Proc. LICS '87

On the Formal Semantics of Statecharts

(Extended Abstract) †

D. Harel[1,2,3] A. Pnueli[1,2,4] J. P. Schmidt[1,5] R. Sherman[1,6]

Abstract: Statecharts have been introduced recently [H] as a visual formalism for specifying the behavior of complex reactive systems. They extend classical state transition diagrams in several ways, while retaining, and even enhancing, their visual appeal. In particular, statecharts cater for hierarchical descriptions, high-level and low-level events, and notably, multi-level concurrency with a broadcast communication mechanism that can give rise to chain-reaction effects. In this paper we provide a formal syntax and (operational) semantics for statecharts. The semantics appears to be novel in a number of ways, among which are its treatement of shared variables, chain reactions and simultaneous multiple messages.

1. Reactive systems and State Diagrams

The literature on software and systems engineering is almost unanimous in recognizing the existence of a major problem in the specification and design of large and complex *reactive systems*. A reactive system (see [P, HP]), in contrast with a *transformational system*, is characterized by being, to a large extent, event-driven, continuously having to react to external and internal stimuli. Examples include telephones, communication networks, computer operating systems, avionics systems, VLSI circuits, and the man-machine interface of many kinds of ordinary software. The problem is rooted in the difficulty of describing reactive behavior in ways that are clear and realistic, and at the same time formal and rigorous, sufficiently so to be amenable to detailed computerized simulation. The behavior of a reactive system is really the set of allowed sequences of input and output events, conditions, and actions, perhaps with some additional information such as timing constraints.

Notable among the solutions proposed for this problem are Petri nets [R], CSP [Ho] CCS [M], ESTEREL [BC], and Temporal Logic [P]. A more recent approach involves *statecharts* [H], which constitute an attempt to revive the classical formalism of finite state machines (FSM's) and their visual counterpart, state transition diagrams, and make it fitting for use in large and complex applications. Indeed, people working on the design of really complex systems have all but given up on the use of conventional FSM's and their state diagrams, and for several reasons.

1. State diagrams are 'flat'. They provide no natural notion of depth, hierarchy or modularity, and therefore do not support stepwise, top-down or bottom-up development.
2. State diagrams are uneconomical when it comes to transitions. An event that causes the very same transition from a large number of states, such as a high-level interrupt, must be attached to each of them separately, resulting in an unneccesary multitude of arrows.
3. State diagrams are extremely uneconomical, indeed quite infeasible, when it comes to states. As the system under description grows linearly, the number of states grows exponentially, the conventional FSM formalism forcing one to explicitly represent them all.
4. Finally, state diagrams are inherently sequential in nature, and do not cater naturally for concurrency.

There have been attempts to remove some of these drawbacks, by using various kinds of hierarchical or communicating state machines, but we consider these solutions to be unsatisfactory. Typically, the hierarchies are merely structural and do not provide real savings in the size of the resulting description, and the communication is usually one-to-one, being channel- or processor-based, allowing for only a single set of communicating machines on the highest level of the description. Moreover, for the most part, these extensions are not diagrammatic, and hence one loses the advantages that a visual medium might offer.

2. Statecharts

Statecharts were proposed to overcome these drawbacks while preserving, and even enhancing, the visual appeal of conventional state diagrams. We shall only review

[1] Ad Cad, Inc., Cambridge, MA.
[2] The Weizmann Inst., Rehovot, Israel.
[3] Carnegie-Mellon Univ., Pittsburgh, PA.
[4] Univ. of Texas, Austin, TX.
[5] Courant Inst., NYU, New York.
[6] USC/ISI, Marina del Ray, CA.

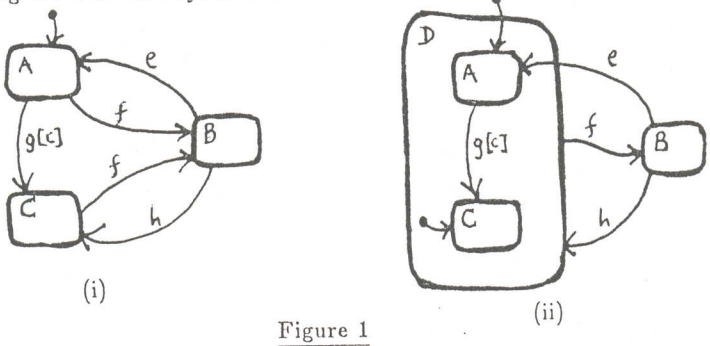
† See note at the end of the paper.

the essentials of the formalism very briefly here, referring the reader to [H] for more details and examples. More examples, as well as several related developments, appear in [BDH, DH1, DH2]. In a nutshell we might say:

statecharts = state-diagrams + depth +

orthogonality + broadcast-communication

The idea of depth is illustrated in Fig. 1, where (ii) may replace (i). The symbols e, f, g, and h stand for events that trigger the transitions, and the bracketed c is a condition. Thus, $g[c]$ triggers the transition from A to C if and when g occurs, but only if c is true at that time. State D is the XOR of A and C, so that being in D is tantamount to being either in A or in C, but not in both. The main point here is the f-arrow, which leaves the contour of D and hence, by definition, applies to both A and C, as in (i). This simple idea, when applied to large collections of states in a multi-level manner, overcomes points 1 and 2 above. Notice the way the small *default arrows* depend on their scope: in (i) state A is singled out as being the default, or start-state, of the three, a fact represented in (ii) by the top default arrow; the bottom one, however, states that C is default among A and C if we are already in D, and hence alleviates the need for continuing the h-arrow beyond D's boundary.

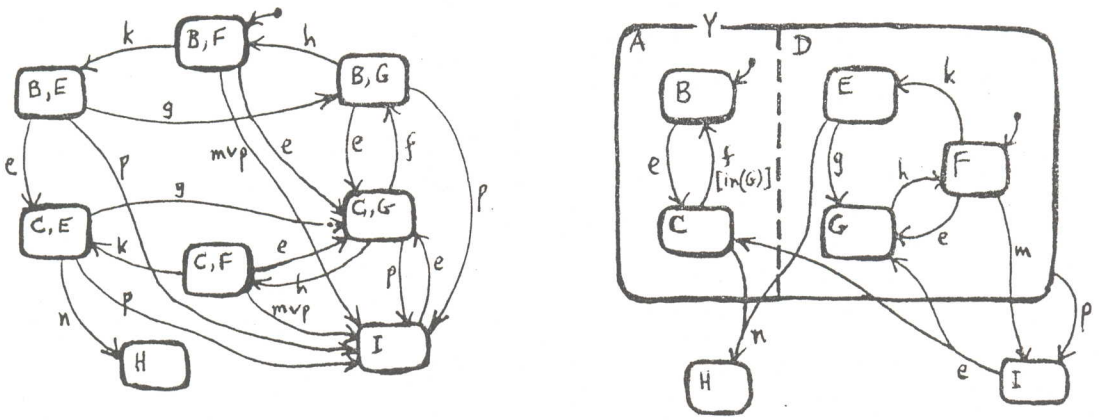


(i) (ii)

Figure 1

Orthogonality is the dual of the XOR decomposition of states, being, in essence, an AND decomposition. Fig. 2 illustrates this. In (ii) state Y consists of two orthogonal components, A and D, related by AND: to be in Y is tantamount to being in both A and D, and hence the two default arrows. The intended semantics of (ii) is given by its equivalent 'flat' version (i), which represents a sort of automata product. Notice the simultaneity of transitions that takes place when event e occurs in state configuration (B, F), and the merging and splitting transitions that lead to and from Y. Notice also the special condition $in(G)$ attached to the f-transition from C, and the way it is reflected in (i). Fig. 2 is, of course, somewhat misleading, since (ii) may not seem to be any better than (i). However, this is obviously the heart of the exponential blow-up problem, the number of states in the (i) version being the product of the numbers in the (ii) version. If the orthogonality construct is used often, and on many levels, difficulties 3 and 4 above are overcome in a reasonable way, as can be further observed by studying the examples in the aforementioned references.

