Técnicas Formais e Informais: Um Estudo sobre Integração usando Statecharts

Carlos Henrique C. Duarte   Edward H. Haeusler

[carlos,hermann]@inf.puc-rio.br

Departamento de Informática
Pontifícia Universidade Católica do Rio de Janeiro
R. Marquês de S. Vicente, 225, Rio de Janeiro, RJ, 22453, Brazil

Resumo

Grande esforço tem sido feito buscando determinar técnicas que tornem o desenvolvimento de programas tratável. Statecharts é um formalismo visual que tem sido bastante utilizado, procurando eliminar os problemas normalmente encontrados nesta atividade. Este trabalho trata o problema de maneira diferente, integrando desta técnica formal com outra informal. O raciocínio formal subjacente a este processo de integração também é apresentado, mostrando que um método para integração pode ser estabelecido, baseado na escolha de um mesmo modelo semântico para dar significado às linguagens de especificação utilizadas.

Abstract

Large efforts have been done to find techniques which turn software development tractable. Statecharts is a visual formalism that have been widely used, intending to eliminate the problems in this activity. This paper presents a different solution for the problem, by integrating this formal technique to other informal one. Either, formal rationale about the integration is presented, showing that an integration method can be established, adopting the same semantic model to assign meaning to the specification languages used.

1 Introdução

Atualmente, tornou-se um desafio encontrar técnicas que permitam que programas sejam desenvolvidos de maneira tratável. Usando as propostas existentes até o momento, ainda existe espaço para que sejam criadas especificações bastante incompletas. Além disso, estas técnicas não impedem que a construção de programas se torne muito complexa, naqueles casos onde existem restrições de concorrência ou de tempo real, por exemplo.

Statecharts é uma técnica visual proposta para atacar estes problemas. São extensões dos diagramas de estados e transições que permitem a especificação de estados compostos, possuindo conjuntos de estados componentes que especificam de forma explícita processos seqüenciais ou concorrentes, e não somente sob a forma de não-determinismos. Além disso, em Statecharts, transições estão associadas a eventos de entrada e de saída, onde estes últimos interferem na ocorrência dos primeiros. Estas extensões diminuem a complexidade das especificações e permitem que os sistemas sejam modelados de forma mais adequada.

Além de possuir estas características, Statecharts é uma técnica formal [Har80]. Em oposição à definição adotada por muitos autores, que associam o termo formal apenas ao caráter "mathematical-sound" das especificações, considera-se que o uso deste termo indica que todos os elementos da linguagem de especificação possuem semântica formal, definida através de algum modelo semântico. Por outro lado, técnicas informais são aquelas que possuem uma linguagem onde existem elementos que não tem significado fixo definido pela semântica.

A vantagem das técnicas formais de especificação sobre as informais é a forma rigorosa com que o problema é tratado, não permitindo que sejam feitas definições ambiguas ou vagas, e levando à construções que são computáveis, devido a sua natureza formal. Entretanto, a consideração de que técnicas formais são ideais para desenvolvimento de programas é falaciosa, em muitos casos. Usando técnicas informais, estratégias de abstração geralmente permitem que alguns aspectos sejam dei-xados de lado, e ambiguidades são toleradas durante o processo de desenvolvimento. Isto ocorre por uma questão organizacional (divisão de complexidade) e econômica (alto tempo gasto na construção de elaborações formais). Portanto, a integração entre estas técnicas é, em muitos casos, aconselhável.

Este artigo apresenta o racionínio formal subjacente a uma integração entre técnicas de desenvolvimento de programas, realizada em [CHCDL92]. Generalizando estas idéias, pretende-se descrever aqui um método para integração entre técnicas formais e informais, usando como exemplo Statecharts e Diagramas de Fluxo de Dados (D.F.D.s) [DeM89]. Este método fundamenta-se na introdução de redundâncias de informação e de restrições de integridade entre as especificações criadas usando as duas técnicas, e ainda na criação de uma semântica formal para ambas usando um mesmo modelo semântico.

Na próxima seção a semântica de Statecharts é discutida e definida formalmente, usando prosseguimentos [Jon91]. Na seção 3 é feita a integração de Statecharts com D.F.D.s, e o enfoque utilizado é analisado em comparação com outros trabalhos. Ao final do texto são apresentadas as conclusões do trabalho.

2 Semântica para Statecharts

Diagramas de Estados e Transições sempre foram usados em Engenharia de Software, principalmente na especificação de interfaces, programas, linguagens e protocolos. Statecharts podem ser utilizados (pelo menos) com os mesmos objetivos, já que são uma extensão desta técnica.

Entretanto, em cada um desses casos, o significado dos termos desta linguagem de especificação pode sofrer alterações. Por exemplo, um estado pode representar um estágio do diálogo entre o usuário e o programa [CHCDL92], ou uma etapa em um modelo do processo de desenvolvimento de programa [CMO92]. Portanto, antes de qualquer formalização, é necessário que a utilização da técnica (e a semântica da linguagem de especificação) esteja bem definida.

Na primeira parte desta seção, a técnica Statecharts é definida precisamente, de maneira pragmática. Em seguida a sintaxe e a semântica formal da linguagem de especificação são definidas. Todas estas definições são feitas de acordo com as características prescritas pelo método apresentado na próxima seção.

2.1 Sintaxe e Semântica informal

Por ser uma técnica visual, Statecharts possuem uma linguagem de especificação gráfica, composta basicamente por estados (cada um possuindo um nome) e transições (associadas a condições e eventos de saída). A figura 1 exemplifica uma construção genérica deste tipo.

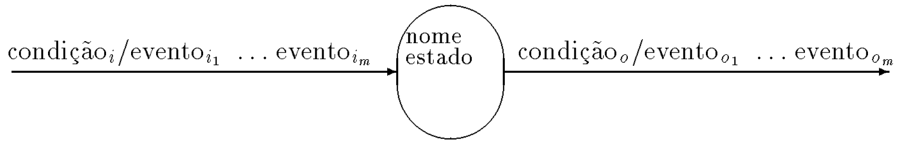

Figura 1: Estados e transições

Condições representam o teste se algum evento de entrada ocorreu, ou expressões lógicas que agrupam conjuntos desses testes. Eventos de entrada representam o resultado de uma interação do programa com o ambiente. Eventos de saída correspondem a alteração de alguma variável de comunicação ou a ativação de uma atividade do sistema. Decorre daía seguinte restrição de integridade:

> **Restrição 1**:
> As atividades ativadas como eventos de saída não possuem interação com o ambiente ou são representadas pelo estado composto de destino da transição.

