Formalization of UML Statechart Diagrams by Hierarchial Transition Systems

Juha Kortelainen[1] and Elina Kuosmanen[2]

[1] University of Oulu, Department of Information Processing Science, Finland,
jkortela@tols16.oulu.fi,
[2] Nokia Mobile Phones, P.O.Box 300, FIN-90401 OULU,
Elina.Kuosmanen@nokia.com,

Abstract. The Uni¯ed Modeling Language (UML) is a standardized notation for describing software systems. The behavior of the system is captured by statechart diagrams, a visual variant of state machines allowing hierarchy and parallelism on states and transitions. The o±cial document [13] speci¯es the dynamic semantics of UML only informally. In this paper we use tools provided by classical automata theory to create a foundation for a formal semantics of UML statecharts. Our work is partially based on the ideas presented earlier by [9] and [12].

1 Introduction

To model concurrent, distributed, real-time, reactive and/or embedded (software) systems is an important issue of contemporary software engineering and computer science. (Semi-)formal modeling languages such as UML ([1], [2], [5],[13]) and STATEMATE toolset ([6], [7], [8]) are created for this purpose. These languages share the following characteristics:

1. They are graphical allowing an 'easy-to-start' approach to modeling.
2. The target system can be described from several viewpoints; di®erent diagrams are applied for the description of views.
3. A gradual sharpening of the model is possible; developing can start from an simple system and then in a stepwise manner more rigorous structures can be achieved.

In the core of these visual modeling languages are statecharts, by which the most important property of the target system, the behaviour of it can be characterized. The statechart diagrams possess important features; the states can

1. be orthogonal (concurrent, parallel),
2. have a multi-level (hierarchical) structure.

Statechart languages were originally designed for everyday industrial use. De¯ning statechart semantics proved to be more complicated than was expected. An e®ort to de¯ne a rigorous semantics for UML statecharts is made in [3], [4], [9], [11], [10], and [15]. For Harel statecharts over twenty di®erent semantics have been proposed ([14], [7]). The most important of them, the STATEMATE semantics, is applying hierarchical automata, sharply formalized in [12]. The main di®erences between classical statecharts and UML statecharts are the following. In UML

² only one input event chosen by the dispatcher is processsed at each point of time;
² interlevel transitions are allowed;
² the trigger part of the transition contains at most one event, negations of events are not allowed;
² if the input event does not enable any transitions, it anyway is consumed;
² in the case of con°icting transitions, the lower level transition has the priority over the upper level one; and
² entry/exit transitions are associated to states.

In UML statechart semantics research modularity and compositionality are often emphasized ([3], [9], [15], [10]). However, since UML allows interlevel transitions, full modularity in UML statecharts and in its semantics is impossible to obtained. On the other hand strong structurality and compositionality in semantic rules easily add unnecessary redundancy, a fact which in concurrent systems easily leads to state explosion.

By combining elements of traditional automata theory we introduce a structure called hierarchical transition system and apply it to model UML state diagrams. Through de¯ning the concept of computation of a hierarchial transition system, we obtain a formal operational semantics for UML statecharts. As becomes clear of the previous, the emphasis is on UML statecharts, with some changes also the STATEMATE semantics can be produced. Our approach is partially based on ideas presented in [9], [10], [12]. The present formalization catches only basic properties of UML statecharts; to model all existing features of this versatile speci¯cation language it has to be developed further.

Our contribution compared to earlier research in this area

² the hierarchial transition system is straighforward to construct from the corresponding state diagram;
² a gradual sharpening of the hierarchial transition system can be carried out simultaneously with the stepwise development of the respective state diagram ;
² the interlevel transitions are not transferred to local ones;
² di®erent priority schemes (other than in UML or classical statecharts can be supported); and
² e±ciency: fast algorithms exist to determine the maximal priority respecting subsets of transitions
² the semantics at present is easy to understand.

The rest of the paper is organized as follows. In Section 2 the concept of hierarchial digraph and certain important state relations as well as their properties are introduced. In the third section we give a structure to a hierarchial transition system, an automata-theoretic model for UML statechart diagrams. The fourth section is devoted to conluding remarks and topics of further work.

2 Preliminaries

In the following some preliminaries, de¯nitions and basic results are given.

A digraph is an ordered pair $D = (V; R)$, where

² V is a nonempty set, the vertices of D
² R µ V £ V is a binary relation of V, the edges of D

An (n-layer) hierarchical digraph, abbreviated as (n-)higraph, is de¯ned as follows.

