第32卷 第22期 计 算 机 工 程 2006年11月
Vol.32 No.22 Computer Engineering November 2006
·博士论文· 文章编号：1000—3428(2006)22—0043—03 文献标识码：A 中图分类号：TP301

基于LTS的Statecharts操作语义研究

钱俊彦，赵岭忠
(桂林电子工业学院计算机系，桂林 541004)

摘 要： Statecharts是一种用于复杂反应式系统行为的可视化规格语言。该文提出了一种基于标签变迁系统(LTS)的Statecharts操作语义描述方法，介绍了Statecharts及其项语法和一步语义，并基于进程代数描述Statecharts的并发行为，使用结构化的操作语义SOS规则描述Statecharts的组合语义，从而得到相应的LTS。

关键词： Statecharts；操作语义；标签变迁系统

Study on Operation Semantics of Statecharts Based on LTS

QIAN Junyan, ZHAO Lingzhong
(Computer Department, Guilin University of Electronic Technology, Guilin 541004)

【Abstract】 Statecharts that extends the finite state machine is a visual language for specifying the behavior of complex reactive system. The state of timed Statecharts is represented by inductive term from kind of term algebra, and a step semantics of timed Statecharts is briefly introduced. Based on process algebra, this paper discusses concurrent behavior for Statecharts by concurrent interleaving sequences. It describes a compositional approach for formalizing the Statecharts semantics directly on sequences of micro steps using labeled transition systems as semantics domain. The results suggest that a concise compositional semantics of timed Statecharts is basal and helpful for model checking Statecharts.

【Key words】 Statecharts; Operation semantics; Labeled transition systems(LTS)

Statecharts[1,2]是一种用于复杂反应式系统行为的可视化规格语言，在有限状态图的基础上增加了一些特性，简而言之：Statecharts=状态图+深度+正交性+广播通信。Statecharts是UML中的一个重要组成部分，主要被用于规约复杂反应式系统，诸如规约通信协议和数字控制单元。

由于反应式系统往往对正确性要求高，因此对Statecharts构造形式化的语义显得非常重要，它不仅有利于描述对象行为的准确性和无二义性，而且有助于系统的正确性和安全性的形式化验证。对于Statecharts形式化语义的研究，人们已经做了大量工作[3~8]，本文通过项代数[7]来表示Statecharts的状态，基于进程代数描述Statecharts的并发行为，然后使用结构化的操作语义(structural operational semantics，SOS)规则描述Statecharts的组合语义，把Statecharts映射为标签变迁系统LTS。

1 Statecharts项语法

图1所示的Statecharts中，圆角矩形框表示状态来描述系统的操作，状态n_1中间用虚线分成两个并发的子状态：n_2和n_3，类似于n_1状态的正交组件称为与状态。状态n_2又可分解为n_4和n_5两个状态，因为模型仅仅能处在n_4和n_5两个状态中的一个，所以n_2被称为或状态。如果状态不能分解成与状态或者或状态，则称作基状态，比如状态n_4和n_5。

为了便于描述和表达，假定迁移名和状态名是独一无二的。采用项[6,7]来描述Statecharts。设定N是Statecharts状态名集合，T是Statecharts迁移名集合，Π是Statecharts事件集合。对任意事件e∈Π，假设¬¬e=def e，并且E⊆Π∪{¬e| e∈Π}，那么有¬E=def {¬e| e∈Π}。Statecharts项集合SC定义如下：

(1)基状态：如果n∈N，那么s = [n]是一个Statecharts项；

(2)或状态：假定n∈N，Statecharts项s1,…,sk，其中k>0，定义向量$\vec{s}$ =def(s1,…,sk)，ρ=def {1,…, k}且l∈ρ，迁移集合$T\subseteq T\times\rho\times 2^{\Pi\cup\neg\Pi}\times 2\Pi\times\rho$，则s=[n: $\vec{s}$;l;T]是一个Statecharts项，其中s1,…,sk是状态s的子状态，s1是缺省状态，而sl是状态s当前的活动的子状态，迁移集合T包含状态s的所有子状态相互连接的迁移；