Em uma transição, nada impede, por exemplo, que a não ocorrência de um evento ($\neg\, a$) seja usada como condição de disparo, e ao mesmo tempo que este (a) seja gerado como evento de saída. Já que considera-se instantânea a ocorrência de transições, uma escolha cuidadosa da semântica deve ser feita, procurando evitar que sejam especificados paradoxos [HdR91], em casos como este. A solução para este problema é dada separando em passos a execução dos processos especificados pelo Statechart, onde cada passo é o conjunto de todos os disparos de transição e as alterações correspondentes dos estados do sistema. Neste trabalho (como em [HRdR92]), ($\neg\, a$) representa que o evento (a) nunca ocorre neste passo, e assim a transição mencionada acima nunca será disparada.

Existem dois tipos de estados compostos:

- OR: o estado contém um conjunto de estados constituintes que representam o comportamento de um processo sequencial, e
- AND: o estado é dividido em partições ortogonais, que são conjuntos disjuntos de estados constituintes, cada qual representando um processo sequencial. O conjunto de partições representa a ocorrência simultânea dos respectivos processos.

Nestes dois casos, tanto estados como partições devem possuir um nome que os identifique unicamente. Na figura 2 são representados estes casos.

Toda especificação ou estado composto possui um estado inicial (chamado default). É possível também especificar que inicial é o último estado pelo qual o processo especificado passou, antes que o controle fosse desviado para outro ponto do programa. Esta característica é chamada história [MM91], e pode ser propagada para todos os estados que constituem algum outro, contido no estado composto em questão. Tais especificações devem ser representadas como na figura 3.

2.2 Semântica Formal

Para dar uma semântica formal a Statecharts, utiliza-se aqui semântica denotacional [All86]. Embora outras definições denotacionais existam (p. ex. [HRdR92]), neste trabalho consideram-se também as extensões à linguagem, introduzidas devido a necessidade de integração com outra

Figura 2: Estados Compostos: (a) OR (b) AND

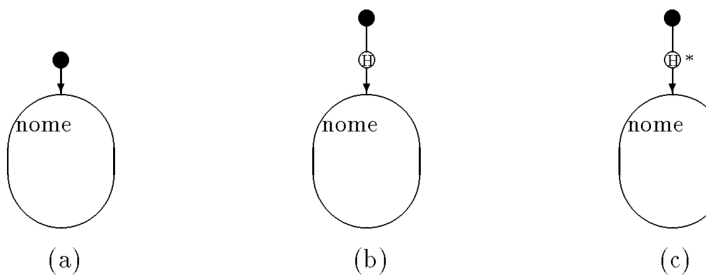

Figura 3: Estado (a) default (b) com história (c) e propagação

técnica. Para que haja uma simplificação nas definições, não são considerados os conceitos de história e propagação.

Numa definição denotacional é necessário escolher meta-variáveis para representar os objetos nas especificações, e definir qual a estrutura desses objetos em forma textual. Esta estrutura textual é chamada sintaxe abstrata da linguagem de especificação. Tal sintaxe é usada na criação de equações semânticas, que associam uma especificação a seu significado, dado em função de domínios semânticos. Estes últimos dão significado intensional aos programas especificados. A seguir a semântica formal de Statecharts é descrita em detalhes usando estas definições.

Meta-variáveis

Entre as meta-variáveis, Ψ representa um statechart completo. Δ representa uma transição, com todas as informações associadas a ela, que são expressões (Θ) e eventos de saída (Π). Todas as informações adicionais, como nomes de estado, de evento e de variáveis são representadas por Ξ.

$$
\begin{array}{lcllcl}
\Psi & \in & \mathbf{ESP} & = & \text{especificações} \\
\Delta & \in & \mathbf{DEC} & = & \text{transições} \\
\Theta & \in & \mathbf{EXP} & = & \text{expressões} \\
\Pi & \in & \mathbf{EVT} & = & \text{eventos} \\
\Xi & \in & \mathbf{VAR} & = & \text{variáveis}
\end{array}
$$

Sintaxe Abstrata

Como a linguagem associada a técnica Statecharts é grafica, para que seja criada uma definição semântica, associa-se cada objeto do espaço visual a um objeto textual. Assim, a sintaxe abstrata colocada abaixo é equivalente a utilizada na construção de diagramas [HRdR92], e permite que raciocínios formais sejam feitos na forma textual convencional.

$$I ::= \Xi_{tsn_{i_1}} \ldots \Xi_{tsn_{i_m}}$$
$$O ::= \Delta_1 \ldots \Delta_n$$

$$\Pi_{in} ::= \lambda \mid in(\Xi_{evt}) \mid tm(\Xi_{evt}, \mathbf{N}) \mid \Xi_{var} = \mathbf{N}$$
$$\Pi_{out} ::= \Xi_{var} \leftarrow \mathbf{N} \mid exec(\Xi_{atv}) \mid out(\Xi_{evt})$$
$$\Pi ::= \Pi_{in} \mid \Pi_{out}$$

$$\Theta ::= \Pi_{in} \mid \Xi_{var} = \mathbf{N} \mid \neg\, \Theta \mid \Theta_1 \vee \Theta_2 \mid \Theta_1 \wedge \Theta_2$$

$$\Delta ::= (\Xi_{tns}, \Theta / \Pi_{out_1} \ldots \Pi_{out_m})$$

$$\Psi ::= \Psi_{Disj} \mid \Psi_{Conj}$$
$$\Psi_{Disj} ::= \Psi_{Prim} \mid Or(\Psi_{Disj_1}, \Psi_{Disj_2}) \mid Connect(\Psi_{Disj}, \Xi_{tns_1}, \Xi_{tns_2})$$
$$\Psi_{Conj} ::= \Psi_{Default} \mid And(\Psi_{Default}, \Psi_{Conj})$$
$$\Psi_{Prim} ::= \Psi_{Basic} \mid \Psi_{Default} \mid HiCl(\Psi, \Xi_{evt})$$
$$\Psi_{Default} ::= Stat(\Psi_{Basic}, \Psi, \Xi_{tns})$$
$$\Psi_{Basic} ::= [I, O, \Xi_{sta}]$$

Considera-se uma especificação básica Ψ_{Basic} um estado de nome Ξ_{sta} com um conjunto de nomes de transições de entrada $\Xi_{i_1}, \ldots, \Xi_{i_m}$ e conjunto de declarações de transições de saída $\Delta_1 \ldots \Delta_n$. Este elemento é usado indutivamente na construção de equações que dão significado ao Statechart.