Another feature of statecharts is the ability to 'remember' a previous visit to a state. Fig. 3 shows a *history entrance* into A, which overrides the default arrow that leads to D. The default is taken when the entrance is via the



(i) (ii)

Figure 2

f-arrow, whereas if A is entered via the e-arrow the state entered is that one of B, C or D that the system happened to be in when it was most recently in A. If the state thus entered has its own substates, then the default arrows that must appear therein are used to determine the lower-level states actually entered.

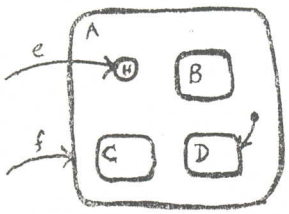


Figure 3

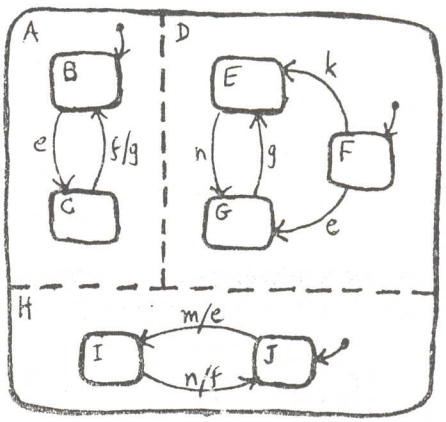


Figure 4

Now, Figs. 1–3 do not contain any outputs, and hence, so far, orthogonal components can synchronize only through common events (like e in Fig. 2), and can affect each other only through *in*(*state*) conditions. The real subtlety of the way statecharts model concurrency is in their *output* events. Here statecharts can be viewed as an extension of Mealy machines, as we allow output events, which we call *actions*, to be attached optionally to the triggering event along a transition. However, in contrast with conventional Mealy machines, an action appearing along a transition in a statechart is not merely sent to the 'outside world' as an output. Rather, it can, and typically *will*, affect the behavior of the statechart itself in orthogonal components. This is achieved by a simple broadcast mechanism: just as the occurrence of an external event causes transitions in all components at which it is relevant (see Fig. 2, for example), if event e occurs and a transition labelled e/f is taken, the action f is immediately activated, and is regarded as an (internal) event, possibly causing further transitions in other components.

Fig. 4 shows a simple example of this. If we are in (B, F, J) and along comes the external event m, the next configuration should be (C, G, I), by virtue of e being generated in H and triggering the two transitions in components A and D. This is a *chain reaction* of length 2. If now external event n occurs, the new configuration will be (B, E, J), by virtue of a similar chain reaction of length 3.

This concludes our brief discussion of the main features of the language, as discussed at length in [H]. Statecharts have been used in the design of a complex avionics system in the Israel Aircraft Industries, and are currently being used experimentaly in a number of additional industrial projects. Statecharts are also at the heart of a commercial product, STATEMATE1, that is intended as a graphical working environment for the engineers involved in reactive systems. Such an implementation is impossible without a formal semantics, and the current paper actually grew out of our work on STATEMATE1.

3. The Semantics: A Discussion

It is quite obvious that even the basic features presented here raise interesting semantic problems. For one, an instantaneous description of the system specified by a statechart must contain a state configuration whose length is not fixed, but changes as orthogonal states are entered and left. Moreover, we would like chain reactions to satisfy two seemingly contradictory properties. On the one hand, they should ideally take 'zero time'; that is, the new state configuration ought to be obtained in the very next step, so that other external events will not be able to interrupt, and perhaps alter, the inevitable chain reaction. On the other hand, the *order* of the transitions taking place within a chain reaction is important, even though they really do not take time. This is one of the main issues that our semantics addresses.

The appendix to this extended abstract contains the formal syntax and the operational semantics that we propose. It first presents a detailed syntax of a rather powerful version the statecharts language, complete with compound events and conditions, and shared variables that can be assigned to and tested. Thus, $e \wedge f$ and $e \vee f$ are events, and similarly for conditions (which are also closed under negation). Apart from e's and f's, the actions that can appear to the right of the '/' along transitions include $c :=$*true* and $c :=$*false* for a condition c, and $v := \tau$ for a variable v and an appropriate algebraic expression τ. In addition, there are several special kinds of events and conditions. For example, $en(s)$ and $ex(s)$ are events that occur upon entering and exiting state s, respectively, and $in(s)$ is the corresponding condition. Also, for expressions τ and σ we have conditions such as $\tau = \sigma$ and $\tau < \sigma$, as well as the event $ch(v)$, which occurs when v changes value. Any condition gives rise to the two events $tr(c)$ and $fs(c)$, which occur when c changes from false to true and vice versa. (Thus, for example, the events $en(s)$ and $tr(in(s))$ will always occur

together.) Also, any event gives rise to the condition $ny(e)$, which stands for 'e has *not-yet* occurred.' This condition, as we shall see later, refers only to what is happening inside the currently evaluated chain reaction. And so, while events are not closed under negation, one *can* state that an event has not yet occured since the beginning of the latest chain reaction. Another special condition is $cr(c)$, which stands for 'the current value of c', and refers to the value of condition c within the current chain reaction. Similarly, we have $cr(v)$ for a variable or expression v, representing its current value within a chain reaction.

The reader who is well-versed in the semantic difficulties raised by such languages as CCS and CSP will no doubt realize the potential difficulties this syntax raises. Not only do we have high level events (like f in Fig. 1(ii) and p in Fig. 2(ii)), concurrency on all levels, and defaults and history entrances that bubble control down to the states on the lowest levels, but we also have simultaneous reactions to events broadcast throughout the system, and shared variables that can, in principle, be assigned simultaneously, possibly generating conflicting values. Some of the decisions that have to be made here involve protecting chain reactions from incoming external events, while complying with the order implicit in them, and resolving conflicts, either by an undefined outcome or by nondeterminism.

The bulk of the appendix is devoted to presenting a formal operational semantics to this version of statecharts. Our approach can be superficially described as follows. To make the explanation simple we shall assume that we are in a configuration C and that a single external event e that has just occurred. Our intention is to define the set of next configurations, $next(\mathbf{C})$, the elements of which are the legal outcomes of applying all transitions triggered by e, followed by all their consequences, including newly enabled transitions, followed by all *their* consequences, etc. (This sequence will not be able to become infinite, as a study of the semantics reveals.)

Given C and e, we first find all transitions T that are 'relevant' to the event e in configuration C, in the sense that (i) T originates in some ancestor of a subconfiguration of C in the state-tree, and (ii) its triggering event indeed occurs when e does. In doing so one must keep track of those T's that are mutually inconsistent, and therefore cannot be taken simultaneously; they will give rise to nondeterminism, as consistent ones must be taken together. Figuring out (i) above entails computing appropriate *least common ancestors* (LCA's) in the state-tree, and (ii) requires a careful definition, mutually inductive in events and conditions, and further complicated by the ny and cr constructs. Haven found such a T, we must 'apply' it, with all its consequences.

