COUPLING TEMPORAL CONSTRAINTS WITH REQUIREMENTS ANALYSIS IN CYBER-PHYSICAL SYSTEMS

Hua Wang and Jian Yu

School of Information and Electronic Engineering, Zhejiang University of Science and Technology, China
Email: {wanghua96@126.com}

Abstract

Cyber-Physical System (CPS) requires networks with temporal semantics. The correct functioning of CPS depends crucially on temporal considerations, especially for time-critical requirements. It is an urgent need to couple temporal semantics into the requirement analysis in developing CPS. There have been past attempts at coupling temporal into designing reactive systems. Most of the current proposals, however, suffer from limitations when considered in CPS scenarios by ignoring temporal aspects of the behavior that are essential to the system. This leads naturally to the use of temporal automaton for modeling and verification of requirements of CPSs. The proposed method is simple, yet powerful to express the temporal semantics of CPS. We demonstrate the effectiveness of the proposed model by an engineering example.

Key Words: Cyber-Physical System, Temporal Semantics, Temporal Transition Table, Clock Skeleton.

1. Introduction

Cyber-Physical System (CPS) is a mechanism in which physical components and software systems are tightly intertwined, applied in many important application domains such as medical health, navigation, transportation, energy, and environmental monitoring systems, etc. [1]. Temporal is an important property in modeling CPS. Although quantitative time is not present in today’s computing and networking abstractions, which has had many benefits as ever before, it is eventually opposite effective when reasoning about systems that must communicate with physical components, say CPS. Existing techniques for computing relational abstractions are time-agnostic: the relationship between the state transition of the target system and the elapsed time is absent, such as [2,3]. CPS requires networks with temporal semantics. The correct functioning of CPS depends crucially on temporal considerations, especially for time-critical requirements. Raising time to first class citizen status in modeling and implementations of CPS are found in [4-6]. In essence, CPS is the convergence of computation, communication and control, involving complex interactions between software and physical elements. To this end, CPS calls for being self-adaptive to the variation of environment. Therefore, there is an urgent need to couple temporal semantics into the requirement analysis in developing CPS. From the perspective of the interactive environment, the key features are its perception and control, which is time-sensitive. Endowing CPS with temporal semantics can reason about temporal influence on the evolution of CPS.

There have been past attempts at coupling temporal into designing reactive systems. An approach that integrates temporal models into functional analysis with regards to closed-loop properties of control software was proposed in [7]. Ontologies are employed to reason about the temporal properties of CPS and temporal semantics were discussed in [8]. Derler and her colleagues considered to leverage clock synchronization in order to develop an entire distributed system as a synchronous system with one logical clock domain using PTIDES, an event-triggered programming model [9]. Temporal definition language, such as [10-12], provides a potential method to describe the temporal semantics at language-level. The work [13] allows for verification and control from specifications given as Linear Temporal Logic (LTL) formulae over linear predicates in the state variables. Researchers argued that temporal is not just a quality factor, but also a correctness one; particularly, the temporal semantics in models and programs must be preserved during compilation [14]. This leads naturally to the use of temporal automaton for modeling and verification of requirements of CPSs. An optimization approach was proposed with end-to-end response time constraints in a multi-resource CPS [15]. Time-aware relational abstraction overcomes shortcomings of relational abstraction procedure based on exploiting the eigenstructure of the matrix of the linear ordinary differential equations [16].

Most of the current proposals, however, suffer from limitations when considered in CPS scenarios by ignoring aspects of the behavior that are essential to the system. Better specification and verification of temporal requirements are desired to decide whether the temporal constraints for a particular adaptation hold. As a result, an automata model is employed to consider the behavior of
CPS over time to tackle the above predicament. The proposed method is simple, yet powerful to express the temporal semantics of CPS. The rich support for specification and verification of automaton theory paves a way to reconcile system behaviour with the requirements model of CPS. We demonstrate the usefulness of the proposed model by an engineering example.

