Timed Transition Activation Semantics in Statecharts

Gregory W. Bond
AT&T Labs—Research, Florham Park, NJ 07932, USA

Abstract. We propose an intuitive semantics for timed transition activation in hierarchical state machine languages like UML Statecharts or Harel Statecharts. The semantics presented here explain precisely when the timer implicitly associated with a timed transition is activated, and possibly deactivated, during a machine’s execution. In addition to the semantics, a decision procedure is given which serves to determine if a timed transition should be activated. These semantics serve as a necessary contribution towards the goal of formalizing Statecharts languages, and towards the ultimate goal of performing quantitative, temporal analysis of unrestricted Statecharts models.

1 Introduction

We propose an intuitive semantics for timed transition activation in hierarchical state machine languages like UML Statecharts [2], Harel Statecharts [3], or ECharts [1]. The semantics presented here explain precisely when the timer implicitly associated with a timed transition is activated, and possibly deactivated, during a machine’s execution. The semantics have been formulated to agree with one’s intuition of how timed transitions should work and clarify the cases where intuition does not suffice. The semantics is not a quantitative, temporal semantics dealing with timer durations; frameworks already exist to capture this aspect of timed, albeit restricted, forms of Statecharts [4, 5, 8]. The semantics presented here address the heretofore neglected practical possibilities of inter-level transitions, transitions spanning orthogonal regions, and transitions to history pseudostates. These semantics serve as a necessary contribution towards the goal of formalizing Statecharts languages, and towards the ultimate goal of performing quantitative, temporal analysis of unrestricted Statecharts models. The semantics, while simple, are surprisingly subtle, particularly when applied to cases of inter-level or self-loop transitions, or in the context of history pseudostates. These semantics, and the associated decision procedure, have been incorporated into ECharts, an extension of the UML Statecharts that is used for the design and implementation of a number of telecommunication projects at AT&T.

2 Related Work

While real-time semantics have been presented for restricted Statecharts dialects, for example [4, 5, 8], none that we are aware of have addressed the interactions
of inter-level transitions, transitions spanning orthogonal regions, and transitions to history pseudostates. In order for any real-time semantics to extend to unrestricted Statecharts, it will be necessary to incorporate a timed transition activation semantics such as the one presented here.

3 Preliminaries

Figure 1 serves as a legend to the notation we will use in the examples that follow. A transition with a super-label \( t \) denotes a timed transition. A transition with no super-label denotes any transition, possibly a timed transition. In the examples in this paper, no transitions include guard conditions, nor do we take into account any transition priority ordering scheme. A transition with a heavy line denotes a transition that has just fired. Usually we also identify a firing transition with a super-label \( f \).

It will be important in our discussion to distinguish between the three cases in Figure 2. In order to do so, one must associate a transition with a particular nested state machine. For example, Figure 2(a) graphically indicates that the transition with sub-label \( n_1 \) is associated with a higher level machine than the transition with sub-label \( n_2 \). While the graphical notation serves to resolve relative transition depths, we will adopt an alternative textual notation in the form of transition labels to serve the same purpose. The textual notation will be used in the formalization of the semantics presented later in the paper. The notation also permits us to aggregate different cases into a single diagram in the interests of conserving space. A sub-label \( n \geq 0 \) denotes the relative depth of the machine that a transition was defined for, where a transition defined for the top-level machine has a depth of 0. Therefore, in Figure 2(a), the graphical notation indicates that \( n_2 > n_1 \), in Figure 2(b), \( n_1 > n_2 \), and in Figure 2(c), \( n_1 = n_2 \). However, by taking a small liberty with the graphical notation and using the textual notation alone, we are able to represent all three cases with the single diagram Figure 2(c), simply by considering the the relative values of \( n_1 \) and \( n_2 \).

![Fig. 1. Legend](image)

In this discussion we assume that each timed transition has a timer and an expiry flag associated with it. A timed transition is always in either an active or an inactive state. When a transition is active, its counter is counting or its
counter has expired resulting in its expiry flag being set. If a transition’s expiry flag is set and its source state is satisfied by the current machine state, and its guard conditions are satisfied, then the transition becomes enabled for firing. Whether or not the transition actually fires depends on the transition firing semantics of the Statecharts dialect in question, which is a topic outside the scope of this paper. When a transition is inactive, its counter is not counting and its expiry flag is clear.