Nesta sintaxe aparecem ainda construções que não foram definidas anteriormente. $Stat(\Psi_{Basic}, \Psi, \Xi)$ indica que Ψ_{Basic} é um estado que possui Ψ como constituinte e Ξ como transição inicial (default). $Connect(\Xi_{Disj}, \Xi_{tsn_1}, \Xi_{tsn_2})$ representa a conecção da transição de saída Ξ_{tsn_2} com a transição de entrada Ξ_{tsn_1}, ligadas à especificação Ψ_{Disj}. Já *HlCl* indica que todas as gerações do evento Ξ a partir de Ψ serão ocultadas dentro dele, tornando este estado insensível à ocorrência deste evento externamente. *And* e *Or* tem significado similar ao definido na seção anterior.

Entre as condições associadas às transições, λ é um evento que ocorre a cada passo. *tm* representa a ocorrência de um time-out, gerado pela não ocorrência do evento Ξ nos últimos **N** passos. Os eventos de saída permitidos são a geração de um evento Ξ qualquer, a atribuição de um valor a uma variável de comunicação ou a execução de uma atividade, especificada em um D.F.D. (de acordo com a restrição 1 colocada na seção anterior).

Domínios Semânticos

Domínios semânticos são objetos que representam como funcionam os programas especificados corretamente usando a sintaxe abstrata da linguagem. Utiliza-se neste trabalho os domínios listados abaixo, que são também os utilizados normalmente para dar significado às linguagens formais convencionais (a exceção de **RES**).

$$
\begin{array}{lllll}
\alpha & \in & \mathbf{LOC} & = & \{\mathrm{N}\} \\
\zeta & \in & \mathbf{ARQ} & = & \mathbf{LOC} \rightarrow \mathbf{TIE} \\
\tau & \in & \mathbf{TIE} & = & \{\mathrm{N}\} + \{\mathrm{B}\} + \mathbf{ARQ} \\
\sigma & \in & \mathbf{STA} & = & \mathbf{LOC} \rightarrow (\mathbf{TIE} + \mathsf{livre}) \\
\delta & \in & \mathbf{RES} & = & \mathbf{STA} \rightarrow \mathcal{P}(\ \mathbf{STA} \times \mathbf{RES}\ ) \\
\upsilon & \in & \mathbf{COD} & = & \mathbf{AMB} \rightarrow (\ \mathbf{STA} \rightarrow \mathbf{LOC}\ ) \\
\pi & \in & \mathbf{TRA} & = & \mathbf{COD} \rightarrow (\ \mathbf{TIE} \rightarrow (\ \mathbf{STA} \rightarrow \mathbf{RES}\ )\ ) \\
\rho & \in & \mathbf{AMB} & = & \mathbf{LOC} \rightarrow (\ \mathbf{TIE} + \mathbf{COD} + \mathbf{TRA})
\end{array}
$$

Devido ao fato de cada statechart especificar um programa concorrente, é necessário o uso de um domínio semântico chamado prosseguimento (**RES**) [Jon91]. Cada instância deste é composta por um estado e pelo conjunto de estados que possivelmente irão suceder a este, se assemelhando a uma árvore de execução, onde cada nó é um estado pelo qual o programa passa. Em cada passo no caminhamento através dessa "árvore" é determinada a computação que está para ser executada, considerando a possibilidade de interferência externa. Por exemplo, *stop* é o prosseguimento que representa um programa que não faz nada, e é definido por

$$stop \triangleq \lambda\sigma \cdot \{\,\}$$

stop define que, para qualquer estado, nada ($\{\,\}$) ocorrerá em seguida.

Usando prosseguimentos, o significado de uma espeficicação é dado associando-a ao conjunto dos possíveis estados ao final da execução do programa correspondente. Além disso, este domínio semântico permite que seja representada interferência entre processos de uma forma natural. Isto é necessário, por exemplo, no caso em que a ocorrência de eventos de saída associados a uma partição de um estado AND influencia (e interfere) no comportamento dos outros componentes ortogonais pertencentes ao mesmo estado composto.

Equações Semânticas

Os componentes de um statechart são associados a seu significado através de equações semânticas. Cada uma destas possui uma assinatura onde o domínio é algum objeto do mundo sintático, definido em termos de meta-variáveis, e onde a imagem é algum objeto do mundo semântico, composto por domínios semânticos. A definição da equação é feita por casos, associando cada elemento da sintaxe abstrata ao respectivo significado, usando o princípio da funcionalidade.

No modelo semântico apresentado a seguir, $\mathcal{M}$ é a equação que associa uma especificação inteira a seu significado. Aí aparece o nome $(*)$, que representa a transição para o estado default.

$$\mathcal{M} : \mathbf{ESP} \rightarrow \mathbf{STA} \rightarrow \mathcal{P}(\mathbf{STA}_\perp)$$

$$\begin{aligned} \mathcal{M}[\![[I, O, \Xi_{sta}]]\!]\sigma = \ & \mathsf{let}\ \rho = \Pi_{2,1}(\mathcal{D}[\![[I, O, \Xi_{sta}]]\!] < \lambda\rho \cdot \perp, \sigma >)\ \mathsf{in} \\ & \mathsf{let}\ \rho' = \rho[* \leftarrow \rho(\Xi_{sta})]\ \mathsf{in} \\ & \quad \mathit{flatten}(\rho'(*)_{\mathbf{TRA}} < 0, \sigma >) \end{aligned} \tag{2.1}$$

A equação $\mathcal{D}$ associa estados ou eventos aos respectivos significados. Para cada estado complexo é criado um ambiente (ρ) para a execução dos processos associados, onde são criadas transições (π) a nível semântico, representando as alterações de estado ocorridas devido à computação destes processos.