Here is where the semantics becomes really intricate. We proceed by defining *micro-steps*, with a 'real' step being a maximal sequence of them. Micro-steps thus capture the internal order in which the simultaneous transitions and actions of a single step are carried out. Micro-steps are not directly transparent to the user, although, as we exemplify below, the ny and cr constructs can be used to exploit them beneficially. In a micro-step the system nondeterministically selects any consistently executable subset of the enabled transitions (in particular, a subset with no conflicting assignment statements), and executes them at one and the same time, adding their immediate consequent actions to the events and transitions awaiting execution, and hands this remaining set to the next micro-step for execution. Note that history and default entrances in states that appear in the target sub-configurations of the transitions that are taken have to be taken into account, and their effects on the lowest-level states have to be calculated. A sequence of micro-steps terminates, and thus becomes an externally observable step, if and when there are no posible consistent choices left. Hence the term 'maximal'.

During micro-step execution, $cr(c)$ and $cr(v)$ are updated to contain the current values of conditions, variables and expressions, and $ny(e)$ is updated to reflect whether or not e has already been executed in the present step, i.e., since the first micro-step in the sequence. In contrast, the status and values of events, conditions and expressions that do *not* involve cr and ny do not noticably change within the micro-steps of a single step. Thus, cr and ny are the only mechanisms for detecting the order in which micro-steps are carried out inside a step.

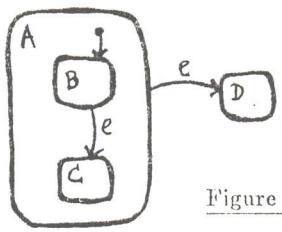


Figure 5

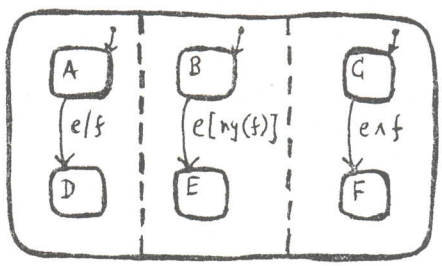


Figure 6

4. Some Examples

A few simple examples are now in order. Fig. 5 shows a simple kind of 'structural' nondeterminism, where if in B and e occurs the two possible outcomes are C and D. Fig. 6 shows nondeterminism caused by the freedom of selecting subsets in micro-steps. Here, if in (A, B, C), event e can cause the system to reach (D, B, F) if the transitions are considered in the order $A \rightarrow D, B \rightarrow E, C \rightarrow F$, since the condition in the second one will be false. However, (D, E, F) is also possible, if the transitions are considered in

the order $B \rightarrow E, A \rightarrow D, C \rightarrow F$. Note that the $e \wedge f$ event occurs in both cases, as the conjunction is not sensitive to the ordering within a step.

The definition of conflicting actions forces an order to be put on two assignments to the same variable, but not on assignments to different variables. Thus, in Fig 7, if the initial values of x and y are both 0, then while the final value of both will be 1 after the step resulting from e in configuration (A, B, C), there are two nondeterministic resulting configurations: (D, E, C) (if both leftmost transitions are considered together, since x and y havn't just become equal; they were equal all along), and (D, E, F) (if they are taken separately, since one of the two variables became 1 while the other remained 0, thus making $tr(x = y)$ happen).

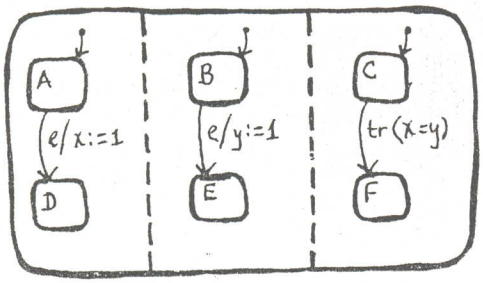


Figure 7

Here are two examples of using the *cr* and *ny* constructs to one's advantage. The first is in forcing *exclusivity* of the effect of an event e, and the second is in specifying *priority* of e over f in case they both occur simultaneously. Fig. 8 shows how the two conflicting assignments force the micro-steps to be taken in sequence, with the consequence that once one of them is taken in the first micro-step the guarding condition on the second one prevents *it* from being taken, as it tests the system's current whereabouts in the first component. The result is that when e occurs in (A, B) the systems reacts nondeterministically, entering either (A, D) (with $x = 2$) or (C, B) (with $x = 1$). Notice that omitting the *cr*'s gives a quite different behavior, with the final configuration being deterministically (C, D), but the final value of x being similarly 1 or 2.

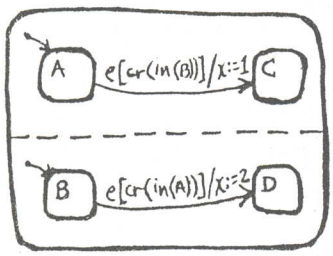


Figure 8

In Fig. 9 we are interested in specifying that in state A event e leads to B and f leads to C, but that if e and f occur *simultaneously* the e-transition is to have priority. This is achieved by the *ny* operator as illustrated.

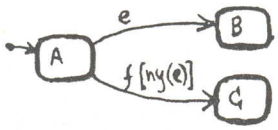


Figure 9

Finally, a somewhat more elaborate kind of priority specification appears in Fig. 10. Here we want to specify that if e occurs when in (E, C), i.e., there is a nondeterministic choice to be made in going to F or B in the top component, then in the bottom component we want to go from C to D only if the former choice was taken. Of course, this might be achieved simply by eliminating the '$/f$' action from the arrow leading to B, but for the sake of the example we may assume that f has other purposes, and has to be carried out whenever e occurs. We could have used a new action symbol, but the example shows how the *cr* alone can be used to achieve th edesired effect.

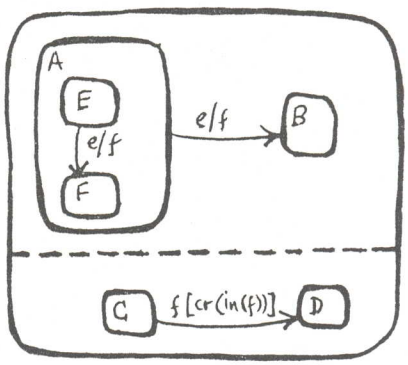


Figure 10

It should be noted that statecharts with no appearences of *cr* and *ny* behave in the intuitive way expected. Thus, the semantics prescribes the behavior described above for the likes of Figs. 1–4. An interesting question is whether this semantics can be easily extended to deal with a number of additional features described informally in [H]. One of the most promising of these is *overlapping states*, the importance of which has been argued in [DH1]. A preliminary attempt at providing a formal syntax and semantics for statecharts with overlapping states appears in [K], but it does not contain any actions, assignments, special events and conditions, or the special constructs *ny* and *cr*. It would be interesting to know if the two semantics are compatible on the rather primitive version obtained by taking the common features only, and whether our semantics can be easily extended to deal with overlapping states using the ideas of [K].

Appendix : Formal Syntax and Semantics

1. Syntax

The syntax of statecharts is defined over the following basic sets of elements: states, transitions, primitive events, primitive conditions and variables. Using these basic sets of elements we define the extended sets of: events, conditions, expressions and labels and the interrelations connecting them.

a. **States** : The set of *states*, S, is defined together with a hierarchy function ρ, a type function ψ, a set of history symbols H and a default function δ.

