1 Introduction

Constantly increasing complexity of computer systems poses big challenges for designers. Extensive testing has proven ineffective and inappropriate [8]. Model Checking [9] and HOL [10] have been proposed and successfully used in the designing process especially in problem areas as real-time, safety-critical and reactive systems.

There are several other formal frameworks available, like CSP [11] for example, but the choice of ITL as a universal specification language was due to the following reasons; it is simple, flexible and has an executable subset giving the basis for both formal proof of the validity of the system design as well as simulation, animation and rapid prototyping in TEMPUR A [3, 7].

The paper presents a unified model of a computer system providing a specification language for formal co-design and shows that with a proper interpretation and definition of events, ITL can be successfully used as an event-reasoning notation, in spite of being a state-based and discrete formalism.

2 Projection and compositionality in ITL

This section gives a very short introduction to ITL’s capabilities of reasoning about compositionality and projected time. For more details we will forward the reader to [1, 2, 3, 7].

Compositional reasoning A system Sys can be modelled by several formulae of the form:

\[(\ddagger) \quad w \land A s \land Sys \Rightarrow Co \land fin w\]

where \(w\) and \(w'\) are non-temporal ITL state formulas while \(A s\) and \(Sys\) are Co arbitrary formulas (possibly) having temporal constructs. The formula \(As\) is often called an assumption and Co a commitment. The informal meaning is that if \(w\) describes the first state of an interval \(\sigma\) where \(A s\) and \(Sys\) are active, then \(Co\) also holds and \(w'\) describes interval’s final state.

Projected time In [2] Moszkowski gives a simple extension of ITL capable of reasoning about concurrent systems with different time granularities. Here we will give the syntax and the semantics of the \(\forall\) projection operator in ITL.

\[
\mathcal{M}_0 \llbracket \forall f_i \forall f_2 \rrbracket = \text{tt} \iff \\
\text{for some } n \geq 0, \sigma' \text{ and } l_0, \ldots, l_n:\n\quad 0 = l_0 < \ldots < l_n = |\sigma'|, \\
\quad \text{and for each } i < n, \mathcal{M}_{\sigma_{l_i: l_{i+1}}} \llbracket f_i \rrbracket = \text{true}, \\
\quad \text{then } \mathcal{M}_{\sigma'} \llbracket f_2 \rrbracket = \text{true},
\]

where \(\sigma' = n\) and for all \(i \leq n, \sigma_i' = \sigma_i\).

In other words \(\mathcal{M}_{\sigma_{l_i: l_{i+1}}} \llbracket f_i \rrbracket \Rightarrow \mathcal{M}_{\sigma'} \llbracket f_2 \rrbracket\), or informally, if \(f_1\) holds in the subintervals bounded by the states of \(\sigma'\), then \(f_2\) holds in \(\sigma\). For an extensive reading on \(\forall\) we will refer to [2, 3]

3 Event reasoning in ITL

We take the view that a system has a very tight relation and communicates with its environment through a very well defined interface.

Signal and Event Let \(T\) be linear time domain with smallest element \(t_0 = 0\). \(\mathcal{N}\) be the set of non-negative integers. \(\mathcal{N}^*\) be the set of positive integers and \(\mathcal{V}\) be a set of values. Consider a system Sys which has a number of inputs and a number of outputs (signals).

Definition 1 Let \(s : T \rightarrow \mathcal{V} \times T\), \(\tau : \mathcal{V} \times T \rightarrow T\) and \(\nu : \mathcal{V} \times T \rightarrow \mathcal{V}\) be three total functions, \(t, t_1, t_2 \in T\) and \(v \in \mathcal{V}\). If axioms 1, 2 and 3 hold, then we will call \(s\) a signal, \(\tau — \text{the timestamp of } s\) and \(\nu — \text{the value of } s\).

Axiom 1 \(s(t_1) = (\nu(s(t_1)), \tau(s(t_1)))\).

Axiom 2 Let

\[
\text{stable}(s, t_1, t_2) = \exists \varepsilon > 0 \cdot (\exists \delta \in [t_1, t_2] \cdot t \in T) \land \\
(\forall t \in [t_1, t_2] \cdot \varepsilon \cdot t \in T \Rightarrow \tau(s(t)) = t + 1). \quad \tau(s(t_1)) = \begin{cases} 
\{ t_1, \text{ where stable}(s, t_1, t_1) \}. 
\end{cases}
\]
Axiom 3 \( \tau(s(t_1)) = \tau(s(t_2)) \Rightarrow s(t_1) = s(t_2) \).