$$\mathcal{D} : (\mathbf{ESP} + \mathbf{EVT}) \rightarrow (\ (\ \mathbf{AMB} \times \mathbf{STA}\ ) \rightarrow (\ \mathbf{AMB} \times \mathbf{STA}\ )\ )$$

$$\begin{aligned} & \mathcal{D}[\![[\Xi_{tsn_{i_1}} \ldots \Xi_{tsn_{i_m}}, \Delta_1 \ldots \Delta_n, \Xi_{sta}]]\!]\rho\sigma = \\ & \quad \mathsf{let}\ \pi(\upsilon)\tau = \\ & \qquad \mathbf{cond}(\ n \geq 1 \rightarrow (\mathsf{let}\ \rho''' = \ \Pi_{2,2}(\mathcal{D}[\![\Theta_1]\!] \circ \mathcal{D}[\![\Pi_{out_{1_1}}]\!] \circ \ldots \circ \mathcal{D}[\![\Pi_{out_{1_y}}]\!] \circ \\ & \qquad\qquad\qquad\qquad \vdots \\ & \qquad\qquad\qquad\qquad \circ \mathcal{D}[\![\Theta_n]\!] \circ \mathcal{D}[\![\Pi_{out_{n_1}}]\!] \circ \ldots \circ \mathcal{D}[\![\Pi_{out_{n_z}}]\!]\rho\sigma)\mathsf{in} \\ & \qquad\qquad\qquad \mathsf{let}\ \rho' = \mathbf{cond}(\ n \geq 1 \rightarrow \rho''', \\ & \qquad\qquad\qquad\qquad \mathsf{true} \rightarrow \rho)\ \mathsf{in} \\ & \qquad\qquad\qquad\quad \mathsf{let}\ \delta_k = \mathcal{C}[\![(\Xi_{tsn_{o_k}}, \Theta_k / \Pi_{out_{k_1}} \ldots \Pi_{out_{k_x}})]\!]\tau\rho'\sigma' \\ & \qquad\qquad\qquad\qquad /*\ \Delta_k = (\Xi_{tsn_{o_k}}, \Theta_k / \Pi_{out_{k_1}} \ldots \Pi_{out_{k_x}}) * /\ \mathsf{in} \\ & \qquad\qquad\qquad\qquad \mathsf{let}\ \pi_k(\upsilon)\tau\sigma' = \delta_k \circ (\rho(\Xi_{tsn_{o_k}})(\upsilon)\sigma'),\ k = 1, \ldots, n\ \mathsf{in} \\ & \qquad\qquad\qquad\qquad\quad \pi_1(\upsilon)\tau \ || \ldots || \ \pi_n(\upsilon)\tau), \\ & \qquad\quad \mathsf{true} \rightarrow \mathit{stop})\ \mathsf{in} \\ & \quad \mathsf{let}\ \rho'' = \rho[\Xi_{tsn_{i_j}} \leftarrow \pi],\ j = 1, \ldots, m\ \mathsf{in} \\ & \qquad < \rho''[\Xi_{sta} \leftarrow \pi], \sigma > \end{aligned} \tag{2.2}$$

$$\mathcal{D}[\![\mathit{Connect}([I, O, \Xi_{sta}], \Xi_{tsn_{in}}, \Xi_{tsn_{out}})]\!]\rho\sigma = < \rho[\Xi_{tsn_{out}} \leftarrow \rho(\Xi_{tsn_{in}})], \sigma > \tag{2.3}$$

$$\begin{aligned} \mathcal{D}[\![\mathit{And}([I_1, O_1, \Xi_{sta_1}], [I_2, O_2, \Xi_{sta_2}])]\!]\rho\sigma = \ & \mathsf{let}\ \pi(\upsilon)\tau = \rho(\Xi_{sta_1})(\upsilon)\tau \ || \ \rho(\Xi_{sta_2})(\upsilon)\tau\ \mathsf{in} \\ & < \rho[\Xi_{sta_1} \leftarrow \pi, \Xi_{sta_2} \leftarrow \pi], \sigma > \end{aligned} \tag{2.4}$$

$$\begin{aligned} \mathcal{D}[\![\mathit{Or}([I_1, O_1, \Xi_{sta_1}], [I_2, O_2, \Xi_{sta_2}])]\!]\rho\sigma = \ & \mathsf{let}\ \pi(\upsilon)\tau = \rho(\Xi_{sta_1})(\upsilon)\tau \cup \rho(\Xi_{sta_2})(\upsilon)\tau\ \mathsf{in} \\ & < \rho[\Xi_{sta_1} \leftarrow \pi, \Xi_{sta_2} \leftarrow \pi], \sigma > \end{aligned} \tag{2.5}$$

$$\begin{aligned} \mathcal{D}[\![\mathit{Stat}([I_1, O_1, \Xi_{sta_1}], [I_2, O_2, \Xi_{sta_2}], \Xi_{tsn})]\!]\rho\sigma = \ & \mathsf{let}\ \rho' = \rho[* \leftarrow \pi]\ \mathsf{in} \\ & \mathsf{let}\ \pi(\upsilon)\tau = \ \rho'(\Xi_{sta_1})(\upsilon)\tau \ || \\ & \qquad\qquad \rho(\Xi_{sta_2})(\upsilon)\tau\ \mathsf{in} \\ & \quad < \rho[\Xi_{sta_1} \leftarrow \pi], \sigma > \end{aligned} \tag{2.6}$$

$$\mathcal{D}[\![HiCl([I,O,\Xi_{sta}],\Xi_{evt})]\!]\rho\sigma = \text{let } \rho' = \rho[\Xi_{evt} \leftarrow \upsilon'] \text{ in} \\ \text{let } \upsilon' = \mathbf{cond}(\ \rho' \rightarrow \lambda\sigma' \cdot \perp \ \text{ in} \\ \text{true} \rightarrow \perp) \\ \text{let } \pi(\upsilon)\tau = \rho'(\Xi_{sta})(\upsilon)\tau \text{ in} \\ < \rho[\Xi_{sta} \leftarrow \pi], \sigma > \tag{2.7}$$

$$\mathcal{D}[\![in(\Xi_{evt}) \mid tm(\Xi_{evt},\mathbf{N}) \mid out(\Xi_{evt})]\!]\rho\sigma = \text{let } \upsilon = \lambda\rho' \cdot \lambda\sigma' \cdot \perp \text{ in} \\ < \rho[\Xi_{evt} \leftarrow \upsilon], \sigma > \tag{2.8}$$

$$\mathcal{D}[\![\lambda \mid \Xi_{var} \leftarrow \mathbf{N} \mid \Xi_{var} = \mathbf{N} \mid exec(\Xi_{proc})]\!]\rho\sigma = < \rho, \sigma > \tag{2.9}$$

O significado das transições em uma especificação é dado pela equação $\mathcal{C}$. Nesta, $\tau+1$ representa a contabilização de um passo de execução causado pela ocorrência da transição. Para que cada passo ocorra, a condição associada à transição deve ser satisfeita, o que é verificado na equação $\mathcal{E}$.