(3)与状态：如果n∈N，Statecharts项s1,…,sk，k>0，向量$\vec{s}$ =def(s1,…,sk)，那么s=[n: $\vec{s}$]是一个Statecharts项，其中s1,…,sk是状态s的并发子状态。

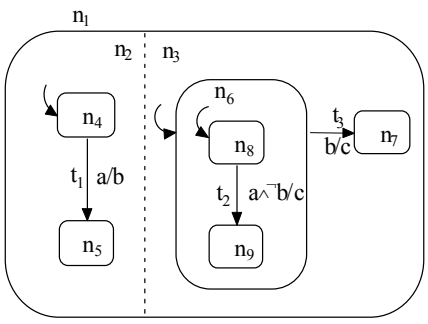

图1 Statecharts的简单例子

对于或状态项[n: $\vec{s}$;l;T]的迁移可用如下形式来表示：$\hat{t}$ =⟨t, i, E, A, j⟩，其中t是迁移名，即name($\hat{t}$) =def t；si是$\hat{t}$ 的源状态，即source($\hat{t}$) =def si；E是$\hat{t}$ 的触发器，即trg($\hat{t}$)

基金项目： 广西自然科学基金资助项目(0542036)
作者简介： 钱俊彦(1973－)，男，讲师、博士生，主研方向：软件工程，模型检验，嵌入式实时系统；赵岭忠，博士生
收稿日期： 2005-11-28 **E-mail：** qjy2000@gliet.edu.cn

=def E；A 是 $\hat{t}$ 的动作，即 Act($\hat{t}$) =def A；sj 是 $\hat{t}$ 的目标状态，即 target($\hat{t}$) =def sj。通过以上描述，图 1 中的 Statecharts 项描述如下所示：

s1 = [n1: (s1, s2)]
s2 = [n2: (s4, s5), 1, {t1, 1, {a}, {b}, 2}]
s3 = [n3: (s6, s7), 1, {t3, 1, {b}, {c}, 2}]
s6 = [n6: (s8, s9), 1, {t2, 1, {a∧ ¬b}, {c}, 2}]
s4= [n4] s5=[n5] s7=[n7] s8=[n8] s9=[n9]

2 Statecharts 一步语义

上面描述了Statecharts项语法，以下将参考文献[8]来表述Statecharts一步语义，它是在Pnueli和Shalev[4]语义的基础上略作了一些变化。从一个稳定的状态格局到达下一个稳定的状态格局，称为一个宏步。精确地说，一个宏步是一系列满足相关性、一致性、触发性和因果性的微步(或迁移)的最大集合。给定s∈SC，t是s的一个迁移，T是s的一个迁移集合，以及E⊆Π，那么迁移允许需要满足下式：

En(s, E, T) =def relevant(s)∩consistent(s, T)∩triggered(s, E∪$\bigcup_{t\in T}$**act**(t))

其中：

· relevant(s)表示一系列迁移，迁移的源状态在当前状态格局中；

· consistent(s, T)表示一系列与 T 没有任何冲突的迁移；

· triggered(s, E)表示一系列迁移，它的触发满足事先规定好的 E 事件集；

· act(t)表示由迁移 t 产生的事件。

通过上述定义，允许迁移的构造算法为：

```
function step_construction(s, E);
var T :=Ø;
while T⊆En(s, E, T) do
    choose t∈En(s, E, T)\T;
    T := T∪{t};
od;
return T;
```

给定一个 Statecharts 项 s 和事件集合 E，通过算法 step_construction 计算出迁移集合 T*，执行 T*后产生事件 A=$\bigcup_{t\in T^*}$**act**(t) ，并改变每一个或状态[n: $\vec{s}$;l;T]中的索引 l，到达 Statecharts 项 s′，即 s $\xrightarrow{E\atop A}$ s′，简写为(s, E, A, s′)。