Let \( J = \{ i_j : 0 \leq j < n, n \in \mathbb{N}^+ \} \) be a non-empty set of input signals and \( O = \{ o_j : 0 \leq j < m, m \in \mathbb{N} \} \) be a (possibly empty) set of output signals and \( J \cap O = \emptyset \). \( J \) and \( O \) form the interface between the system and its environment. It is clear that definition 1 is too general and does not comply with the final variability principle, so we will restrict it a bit. Let \( P^{\text{max}} \geq P^{\text{min}} > 0 \) for which \( \forall t \in T \cdot i + \min T + P^{\text{min}} \in T \land i + P^{\text{max}} \in T \) hold and let us denote
\[
\text{nwe}(i, t) = \forall t' \in [i, i + P^{\text{max}}) \cdot t' \in T \Rightarrow \text{nwe}(i, t) = i(t) \]

and pronounce “\( i \) changes at time \( t \)”.

Note here that all signals change at time \( t_0 = 0 \), i.e. initialisation occurs at \( t_0 \).

Definition 2 Let \( i \in J \) be a signal and \( t' \in T \) such that \( \text{nwe}(i, t) \). If

1. \( \forall t \in T \cdot t' \leq t + P^{\text{min}} \Rightarrow i(t) = i(t') \) and
2. \( \exists t \in T \cdot t' \leq t + P^{\text{max}} \Rightarrow i(t) \neq i(t') \).

then we call \( i \) a sporadic signal. \( P^{\text{min}} \) and \( P^{\text{max}} \) will be called minimum and maximum periods of \( i \), correspondingly.

Definition 3 Let \( i \in J \) be a sporadic signal, and let \( P^{\text{min}} \) and \( P^{\text{max}} \) be its periods. If \( P^{\text{min}} = P^{\text{max}} = T \), then \( i \) is called a periodic signal with period \( T \).

Axiom 4 All input signals of Sys are sporadic.

Such an axiom guarantees the validity of the final variability assumption [6].

Definition 4 Let \( t \in T \) be a time instant. An input action at time \( t \) is a nonempty subset
\[
I \subseteq \{ i(t) : i \in J \text{ and } \text{nwe}(i, t) \}.
\]

and an output action at time \( t \) is a nonempty subset
\[
O \subseteq \{ o(t) : o \in O \text{ and } \text{nwe}(o, t) \}.
\]

Definition 5 Let \( t \in T \) be a time instant. We say that event \( E \) occurs at time \( t \) if \( E \) is the maximal\(^1\) action at time \( t \).

Periodic input signals Obviously, even if all system’s signals are periodic we may still hit a very sporadic picture. Consider Sys with input interface \( J = \{ p_0, p_1 \} \) where \( p_0 \) and \( p_1 \) are periodic and \( T_{p_0} = \pi \) and \( T_{p_1} = 1 \). In figure 1(a) we have shown a time axis and the moments in time at which \( p_0 \) and \( p_1 \) occur, while in figure 1(b) we have “rolled” the time axis in a circle \( c \) with length \( T_{p_0} \) and all moments in time when \( p_0 \) occurs have been projected into point \( A \).

\(^1\)Maximal in terms of \( c \).

It is not very difficult to show that, if signals’ periods are not relatively rational, the projection of all moments in time when \( p_1 \) occurs is the whole \( c \).

Let us now assume the only realistic scenario when computation does take time, i.e. the system reacts to the environment within certain amount of time. As we saw from the above, there will be many cases when the system tries to react to the event \( \{ p_0 \} \) and the next event \( \{ p_1 \} \) occurs before the computation finishes, i.e. the system misses a deadline.

Now, it is not very difficult to see that if the periods of the signals are relatively rational numbers, then there will be a moment \( x > 0 \) when all signals occur again and we can even define the smallest \( x \) as the least common multiple of the periods of the signals. If we denote the set of the signal’s periods as \( \Omega \), i.e. \( \Omega = \{ T_{p_i} : p_i \in J \} \) then \( x = \text{LCM}(\Omega) \).