$$\mathcal{C} : \mathbf{DEC} \rightarrow (\ \mathbf{TIE} \rightarrow (\ \mathbf{AMB} \rightarrow \mathbf{RES}\ )\ )$$

$$\mathcal{C}[\![(\Xi_{tsn},\Theta/)]\!]\tau\rho = \lambda\sigma \cdot \mathbf{cond}(\ \mathcal{E}[\![\Theta]\!]\tau\rho\sigma \rightarrow \{(\sigma, \rho(\Xi_{tsn})_{\mathbf{TRA}} < \tau+1, \sigma >)\} \\ \text{true} \rightarrow \{(\sigma, stop)\}) \text{ in} \tag{2.10}$$

$$\mathcal{C}[\![(\Xi_{tsn},\Theta/\Xi_{var} \leftarrow \mathbf{N})]\!]\tau\rho = \lambda\sigma \cdot \mathbf{cond}(\ \mathcal{E}[\![\Theta]\!]\tau\rho\sigma \rightarrow \{(\ \sigma[\rho(\Xi_{var}) \leftarrow \mathbf{N}], \\ \rho(\Xi_{tsn})_{\mathbf{TRA}} < \tau+1, \sigma >)\ \} \\ \text{true} \rightarrow \{(\sigma, stop)\}) \tag{2.11}$$

$$\mathcal{C}[\![(\Xi_{tsn},\Theta/out(\Xi_{evt}))]\!]\tau\rho = \text{let } \upsilon' = \rho(\Xi_{evt})[\rho \leftarrow \lambda\sigma' \cdot \tau] \text{ in} \\ \text{let} \rho' = \rho[\Xi_{evt} \leftarrow \upsilon'] \text{ in} \\ \lambda\sigma \cdot \mathbf{cond}(\ \mathcal{E}[\![\Theta]\!]\tau\rho\sigma \rightarrow \{(\sigma, \rho'(\Xi_{tsn})_{\mathbf{TRA}} < \tau+1, \sigma >)\}, \\ \text{true} \rightarrow \{(\sigma, stop)\}) \tag{2.12}$$

$$\mathcal{C}[\![(\Xi_{tsn},\Theta/exec(\Xi_{atv}))]\!]\tau\rho = \lambda\sigma \cdot \text{let } \delta = \mathbf{cond}(\ \lambda\Xi'_{sta} \cdot (\rho(\Xi'_{sta}) = \rho(\Xi_{atv})) \equiv \text{false} \rightarrow \\ \rho(\Xi_{atv})_{\mathbf{TRA}} \circ \rho(\Xi_{sta})_{\mathbf{TRA}} < \tau+1, \sigma > \\ \text{true} \rightarrow \rho(\Xi_{atv})_{\mathbf{TRA}} < \tau+1, \sigma >) \text{ in} \\ \mathbf{cond}(\ \mathcal{E}[\![\Theta]\!]\tau\rho\sigma \rightarrow \{(\sigma,\delta)\}, \\ \text{true} \rightarrow \{(\sigma, stop)\}) \tag{2.13}$$

$$\mathcal{C}[\![(\Xi_{tsn},\Theta/\Pi_1 \ldots \Pi_m)]\!]\tau\rho = \mathcal{C}[\![(\Xi_{tsn},\Theta/\Pi_1)]\!]\tau\rho \mid\mid \ldots \mid\mid \mathcal{C}[\![(\Xi_{tsn},\Theta/\Pi_m)]\!]\tau\rho \tag{2.14}$$

Na semântica formal de Statecharts podem ser observadas três formas de paralelismo: um entre estados (explícito), que representa a ocorrência simultânea destes; outro entre transições (implícito), que representa não-determinismos; e ainda outro entre eventos (indireto), que representa a possibilidade destes ocorrerem simultaneamente. Isto pode ser observado nas equações (2.1), (2.3) e (2.14), respectivamente.

$exec(\Xi_{atv})$ significa que a atividade Ξ_{atv}, especificada no D.F.D. associado, deve ocorrer. Na equação (2.13) é verificado se esta atividade tem comportamento detalhado pelo estado de destino da transição onde *exec* aparece. Se este for o caso, o termo é associado ao significado dado a este estado (representado por $\rho(\Xi_{sta})$; caso contrário (Ξ_{atv}) é uma atividade sem diálogo). Senão *exec* é associado à computação de Ξ_{atv} seguida da computação de Ξ_{sta}. Este último caso é representado pela composição de funções ($\circ$) mostrada na equação.

$$\mathcal{E} : \mathbf{EXP} \rightarrow (\ \mathbf{TIE} \rightarrow (\ \mathbf{AMB} \rightarrow (\ \mathbf{STA} \rightarrow \mathbf{TIE}\ )\ )\ )$$

$$\mathcal{E}[\![\lambda]\!]\tau\rho\sigma = \text{true} \tag{2.15}$$

$$\mathcal{E}[\![in(\Xi_{evt})]\!]\tau\rho\sigma = ((\lambda\rho' \cdot (\rho(\Xi_{evt})(\rho') \neq \lambda\sigma' \cdot \tau)) \equiv \text{true}) \tag{2.16}$$

$$\mathcal{E}[\![tm(\Xi_{evt},\mathbf{N})]\!]\tau\rho\sigma = ((\lambda\rho' \cdot (\lambda\rho'' \cdot \lambda\sigma' \cdot \tau\text{-}\mathbf{N}) > \rho(\Xi_{evt})(\rho')) \equiv \text{true}) \tag{2.17}$$

$$\mathcal{E}[\![\Xi_{var} = \mathbf{N}]\!]\tau\rho\sigma = (\sigma(\rho(\Xi_{var})) \equiv \mathbf{N}) \tag{2.18}$$

$$\mathcal{E}[\![\neg\ \Theta]\!]\tau\rho\sigma = \neg\ \mathcal{E}[\![\Theta]\!]\tau\rho\sigma \tag{2.19}$$

$$\mathcal{E}[\![\Theta_1 \vee \Theta_2]\!]\tau\rho\sigma = \mathcal{E}[\![\Theta_1]\!]\tau\rho\sigma \vee \mathcal{E}[\![\Theta_2]\!]\tau\rho\sigma \tag{2.20}$$
$$\mathcal{E}[\![\Theta_1 \wedge \Theta_2]\!]\tau\rho\sigma = \mathcal{E}[\![\Theta_1]\!]\tau\rho\sigma \wedge \mathcal{E}[\![\Theta_2]\!]\tau\rho\sigma \tag{2.21}$$