3 基于 LTS 的 Statecharts 组合语义

为了描述 Statecharts 语义，使用进程代数来描述 Statecharts 的并发行为，在此基础上，通过状态转换规则，把 Statecharts 转换为标签变迁系统 LTS。LTS 是一个四元组(S, Σ,→, s0)，其中 S 表示状态集合，Σ表示标签集合，→⊆ S×Σ×S 表示转换关系，s0 表示初始状态。给定一个转换(s, σ, s′)∈→，其中σ∈Σ，通常简写为 s $\xrightarrow{\sigma}$ s′。

3.1 Statecharts 的组合操作语义

在定义状态转换操作规则之前，首先定义 $\vec{s}_{l\to s'}=_{\text{def}}(s_1,...,s_{l-1},s',s_{l+1},...,s_k)$，其中 1≤l≤k，且 s′∈SC。定义操作规则中需要用到的 Statecharts 缺省状态 default 函数，对于基状态 default([n])=def[n]，或状态 default([n: $\vec{s}$;l;T]) =def default(s1)，与状态 default([n: $\vec{s}$])=def$\bigcup_{1\le i\le k}$ default(si)。

如果考虑历史状态情况，可分为两种情况：深度历史和浅度历史。定义历史状态标志τ∈{none，deep，shallow}，其中 none 表示不考虑历史状态的通常情况，deep 表示深度历史，shallow 表示浅度历史。缺省状态函数 default(s)用 default(τ, s)的形式来代替。对于不考虑历史状态和深度历史的情况，处理相对比较简单，可分别定义 default(none，s)=def default(s)，default(deep, s) =def s。对于浅度历史的情况，缺省状态函数 default 需根据不同状态类型进行重新定义，基状态 default(shallow, [n])=def[n]，或状态 default(shallow, [n: $\vec{s}$;l;T]) =def [n: $\vec{s}_{[l\to\text{default}(s_l)]}$;l;T]，与状态 default(shallow, [n: $\vec{s}$]) =def default(shallow, $\vec{s}$)。

转换关系 → 采用plotkin[10]的结构化操作语义SOS风格来描述，SOS规则的形式如下：

规则名 $\dfrac{\text{前提}}{\text{结论}}$ 或 规则名 $\dfrac{\text{前提}}{\text{结论}}$ (边缘条件)

(边缘条件)

根据上述的项语法和一步语义，以及 Statecharts 的进程代数描述，定义 Statecharts 或状态和与状态的状态转换操作规则。对于或状态，可分为两种情况：一种情况表示从某个或状态转换到另一个兄弟状态，如 OR-1 所示，OR-1 规则描述或状态[n: $\vec{s}$;i;T]迁移 t∈T 的一个执行情况，其中 source(t) = si，target(t) = sl，表示当前活动子状态为 si 转换到当前活动子状态为 sl 的或状态[n: $\vec{s}_{[l\to\text{default}(\tau,s_l)]}$;l;T]；

OR-1：

$$\frac{-}{[n:\vec{s};i;T]\xrightarrow[\text{act}(t)]{\text{trg}(t)}_{\{\text{name}(t)\}}[n:\vec{s}_{[l\to\text{default}(\tau,s_l)]};l;T]\ (t\in T)}$$

另一种情况表示某个或状态的子状态之间相互转换，如 OR-2 所示。OR-2 规则描述当前活动子状态为 s_l 的或状态 $[n:\vec{s};l;T]$ 转换到具有同样序号的当前活动子状态为 s_l' 的或状态 $[n:\vec{s}_{[l\to s_l']};l;T]$；

OR-2：

$$\frac{s_l\xrightarrow[A]{E}_L s_l'}{[n:\vec{s};l;T]\xrightarrow[A]{E}_L[n:\vec{s}_{[l\to s_l']};l;T]}$$

