g_2) \land \text{tp}(\Diamond g_3)) \lor \text{tp}(\Diamond g_3) \\
    = \text{tp}(\Diamond g_2) \lor \text{tp}(\Diamond g_3) \lor \text{tp}(g_2) \lor \text{tp}(g_3)
    \]
    This can be simplified furthermore, by reducing $\text{tp}(\Diamond g_2)$ and $\text{tp}(\Diamond g_3)$.
  - or $d_{23} = g_{23} \implies$ (delayable transition). The reader can verify
    \[
    \text{tp}(d_{23}) = (\text{tp}(\neg g_1) \lor \text{tp}(\neg g_2) \lor \text{tp}(g_1) \lor \text{tp}(\Diamond g_2)) \\
    \land (\text{tp}(\neg g_2) \lor \text{tp}(\neg g_1) \lor \text{tp}(g_2) \lor \text{tp}(\Diamond g_1))
    \]

4 Discussion

The paper studies relationships between progress functions and deadline conditions or invariants used in hybrid systems to specify when continuous evolution can take place. Progress functions are more general and their explicit knowledge is important for analysis and simulation. Deadline conditions or equivalently invariants, are easier to specify as they express constraints on the states without explicitly mentioning time.

The results show that any “reasonable” time progress function can be generated by using deadline conditions or invariants. However, for this correspondence to be a homomorphism, it is necessary to restrict to deadline conditions with finite variability. In this case and under some closeness conditions, it is possible
to compute compositionally progress functions corresponding to deadline conditions that are formulas with conjunction, disjunction and modal operators.

Apart from their theoretical interest, the results can find an application in a framework for the compositional specification of hybrid systems, currently under study.

Acknowledgement: We thank Sergio Yovine and Oded Maler for constructive critiques of the ideas developed in the paper.

References