Nestas definições utilizamos as seguintes funções [Jon91]:

$$\mathit{flatten}\colon \mathbf{RES} \rightarrow \mathbf{STA} \rightarrow \mathcal{P}(\mathbf{STA}_\perp)$$
$$\begin{aligned}\mathit{flatten}(\delta) \triangleq \lambda\sigma\cdot\ & (\{\sigma' \mid (\sigma', \mathit{stop}) \in \delta(\sigma)\} \cup \\ & \{\mathit{flatten}(\delta')(\sigma') \mid (\sigma', \delta') \in \delta(\sigma) \wedge \delta' \neq \mathit{stop}\})\end{aligned}$$

$$\_ \mid\mid \_\colon \mathbf{RES} \times \mathbf{RES} \rightarrow \mathbf{RES}$$
$$\delta_1 \mid\mid \delta_2 \triangleq \lambda\sigma\cdot(\delta_1 \ \mathsf{before}\ \delta_2)\sigma \ \cup (\delta_2 \ \mathsf{before}\ \delta_1)\sigma$$

$$\_ \ \mathsf{before}\ \_\colon \mathbf{RES} \times \mathbf{RES} \rightarrow \mathbf{RES}$$
$$\begin{aligned}\delta_1 \ \mathsf{before}\ \delta_2 \triangleq\ & \lambda\delta\cdot(\{(\sigma, \delta_2) \mid (\sigma, \mathit{stop}) \in \delta\} \cup \\ & \{(\sigma, \delta_r \ \mathsf{before}\ \delta_2), (\sigma, \delta_2 \ \mathsf{before}\ \delta_r) \mid (\sigma, \delta_r) \in \delta \wedge \delta_r \neq \mathit{stop}\}) \circ \delta_1\end{aligned}$$

A semântica definida aqui é equivalente à apresentada em [HRdR92], que utiliza como domínio semântico histórias da computação de um processo. Entretanto, o modelo apresentado aqui contempla também as construções que se referem à integração com D.F.D.s. Além disso, utiliza-se de um estilo mais difundido, facilitando traduções para código executável e permitindo que sejam feitas comparações formais com outros modelos [All86].

3 Integração com outra técnica

Conforme discutido no início do trabalho, existem várias motivações para que uma integração entre técnicas formais e informais seja relizada. O método sistemático que conduz esta integração é apresentado a seguir.

Neste método, a integração deve seguir as seguintes etapas:

1. Estudo e estabelecimento do significado das linguagens envoldidas;
2. Introdução de redundâncias de informação e restrições de integridade a nível de especificação;
3. Definição do modelo semântico para as linguagens das técnicas envolvidas.

Até agora, todas etapas acima foram apresentadas na seção 2, considerando que Statecharts seriam integrados com outra técnica que permitisse a especificação das atividades em um sistema. A forma de redundância de informação incluida neste formalismo foi a representação do nome da atividade que seria ativada no caso de uma transição ocorrer (*exec*), e a restrição imposta foi que esta atividade não poderia possuir interação com o ambiente. Um modelo semântico baseado em prosseguimentos foi então apresentado.

Entre as técnicas informais que permitem a especificação das atividades de um sistema, D.F.D.s foram escolhidos neste exercício de integração, por permitirem a especificação de aspctos do sistema que não podem ser representados usando apenas Statecharts. Uma descrição informal da semântica desta técnica não é apresentada aqui, já que esta possui uma utilização bastante difundida.

Como ténica informal, D.F.D.s ocupam uma papel secundário na integração, já que não possuem uma linguagem de especificação com tantos elementos formalizáveis quanto Statecharts. Dessa forma, as etapas 1, 2 e 3 do método em questão não são tão complexas quanto as da técnica formal. Esta etapas são apresentadas ao longo desta seção.

3.1 Semântica Formal

Serão usadas as mesmas equações e domínios semânticos, para que as linguagens das duas técnicas possam ser entendidas e integradas de maneira uniforme. As meta-veriáveis tem um significado particular a estas técnica e a sintaxe abstrata devem ser outra, já que representa outro tipo de especificação.

Meta-variáveis

$$
\begin{array}{lcccl}
\Psi & \in & \mathbf{ESP} & = & \text{especificações (D.F.D.s)} \\
\Delta & \in & \mathbf{DEC} & = & \text{declarações (variáveis)} \\
\Xi & \in & \mathbf{VAR} & = & \text{variáveis}
\end{array}
$$

Sintaxe Abstrata

Da mesma forma que em Statecharts, D.F.D.s possuem uma sintaxe gráfica, que está associada á sintaxe abstrata apresentada abaixo:

$$\Delta ::= \Xi_1 \ldots \Xi_m$$

$$
\begin{array}{ll}
\Psi ::= & Actv(\Delta_i, \Delta_o, \Xi_{atv}) \mid Ent(\Delta_i, \Delta_o, \Xi_{ent}) \mid Strg(\Delta_i, \Delta_o, \Xi_{stg}) \mid \\
& \Psi_{atv_1} \ldots \Psi_{atv_k} \;\; \Psi_{stg_1} \ldots \Psi_{stg_m} \;\; \Psi_{ent_1} \ldots \Psi_{ent_n} \mid \\
& Level(\Xi_{atv}, \Psi)
\end{array}
$$

Nesta sintaxe, *Actv* representa as especificação de atividades, *Ent* a especificação de entidades externas e *Strg* a especificação de depósitos de dados do sistema. $Level(\Xi_{atv}, \Psi)$ indica que a especificação Ψ descreve o detalhamento da atividade Ξ_{atv} (explosão).

Equações Semânticas

$$\mathcal{D}[\![\Xi_1 \ldots \Xi_m]\!]\rho\sigma = \mathsf{let}\;\; \sigma(\alpha_i) = \mathsf{livre}\;\mathsf{in} \quad < \rho[\Xi_i \leftarrow \alpha_i], \sigma[\alpha_i \leftarrow \perp] >, i = 1, \ldots, m \tag{3.1}$$

$$\mathcal{D}[\![Actv(\Delta_i, \Delta_o, \Xi_{atv})]\!]\rho\sigma = \mathsf{let}\;\; \sigma' = \Pi_{2,2}(\mathcal{D}[\![\Delta_i]\!] \circ \mathcal{D}[\![\Delta_o]\!]\rho\sigma)\;\mathsf{in}\;\; \mathsf{let}\;\; \pi(\upsilon)\tau\sigma = \rho(\Xi_{atv})(\upsilon)\tau\sigma'\;\mathsf{in}\;\; \mathsf{let}\;\; \rho' = \rho[\Xi_{atv} \leftarrow \pi]\;\mathsf{in}\;\; < \rho', \sigma' > \tag{3.2}$$

$$\mathcal{D}[\![Ent(\Delta_i, \Delta_o, \Xi_{ent})]\!]\rho\sigma = \mathcal{D}[\![\Delta_i]\!] \circ \mathcal{D}[\![\Delta_o]\!]\rho\sigma \tag{3.3}$$