为了描述与状态多个迁移允许并发执行的情况，使用进程代数[9]的归并运算符||、左归并运算符⌊及通信归并运算符|。进程x||y表示该进程并发执行进程x和进程y；x⌊y与前者类似，不过限制第 1 步必须从进程x开始；x|y表示x与y需进行同步通信。图 1 所示的状态n_1，包含两个并发的子状态n_2和n_3。假设状态格局处于n_4和n_8，此时事件a和b满足，根据迁移允许的公式，则迁移t_1和t_3允许，由于t_1和t_3分别属于并发状态n_2和n_3，因此二者并发执行，描述为$t_1||t_3$，根据上述进程代数并发的公理，从而使之交错，则$t_1||t_3=(t_1⌊t_3+t_3⌊t_1)+t_1|t_3$。AND规则描述与状态$[n:\vec{s}]$中诸如sm的所有子状态下的迁移允许，表示|M|个子状态并发执行，使用进程代数中的归并进行描述[n: $\vec{s}_1'$]||…||[n: $\vec{s}'_{|M|}$]，从而使之交错。

AND：

$$\frac{(\forall m\in M: s_m\xrightarrow[Lm]{Em\atop Am}s_m')\wedge(\forall i,j\in M:(\neg\text{trg}(t_i))\cap\text{act}(t_j)=\varnothing)}{[n:\vec{s}]\xrightarrow[\bigcup_{m\in M}A_m]{(\bigcup_{l\in H}(E_l\setminus\bigcup_{j=1}^{l-1}\text{act}(t_j)))}\bigcup_{m\in M}L_m\ [n:\vec{s}_1']||...||[n:\vec{s}'_{|M|}]}$$

$$(M\subseteq K=\{1,...,k\},\ H=\{1,...,|M|\})$$

3.2 Statecharts宏步表示

给定一个 Statecharts 项表示 SC，可用映射函数ψ:SC→LTS 等价为标签变迁系统 LTS。假设 p 是一个

Statecharts 项表示 SC，定义 C(p)、Σ(p)、→p 和 Δp 分别是 LTS ψ(p)的状态集合、标签集合、转换关系和初始状态。其中 C(p)描述 Statecharts 项集合，可定义为 C([n]) = {{n}}，C([n: $\vec{s}$;l;T]) = $\bigcup_{1\le i\le k}$ {{n}∪qi | qi∈C(si)}，C([n: $\vec{s}$]) = {{n}∪$\bigcup_{1\le i\le k}$ qi | qi∈C(si)}；Σ(p) = $2^{\Pi p\cup\neg\Pi p}\times 2^{\Pi p}$ 描述 Statecharts 的事件和动作，可写为事件/动作；→p⊆C(p)×Σ(p)×C(p)描述 Statecharts 的顺序微步，转换规则如 3.1 节所述；Δp 描述 Statecharts 初始项集合，定义为 Δ([n]) = {n}，Δ([n: $\vec{s}$;l;T]) = {n}∪s1，Δ([n: $\vec{s}$]) = {n}∪$\bigcup_{1\le i\le k}$ si。

通过上述的定义，可把 Statecharts 等价为标签变迁系统 LTS。让 LTS 的状态来描述 Statecharts 项集合，LTS 的标签描述 Statecharts 的事件和动作，以及 LTS 的转换描述 Statecharts 的顺序微步，从而得到对应于 Statecharts 的 LTS 模型(C(p), Σ(p), $\rightarrow_p$, Δp)，其中：

(1) C(p)是一个状态集合，描述 Statecharts 的项集合，且 C(p)⊆ 2p；

(2) Σ(p)： $2^{\Pi p\cup\neg\Pi p}\times 2^{\Pi p}$ 是一个标签集合，描述 Statecharts 的事件和动作；

(3) →p ⊆C(p)×($2^{\Pi p\cup\neg\Pi p}\times 2^{\Pi p}$)×C(p)表示转换关系，描述 Statecharts 的顺序微步；