The *hierarchy function* $\rho : S \rightarrow 2^S$, defines for each state its *substates*. If $\rho(x)=\rho(y)$ then $x=y$. There exists a unique state $r \in S$ such that $\forall s \in S \; r \notin \rho(s)$, r is the *root* of the statechart. ρ^*, ρ^+ are extensions of ρ defined by : $\rho^*(s)=\bigcup_{i \geq 0} \rho^i(s)$, $\rho^+(s)=\bigcup_{i \geq 1} \rho^i(s)$.

The *type function* $\psi : S \rightarrow \{AND, OR\}$ defines for each state its type. If $\rho(s) \neq \emptyset$ and $\psi(s)=OR$ then $\rho(s)$ is a *xor* decomposition of s. This means that when the system is in a state s it is in one and only one of the substates of s. If $\rho(s) \neq \emptyset$ and $\psi(s)=AND$ then $\rho(s)$ is an *and* decomposition of s. This means that when the system is in a state s it is simultaneously in all the substates of s.

The set of *history symbols*, H, is related to the set of states by a function $\gamma : H \rightarrow S$ such that : $\gamma(h_1)=\gamma(h_2)$ implies $h_1=h_2$ and $\gamma(H)$ is a subset of the set of OR states, that is only an OR state may have a history symbol associated with it. It is useful to define $\omega : S \cup H \rightarrow S$ by : $\omega(x)$ is x if $x \in S$ and $\omega(x)$ is $\gamma(x)$ if $x \in H$.

The *default function*, $\delta : S \rightarrow 2^{S \cup H}$, defines for a state s a set of states and history symbols which are contained in the state. If $x \in \delta(s)$ then: for $x \in S, x \in \rho^+(s)$, and for $x \in H, \gamma(x) \in \rho^*(s)$.

$\delta(s)$ is the *default set for* s.

b. **Expressions** : The set of variables is denoted by V_p. The set of *expressions*, V, is defined inductively as follows :

1. If k is a number then $k \in V$.
2. If $v \in V_p$ then $v \in V$.
3. If $v \in V$ then $current(v) \in V$.
4. If $v_1, v_2 \in V$ and op is an algebraic operation then $op(v_1, v_2) \in V$.

$current(v)$ is abbreviated to $cr(v)$.

c. **Conditions** : The set of primitive conditions is denoted by C_p. The set of *conditions*, C, is defined inductively as follows :

1. $T, F \in C$, T, F stand for *true*, *false*, respectively.
2. If $c \in C_p$ then $c \in C$.
3. If $s \in S$ then $in.(s) \in C$.
4. If $e \in E$ then $not\_yet(e) \in C$.
5. If $u, v \in V$, $R \in \{=, >, <, \neq, \leq, \geq\}$ then $u \; R \; v \in C$.
6. If $c \in C$ then $current(c) \in C$.
7. If $c_1, c_2 \in C$ then $c_1 \vee c_2, c_1 \wedge c_2, \neg c_1 \in C$.

$current(c)$ is abbreviated to $cr(c)$, $not\_yet(e)$ is abbreviated to $ny(e)$.

d. **Events** : The set of primitive events is denoted by E_p. The set of *events*, E, is defined inductively as follows :

1. $\lambda \in E$, λ is the *null event*.
2. If $e \in E_p$ then $e \in E$.
3. If $c \in C$ then $true(c), false(c) \in E$.
4. If $v \in V$ then $changed(v) \in E$.
5. If $s \in S$ then $exit(s), entered(s) \in E$.
6. If $e_1, e_2 \in E$ then $e_1 \vee e_2, e_1 \wedge e_2 \in E$.
7. If $e \in E, c \in C$ then $e[c] \in E$.

$true(c), false(c)$ are abbreviated to $tr(c), fs(c)$, respectively. $exit(s), entered(s)$ are abbreviated to $ex(s), en(s)$, respectively. $changed(v)$ is abbreviated to $ch(v)$. e is *atomic* if e is of the form 1-5.

e. **Actions** : The set of *actions*, A, is defined inductively as follows :

1. $\mu \in A$, μ is the *null action*.
2. If $c \in C_p, d \in C$ then $c := d \in A$.
3. If $v \in V_p, u \in V$ then $v := u \in A$.
4. If $a_i \in A$, $i=0,\ldots,n$ then $a_0; \cdots; a_n \in A$

$a \in A$ is *atomic* if it is of the form 1-3.

f. **Labels** : The set of *labels*, L, is the set of pairs $E \times A$, and for $l=(e, a)$ we write e/a. Informally, if e/a is a label of a transition t, then t is triggered by e and a is executed when t is taken.

g. **Transitions** : The set of *transitions*, T, is defined as the set of triples $T \subset 2^S \times L \times 2^{S \cup H}$. A transition $t=(X, l, Y)$ is composed of: a *source set* X, a *target set* Y, denoted

$source(t)$,$target(t)$, respectively, and a label l. Informally, if $l=e/a$, the system is in X and e occurs then t is enabled and can be taken. If t is taken, a is executed and the system is than at Y.

2. Notations and Definitions

Following are formal definitions of basic terms used in the next section. In appendix A we give diagrams illustrating these definitions.

a. A state s is *basic* if $\rho(s)=\emptyset$.

b. For a set of states X, the *Lowest Common Ancestor* of X, denoted $LCA(X)$ is defined as follows: $LCA(X)=x$ iff :

1. $X\subseteq\rho^*(x)$
2. $\forall s\in S \quad X\subseteq\rho^*(s)\Rightarrow x\in\rho^*(s)$

For a set of states X, the *strict Lowest Common OR Ancestor* of X, denoted $LCA^+(X)$ is defined as follows: $LCA^+(X)=x$ iff :

1. $X\subseteq\rho^+(x)$
2. $\psi(x)=OR$
3. $\forall s\in S$ if $\psi(s)=OR$ then $X\subseteq\rho^*(s)\Rightarrow x\in\rho^*(s)$

c. For a transition $t\in T$, $LCA(t)$ is defined as the LCA^+ of its source and target sets of states. That is if $t=(X,l,Y)$ then $LCA(t)=LCA^+(X\cup\omega(Y))$.

d. Two states x,y are *orthogonal*, denoted $x\perp y$, if either $x=y$ or their LCA is an AND state, that is $\psi(LCA(\{x,y\}))=AND$.

e. A set of states X, is an *orthogonal set* if $\forall x,y\in X \quad x\perp y$. Note that $\{x\}$ is an orthogonal set for every $x\in S$.

f. A set X is an *orthogonal set relative to* s if:

1. $X\subset\rho^*(s)$
2. X is an orthogonal set

g. A set X is a *maximal orthogonal set relative to* s if :

1. X is an orthogonal set relative to s.
2. $\forall y\in\rho^*(s),\ y\notin X\Rightarrow X\cup\{y\}$ is not orthogonal.

Note that for every $s\in S$, $\{s\}$ is a maximal orthogonal set relative to s.

h. A *state configuration of* s is an orthogonal set relative to s all of whose members are basic states.

i. A *maximal state configuration of* s is a maximal orthogonal set relative to s all of whose members are basic states.