2. The temporal automata

1.1 Modeling temporal

**Definition 1.** A Transition Table is a quadruple \( M = (K, \Sigma, S_0, \Delta) \), where \( K \) is a finite set of automaton states, \( \Sigma \) is an input alphabet table, \( S_0 \subseteq K \) is a set of initial states, and \( \Delta \subseteq K \times K \times \Sigma \) is a set of transitions. The automaton starts in an initial state, and if \( (s, s', \sigma) \in \Delta \) then the automaton can change its state from \( s \) to \( s' \) reading the input symbol \( \sigma \). A run of \( M \) over \( \sigma \) is defined as

\[
r : s_0 \otimes \sigma_1 \rightarrow s_1 \otimes \sigma_2 \rightarrow \cdots
\]

for a word \( \sigma = \sigma_1 \sigma_2 \cdots \) over the alphabet \( \Sigma \), provided \( s_0 \in S_0 \) and \( (s_{i-1}, s, \sigma_i) \in \Delta \) for all \( i \geq 1 \). The set \( \text{inf}(r) \) consists of the states \( s \in K \) such that \( s = s_i \) for infinitely many \( i \geq 0 \).

For an additional set \( F \subseteq K \) of final states, a run \( r \) of \( M \) over a word \( \sigma \in \Sigma^* \) is an acceptable run if \( \text{inf}(r) \cap F \neq \emptyset \).

Temporal words are defined by assigning a real-valued time with each symbol in a word so that a behavior of a CPS system corresponds to a temporal word over the alphabet of even ts. A word \( a \) is coupled with a temporal sequence \( \tau \) defined as below:

**Definition 2.** A Temporal Sequence \( \tau = \tau_1 \tau_2 \cdots \) is an infinite sequence of time values \( \tau_i \in \mathbb{R} \). The set of nonnegative real numbers, \( \mathbb{R}_+ \), is the time domain, satisfying the following constraints:

1. \( \tau_i < \tau_{i+1} \), i.e., for all \( i \geq 1 \), \( \tau \) is increases strictly monotonically.
2. For each \( \tau \in \mathbb{R}_+ \), there is some \( i \geq 1 \) such that \( \tau_j > \tau \).

**Definition 3.** A Temporal Event over an alphabet \( \Sigma \) is an ordered pair \( (\sigma, \tau) \) where \( \sigma = \sigma_1 \sigma_2 \cdots \) is an infinite event over \( \Sigma \) and \( \tau \) is a temporal sequence, i.e., the time at which it occurs. A temporal language over \( \Sigma \) is a set of temporal words over \( \Sigma \).

A temporal event \( (\sigma, \tau) \) is treated as an input to an automaton and presents the symbol \( \sigma_i \) at time \( \tau_i \). In our context, each symbol \( \sigma_i \) is interpreted to indicate an event occurrence and the corresponding \( \tau_i \) is explained as the time of occurrence of \( \sigma_i \).

Especially, one could allow a system to assign the same time value to quite a little successive events in the sequence. One hereto could ameliorate slightly the meaning of temporal word to require a temporal sequence to increase only monotonically, i.e., demand for all \( i \geq 1 \), \( \tau_i \leq \tau_{i+1} \). And all conclusions is also applicable in the alternative model. The logic language -level operations such as union, intersection, and complementation could be applied for temporal language as usual.

**Definition 4.** For a temporal language \( L \) over \( \Sigma \),\( Untemporal (L) \) is the \( \omega - \) language consisting of \( \sigma \in \Sigma^* \) such that \( (\sigma, \tau) \in L \) for some temporal sequence \( \tau \).

Next, we can bring time constraints into the transition tables in order to read temporal words as a natural extension of untemporal transition tables. The state transition relies on the input symbol read according to the transition tables as indicated above. When considering time constraints, we want the transition to be built also on the time of the event occurrence bound up with the times of the preceding event occurrence. Therefore, a real-numbered clock is coupled with each state-transition entry in the transition table. Thus, the state transition is allowed only if the current clock value satisfies the constraint while presenting the desired input symbol, i.e., the expected event occurrence. Zero can be assigned to a clock accompanied by any state transition. Consider the following example for the Temporal Transition Table in Fig. 1.