On other hand, events that take place in \( [0, \text{LCM}(\Omega)] \) will take place in every interval \( \lceil n \text{LCM}(\Omega) \rceil \) in other words, as far as the events, alone, are concerned, all intervals of this kind are equivalent. This allows us to use these moments in time when a signal changes as special time points which will give us a “time grid” and the required discretisation of the (eventually) continuous time.

(pseudo) Sporadic input signals As we saw in the previous section, periodic signals with appropriate periods have nice discrete time behaviour. Unfortunately, this nice picture breaks when sporadic signals, as they were defined in definition 2, are considered. On figure 2 we have given a small example illustrating the problem.

Suppose we have a system with two input signals; a periodic signal \( p \) with period \( T_p \) and a sporadic one \( s \).
with periods $P_{s}^{min}$ and $P_{s}^{max}$ correspondingly. According to definition 2, $s$ may change its value at any arbitrary moment in the closed interval denoted with $\Delta t$ on the figure. Obviously this kind of possible behaviour can break any time discretisation fixed in advance and this is simply because definition 2 allows $s$ to change in any point at the interval $\Delta t$.

One way of avoiding the problem is to restrict the sporadic signals and to allow them to change at time points that are part of the time grid defined by the periodic signals of the system $Sys$. The following definition formalises the approach.

**Definition 6** Let $t_{0} \in T$ be a moment of occurrence (change) of the sporadic signal $s \in S$, $P_{s}^{min}, P_{s}^{max}$ be its periods, $DT$ be the set of time instances when a periodic signal occurs and

$$ST_{s}(t_{0}) = \{[t_{0} + P_{s}^{min}, t_{0} + P_{s}^{max}] \cap DT\} \cup \{t \in DT : \mu(t \geq t_{0} + P_{s}^{max})\}.$$  

If $\forall t \in [t_{0} + P_{s}^{min}, P_{s}^{max}) \land \text{new}(s, t) \Rightarrow t \in ST_{s}(t_{0})$, then $s$ will be called a *pseudo sporadic signal*.

### 4 Compositionality and projection

In this section we give a unified and compositional ITL framework for modelling systems interacting with environment through events. The compositional properties, this model has, appear to be very effective when we consider the formal validity of complex systems with many co-operating submodules.

Suppose a system $Sys$ that has a number of periodic input signals $S \neq \emptyset$ whose periods are relatively rational numbers and a number of pseudo sporadic input signals. As it was shown in [1], system’s behaviour can be expressed easily using the assumption — commitment strategy. In our framework it can be specified by

$$e_{i}(\bar{U}) \land As(\bar{U}) \land$$

$$\{\exists \bar{I}_{\text{old}}, \bar{O}_{\text{old}}, \bar{O}_{\text{new}} \land (\ddagger) \}$$

$$\{ \bar{I}_{\text{old}} = \bar{I} \land \bar{O}_{\text{old}} = \bar{O} \land \text{Sys}(\bar{I}, \bar{O}_{\text{new}}) \land$$

$$\bar{O}_{\text{new}} \leftarrow \bar{O}_{\text{new}} \land \text{len}(\bar{I}) \} \Rightarrow Co$$

where

- $e_{i}$ is a “state” ITL formulae, i.e. it is a non-temporal formula over the variables modelling the input signals of the system and it models one possible input event. Note here that if $e_{i}$ is false, then the system is not executed, i.e. it stays as a guard for the “execution” of the formula behind it.
- $As(\bar{U})$ is a temporal formula representing an assumption over the input signals such as stability over the whole interval when the system is active.

- $Sys(\bar{I}, \bar{O}_{\text{new}})$ is a temporal formula describing the behaviour of the system.
- $\text{len}(\bar{I})$ gives the number of states between the current and the next input event.
- $Co$ is a temporal formula giving an on-going property of the system such as that of a scheduler.
- $e_{o}$ is a non-temporal formula describing the expected output event which will be issued when the next input event occurs.

The informal semantics of $(\ddagger)$, which we can call a component, is as follows: if an input event $e_{i}$ happens and the assumption $As$ holds for the input variables and the system $Sys$ is operating over the interval defined by the $\sqcap$ operator, then the output event $e_{o}$ will be issued at the end of the interval and the property $Co$ will hold, for example — the order of the execution of the submodules of the system $Sys$.