j. Given a sequence of maximal state configurations relative to the root, $(X_0,\ldots,X_n)$, and a history symbol h with $\gamma(h)=s$ the *history function* $\tau(h,(X_0,\ldots,X_n))$ defines the substate of s which is the "last visit" of the system in s according to $(X_0,\ldots,X_n)$. Formally, let $I=\{i \mid \rho^*(s)\cap X_i\neq\emptyset\}$. If $I=\emptyset$ then $\tau(h,(X_0,\ldots,X_n))=\delta(s)$, otherwise let j be the maximal number in I. Then since s is an OR state there exists a unique $s'\in\rho(s)$ such that $\rho^*(s')\cap X_j\neq\emptyset$, we define $\tau(h,(X_0,\ldots,X_n))=s'$.

k. The target set of a transition may contain non basic states and history symbols. Each step of the system is defined by a set of transitions which should define a maximal state configuration of the root state. Thus, we have to define a state configuration corresponding to a target set of a transition. The function $C(s,X,(X_0,\ldots,X_n))$ where: X is an orthogonal set realtive to s and $(X_0,\ldots,X_n)$ is a sequence of maximal state configurations of the root state, is defined recursively. $C(s,X,(X_0,\ldots,X_n))$ is a maximal state configuration of s computed by repeatedly applying the default and history functions and completing the orthogonal set into a maximal orthogonal set by taking the defaults in orthogonal components. For the formal definition of $C(s,X,\overline{X})$ see appendix B.

l. The *initial state configuration* X_0, is defined to be $C(root,\{root\},\emptyset)$. The initial state configuration is a maximal state configuration of the root state.

m. A set of transitions Υ is *structurally consistent* if $\forall t_1,t_2\in\Upsilon$ $LCA(t_1)\perp LCA(t_2)$. A consistent set of transitions can be taken simultaneously.

n. A transition $t=(Y,l,Z)$ is *structurally relevant* to a state configuration X if for every $y\in Y$ there exists $x\in X$ such that $x\in\rho^*(y)$. If t is structurally relevant to X and the system is in X then t can be taken.

3. Semantics

The semantics is based upon a sequence of time *instant* $\{\sigma_i\}_{i\geq 0}$, corresponding to the sampling rate of the *System Under Desciption* (SUD). The basic time *intervals* are defined by $I_i=[\sigma_i,\sigma_{i+1})$. At σ_{i+1} the SUD *reacts* to *external stimuli* occuring in the interval I_i. The semantics of statecharts is defined by providing the formal definition for changes occuring in the SUD as a reaction to external stimuli. An *external stimulus* associated with σ_{i+1} is a triple (Π,Θ,ξ) where : Π is

set of external primitive events occuring in I_i, Θ is a set of external primitive conditions whose value is *true* at $[\sigma,\sigma_{i+1})$ for some $\sigma \geq \sigma_i$, and ξ is a function, determined by the external environment, such that for a variable v, $\xi(v)=x$ if v's value is x in $[\sigma,\sigma_{i+1})$ for some $\sigma \geq \sigma_i$. A *system configuration* associated with an instant σ_{i+1} is a tuple (X,Π,Θ,ξ) where X is a maximal state configuration of the root state and (Π,Θ,ξ) is an external stimulus associated with σ_{i+1}.

The system reaction at some instant is composed of the set of transitions taken at that instant and the set of events generated when these transitions are taken. Thus, a *system reaction* is a pair (Υ,Π^g) where Υ is a set of transitions called a *step* and Π^g is the set of atomic events *generated* by Υ. Given a system configuration $SC=(X,\Pi,\Theta,\xi)$ we define in this section the set $\{SR=(\Upsilon,\Pi^g)\}$ of *possible system reactions* to SC

Intuitively a step is a set of consistent transitions that are structurally relevant to a given system configuration and enabled under the given external stimuli. The set of generated events is the set of events occuring as a result of taking the step's transitions and executing the actions associated with these transitions. All the transitions participating in a step Υ are taken simultaneously. In our formal definition of a step we sequence the set of taken transitions into *micro-steps* each of which is a subset of Υ. Thus, a step is defined as a sequence of micro-steps. It is important to understand that although micro-steps are essential for the precise definition and understanding of the semantics they are considered as an internal mechanism for computing steps and actually should be hidden from the user. Sophisticated users may use their knowledge about this internal mechanism of micro-steps to define complicated and unusual system reactions. However, in the simplest and most often used cases the system reaction as defined in the following is identical to the intuitive understanding of a step, above. Because of the very rich syntax allowing shared variables and simultaneous assignments to variables in actions the terms "consistent" and "enabled" in the above naive definition should be defined very carefully, and this is the reason for introducing the concept of micro-steps. The reader should remember this simple definition as an intuitive understanding of a step.

Definition : Given a system configuration $SC=(X,\Pi,\Theta,\xi)$ we define below an extension of ξ to V, and we say that v is *evaluated to* x *under* SC if $\xi(v)=x$. ξ is extended to V as follows:

1. If k is a number then $\xi(k)=k$.
2. If $v_1,v_2 \in V$ then $\xi(op(v_1,v_2))=op(\xi(v_1),\xi(v_2))$.
3. If $v \in V$ then $\xi(cr(v))=\xi(v)$.

Definition : Given a system configuration $SC=(X,\Pi,\Theta,\xi)$, SC *satisfies* $c \in C$, denoted $SC \rightarrow c$, is defined inductively as follows :

1. $SC \rightarrow T$, $not(SC \rightarrow F)$.
2. If $c \in C_p$ then $SC \rightarrow c$ iff $c \in \Theta$.
3. If $s \in S$ then $SC \rightarrow in(s)$ iff $X \cap \rho^*(s) \neq \emptyset$
4. If $e \in E$ then $SC \rightarrow ny(e)$ iff $not(SC \rightarrow e)$.
5. If $u,v \in V, R \in \{=,>,<,\neq,\geq,\leq\}$ then $SC \rightarrow u\ R\ v$ iff $\xi(u)\ R\ \xi(v)$.
6. If $c \in C$ then $SC \rightarrow cr(c)$ iff $SC \rightarrow c$.
7. If $c_1,c_2 \in C$ then
   - 7.1. $SC \rightarrow c_1 \wedge c_2$ iff $SC \rightarrow c_1$ and $SC \rightarrow c_2$.
   - 7.2. $SC \rightarrow c_1 \vee c_2$ iff $SC \rightarrow c_1$ or $SC \rightarrow c_2$.
   - 7.3. $SC \rightarrow \neg c_1$ iff $not(SC \rightarrow c_1)$.

Definition : Given a system configuration $SC=(X,\Pi,\Theta,\xi)$, $e \in E$ *occurs at* SC, denoted $SC \rightarrow e$, is defined inductively as follows :

1. $SC \rightarrow \lambda$.
2. If $e \in E_p$ then $SC \rightarrow e$ iff $e \in \Pi$.
3. If $c \in C$ then $not(SC \rightarrow tr(c)), not(SC \rightarrow fs(c))$.
4. If $v \in V$ then $not(SC \rightarrow ch(v))$.
5. If $s \in S$ then $not(SC \rightarrow ex(s)), not(SC \rightarrow en(s))$.
6. If $e_1,e_2 \in E$ then
   - 6.1. $SC \rightarrow e_1 \wedge e_2$ iff $SC \rightarrow e_1$ and $SC \rightarrow e_2$.
   - 6.2. $SC \rightarrow e_1 \vee e_2$ iff $SC \rightarrow e_1$ or $SC \rightarrow e_2$.
7. If $e \in E, c \in C$ then $SC \rightarrow e[c]$ iff $SC \rightarrow e$ and $SC \rightarrow c$.