$$\mathcal{D}[\![Strg(\Delta_i, \Delta_o, \Xi_{stg})]\!]\rho\sigma = \mathsf{let}\;\; \sigma(\alpha) = \mathsf{livre}\;\mathsf{in}\;\; \mathsf{let}\;\; \rho' = \rho[\Xi_{stg} \leftarrow \alpha]\;\mathsf{in}\;\; \mathsf{let}\;\; \sigma' = \Pi_{2,2}(\mathcal{D}[\![\Delta_i]\!] \circ \mathcal{D}[\![\Delta_o]\!]\rho'\sigma)\;\mathsf{in}\;\; < \rho', \sigma'[\alpha \leftarrow \lambda\alpha' \cdot \perp] > \tag{3.4}$$

$$\mathcal{D}[\![Level(\Xi_{atv}, \Psi)]\!]\rho\sigma = \mathsf{let}\;\; < \rho', \sigma' > = \mathcal{D}[\![\Psi]\!]\rho\sigma\;\mathsf{in}\;\; \mathsf{let}\;\; \pi(\upsilon)\tau\sigma = \rho'(\Xi_{atv})(\upsilon)\tau\sigma\;\mathsf{in}\;\; < \rho'[\Xi_{atv} \leftarrow \pi], \sigma' > \tag{3.5}$$

$$
\begin{aligned}
&\mathcal{D}[\![\Psi_{atv_1} \ldots \Psi_{atv_k}\;\; \Psi_{stg_1} \ldots \Psi_{stg_m}\;\; \Psi_{ent_1} \ldots \Psi_{ent_n}]\!]\rho\sigma = \\
&\quad \mathsf{let}\ \rho' = \Pi_{2,1}(\ \mathcal{D}[\![\Psi_{atv_1}]\!] \circ \ldots \circ \mathcal{D}[\![\Psi_{atv_k}]\!] \circ \quad \mathsf{in} \\
&\qquad\qquad \mathcal{D}[\![\Psi_{stg_1}]\!] \circ \ldots \circ \mathcal{D}[\![\Psi_{stg_m}]\!] \circ \\
&\qquad\qquad \mathcal{D}[\![\Psi_{ent_1}]\!] \circ \ldots \circ \mathcal{D}[\![\Psi_{ent_n}]\!]) \\
&\quad \mathsf{let}\ \sigma_i' = \ \sigma[\rho'(\Xi_{atv_{i_{I_z}}}) \leftarrow \mathbf{cond}(\ \sigma(\rho'(\Xi_{stg_x}))(\Xi_{atv_{i_{I_z}}}) = \perp \rightarrow \sigma(\rho'(\Xi_{atv_{i_{I_z}}})), \quad \mathsf{in} \\
&\qquad\qquad\qquad \mathsf{true} \rightarrow \sigma(\rho'(\Xi_{stg_x}))(\Xi_{atv_{i_{I_z}}}))], \\
&\qquad x = 1, \ldots, m, z = 1, \ldots, e \\
&\quad \mathsf{let}\ \pi_i(v)\tau\sigma = subst(\rho'(\Xi_{atv_i})(v)\tau\sigma_i')\ \mathsf{in} \\
&\qquad < \rho'[\Xi_{atv_i} \leftarrow \pi_i], \sigma >, i = 1, \ldots, e \\
&\quad /*\ \Psi_{atv_i} = Actv(\Xi_{atv_{i_{I_1}}} \ldots \Xi_{atv_{i_{I_e}}}, \Xi_{atv_{i_{O_1}}} \ldots \Xi_{atv_{i_{O_f}}}, \Xi_{atv_i})\ */
\end{aligned}
\tag{3.6}
$$

Na equação acima usamos:

$$
\begin{aligned}
&subst\text{: } \mathbf{RES} \rightarrow \mathbf{RES} \\
&subst \triangleq \lambda\delta \cdot (\ \{(\sigma'[\rho'(\Xi_{stg_x}) \leftarrow \zeta_y], stop) \mid (\sigma', stop) \in \delta \wedge \\
&\qquad \zeta_y = \rho'(\Xi_{stg_x})[\Xi_{atv_{i_{O_w}}} \rightarrow \mathbf{cond}(\ \sigma'(\rho'(\Xi_{stg_x}))(\Xi_{atv_{i_{O_w}}}) = \perp \leftarrow \Xi_{atv_{i_{O_w}}}, \\
&\qquad\qquad\qquad \mathsf{true} \leftarrow \sigma'(\rho'(\Xi_{stg_x}))(\Xi_{atv_{i_{O_w}}})]\} \\
&\qquad \cup \{subst(\delta') \mid (\sigma', \delta') \in \delta\}), w = 1, \ldots, f
\end{aligned}
$$

Nas definições acima, $\mathcal{D}$ tem o mesmo papel que na seção anterior, que é associar cada especificação à seu significado, dado em função de prosseguimentos.

Pode-se observar nestas equações que variáveis Ξ estão sempre associadas ao ambiente onde o respectivo objeto se encontra, representando a definição e a associação destas aos pontos do programa onde são usadas. A equação (3.6) diz, além disso, quais variáveis serão usadas na computação do processo associado à atividade Ξ_{atv} e como é feita a atualização dos arquivos utilizados após este computo.

Existe ainda uma restrição de consistência entre as duas semânticas, para que a integração seja correta:

Restrição 2:
$\forall \Xi_{atv} \cdot \exists!\, \Xi_{sta} \cdot \rho(\Xi_{atv}) \equiv \rho(\Xi_{sta})$

3.2 Discussão

Observando a literatura sobre integração entre técnicas de desenvolvimento de programas pode-se constatar uma grande ausência de métodos para o tratamento deste problema. Integrações particulares seguem geralmente duas tendências: transpor os conceitos definidos na especificação informal para outra formal através de alguma forma de mapeamento, ou dar semântica formal a todas as construções que tenham significado fixo na linguagem usada pela técnica informal.

A primeira tendência acarreta em integrações onde o processo de tradução pode levar a inconsistências ou mesmo erros, pois geralmente estes mapeamentos são informais. Em alguns casos a integração não é nem clara o suficiente para garantir se uma das técnicas complementou a outra. Isto é o que ocorre em [TvKP90].

Por outro lado, a tendência oposta tem como fundamentação o modelo formal, de onde pode-se constatar, por exemplo, a complementariedade das informações especificadas. Este é um objetivo em alguns trabalhos de pesquisa, como o de Colleman et alli: "Trabalho adicional é requerido para colocar Objectcharts sobre uma fundamentação semântica sólida" [CHB92].