Let m 2 $\mathbb{N}_+$ and $D_1; D_2; :::; D_m$ be digraphs with pairwise disjoint vertex sets. Then f$D_1; D_2; ::: D_m$g is a 1-layer hierarchical digraph with m parallel components.

Let n 2 $\mathbb{N}$; n ¸ 2. An n-layer hierarchical digraph is an ordered tuple $HD = (V; D; h)$ where

(i) V is a ¯nite set, the vertices of HD;
(ii) D = f$D_{11}; :::; D_{1m_1}; D_{21}; :::; D_{2m_2}; :::; D_{n1}; :::; D_{nm_n}$g is a ¯nite set of digraphs $D_{ij} = (V_{ij}; R_{ij})$ where $j = 1; 2; :::; m_i$, m_i 2 $\mathbb{N}_+$, and $i = 1; 2; :::; n$; moreover, the vertice sets

$$V_{11}; V_{12}; :::; V_{1m_1}; V_{21}; V_{22}; :::; V_{2m_2}; :::; V_{n1}; V_{n2}; :::; V_{nm_n}$$

form a partition of V, i.e., they are nonempty, pairwise disjoint and their union is equal to V; and
(iii) h is the re¯nement function of HD, a mapping: V ! $P(D)$ [3] such that
a) for each i 2 f1; 2; :::; n ¡ 1g

$$\bigcup_{j=1}^{m_i} \bigcup_{v 2 V_{ij}} h(v) = fD_{i+1;1}; D_{i+1;2} :::; D_{i+1;m_{i+1}}g \qquad (1)$$

b) $\bigcup_{j=1}^{m_n} \bigcup_{v 2 V_{nj}} h(v) = ;$; and
c) for each distinct u and v in V we have $h(u)$ \ $h(v) = ;$.

Let the set that contains all edges of the n-higraph be $R = \bigcup_{i=1}^{n} \bigcup_{j=1}^{m_n} R_{ij}$.

The hierarchical structure of a n-higraph $HD = (V; D; h)$ is evident: the digraphs $D_{i1}; D_{i2}; :::; D_{im_i}$ form i^{th} layer of HD $(i = 1; 2; :::; n)$; the vertices of the j^{th} layer digraphs are mapped into a (possibly empty) set of digraphs in the $(j + 1)^{th}$ layer $(j = 1; 2; :::; n$ ¡ $1)$; each vertice of any n^{th} layer digraph is mapped into an empty set and the sets in $h(u)$, $h(v)$ disjoint if u and v are distinct vertices.

Less formally, the vertices of the i^{th} layer the digraphs are in a more detailed fashion characterized by $(i + 1)^{th}$ layer digraphs; for each v 2 V, $h(v)$ is the re¯nement of v. If $h(v)$ contains more than one digraph, these are said to be parallel to each other.

[3] For each set X, we denote by $P(X)$ the set of all subsets of X.

We make the convention that the $(i + 1)^{th}$ layer is lower in the hierarchy than the i^{th} layer.

It had been quite possible to de¯ne hierarchial digraphs through the concept of (hierarchial) digraph term (analogously to statechart term used when formalizing traditional statechart diagrams). However, the approach chosen here, while being at the ¯rst sight maybe a bit complicated, gives a concrete hierarchial structure to the digraph (and thus to the respective UML statechart) and is thus well justi¯ed.

Let $HD = (V; D; h)$ be as above. The relationships between vertices of HD can basically be characterized by three mutually disjoint binary relations of V, namely by ancestor or vertical relation (¢), horizontal relation (2), and parallel relation (k). These are constructed in the following.

The re¯nement function h splits vertices into more detailed entities, digraphs, and produces a tree structure between corresponding vertices. Let ® µ V £ V be a binary relation such that $(u; v)$ 2 ® if v is a vertex of some digraph in $h(u)$. The relation ® is certainly irre°exive.

Let ¢ (£, resp.) be the transitive (re°exive and transitive, resp.) closure of ®. Call ¢ the ancestor relation of HD. If u ¢ v, we say that u is an ancestor of v (or that v is a decendant of u)(in HD).

Clearly u ¢ v if and only if there exist k 2 $\mathbb{N}_+$ and vertices $u_0; u_1; :::; u_k$ such that $u = u_0; v = u_k$ and u_{i+1} is a vertex of a digraph in $h(u_i)$ for $i = 0; 1; :::; k$ ¡ 1.