![Fig. 1: Example of a temporal transition table](image)

The start state is \( s_0 \). There are two clocks \( ckg \) and \( \alpha ckg \). The edge connecting \( s_0 \) to \( s_1 \) is annotated with the form “\( \alpha ckg \) = 0” that represents reading the symbol \( \alpha \) and resetting the clock \( ckg \). Similar considerations apply to the other clock \( ckg \), which is reset when the automaton reads symbol \( b \) and the current state is \( s_1 \). The annotation of the check \( (ckg < 2) \) associated with the self-transition \( s_1 \), requiring an input symbol \( c \) ensures that the delay between \( c \) and the preceding \( \alpha \) is less than 2. There is a worth noting that no explicit bounds are placed on the time difference between \( b \) and the preceding \( \alpha \), or \( d \) and the preceding \( c \). This presents a crucial benefit of setting multiple clocks independently of each other. The temporal transition table in Fig. 1 accepts the language
\[ L_1 = \{((abcd)^n, \tau) \mid \forall i.((\tau_{4i+3} < \tau_{4i+1} + 2) \land (\tau_{4i+4} > \tau_{4i+2} + 3))\} \]

Also note that the clocks \( c_k \) and \( c_l \) are used independently; therefore, the language \( L_1 \) is the intersection of the two languages \( L_x \) and \( L_y \), each of which can be expressed by an automaton employing just one clock, defined as follows.

\[ L^*_1 = \{((abcd)^n, \tau) \mid \forall i.((\tau_{4i+3} < \tau_{4i+1} + 2))\} \]

\[ L^*_2 = \{((abcd)^n, \tau) \mid \forall i.((\tau_{4i+4} > \tau_{4i+2} + 3))\} \]

### 1.1 Temporal Transition Table

We are now in a position to define the Temporal Transition Table. Notice how, in the transition, temporal constraints could be applied into the definition. Before diving into how this is done, we need to make a distinction between those times at which events are available and those at which they can become ready to be available. In this sense, two ways are needed to record event that might occur: when a process can communicate an event, we use \( a \) to indicate it at any time for \( a \in \Sigma \) while use the special notation \( a^* \) to specify an event on the moment when it become available. Thus, we have the notation \( \Sigma = \Sigma \cup \{a \in \Sigma\} \) and \( \tilde{a} \) is a typical member in \( \Sigma \). The sequence of observable events in a process is defined as \( \text{trace} \). For instance, pressing the brake pedal in an automobile electronics system or opening automatically the gate at a railroad crossing could be identified as an event. Now we have the definition of temporal traces.

**Definition 5.** All temporal events in set \( \Sigma = A \cdot \Sigma \), where \( A \in [0, \infty) \) is the global clock skeleton. The set of all \textbf{temporal traces} is

\[ \Sigma^\infty = \{s \in \Sigma^* \mid \text{if} \ (\tilde{e}, \tau) \text{ precedes} (\tilde{e}', \tau') \text{ in} \ s \text{, then} \ \tau \leq \tau'\} \]

For \( s \in \Sigma^\infty \), interleaving \( s \) and \( \kappa \) is defined to be the set of all temporal traces in \( \Sigma^\infty \). We note the set of all stable values as \( \beta \) and have

\[ \beta_{stab} = \{s \in (\Sigma^\infty)_{\omega} \mid \exists e \in H(s, e) \in K \} \text{, for} \ K \in (\Sigma^\infty)_{\omega} \text{, and} \]

\[ H = [0, \infty) \cup \{\infty\} \].

**Definition 6.** All temporal events in temporal transition table can be formulated by two types of events, \textit{basic} temporal event and \textit{composite} temporal event. Basic temporal events are an atomic observable occurrence, while the composite temporal event is logical operation of basic or other composite events using the following operators extended from our previous work by coupling the temporal semantics as in Table 1:

**Table 1:** Temporal Event Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Predicate</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intersection</td>
<td>( \text{AND}(e_1 \mid \tau_1, e_2 \mid \tau_2, ...) )</td>
<td>The intersection operation of temporal events</td>
</tr>
<tr>
<td>Union</td>
<td>( \text{OR}(e_1 \mid \tau_1, e_2 \mid \tau_2, ...) )</td>
<td>The union operation of temporal events</td>
</tr>
<tr>
<td>Exclusive OR</td>
<td>( \text{XOR}(e_1 \mid \tau_1, e_2 \mid \tau_2, ...) )</td>
<td>Mutually exclusive temporal events</td>
</tr>
<tr>
<td>Sequence</td>
<td>( \text{SEQ}(e_1 \mid \tau_1, e_2 \mid \tau_2, ...) )</td>
<td>Events occurrence in accordance with a certain sequence</td>
</tr>
<tr>
<td>Parallel</td>
<td>( \text{SEQ}(e_1 \mid \tau_1, e_2 \mid \tau_2, ...) )</td>
<td>Composite running behavior of all the concurrent processes</td>
</tr>
<tr>
<td>Negative</td>
<td>( \text{NEG}(e \mid \tau) )</td>
<td>The opposite of an temporal event</td>
</tr>
<tr>
<td>Quantification</td>
<td>( \text{ANY}(n, e \mid \tau) )</td>
<td>The repeated occurrence of ( n ) times</td>
</tr>
</tbody>
</table>

**Definition 7.** A **Temporal Transition Table** is a tuple \( M_{\Delta} = (\Sigma, S, C, A, \Delta) \), where

- \( \Delta \) is a finite set of automaton states,
- \( \Sigma \) is an input alphabet table,
- \( S \subseteq \Sigma \) is a set of initial states,
- \( C \subseteq \Sigma \) is a finite set of clocks, and
- \( \Delta \subseteq \{\Sigma \times \Sigma \times 2^\Delta \times \Phi(C)\} \) is a set of transitions.

The transition set \( \Delta \) adds temporal semantics indicating the automaton can alter its state from \( s \) to \( s' \) while reading the input symbols from \( a \) if satisfying \((\Sigma \times \Sigma) \times \Phi(C) \epsilon \Delta\), where \( \Phi \subseteq C \) and \( \ell \) is a temporal constraint over \( C \). The set \( C \), which is necessary in a temporal transition table, records each clock at time \( \tau \), when an event is observed. Once started, the temporal automation initializes all clocks to zero. The values of all clocks change as time goes to reflect the passage time accompanying the state transition. A temporal event \( (\tilde{e}, \tau) \) records the time at which it occur \( s \). For the purpose of retrofitting CPS with temporal properties, a globally consistent virtual clock skeleton is invented to ensure all clocks advance at the identical rate. Taking some transition of the form \((\Sigma \times \Sigma \times \Phi(C) \epsilon \Delta\), the system changes state from \( s \) to \( s' \) on the input value \( a \), i.e., the desired event occurrence if current clock values meet the temporal constraint \( \ell \). It is considerable that a temporal transition table \( M_{\Delta} \) could have the corresponding a transition table \( M \) by setting the clock set to be empty set and substituting the transition \((s, s', a, \emptyset, \text{true}) \) for \((s, s', a)\).

**2.3 Verification of implementation**

Assume without loss of generality that all temporal events are considered to be instantaneous and asked for the cooperation while communicating. The earliest and latest times of any temporal event are noted as \( \text{begin} (\tilde{e}) \) and \( \text{end} (\tilde{e}) \) respectively. We consider a classical modeling
technique Communication Sequential Processes (CSP) proposed in [18], which is employed to prove correctness of the temporal automaton. The verification of correctness comes down to an inclusion problem [19], namely whether the implementation $I$ meets the specification $S$, noted as $I \subseteq S$. The implementation is given as temporal processes and could be considered as composition of several components, each of which is treated as a temporal process $P_i$. At the same time, the specification of the CPS is described as a temporal language $L$. Now the verification problem is posed as the result of $(|| P_i ||_L) \subseteq L$ is true or false. Temporal processes are considered as a set of temporal traces. As a first step, we should find a way to extend untimed processes to temporal processes as illustrated in Fig. 2.

![Algorithm 1: GetTemporalProcess()](image)

**Fig. 2**: Algorithm for getting temporal processes.

We now show how to check the inclusion of the temporal traces of $(|| P ||)$ in $L$. For this purpose, a region automaton [20] is employed to correspond the product of temporal transition tables. The algorithm **CheckInclusion()** constructs the region automaton to reason about the correctness of the model as shown in Fig. 3. The return value flag shows the desired correctness holds if flag is true, vice versa.