Given a system configuration SC as above, the SUD reaction is composed of a sequence of micro steps. The first micro step is defined as a set of transitions whose trigger occurs at SC. This first micro step results in a *micro system configuration* to which the SUD may react by another micro step and so forth. A system configuration is a full description of the SUD status in a time instance. The micro system configuration is an abstract term in the sense that it does not describe an actual status of the SUD. In the following it is defined as an extension of the system configuration term, used to defined the abstract SUD status *between* micro steps. Note that by the following definition the only operators that are "sensitive" to the micro step mechanism are the cr, ny operators. A user who does not use these operators will in most cases get a system reaction corresponding to the simple intuitive definition introduced in the begining of this section. In the following, we formally define the micro system configuration term and the terms of satisfy and occur for conditions and events, with respect to a micro system configuration.

Definition : A *micro system configuration* μSC with respect to a system configuration $SC=(X,\Pi,\Theta,\xi)$ is a quadruple $\mu SC=(\mu X,\mu\Pi,\mu\Theta,\mu\xi,\mu Y)$ where :

1. μX is a partial state configuration, $\mu X \subseteq X$.
2. $\mu\Pi \subset E_p \cup \{ex(s),en(s) | s \in S\}$, $\Pi \subseteq \mu\Pi$.
3. $\mu\Theta \subseteq \{cr(c) | c \in C_p\}$.
4. $\mu\xi$ is a function assigning values to current-variables,
5. μY is a partial state configuration. i.e. $\mu\xi$ is a function from $\{cr(v) \mid v \in V_p\}$ to the set of numbers.

Definition : Given $\mu SC=(\mu X,\mu\Pi,\mu\Theta,\mu\xi,\mu Y)$ with respect to a system configuration $SC=(X,\Pi,\Theta,\xi)$, we define below an extension of $\mu\xi$ to V, and we say that v is *evaluated to* x *under* μSC if $\mu\xi(v)=x$. $\mu\xi$ is extended to V as follows:

1. If k is a number then $\mu\xi(k)=\mu\xi(cr(k))=k$.
2. If $v\in V_p$ then $\mu\xi(v)=\xi(v)$.
3. If $v_1,v_2\in V$ then $\mu\xi(op(v_1,v_2))=op(\mu\xi(v_1),\mu\xi(v_2))$.
4. If $v_1,v_2\in V$ then $\mu\xi(cr(op(v_1,v_2)))=op(\mu\xi(cr(v_1)),\mu\xi(cr(v_2)))$.

Note that for any cr-free expression v, $\mu\xi(v)=\xi(v)$.

Definition : Given $\mu SC=(\mu X,\mu\Pi,\mu\Theta,\mu\xi,\mu Y)$ with respect to a system configuration $SC=(X,\Pi,\Theta,\xi)$, μSC *satisfies* $c\in C$, denoted $\mu SC\rightarrow c$, is defined inductively as follows :

1. $\mu SC\rightarrow T$, *not* $(\mu SC\rightarrow F)$.
2. If $c\in C_p$ then:
   - 2.1. $\mu SC\rightarrow c$ iff $SC\rightarrow c$.
   - 2.2. $\mu SC\rightarrow cr(c)$ iff $cr(c)\in\mu\Theta$.
3. If $s\in S$ then:
   - 3.1. $\mu SC\rightarrow in(s)$ iff $SC\rightarrow in(s)$
   - 3.2. $\mu SC\rightarrow cr(in(s))$ iff $\mu X\cap\rho^*(s)\neq\emptyset$ or $\mu Y\cap\rho^*(s)\neq\emptyset$
4. If $e\in E$ then:
   - 4.1. $\mu SC\rightarrow ny(e)$ iff *not* $(\mu SC\rightarrow e)$.
   - 4.2. $\mu SC\rightarrow cr(ny(e))$ iff $\mu SC\rightarrow ny(e)$.
5. If $u,v\in V, R\in\{=,>,<,\neq,\geq,\leq\}$ then:
   - 5.1. $\mu SC\rightarrow u\ R\ v$ iff $\mu\xi(u)\ R\ \mu\xi(v)$.
   - 5.2. $\mu SC\rightarrow cr(u\ R\ v)$ iff $\mu SC\rightarrow cr(u)\ R\ cr(v)$.
6. If $c_1,c_2\in C$ then :
   - 6.1. for $c_1/\backslash c_2\in C$:
     - 6.1.1. $\mu SC\rightarrow c_1/\backslash c_2$ iff $\mu SC\rightarrow c_1$ and $\mu SC\rightarrow c_2$.
     - 6.1.2. $\mu SC\rightarrow cr(c_1/\backslash c_2)$ iff $\mu SC\rightarrow cr(c_1)$ and $\mu SC\rightarrow cr(c_2)$.
   - 6.2. for $c_1\backslash/c_2\in C$:
     - 6.2.1. $\mu SC\rightarrow c_1\backslash/c_2$ iff $\mu SC\rightarrow c_1$ or $\mu SC\rightarrow c_2$.
     - 6.2.2. $\mu SC\rightarrow cr(c_1\backslash/c_2)$ iff $\mu SC\rightarrow cr(c_1)$ or $\mu SC\rightarrow cr(c_2)$.
   - 6.3. for $\neg c_1\in C$:
     - 6.3.1. $\mu SC\rightarrow\neg c_1$ iff *not* $(\mu SC\rightarrow c_1)$.
     - 6.3.2. $\mu SC\rightarrow cr(\neg c_1)$ iff *not* $(\mu SC\rightarrow cr(c_1))$.

Note that for cr,ny-free condition c, $\mu SC\rightarrow c$ iff $SC\rightarrow c$.

Definition : Given two micro system configurations $\mu SC_1=(\mu X_1,\mu\Pi_1,\mu\Theta_1,\mu\xi_1,\mu Y_1)$, $\mu SC=(\mu X,\mu\Pi,\mu\Theta,\mu\xi,\mu Y)$, both with respect to a system configuration $SC=(X,\Pi,\Theta,\xi)$ we say that μSC is a *possible successor* of μSC_1 if:

1. $\mu X\subseteq\mu X_1$.
2. $\mu\Pi_1\subseteq\mu\Pi$.
3. $\mu Y_1\subseteq\mu Y$.

Definition : Given $\mu SC=(\mu X,\mu\Pi,\mu\Theta,\mu\xi,\mu Y)$ which is a possible succesor of $\mu SC_1=(\mu X_1,\mu\Pi_1,\mu\Theta_1,\mu\xi_1)$ both with respect to a system configuration $SC=(X,\Pi,\Theta,\xi)$, $e\in E$ *occurs at* μSC, denoted $\mu SC\rightarrow e$, is defined inductively as follows :