Let 2 be the binary relation of V such that u 2 v if there exist i 2 f1; 2; :::; ng and j 2 f1; 2; :::; m_ig and distinct vertices $u_1; v_1$ 2 V_{ij} such that u_1 £ u and v_1 £ v. By de¯nition, 2 is certainly symmetric and irre°exive. Call 2 the horizontal relation of HD. If u 2 v, we say that u and v are horizontal vertices.

Two vertices are horizontal, if they belong to the same digraph or if they are decendants of two vertices that belong to the same digraph.

Finally the parallel relation k of HD is de¯ned. Let u k v if there exist vertices u_1 and v_1 such that u_1 £ u and v_1 £ v and either

1. u_1 and v_1 are vertices of distinct digraphs in f$D_{11}; D_{12}; :::; D_{1m_1}$g; or
2. there exists a y 2 V such that u_1 and v_1 are vertices of distinct digraphs in $h(y)$.

Certainly also k is irre°exive and symmetric. If u k v, we say that u and v are parallel vertices.

Parallel vertices belong to the di®erent digraphs in the ¯rst level or belong to the di®erent digraphs in a re¯nement of their common ancestor. Decendants of parallel vertices are also parallel, hence if u k z and u £ u_1 and z £ z_1, then u_1 k z_1.

The following theorem expresses formally the fact that two distinct vertices in a higraph are exclusively either in a ancestor-decendant relation or horizontal or parallel: they form a partition of the set V £ V.

Theorem 1. Let $HD = (V; D; h)$ be a higraph. The binary relations £, ¢ $^{¡1}$, 2 and k are pairwise disjoint and V £ V = £ [¢ $^{¡1}$ [2 [k.

Proof. The proof follows directly from the de¯nitions of the relations.

3 Constructing the hierarchical transition system

Now we introduce the concept of a hierarchical transition system. Informally it is a hierarchical digraph enriched with (interlevel) transitons between sets of vertices.

An (n-layer) hierarchical transition system (n-hts) can be presented as an eighttuple $HTS = (HD; I; E; F; T; sel; join)$ where

(i) $HD = (V; D; h)$ is an n-higraph; the hierarchical digraph of HTS; V is the set of states;
(ii) I is a ¯nite set of enter states, it is a subset of V such that I contains exactly one vertex s from each D in D; s is a enter state of D;
(iii) E is a ¯nite set of events;
(iv) F is a ¯nite set of guard functions;
(v) T is a ¯nite set of transitions;
(vi) sel is the selection relation; and
(vii) $join$ is the join relation.

As can be seen in the characterization above, the set of states of HTS is exactly the set of vertices of the underlying hierarchial digraph. We make the convention that all the concepts and relations concerning the vertices of a higraph are directly generalized to involve the states of an hts. We can thus talk, for instance, about horizontal, vertical or parallel states of an hts.

Before giving a detailed description of each of the entities in the de¯nition of a hierarchial transition system HTS, let us have a general overview of its functioning. One step of HTS consists of the following instantaneous phases: HTS is in one of its global states; using the selection relation a (trigger) event is picked up from the environment and o®ered to HTS which reacts by running a maximal set of acceptable transitions; the global state changes; the aforementioned transitions induce a bunch of new events that are stored to the environment.

The more rigorous description of enter state, event, guard function, transition, selection relation and join relation is as follows.

Let V, D, R and h in $HD = (V; D; h)$ be exactly as in the de¯nition of n-layer hierarchical digraph.

The Enter States. The enter state, as its very name declares, is the default initial state of a digraph, a state ¯rst entered when the digraph takes an active role in a computation of HTS. Thus the set

$$I = fs_{11}; s_{12}; \ldots; s_{1m_1}; s_{21}; s_{22}; \ldots; s_{2m_2}; \ldots; s_{n1}; s_{n2} \ldots; s_{nm_n}g$$

is such that s_{ij} is in the vertice set V_{ij} of the digraph D_{ij} 2 D for each $j = 1; 2; \ldots; m_i$, $i = 1; 2; \ldots; n$. Naturally s_{ij} is called the enter state of the digraph D_{ij}.

The Events. Events are the activating entities of HTS. Let $E = fe_1; e_2; \ldots; e_rg$ where r 2 $I\!N_+$.

In UML the trigger event is picked from the environment; neither how this picking happens nor the (data) structure of the environment is rigorously speci¯ed. We de¯ne an environment of HTS to be any element in the set $E^¤$ [4]. Denote by $\mathcal{E}$ be the set of all environments of HTS; thus $\mathcal{E} = E^¤$.