4 Motivating Examples

We now present a number of examples to hone one’s intuition. Consider again Figure 1 which serves as a simple, introductory example. By virtue of f firing, the machine arrives in a state that satisfies the source state of the timed transition t and, therefore, we expect that t should be active. If t was inactive prior to f firing then t is activated when f fires, meaning that t’s timer commences counting. As we will soon see, it is also possible for t to be active prior to f firing. In this example, if t was already active then the expected behavior is that it will first be deactivated, meaning that t’s counter is stopped and cleared and its expiry flag is cleared. Once t is deactivated, it is activated. We refer to deactivation followed by activation as reactivation. In some cases it may also be desirable for an already active transition to simply remain active after a transition fires, that is to say if t is active prior to f firing, and then t is activated when f fires, then t remains active.

In Figure 3(a), a transition f fires whose source and target states are deeper than the source state of the timed transition t. First one should realize that t must be active when f fires since t’s source state is satisfied in f’s source state. So the question arises: When f fires, should t remain active or should it be reactivated? Furthermore, one must consider if the answer is affected by the relative transition depths. To answer this question, consider what one wants to accomplish with a timed transition. In this case, t should be enabled if its timer expires while the machine remains in sub-state s1 of state s0. This behavior is desired regardless of what sub-state s1 might be in. Now let us consider this observation in light of relative transition depths. If n2 = 2, then the source state that is explicitly referenced by t is not updated as a result of f firing, so it would seem reasonable for t to remain active. However, if n2 < 2 then f’s target must
cross the boundary of state $s_1$, the same state that is explicitly referenced by $t$'s source state. In this case, one would expect $t$ to be reactivated.

In the next example, shown in Figure 3(b), we consider a variation of the previous example in the context of orthogonal states. Once again, $t$ must be active both prior to $f$ firing and after $f$ firing, but whether or not $t$ is reactivated when $f$ fires depends on the relative depths of $t$ and $f$. If $n_2 = 1$ then $f$'s target state $s_2$ is independent of $t$'s source state, so it should follow that $t$ remains active. If $n_2 = 0$, however, then state $s_{11}$ would be explicitly re-entered and state $s_{12}$ would be implicitly re-entered when $f$ fires, so it follows that $t$ should be reactivated since $t$'s source state includes $s_{12}$.

![Fig. 3. Firing transition target state versus timed transition source state](image)

In the next two examples, shown in Figures 4(a) and (b), we assume that the state of the machine prior to the transition firing is sub-state $s_2$ of state $s_1$. Therefore, the timed transition is active in this state. The situation depicted in Figure 4(a) is similar to that shown in Figure 1. Since $f$'s target is the same as $t$'s source, then $t$ is reactivated when $f$ fires. This should be the case regardless of relative transition depths. A variation on this example is shown in Figure 4(b). Since $f$'s target state $s_1$ contains a sub-machine, then it is reset to its default initial state $s_2$ when $f$ fires. Since there is an implicit transition to state $s_2$ and since $s_2$ is $t$'s source state, then we would expect $t$ to be reactivated, regardless of relative transition depths.

We now turn our attention to timed transition activation in the presence of history pseudostates. Initially, we consider only the UML Statecharts deep history pseudostate which, when specified as a firing transition’s target state, means that the target machine’s current state reverts to its previous state recursively reverting to the previous states of any sub-states. In Figure 5(a), we assume the state of the machine prior to $f$ firing is sub-state $s_2$ of state $s_1$. The firing transition should therefore cause $s_1$ to revert to its previous sub-state $s_2$. Since $f$'s target is a deep history pseudostate, then we are returning to the previous
machine state. If $n_1 = 1$ and $n_2 = 0$ then we would expect that $t$ remain active and not be reactivated. This behavior is appealing from a design perspective since it supports compositional design. A submachine with a timeout transition may be embedded within the state of a parent machine whose transitions may fire without affecting the temporal behavior of the nested sub-machine. However, if $n_1 \leq n_2$ then $f$'s target state explictly references $t$'s source state in which case we would expect $t$ to be reactivated.