(4) Δp 是初始状态。

综上所述，可得到图 1 中 Statecharts 的 LTS 描述，如图 2 所示。

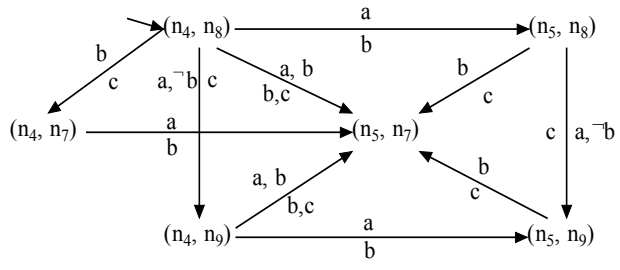

图 2 Statecharts 的 LTS 描述

4 结论

Statecharts 是一种直观、易用且功能强大的可视化规格语言。Statecharts 不仅已成为面向对象软件开发方法的重要组成部分，被用于诸如 OMT、ROOM、UML 中，而且也被集成到一些计算机辅助软件工程环境中，诸如 STATEMATE、ObjectTime Developer、RATIONATROSE 等。由于 Statecharts 应用广泛，研究其精确语义，具有非常重要的意义。

本文基于 Statecharts 状态的项表示，及其一步语义，给出了 Statecharts 的结构化操作语义，为其形式化验证奠定基础。今后的工作将利用本文给出的形式化语义，开发出模型检验工具，应用于实际系统开发中，从而进一步完善我们的工作。

参考文献

1 Harel D. Statecharts: A Visual Formalism for Complex Systems[J]. Science of Computing, 1987, 8(3): 231-274.

2 Harel D, Politi M. Modeling Reactive Systems with Statechart: The Statemate Approach[M]. Mc Graw-Hill, 1996.

3 Harel D, Pnueli A, Schmidt J P, et al. On the Formal Semantics of Statecharts[C]. Proceedings of the 2nd IEEE Symposium on Logic in Computer Science, Ithaca, New York, 1987: 54-64.

4 Pnueli A, Shalev M. What Is in a Step: On the Semantics of Statecharts[C]. Proc. of Theoretical Aspects of Computer Software, 1991: 244-264.

5 Lüttgen G, Beeck M V D, Cleaveland R. Statecharts via Process Algebra[C]. Proc. of the 10th International Conference on Concurrency Theory, 1999: 399-414.

6 Beeck M V D. A Concise Compositional Statecharts Semantics Definition[C]. Proc. of FORTE/PSTV, Kluwer, 2000.

7 Lüttgen G, Beeck M V D, Cleaveland R. A Compositional Approach to Statecharts Semantics[C]. Proceedings of the 8th ACM SIGSOFT International Symposium on Foundations of Software Engineering: Twenty-first Century Applications, 2000.

8 Maggiolo S A, Peron A, Tini S. Equivalences of Statecharts[C]. Proc. of the 7th International Conference on Concurrency Theory, 1996: 687-702.

9 Fokkink W. Introduction to Process Algebra[M]. Springer, 2000.

10 Plotkin G D. A Structural Approach to Operational Semantics[R]. Computer Science Department, Aarhus University, Technical Report: DAIMI FN-19, 1981.

(上接第 9 页)

5 结论

UP 的提出是对 Musa 操作剖面的有力扩展，为操作剖面增加了动态的使用信息，使可靠性测试数据生成更加接近于用户的真实使用。从某种角度来看，Musa 提出的操作剖面可以看作 UP 的一个特例。利用面向对象技术对 UP 的封装，增强了方法的可实施性和灵活性，使信息得以复用，提高了测试效率。目前根据 UP 方法开发出了软件可靠性测试数据自动生成工具 TCS，并应用于实际工程的软件可靠性测试中。

参考文献

1 陆民燕，陈雪松. 软件可靠性测试及其实践[J]. 测控技术，2000, 19(5).

2 Musa J D. Introduction to Software Reliability and Testing, Software Reliability Engineering——Case Studies[C]. Proceedings of the 8th International Symposium on Software Reliability and Testing Course, USA, 1997: 3-12.

3 Musa J D. Software Reliability Engineering[M]. New York: McGraw-Hill, 1998.

4 Michael L. Handbook of Software Reliability Engineering[M]. McGraw-Hill and IEEE Computer Society Press, 1996.

5 Elbaum S, Narla S A. Methodology for Operational Profile Refinement[C]. Proceedings of IEEE Annual Reliability and Maintainability Symposium, 2001: 142-149.