The status quo of the hierarchial transition system has to be registered at each point of time. The concept of a con¯guration (or a global state) of HTS is thus necessary. A con¯guration contains all the active states of the system in a certain phase of the computation. A subset C of V is a con¯guration of HTS if it has the following properties:

1. C contains a unique vertex from each digraph in $fD_{11}; D_{12}; \ldots; D_{1m_1}g$
2. for each s 2 C, such that $h(s)$ is nonempty, the set C contains exactly one vertex from each digraph in $h(s)$.

Let $\mathcal{C}$ be the set of con¯gurations of the HTS. Informally, a con¯guration contains a thread of states from each parallel component of the hierarchial transition system. The initial con¯guration C_I of HTS is the unique con¯guration containing only enter states of digraphs.

The Guard Functions. In general each guard function in F is a mapping from the set $\mathcal{C} £ \mathcal{E}$ into the set $f0; 1g$. Each transition t contains a guard function f and the transition can be ¯red only if $f(C; x) = 1$ where C is the current con¯guration and x is the current environment. In our considerations the guard functions are interpreted to be Boolean expressions concerning state sets and environments of HTS, i.e., expressions where statements of the form s 2 S (where s is some state variable and S some state set variable) and certain statements concerning the environment are combined together using negation, conjunction and/or disjunction. Whether the Boolean expression reaches the value 1 (true) or 0 (false) can be evaluated in each con¯guration and environment.

The Transitions. The set T of transitions is a ¯nite subset of $P(V) £ F £ (E [f²g) £ E^¤ £ P(V)$. For each $t = (X; f; d; w; Y)$ 2 T the following conditions hold.

(i) X is nonempty and it consists of pairwise parallel states; X is the set of source states.
(ii) Y is nonempty and it consists of pairwise parallel states, Y is the set of target states.
(iii) There exists $u; v$ 2 V such that
a) $(u; v)$ is an edge of some digraph in D;
b) for each x 2 X : u £ x; and

[4] For any set X, let $X^¤$ be the set of all ¯nite sequences (words) $x_1x_2 \ldots x_k$ such that k 2 $I\!N$ and x_i 2 X for each i 2 $f1; 2; \ldots; kg$. If in the sequence $x = x_1x_2 \ldots x_k$ we have $k = 0$, then x is the empty word, denoted by ². If $X = fag$, we write $a^¤$ instead of $fag^¤$.

c) for each y 2 Y : v £ y.
Call u the principal source, v the principal target and (u; v) the principal edge of the transition t. Moreover, t is a re¯nement transition of the edge (u; v).

(iv) f is the guard function of t.
(v) d is the trigger (event) of t; it is possible that e = ² in which case no event of E is needed to ¯re t.
(vi) w 2 E¤ is the event sequence induced by t; w = ² indicates the situation that no events are induced by t.

We assume that for each edge (u; v) 2 R there exists at least one re¯nement transition t. Hence jRj · jTj. The re¯nement transitions more rigorously specify the functioning of the corresponding edge. Just as the lower layer digraphs re¯ne the higher level states, one can think that the transitions are a re¯nement of their principal edges. An edge between states tells that an access exists from the source state to the target state; the properties of the re¯nement transitions characterize in a more accurate manner how the status change is performed.

The Selection Relation. Informally, by using the selection relation the trigger event is chosen from the current environment and o®ered to the hierarchial transition system HT S; the environment is then changed accordingly. If the environment is empty (i.e., it equals empty word), then it certainly does not contain any events and nothing is o®ered. Thus we assume that sel µ E¤ £ (E £ E¤) is a relation between the sets E¤ and E £ E¤ such that

² for each x; e_1; e_2; y_1; y_2, if (x; (e_1; y_1)) and (x; (e_1; y_1)) are both in sel, then (e_1; y_1) = (e_2; y_2); and
² for all e 2 E and y 2 E¤, the element (²; (e; x)) is not in sel.

For (x; (e; y)) 2 sel our interpretation is the following: from the environment x the event e is picked and the environment x is changed to y.

The Join Relation. The environment is updated with the (possibly empty) set of event sequences induced by the simultaneous ¯ring of a (maximal) set of (noncon°icting) transitions. So join µ (E¤ £ P_{FIN}(E¤)) £ E¤. [5] is a relation between the sets E¤ £ P_{FIN}(E¤) and E¤ such that for each (x; X) 2 E¤ £ P_{FIN}(E¤), there exists at most one y 2 E¤ such that ((x; X); y) 2 join. For ((x; X); y) 2 join the interpretation is that the environment x and the ¯nite set of event sequences X are joined together to form the environment y.