Now let us consider a variation on the previous example, shown in Figure 5(b). Assume that $n_1 = 1$ and $n_2 = 0$, and that the previous state of $s_1$ was $s_2$ which implies that $t$ was active. The “intuitive” behavior in this case is not as clear as it was in the previous case. One choice would be to maintain consistency with the previous example, leaving $t$ active. On the other hand, some may feel that because the current machine state changed from $s_1$ to $s_3$ at some earlier point in the machine’s execution, then $t$ should be reactivated upon entry to $s_1$, even if entry is via a deep history pseudostate. We have chosen the former interpretation which leaves $t$ active since consistency and simplicity are strong arguments in its favor. Furthermore, if the designer desires that $t$ be reactivated, then $f$ can be changed to target only state $s_1$, or sub-state $s_2$ of $s_1$.

Timed transition activation in the presence of a shallow history pseudostate is a combination of aforementioned approaches which we will formally describe shortly.

Completing our series of examples, we now consider three cases when the firing transition is itself the timed transition. In a manner similar to the example of Figure 1, the timed transition shown in Figure 6(a) is reactivated when it fires because the firing transition target state is equal to its source state. In Figures 6(b) and (c), the transition target is shown to explicitly reference the transition’s source, so, like the examples shown in Figure 5 for $n_2 \leq n_1$, the transition is reactivated.
Fig. 5. Firing transition target is a deep history pseudostate

Fig. 6. Firing transition is a timed transition
5 Formalization

In our discussion of the examples in the previous section, we informally spoke in terms of relative transition depth and the relationship between the timed transition source state and the firing transition target state. These concepts form the basis for our formalization of the activation semantics. We now take a closer look at this relationship.

The examples shown in Figures 7 (a) and (b) are instances of Figure 3(a). Recall that when f’s depth is 2, t is activated, and when f’s depth is 1, t is reactivated. The examples show that reactivation occurs when t’s source state intersects with f’s target state. Note that if t’s source state extended to sub-state s2 of state s1, as shown in Figure 7(c), then t’s source state would intersect f’s target state so t would be reactivated regardless of f’s depth. From these examples, one should note that source and target state intersection is a relation based not only on relative transition depths, but also on the sub-states that a source or target state may reference below their defined depths.

Fig. 7. Intersection of timed transition source state and firing transition target state

5.1 State Configurations

Before we define the intersection relation, it is necessary to formally define the concept of a transition’s source or target state. Following a variation of [6], we define a (source or target) state configuration as a vector of linear terms. Let StateName denote a countable set of state names, ranged over by n, where the distinguished state names H, representing the shallow history pseudostate, and H∗, representing the deep history pseudostate, belong to StateName. Let V represent a countable set of variable names disjoint from StateName, ranged over by X, Y. Let StateCfg denote the smallest set of linear term vectors satisfying the following conditions:

1. \( V \subseteq \text{StateCfg} \),
2. \((n_1(S_1), \ldots, n_k(S_k)) \in \text{StateCfg}, \ k > 0, \ \text{if for all} \ 1 \leq i \leq k, \ n_i \in \text{NameState}, \ \text{and} \ S_i \in \text{StateCfg}\),
3. \(\forall S \in \text{StateCfg}, \ \text{for each variable} \ X \ \text{in} \ S, \ X \ \text{may only occur once in} \ S.\)

In this context, a variable \(X\) represents a term vector, not an individual term. We let \(S\) range over \(\text{StateCfg}\) and we let \(T\) range over \(\text{StateCfg} \setminus V\). When we refer to the depth of a term vector, we mean the depth of the term vector’s tree, where the root of the tree has depth 1. By convention we represent a reference to a primitive state (containing no sub-states) \(n\) with the term \(n(X)\). This formulation is capable of representing state configurations for the common Statecharts dialects, including those spanning orthogonal regions. As an example, Table 1 lists the timed transition source state configuration and the firing transition target state configuration for each of the examples given in the previous section.