We would like to outline the differences between $(†)$ and $(‡)$. As we can see, $w$ from $(†)$ corresponds to $e_{i}(\bar{U})$ though it describes only the “input” part of the state while $w$ specifies the state fully. Obviously $As$ and $Co$ do not differ much from their counterparts, i.e. $As(\bar{U})$ and $Co$.

We can look at

$$\{ \exists \bar{I}_{\text{old}}, \bar{O}_{\text{old}}, \bar{O}_{\text{new}} \land$$

$$[ \bar{I}_{\text{old}} = \bar{I} \land \bar{O}_{\text{old}} = \bar{O} \land \text{Sys}(\bar{I}, \bar{O}_{\text{new}}) \land$$

$$\bar{O}_{\text{new}} \leftarrow \bar{O}_{\text{new}} \land \text{len}(\bar{I}) \} \land$$

$$\forall \square e_{o}(\bar{I}_{\text{old}}, \bar{O}_{\text{old}}, \bar{O}_{\text{new}}) \Rightarrow Co$$

in $(‡)$ as a refinement of $Sys$ from $(†)$ implementing the idea of buffering the inputs and the outputs.

The significant change is the absence of $w$ in $(‡)$ and the reason for this is that, having events, we cannot describe the final state fully; it simply depends heavily on the environment. Instead, we have only provided the “output” part of the next state with $e_{o}(\bar{I}_{\text{old}}, \bar{O}_{\text{old}}, \bar{O}_{\text{new}})$.

**The Flip-Flop example** A simple example which illustrates the use of the proposed approach is the so called SR-Flip-Flop. It is a small hardware system described in [4, 5]. In figure 3, we have shown the structure of the device, i.e. two nand-gates with feedback from the output of each of them to one of the inputs of the other. The full functional behaviour is given by the table in figure 4. Suppose the $nand$-gates are represented as $nand(I_{0}, I_{1}, o) \equiv o := \neg(I_{0} \land I_{1})$. Then we can describe the flip-flop as

$$true \land \text{padded}(\bar{U}) \land$$

$$\{ \exists \bar{I}_{\text{old}}, \bar{O}_{\text{old}}, \bar{O}_{\text{new}} \land$$

$$[ \bar{I}_{\text{old}} = \bar{I} \land \bar{O}_{\text{old}} = \bar{O} \land \text{Sys}(\bar{I}, \bar{O}_{\text{new}}) \land$$

$$\bar{O}_{\text{new}} \leftarrow \bar{O}_{\text{new}} \land \text{len}(\bar{I}) \} \land$$

$$\forall \square e_{o}(\bar{I}_{\text{old}}, \bar{O}_{\text{old}}, \bar{O}_{\text{new}}) \Rightarrow true$$
Figure 3: SR Flip-Flop

![SR Flip-Flop diagram]

<table>
<thead>
<tr>
<th>$i_0$</th>
<th>$i_1$</th>
<th>$o_0$</th>
<th>$o_1$</th>
<th>$o_{0_{\text{new}}}$</th>
<th>$o_{1_{\text{new}}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>·</td>
<td>·</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>·</td>
<td>·</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>·</td>
<td>·</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

*a* could be anything, don’t care

Figure 4: The Flip-Flop function

where

$$\text{Sys}(\tau, \pi) \equiv [\text{nand}(i_0, o_1, o_0) \land \text{nand}(o_0, i_1, o_1)]^*$$

and $f_{ff}$ is the “flip-flop” function which represents the functionality of the flip-flop, i.e., it is a non-temporal syntactical object and, in this case, can be described by the table in figure 4.

5 Conclusions and future work

We gave a unified and simple notion of event and showed how ITL can be used for handling reactive and concurrent systems. Our current work is on an automated system based on PVS for specifying and proving formulas like ($\ddagger$) which will be used for compositional design and refinement of many case studies.

Based on previous research on refinement [12, 13] we aim to develop a set of refinement rules to explore the $V$ projection operator as a major vehicle for refinement. We also aim to incorporate the compositional proof rules given in [1] into the current framework.

Acknowledgements Many thanks must go to Antonio Cau and Hussein Zedan for discussions and advice.

References