O método apresentado aqui procura seguir esta última tendência. Pode-se constatar formalmente nos exemplos que uma técnica complementa a outra. Buscando constatar diferenças entre os modelos semânticos definidos anteriormente, observa-se que:

- **ARQ** é usado apenas no modelo de D.F.D.s;
- variáveis em D.F.D.s são usadas para indicar de onde procedem e qual o destino dos dados manipulados pelas atividades do sistema;
- variáveis em Statecharts são usadas apenas com finalidade de controle, como eventos associados às transições;
- A ordem com que as atividades são processadas pode ser derivada do encadeamento de transições nos Statecharts.

Todas estas constatações (formais, pois foram derivadas dos modelos semânticos) indicam que Statecharts são mais indicados para representar o aspecto comportamental e que D.F.D.s mais indicados para representar o aspecto estrutural do sistema. Isto mostra que estas técnicas são realmente complementares e merem ser usadas de forma integrada.

Outra característica interessante alcançada pela integração é a minimalidade de redundância de informação, já que todas as construções são definidas formalmente, e as interseções desnecessárias podem ser eliminadas.

Entretanto, pode-se argumentar que a integração neste trabalho se dá apenas a nível teórico, o que não é verdade, já que existem meta-compiladores que tomam como entrada definições em semântica denotacional e produzem um interpretador e/ou compilador para a linguagem de especificação desejada [Hae89]. Tomando o produto obtido neste passo de compilação, ao serem fornecidas todas as definições apresentadas neste trabalho como entrada, e usando-o sobre um Statechart e um D.F.D., integrados de acordo com o método apresentado aqui, seria obtido um programa que poderia ser executado.

4 Conclusões e Trabalhos Adicionais

Neste trabalho Statecharts foram escolhidos para realizar um estudo sobre integração entre técnicas de desenvolvimento de programas. Esta escolha se baseou no fato desta técnica propiciar grande facilidade na criação e manutenção de especificações, ser formal, visual e possuir semântica amplamente discutida e formalizada, entre outras características.

Foi apresentada uma semântica formal para Statecharts. Esta semântica foi dada usando prosseguimentos. A preocupação aqui não foi discutir uma definição mais adequada para esta semântica, como em [HdR91], mas sim apresentá-la de forma a permitir uma fácil compreensão e integração com a semântica de outra técnica, construída da mesma maneira.

A integração desta técnica formal com D.F.D.s foi feita usando um método baseado no estabelecimento de redundância de informação e de restrições de integridade. Posteriormente, tal método prescreve que deve ser criado um modelo semântico único, para dar significado a todas as especificações criadas usando tais técnicas.

A contribuição maior deste trabalho foi definir tal método, com base na criação de um modelo semântico único para as linguagens envolvidas. A escolha por esta forma de integração permite que sejam realizados raciocínios, com uma base formal, sobre as características da integração, como complementariedade, redundância de informação e computabilidade efetiva das técnicas envolvidas.

Futuros trabalhos poderiam ser desenvolvidos da mesma forma, considerando outras técnicas de especificação. Poderiam ser utilizados também outros modelos semânticos, indicados para dar semântica tanto à Statecharts quanto a D.F.D.s (Estruturas de Eventos e Redes de Petri [NPW81] associados à formalizações no estilo de [Tse89], por exemplo), e verificar se o raciocício sobre a técnica integrada é tão natural quanto o realizado ao ser utilizada semântica denotacional. Nada impede também que a integração seja realizada entre duas ou mais técnicas quaisquer (formais ou informais). Entretanto, em todos estes exercícios deve se ter sempre em mente que o objetivo principal é especificar o sistema de forma mais completa, em comparação com a especificação obtida através da aplicação de técnicas isoladas.

Agradecimentos

A Michel Heluey Fortuna, que lançou idéias preciosas em discussões anteriores e tornou possível a elaboração de trabalhos como este. Agradecemos também a Luiz Paulo Alves Franca pela ajuda com a bibliografia.

Referências

[All86] Loyd Allison. *A practical introduction to denotational semantics.* Cambridge University Press, 1986.

[CHB92] D. Colleman, F. Hayes, and S. Bear. Introducting objectcharts or how to use statecharts in object oriented design. *IEEE Transactions on Software Engineering*, 18(1):9–18, 1992.

[CHCDL92] Roberto Ierusalimsky Carlos H. C. Duarte and Carlos J. P. Lucena. On the modularization of formal specifications: The NDB example revisited. Technical Report 33/92, Pontifícia Universidade Católica, Rio de Janeiro, Brazil, 1992. Series Monografias em Ciência da Computação.

[CMO92] Bill Curtis, I. Marc, and Jim Over. Process modeling. *Communications of the ACM*, 35(9), September 1992.

[DeM89] Tom DeMarco. *Structured Analysis and System Specification.* 1989.

[Hae89] Edward Hermann Haeusler. Environments for prototyping programming languages: a proposal based on semantical specification. Relatório científico, Departamento de Computação, UFF, 1989. 34 págs.

[Har80] D. Harel. On visual formalisms. *Communications of the ACM*, 31(5):512–530, 1980.

[HdR91] C. Huizing and W. P. de Roever. Introduction to design choices in the semantics of statecharts. *Information Processing Letters*, 37:205–213, 1991.

[HRdR92] J. J. M. Hooman, S. Ramesh, and W. P. de Roever. A compositional axiomatization of statecharts. *Theoretical Computer Science*, (10):289–335, 1992.

[Jon91] C. B. Jones. Interference resumed. Technical Report UCMS-91-5-1, Department of Computer Science, University of Manchester, England, 1991.

[MM91] C. A. A. Meira and P. C. Masiero. Um gerador de aplicações para sistemas reativos. In *V Simpósio Brasileiro de Engenharia de Software*, pages 45–59, October 1991.

[NPW81] M. Nielsen, G. Plotkin, and G. Winskel. Petri-nets, event-structures and domains - part i. *Theoretical Computer Science*, 13:85–108, 1981.

[Tse89] L. Tse, T. H. e Pong. Towards a formal foundation for Demarco data flow diagrams. *The Computer Journal*, 32(1):1–12, 1989.

[TvKP90] H. Toetenel, J. van Katwikj, and Nico Plat. Structured analysis - formal design, using stream & object-oriented formal specification. In *ACM SIGSOFT International Workshop on Formal Methods in Software Development*, pages 118–127, May 1990.