<table>
<thead>
<tr>
<th>Figure</th>
<th>t source configuration</th>
<th>f target configuration</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(s1( X1 ))</td>
<td>(s1( X2 ))</td>
</tr>
<tr>
<td>3(a)</td>
<td>(s0( s1( X1 )))</td>
<td>(s0( s1( s2( X2 ))) )</td>
</tr>
<tr>
<td>3(b)</td>
<td>(s11( X1 ), s12( s4( X2 )))</td>
<td>(s11( s2( X1 ), s12( X1 )))</td>
</tr>
<tr>
<td>4(a)</td>
<td>(s1( s2( X1 )))</td>
<td>(s1( s2( X2 )))</td>
</tr>
<tr>
<td>4(b)</td>
<td>(s1( s2( X1 )))</td>
<td>(s1( X2 ))</td>
</tr>
<tr>
<td>5(a)</td>
<td>(s1( s2( X1 )))</td>
<td>(s1( H' ))</td>
</tr>
<tr>
<td>5(b)</td>
<td>(s1( s2( X1 )))</td>
<td>(s1( H' ))</td>
</tr>
<tr>
<td>6(a)</td>
<td>(s1( X1 ))</td>
<td>(s1( X2 ))</td>
</tr>
<tr>
<td>6(b)</td>
<td>(s1( X ))</td>
<td>(s1( H' ))</td>
</tr>
<tr>
<td>6(c)</td>
<td>(s1( s2( X )))</td>
<td>(s1( H' ))</td>
</tr>
</tbody>
</table>

Table 1. State configurations for examples

The translation from the graphical notation to the term vector-based notation is straightforward. As shown in the entry for Figure 3(b), we adopt two conventions with regards to representing state configurations that reference orthogonal regions. The first convention is that orthogonal regions should be uniquely identified. In Figure 3(b), the two orthogonal regions are labelled \(s11\) and \(s12\). The second convention is that if one or more orthogonal regions of a concurrent machine are referenced by a transition in the graphical notation, then all orthogonal regions must be referenced in the associated state configuration. In Figure 3(b), the orthogonal region labelled \(s11\) is not explicitly referenced by \(t\)’s target, however, it is referenced via the variable \(X_1\) in the associated source state configuration.

5.2 Inference Rules

We present the semantics in the form of auxiliary inference rules that are intended to complement existing structural operational semantics-style Statecharts
language descriptions, for example [7]. The inference rules define the disjoint relations activate and reactivate, where

$activate, reactivate \subseteq StateCfg \setminus \mathcal{V} \times StateCfg \setminus \mathcal{V} \times \mathbb{N} \times \mathbb{N}$.

Here, an element $(T_i, T_f, d_t, d_f)$ belonging to activate or reactivate denotes a timed transition $t$ with source state configuration $T_i$ and depth $d_t$, and a firing transition $f$ with target state configuration $T_f$ and depth $d_f$. An element belonging to activate (reactivate) means that $t$ should be activated (reactivated) when $f$ fires. We assume that $T_i$ and $T_f$ are both satisfied by the same machine state: $T_f$ by virtue of the fact that it specifies the target of the firing transition thereby constraining the new current machine state, and $T_i$ by virtue of the fact that the timed transition is being considered for activation in the machine state resulting from the firing transition.

Initially, we present the rules excluding those involving history pseudostates.

\[ (t_0) \quad activate(n_1(S_1),\ldots,n_k(X_k),\ldots,n_m(X_m)),(n_1(T_1),\ldots,n_k(T_k),\ldots,n_m(T_m)),0,d_f) \]
\[ k,m \geq 1, \quad d_f > 0, \quad T_k \text{ has depth} \geq d_f \]

\[ (t+) \quad activate(T_i,T_f,d_t,d_f) \]
\[ k,m \geq 1, \quad d_t \geq d_f \]

\[ (t-) \quad activate(T_i,T_f,d_t,d_f) \]
\[ d_t > 0, \quad d_t < d_f \]

\[ (xor) \quad reactivate(T_i,T_f,d_t,d_f), \quad (T_i,T_f,d_t,d_f) \notin activate \]

Rule (t0) is an axiom that generates all elements of activate whose timed transition source state refers to any state in orthogonal regions not containing the firing transition target state, and whose firing transition is an arbitrary target state. Rule (t+) generates all elements of activate whose state configurations are nested in parent state configurations. This has the effect of increasing the depths of the timed and firing transitions. Rule (t-) generates all elements of activate whose timed transition source refers to states below depth $d_t$. Rule (xor) is an axiom that defines an element of reactivate to be any element that is not already defined to belong to activate. As we will show shortly, there is a decision procedure for determining an element’s membership in reactivate.