1. $\mu SC\rightarrow\lambda$.
2. If $e\in E_p$ then $\mu SC\rightarrow e$ iff $e\in\mu\Pi$.
3. If $c\in C$ then:
   - 3.1. $\mu SC\rightarrow tr(c)$ iff either $\mu SC_1\rightarrow tr(c)$ or $\mu SC_1\rightarrow cr(\neg c)$ and $\mu SC\rightarrow cr(c)$.
   - 3.2. $\mu SC\rightarrow fs(c)$ iff either $\mu SC_1\rightarrow fs(c)$ or $\mu SC_1\rightarrow cr(c)$ and $\mu SC\rightarrow cr(\neg c)$.
4. If $v\in V$ then $\mu SC\rightarrow ch(v)$ iff either $\mu SC_1\rightarrow ch(v)$ or $\mu\xi_1(cr(v))\neq\mu\xi(cr(v))$.
5. If $s\in S$ then :
   - 5.1. $SC\rightarrow ex(s)$ iff either $\mu SC_1\rightarrow ex(s)$ or $ex(s)\in\mu\Pi$
   - 5.2. $SC\rightarrow en(s)$ iff either $\mu SC_1\rightarrow en(s)$ or $en(s)\in\mu\Pi$.
6. If $e_1,e_2\in E$ then
   - 6.1. $\mu SC\rightarrow e_1/\backslash e_2$ iff $\mu SC\rightarrow e_1$ and $\mu SC\rightarrow e_2$.
   - 6.2. $\mu SC\rightarrow e_1\backslash/e_2$ iff $\mu SC\rightarrow e_1$ or $\mu SC\rightarrow e_2$.
7. If $e\in E, c\in C$ then $\mu SC\rightarrow e[c]$ iff $\mu SC\rightarrow e$ and $\mu SC\rightarrow c$.

Note that since $\lambda[c]\equiv c$ we have $C\subseteq E$.

Lemma : For any $e\in E$, if $\mu SC_1\rightarrow e$ then $\mu SC\rightarrow e$.

In the following, we define the micro step term. Informally, a micro step is a set of transitions whose trigger occur at a given micro system configuration and which can be taken simultaneously without causing any conflicts, either structurally or in values assigned to variables and conditions.

Lemma : If $SC=(X,\Pi,\Theta,\xi)$ is a system configuration then $\mu Sc=(X,\Pi,\Theta,\xi,\emptyset)$ is a micro system configuration with respect to SC. Also, μSC is a possible successor of itself.

Definition : Given a micro system configuration $\mu SC=(\mu X,\mu\Pi,\mu\Theta,\mu\xi,\mu Y)$, two atomic actions a,b are *inconsistent* under μSC if:
either $a\equiv c:=d_1$ and $b\equiv c:=d_2$ for some $c\in C_p$
or $a\equiv v:=u_1$ and $b\equiv v:=u_2$ for some $v\in V_p$
Two atomic actions a,b are *consistent* under μSC if they are not inconsistent.
An action $a=a_0;...;a_n$ is *consistent* under μSC if $\forall i,j$ if $i\neq j$ then a_i,a_j are consistent under μSC.

Note that an atomic action is always consistent.

Definition : Given a set of transitions $\mu\Upsilon=\{t_0,...,t_n\}$, where $t_i=(X_i,(e_i,a_i),Y_i)$, $a=a_0;...;a_n$ is the action *caused* by $\mu\Upsilon$.

Definition : Given a micro system configuration μSC, a set of transitions $\mu\Upsilon$ is *consistent* under μSC if:

1. $\mu\Upsilon$ is structurally consistent.
2. the action caused by $\mu\Upsilon$ is consistent under μSC.

Definition : Given two micro system configuration μSC_1 and $\mu SC=(\mu X,\mu\Pi,\mu\Theta,\mu\xi,\mu Y)$ both with respect to SC, where μSC is a possible successor of μSC_1, and a transition t labeled by $l=e/a$, t is *enabled* under μSC if :

1. t is structurally relevant to μSC.
2. $\mu SC \rightarrow e$.

Definition : Given two micro system configurations μSC_1 and $\mu SC=(\mu X,\mu\Pi,\mu\Theta,\mu\xi,\mu Y)$ both with respect to SC, where μSC is a possible successor of μSC_1 and a nonempty set of transitions $\mu\Upsilon$, $\mu\Upsilon$ is a *micro step* from μSC if :

1. $\mu\Upsilon$ is consistent under μSC.
2. $\forall t\in\mu\Upsilon$, t is enabled under μSC.

A micro step causes changes in the current values of conditions and variables and occurrances of new events. These changes are formally defined as follows :

Definition : Given two micro system configuration μSC_1 and $\mu SC=(\mu X,\mu\Pi,\mu\Theta,\mu\xi,\mu Y)$ both with respect to SC, where μSC is a possible successor of μSC_1 and a micro-step $\mu\Upsilon$ from μSC, where $a=a_0;\ldots;a_n$ is the action caused by $\mu\Upsilon$, with a_i being atomic actions:

1. $\mu\Upsilon$ *generates* $e\in E_p$ iff $a_i\equiv e$ for some i.
2. $\mu\Upsilon$ *generates* $ex(s)$ iff $\rho^*(s)\cap\mu X\neq\emptyset$ and there exists $t\in\mu\Upsilon$ s.t. $s\in\rho^+(LCA(t))$.
3. Given a sequence of state configurations $\overline{X}=(X_0,\ldots,X_n)$ where $X_n=X$, $\mu\Upsilon$ *generates* $en(s)$ iff there exists $t\in\mu\Upsilon$ s.t. $\rho^*(s)\cap C(LCA(t),target(t),\overline{X})\neq\emptyset$ and $s\in\rho^+(LCA(t))$.
4. $\mu\Upsilon$ *assigns* T to $c\in C_p$ iff $a_i\equiv c:=d$ and $\mu SC\rightarrow d$ for some i.
5. $\mu\Upsilon$ *assigns* F to $c\in C_p$ iff $a_i\equiv c:=d$ and $\mu SC\rightarrow\neg d$ for some i.
6. $\mu\Upsilon$ *assigns* x to $v\in V_p$ iff $a_i\equiv v:=u$ and $\mu\xi(u)=x$ for some i.

Lemma : If $\mu\Upsilon$ is a micro step from μSC then :

1. if $\mu\Upsilon$ assigns T (resp. F) to $c\in C_p$ it does not assign F (resp. T) to c.
2. if $x\neq y$ and $\mu\Upsilon$ assigns x to $v\in V_p$ then it does not assign y to v.

Proof : Follows from the consistency of micro-steps.

A micro step also results in a new micro system configuration :

Definition : Let $\overline{X}=(X_0,\ldots,X_n)$ be a sequence of state configurations. Let SC be a system configuration whose state configuration is X_n. Let μSC_1 be a micro system configuration with respect to SC, and let $\mu\Upsilon$ be a micro-step from μSC_1. Then $\mu SC=(\mu X,\mu\Pi,\mu\Theta,\mu\xi,\mu Y)$ is the micro system configuration *reached* by $\mu\Upsilon$ from μSC_1 if:

1. $\mu X=\mu X_1-\mu X_1\cap\{\cup\rho^*(LCA(t))|t\in\mu\Upsilon\}$
2. $\mu\Pi=\mu\Pi_1\cup\{e\mid\mu\Upsilon$ generates $e\}$.
3. $\mu\Theta=\mu\Theta_1\cap\{cr(c)|\,\mu\Upsilon$ does not assign F to $c\}\cup\{cr(c)|\,\mu\Upsilon$ assigns T to $c\}$
4. $\mu\xi(v)=x$ iff $\mu\xi_1(v)=x$ and $\mu\Upsilon$ does not assign any value to v or $\mu\Upsilon$ assign x to v.
5. $\mu Y=\mu Y_1\cup\{\cup C(LCA(t),target(t),\overline{X})|t\in\mu\Upsilon\}$

Lemma : If μSC is reached by $\mu\Upsilon$ from μSC_1 then:

1. for $e\in E_p\cup\{ex(s),en(s)|s\in S\}$, $\mu SC\rightarrow e$ iff either $\mu SC_1\rightarrow e$ or $\mu\Upsilon$ generate e.
2. for $c\in C_p$
   - 2.1. $\mu SC\rightarrow cr(c)$ iff either $\mu SC_1\rightarrow cr(c)$ and $\mu\Upsilon$ does not assign F to c or $\mu SC_1\rightarrow cr(\neg c)$ and $\mu\Upsilon$ assign T to c.
   - 2.2. $\mu SC\rightarrow tr(c)(fs(c))$ iff either $\mu SC_1\rightarrow tr(c)(fs(c))$ or $\mu SC_1\rightarrow cr(\neg c)(cr(c))$ and $\mu\Upsilon$ assign $T(F)$ to c.
3. for $v\in V_p$
   - 3.1. $\mu\xi(cr(v))=x$ iff either $\mu\xi_1(cr(v))=x$ and $\mu\Upsilon$ does not assign any value to v or $\mu\Upsilon$ assign x to v.
   - 3.2. $\mu SC\rightarrow ch(v)$ iff either $\mu SC_1\rightarrow ch(v)$ or $\mu\Upsilon$ assign x to v and $\mu\xi_1(cr(v))\neq x$.

A step is a maximal sequence of micro steps as formally defined in the following :

Definition : Let SC be a system configuration. A *step* Υ from SC is a sequence $(\mu\Upsilon_0,\ldots,\mu\Upsilon_m)$ where :

1. $\mu SC_0=SC$.
2. $\mu\Upsilon_i$ is a micro step from μSC_i, for $i=0,\ldots,m$
3. μSC_{i+1} is the micro system configuration reached by $\mu\Upsilon i$ for $i=0,\ldots,m$.
4. The set $\mu\Upsilon_0\cup\cdots\cup\mu\Upsilon_n$ is a structurally consistent set of transitions.
5. If t is enabled under μSC_{m+1} then $\{t\}\cup\mu\Upsilon_0\cup\cdots\cup\mu\Upsilon_n$ is not structurally consistent (i.e. the sequence is maximal).

A step causes the system to reach a new system configuration, which is actually similar to the last micro system configuration reached by the sequence of micro steps composing the step. Since there are no enabled transitions from this last micro system configuration the SUD "waits" there a time interval for new external stimuli. A step also causes some events to occur. These events together with the new values assigned to conditions and variables are the output of the SUD to the external environment.

Definition : Let SC' be a system configuration and let $\Upsilon=(\mu\Upsilon_0,\ldots,\mu\Upsilon_m)$ be a step from SC' taken at time instant σ_i. $SC=(X,\Pi,\Theta,\xi)$ is the system configuration *reached* by Υ if:

1. $X=\mu X_{m+1}\cup\mu Y_{m+1}$
2. Π is the set of primitive events occuring at time interval i.
3. For $c\in C_p$, c holds at σ_i iff $\mu SC_{m+1}\rightarrow cr(c)$.
   Θ is the set of primitive conditions holding at $[\sigma,\sigma_{i+1})$ for some $\sigma\geq\sigma_i$.
4. For $v\in V_p$, v's value at σ_i is x iff $\mu\xi_{m+1}(cr(v))=x$.
   $\xi(v)=x$ if v's value at $[\sigma,\sigma_{i+1})$ is x for some $\sigma\geq\sigma_i$.

Definition : Let SC be a system configuration at instant i and let $\Upsilon=(\mu\Upsilon_0, \ldots, \mu\Upsilon_m)$ be a step from SC taken at σ_i. The set of *generated events* by Υ is $\Pi^g=\{e \mid e$ generated by $\mu\Upsilon_i$ for some $i\}\cup\{e \mid \mu SC_{m+1}\rightarrow e, e$ atomic $\}$.

Definition : A *run* of the SUD is a sequence $\{(SC_i,\Upsilon_i,\Pi_i^g)\}_{i\geq 0}$ where :

1. $SC_0=(X_0,\Pi_0,\Theta_0,\xi_0)$, X_0 is the initial state configuration, and (Π_0,Θ_0,ξ_0) are the external stimuli occuring at the first time interval I_0.
2. Υ_i is a step taken from SC_i at time instant σ_{i+1} for $i\geq 0$.
3. SC_{i+1} is the system configuration reached from SC_i by Υ_i for $i\geq 0$.
4. Π_i^g is the set of generated events by Υ_i for $i\geq 0$.

Definition : Let SC be a system configuration. The SUD is *non deterministic* in SC if there exist two different reactions $(\Upsilon_1,\Pi_1^g),(\Upsilon_2,\Pi_2^g)$ such that $\Pi_1^g\neq\Pi_2^g$ or $\Upsilon_1\neq\Upsilon_2$.

References

[BDH] A. Bar-Tur, D. Drusinsky and D. Harel, "Using Statecharts for Describing the Communication between Complex Systems". Submitted.

[BC] G. Berry and I. Cosserat, "The ESTEREL Synchronous Programming Language and its Mathematical Semantics", in *Seminar on Concurrency* (S. Brookes and G. Winskel, eds.), Lect. Notes in Comput. Sci., Vol 197, Springer-Verlag, Berlin, 1985.

[DH1] D. Drusinsky and D. Harel, "Using Statecharts for Hardware Description", CS85-06, The Weizmann Inst. of Science.

[DH2] D. Drusinsky and D. Harel, "Statecharts as an Abstract Model for Digital Control Units", CS86-12, The Weizmann Inst. of Science. Submitted.

[H] D. Harel, "Statecharts: A Visual Formalism for Complex Systems", *Science of Computer Programming"* **8** (1987). (Early version appeared as "Statecharts: A Visual Approach to Complex Systems", Weizmann Inst. Tech Report, Feb. 1984.)

[HP] D. Harel and A. Pnueli, "On the Development of Reactive Systems", in *Logic and Models of Concurrent Systems* (NATO ASI Series, Vol. 133, K. R. Apt, ed.), Springer-Verlag, Berlin, 1985, pp. 477-498.

[Ho] C. A. R. Hoare, "Communicating Sequential Processes", *Comm. ACM* **21** (1978), 666-677.

[K] H.-A. Kahana, "Statecharts with Overlapping States", M.Sc. Thesis, Bar-Ilan University, 1986. (In Hebrew.)

[M] R. Milner, *A Calculus of Communicating Systems*, Lecture Notes in Computer Science, Vol. 92, Springer-Verlag, Berlin, 1980.

[P] A. Pnueli, "Applications of Temporal Logic to the Specification and Verification of Reactive Systems: A Survey of Current Trends", in *Current Trends in Concurrency* (de Bakker et al. eds.), Lect. Notes in Comput. Sci., Vol. 224, Springer-Verlag, Berlin, 1986, pp. 510-584.

[R] W. Reisig, *Petri Nets: An Introduction*, Springer-Verlag, Berlin, 1985.

(Note: This paper is actually incomplete, being essentially an extended abstract, and describing only one approach to the formal semantics of statecharts. There are a number of reasons for considering alternative approaches, among which is the desire that a full symmetry exist between the way orthogonal components relate to each other and the way an entire statechart relates to its environment. The present proposal can be seen to fall short of achieving this goal in its entirety. The full version of this paper, which we unfortunately did not manage to complete in time for inclusion in the proceedings, contains a slightly different approach that overcomes this anomaly. On the majority of standard examples, however, both semantics are equivalent.)