![Algorithm 2: CheckInclusion()](image)

**Fig. 3**: Algorithm for reasoning about the correctness.

3. **Case study**

3.1 **Motivating scenario**

To validate the effectiveness of our approach, we show an actual project while employing the proposed approach. We consider an example of a motivating industry project in our previous work [17]. A Pervasive Health Management Director (PHmD) system for providing real-time personal health evaluations based on the medical health knowledge database by analyzing health data collected from remote wireless health devices, such as JOGGER Monitor, Electrocardiogram Measurement Instrument, Blood-glucose Meter and so on. Health service providers supply the health evaluation service based on QoS (Quality of Service) requirements subscribed by different customers who receive relevant services to monitor and improve their health level.

The health management and service system is deployed in Cyber-physical System based on cloud computing environment. The physical information collected from remote devices or wearable textiles is sent to PHmD system by means of wireless sensor networking technology. The expert system in PHmD is employed to analyze and process the individual physiological information applying underlying health knowledge warehouse. Then the risk evaluation is performed to know about the health level of customer. Eventually, home monitor, disease prevention, emergency rescue and remote diagnosis would bear the promising fruits cooperatively.

3.2 **Modeling the PHmD**

To illustrate how the proposed formalism described above works in practice, we use Jogging Monitor to specify the Wearable Device communicating with the Health APP: hosted in wireless cellphones. The data of JOGGER, Body Temperature, Respiratory Frequency and Oxygen Consumption are fetched from the wearable device in every 20 seconds span. Then Vital Signs date are conveyed to the hosting App. If there is any exception after analysis, it will be sent to the jogger. The jogger will act on the results of the analysis report. We consider a simple situation where the APP will alarm when the JOGGER of continue to exceed a threshold value for a duration of 30 seconds. The situation is composed of two components: JOGGER AND APP. The jogger is modeled as an automaton shown in Fig. 4.

![Fig. 4: The component JOGGER](image)
The event set is \(\{\text{approach}, \text{keep}, \text{leave}, E_{\text{ckbp}}\}\). We use \(E_{\text{ckbp}}\) to express the temporal requirements. JOGGER communicates with APP with two events approach and leave. The event keep denotes the event of the increase of the blood pressure above the threshold value and keep for a period of time. JOGGER is required to send the signal keep if it keeps the above-threshold-value for at least 30s. The event leave would be sent if the blood pressure falls within the range of the normal values.

\[
\text{Fig. 5: The component APP}
\]

The component APP is modeled as shown in Fig 5. The event set is \(\{\text{approach}, \text{alarm}, \text{recover}, E_{\text{ckapp}}\}\). We use \(E_{\text{ckapp}}\) to express the temporal requirements. When it receives the signal keep from JOGGER, the component APP replies by sending the signal alarm to JOGGER for a duration of 10s. When it receives the signal leave from JOGGER, the component APP replies with the signal recover to JOGGER within 2s. The whole system is then

\[
[\text{JOGGER } || \text{ APP}]
\]

The union of the event sets of both components is the event set of the entire system. The automaton of the system is built by composing the two components. The correctness requirements of the system is the safety property: when the JOGGER has the blood pressure above the threshold value, it should be alarmed when meeting the temporal constraints, as specified in Fig 6. The event label “recover, ~keep” stands for any event set not containing keep, but containing recover. The safety property implies alarm after keep. The safety property could be checked by employing the algorithm Check Inclusion().

\[
\text{Fig. 6: The safety property}
\]

4. Conclusion and future works

A temporal automata is proposed based on basic automata theory by adding the formal semantics to the definition of timing requirements in CPS. Furthermore, the verification method of implementation of requirements of CPS is put forward according to its specification using communicating sequential processes. Contribution includes: the temporal traces are used to record the event occurrence under the uniform clock skeleton work; temporal event operators provide a way to describe possible composition when needed; verification algorithm is simple but efficient employing region automata.

Future works include the scale of verification model should be controlled. In addition, technology would be borrowed to how to reduce the temporal transition entries in order to scale to complicated application environment. Also, we plan to increase the quantitative relationship at the clock level. Finally, automated tool support of modeling process is under development.

References

[8] Petnga L, Austin M, Ontologies of time and time-based reasoning for MBSE of cyber-physical systems [J], *Procedia Computer Science*, 16(0), 2013, 403-412.