As an example of using these rules, consider the following derivation for the example shown in Figure 7(a).

\[ (t_0) \quad activate(s_2(X_1)),(s_2(s_3(X_2)),0,1) \]
\[ (t+) \quad activate(s_1(s_2(X_1))),0,1) \]
\[ (t-) \quad activate(s_1(s_2(s_3(X_2))),s_1(s_2(s_3(s_3(X_2)))),1,2) \]
\[ reactivate(s_1(s_2(s_3(X_2))),s_1(s_2(s_3(s_3(X_2)))),0,2) \]
The following rules define activation in the context of shallow and deep history pseudostates.

\[
(f_0) \quad \text{activate}((n_1(X_1), \ldots, n_k(T_k) \ldots, n_m(X_m)), (n_1(S_1), \ldots, n_k(H) \ldots, n_m(S_m)), 1, 0),
\]
\[k, m \geq 1, T_k \text{ has depth } = 1\]

\[
(f^*0) \quad \text{activate}((n_1(X_1), \ldots, n_k(T_k) \ldots, n_m(X_m)), (n_1(S_1), \ldots, n_k(H) \ldots, n_m(S_m)), d_t, 0),
\]
\[k, m \geq 1, d_t > 0, T_k \text{ has depth } \geq d_t\]

\[
(f^+) \quad \text{activate}(T_t, T_f, d_t, d_f),
\]
\[k, m \geq 1, d_t > d_f\]

\[
(f^-) \quad \text{activate}(T_t, T_f, d_t, d_f),
\]
\[d_f > 0, d_t > d_f\]

These rules are similar to those already given for the non-history pseudostate case, the major difference being that in the former case, \(dt > df\) and in the latter case, \(dt < df\). Rules \((f0)\) and \((f^*0)\) are axioms generating all elements of \(\text{activate}\) whose firing transition references a shallow or deep history pseudostate, respectively. Rules \((f^+)\) and \((f^-)\) are analogous to rules \((t^+)\) and \((t^-)\), respectively.

5.3 Decision Procedure

We now present decision procedures to determine membership of a given element in \(\text{activate}\) or \(\text{reactivate}\). We base both procedures on the inference rules given in the previous section.

The decision procedure to determine membership in \(\text{activate}\) exploits a number of properties associated with the inference rules. First note that the rules for inferring membership in \(\text{activate}\) apply only for transitions \(t\) and \(f\) such that either \(d_t < d_f\), or \(d_t > d_f\). The decision procedure for membership in \(\text{activate}\) is, therefore, partitioned into three cases: when \(d_t < d_f\), the rules for non-history pseudostates apply, when \(d_t > d_f\), the rules for history pseudostates apply, and when \(d_t = d_f\), no rules apply. The axioms in each rule set serve as a termination condition for its respective case.

An important property of the rule for each case is that their derivations possess a normal form. Consider the case where \(d_t < d_f\). Any derivation for this case begins with the application of rule \((t0)\) and is followed by an arbitrary sequence of applications of rules \((t^+)\) and \((t^-)\). By recognizing that the result of any such sequence ending in the application of \((t^-)\) is equal to the result of a sequence in which all but the final application of \((t^-)\) are removed from the original sequence, then one has the basis for a normal form for derivations. Thus, any derivation has an equivalent normal form that consists of an application of \((t0)\), followed by at least one application of \((t^+)\), followed by at most one application of \((t^-)\), followed by zero or more applications of \((t^+)\). The same
observation holds for the case where $d_t > d_f$. This property serves as the basis for the computation steps followed by decision procedure, namely, that the decision procedure proceeds in the reverse order of a normalized derivation.