Informally, the transition (X; f; e; w; Y) can be ¯red (i.e., it is enabled) if X is a subset of the current con¯guration of HT S, the value of f with the current con¯guration as its argument is 1 (i.e., true) and the trigger e is either ² or the event chosen by the trigger function. If (X; f; e; w; Y) is ¯red then it produces the sequence of events w which is added to the end of the environment. The current con¯guration changes along rules de¯ned later.

[5] For each set X, we denote by P_{FIN}(X) the set of all ¯nite subsets of X.

Let t_1 and t_2 be distinct transitions of HT S with principal sources u_1 and u_2. Then t_1 and t_2 are

1. parallel if u_1 and u_2 are parallel,
2. horizontal if u_1 and u_2 are horizontal,
3. con°icting if either u_1 £ u_2 or u_2 £ u_1.

Remark 1. Two transitions t_1 2 T and t_2 2 T are exclusively either parallel, horizontal or con°icting.

We can model a UML statechart diagram with a hierarchial transition system. The procedure how the hts is constructed from the corresponding UML statechart is not described in this paper. Certainly all the numerous features of the UML statecharts cannot yet be caught by our present hts; e®orts to develop a more developed model are continued in the future.

4 Future work

A formal operational semantics for hierarchial transition systems and thus for a large subset of UML statecharts can be derived on basis of the constructions above. How e±ciently it can be implemented with software tools and exactly how does it di®er from the existing UML statechart semantics are topics of further research.

References

1. Booch G., Rumbaugh J., Jacobson I., The Uni¯ed Modeling Language User Guide, 1998 Reading, Massachusetts, Addison Wesley Longman
2. Douglass , Bruce Powel : Doing Hard Time, Developing Real-Time Systems with UML, Objects, Frameworks, and Patterns, 1999 Reading, Massachusetts, Addison Wesley Longman 2nd ed.
3. Gnesi S., Latella D., and Massink M., Modular semantics for a UML statechart diagrams kernel and its extension to multicharts and branching time model-checking, The Journal of Logic and Algebraic programming 51 (2002), 43-75.
4. Gogolla M., and Presicce, F. P., State diagrams in UML : a forma semantics using graph transformations, in: Proceedings of Workshop on Precise Semantics of Modeling Techniques (PSMT'98), 1998, pp. 55 { 72.
5. Gomaa H., Designing Concurrent, Distributed, And Real-Time Applications With UML ,2000 Upper Saddle River, Addison Wesley Longman
6. Harel D., Statecharts: A visual formalism for complex systems, Science of Computer Programming 8 (1987), 231-274.
7. Harel D., The STATEMATE semantics of statecharts, ACM Transactions on Software Engineering and Methodology 5 (1996), 293- 333. A visual formalism for complex systems, Science of Computer Programming 8 (1987), 231-274.
8. Harel D., and M. Politi, Modeling Reactive Systems with Statecharts, 1998 New York, McGraw Hill

9. Latella D., Majzik I., and Massink M., A simpli¯ed formal semantics for a subset of UML statechart diagrams, Technical Report Hide/T1.2/PDCC/5/v1, ESPRIT Project no. 27439, 1998.
10. Latella D., Majzik I., and Massink M., Towards a formal operational semantics of UML statechart diagrams, in: Ciancarini P., Fantechi A., and Gorrieri R. (eds), Third International Conference on Formal Methods for Open Object-Oriented Distributed Systems, Kluwer Academic Publishers, Dordrecht, 1999, pp. 331-347.
11. Rossi C., Encuso,M., and de Guzm¶an, I. P., Formalization of UML state machines using temporal logic, Software and System Modeling 3 (2004), 31 {54.
12. Mikk E., Lakhnech Y., and Siegel M., Hierarchical automata as model for state-charts, Lecture Notes in Computer Science 1345 (1997), 181- 196.
13. Object Management Group, Inc. OMG Uni¯ed Modeling Language Speci¯cation - version 1.4, 2001
14. von der Beeck M., A Comparison of statechart variants, Lecture Notes in Computer Science 863 (1994), 128-148.
15. von der Beeck M., A structured operational semantics for UML-statecharts, Software and System Modeling 1 (2002), 130 { 141.