The final property we will use is that for any element $(T_t, T_f, d_t, d_f)$ of activate derived by the rules for the case where $d_t < d_f$, there exists exactly one non-variable term in $T_f$. Similarly, for the case where $d_t > d_f$, there exists exactly one non-variable term in $T_t$. This property is used to guide the decision procedure’s search through the solution space.

We now present the decision procedure activate, shown in Figure 8, which returns true if an element $(T_t, T_f, d_t, d_f)$ is a member of activate, otherwise it returns false.

It is easy to see that the decision procedure is divided into the three cases, $d_t = d_f$, $d_t < d_f$, and $d_t > d_f$. Furthermore, for one can see that there are two sub-cases for each of the cases $d_t < d_f$ or $d_t > d_f$. Considering the case $d_t < d_f$, one can see that the sub-case $d_t > 0$ corresponds to an attempt to apply rule $t+$ in reverse and recursively invoke the procedure on the result. For the sub-case $d_t = 0$, we check if the element is an instance of axiom $(t0)$. If it isn’t, then we attempt to apply rule $t-$ in reverse and recursively invoke the procedure on the result. Note that applying $(t-)$ in reverse in this way can only occur once for a given invocation of the procedure since $d_f$ will be equal to 1 the next time this sub-case is considered. Note that this is consistent with following the normal form of a derivation, as discussed above. The explanation for the case $d_t > d_f$ is similar to that for $d_t < d_f$.

The procedure will terminate for any element by virtue of the fact that the transition depths or the state configuration term vector sizes are reduced for each recursive invocation of the procedure. For this reason, the procedure also serves as a decision procedure for reanimate as defined by rule $(xor)$.

6 Conclusions

We have presented an intuitive, formal semantics for timed transition activation in Statecharts dialects supporting timed transitions. The semantics explain how a firing transition can affect a timed transition’s activation state. The semantics address the practical possibilities of inter-level transitions, transitions spanning orthogonal regions, and transitions to history pseudostates. In addition to the semantics, an accompanying decision procedure is given which determines whether or not a timed transition should be activated. The semantics and associated procedure are currently incorporated into ECharts, an extension of the UML Statecharts developed at AT&T. The semantics are an important contribution to the field of Statecharts semantics and serve as the basis for a full temporal semantics.

References

activate(T_t, T_f, d_t, d_f)

if d_t = d_f then
    return false
elif d_t < d_f then
    if d_t > 0 then
        if T_t of the form (n_1(S_1),...,n_k(T_tk),...,n_m(S_m)) and
            T_f of the form (n_1(X_1),...,n_k(T_fk),...,n_m(X_m)) and
            k,m => 1 then
            return activate(T_tk, T_fk, d_t - 1, d_f - 1)
        else
            return false
        endif
    elif d_t = 0 then
        if T_t of the form (n_1(X_1),...,n_k(T_tk),...,n_m(X_m)) and
            T_f of the form (n_1(S_1),...,n_k(T_fk),...,n_m(S_m)) and
            k,m => 1 then
            return true
        elif d_f > 1 then
            return activate(T_t, T_f, d_f - 1, d_f)
        else
            return false
        endif
    endif
elif d_t > d_f then
    if d_f > 0 then
        if T_t of the form (n_1(X_1),...,n_k(T_tk),...,n_m(X_m)) and
            T_f of the form (n_1(S_1),...,n_k(T_fk),...,n_m(S_m)) then
            return activate(T_tk, T_fk, d_t - 1, d_f - 1)
        else
            return false
        endif
    elif d_f = 0 then
        if T_t of the form (n_1(X_1),...,n_k(T_tk),...,n_m(X_m)) and
            T_f of the form (n_1(S_1),...,n_k(H),...,n_m(S_m)) and
            k,m => 1 and
            d_t = 1 then
            return true
        elif T_t of the form (n_1(X_1),...,n_k(T_tk),...,n_m(X_m)) and
            T_f of the form (n_1(S_1),...,n_k(H*),...,n_m(S_m)) and
            k,m => 1 then
            return true
        elif d_t > 1 then
            return activate(T_t, T_f, d_t, d_t - 1)
        else
            return false
        endif
    endif
endif

Fig. 8. The activate decision procedure


