Composicionalidade de Primitivas - Uma Abordagem Categorial

Simone André da Costa[1]
Júlio Henrique Araújo Pereira Machado[2]
Paulo Fernando Blauth Menezes[2]

Resumo: Autômatos não-seqüenciais são apresentados como modelo semântico para a interpretação formal do conceito de primitiva em sistemas operacionais sob a ótica da Teoria das Categorias. Este modelo constitui originalmente um domínio semântico para sistemas reativos distribuídos, concorrentes e comunicantes, onde os estados e transições possuem estrutura monoidal comutativa, que satisfaz a composicionalidade diagonal. Utilizando este domínio, estende-se o conceito de transação típico de sistemas distribuídos, o qual permite agrupar operações de forma que o todo seja uma estrutura atômica, para a composição atômica de primitivas de um sistema operacional. Transações são formalmente definidas através de uma operação de reificação entre autômatos que mapeia transições do autômato origem em transações do autômato destino (mais concreto). Adicionalmente, são introduzidas as construções necessárias para a definição da semântica de uma reificação, especificada como um mapeamento para uma computação.

Palavras-chave: autômatos não-seqüenciais, primitivas, reificação, semântica, teoria das categorias.

Abstract: Nonsequential automata are presented as a semantic model for the formal interpretation of the concept of operating systems primitive under the view of Category Theory. This model originally constitutes a semantic domain for reactive, distributed, concurrent and communicating systems, with a monoidal structure on states and transitions, which satisfies the diagonal compositionality requirement. Based on this domain, the concept of transaction used in distributed system which allows grouping operations into a single atomic structure, is extended to includes the atomic composition of operating systems primitives. Transactions are formally defined through a reification between automata that maps transitions from the source automaton to transactions of the target (more concrete) automata. Additionally, we introduce the necessary constructions for defining the semantics of reification, specified as a mapping to a computation.

Keywords: nonsequential automata, primitives, reification, semantics, category theory.

---

1 Centro de Ciências Exatas e Tecnológicas, UNISINOS {sac@exatas.unisinos.br}
2 Instituto de Informática, UFRGS, Caixa Postal 9999 {jhapm, blauth @inf.ufrgs.br}

1 Introdução

Através dos anos, a estrutura e capacidades disponíveis em sistemas operacionais têm sofrido uma gradual evolução. Em anos recentes, um grande número de novos elementos, tanto de hardware quanto software, foram adicionados aos sistemas operacionais. No campo de hardware surgiram máquinas multiprocessadas, maior rapidez de processamento, conexões a redes de computadores de alta velocidade, entre outros. Na área de software, vemos um aumento no número de sistemas multimídia, Internet e aplicações cliente/servidor. Estas mudanças demandam não só modificações e aperfeiçoamentos às arquiteturas existentes, mas também novos meios de organizar os sistemas operacionais.

Em sistemas distribuídos, por exemplo, é muito usual o conceito de transações atômicas [1], o qual pode ser entendido como a capacidade de agrupar diversas operações de forma que o todo seja completamente realizado ou nenhuma das operações venha a se efetivar.

É possível estender o conceito de transação não somente para operações em uma linguagem de programação ou acesso a banco de dados, por exemplo, mas também para o conceito de primitiva do sistema operacional em si.

Algumas primitivas usuais de sistemas operacionais incluem operações para a troca de mensagens (como "send" e "receive") em sistemas distribuídos, e operações utilizadas em mecanismos de exclusão mútua e sincronização. Exemplos de primitivas implementadas como instruções de máquina não sujeitas à interrupção para a construção de algoritmos de exclusão mútua incluem "test and set" e "exchange", que permitem testar e modificar o conteúdo de uma palavra de memória ou trocar o conteúdo entre duas palavras de forma atômica [2]. Uma possível definição para a primitiva "exchange" seria:

```
void exchange (int register, int memory) {
  int temp;
  temp = memory;
  memory = register;
  register = temp;
}
```

Outro exemplo de seqüência atômica de operações inclui as primitivas para semáforos "V(s)" e "P(s)" [3], onde "P(s)" decrementa o valor do semáforo e testa por valor negativo, bloqueando então o processo, e "V(s)" que incrementa o valor do semáforo e testa se o valor não é positivo para desbloquear o processo. Tais operações são composições seqüenciais de operações de incremento, decremento e teste.

Note que nos exemplos acima, um conjunto de operações básicas são compostas como uma estrutura única atômica. Na visão apresentada neste trabalho, uma primitiva do

kernel de um sistema operacional é apresentada sob uma ótica categorial como uma seqüência de operações obtida por um processo de reificação entre autômatos, a qual a torna, por definição, um único passo indivisível. Além disso, o modelo utilizado permite ainda a especificação de novas primitivas (possivelmente complexas) sobre outras primitivas já existentes.

É importante salientar ainda que este trabalho não trata de aspectos de implementação do conceito de composição de primitivas como uma primitiva, mas apresenta um modelo semântico para sua interpretação baseado em Teoria das Categorias [4,5,6,7].

2 Autômatos Não-Seqüenciais

Autômatos não-seqüenciais [8,9,10] constituem um domínio semântico para sistemas reativos distribuídos e comunicantes, que satisfaz à composicionalidade diagonal. A composicionalidade diagonal [11] corresponde à composicionalidade vertical, uma metodologia de especificação hierárquica a qual permite adicionar estruturas a um sistema concorrente, em diferentes níveis de abstração e a composicionalidade horizontal, pela qual se entende e desenvolve melhor um sistema complexo (entidade estruturada), pois permite considerar as suas diversas componentes separadamente.

Autômatos não-seqüenciais são autômatos especiais onde os estados e as transições possuem estrutura monoidal comutativa e são definidos sobre grafos monoidais comutativos, os quais são grafos reflexivos internos aos monóides comutativos [6,7]. Trata-se de um modelo do tipo não-intercalação (*non-interleaving*).

Assim, um autômato não-seqüencial é um grafo reflexivo com arcos etiquetados, tal que nodos, arcos e etiquetas são elementos de monóides comutativos. O grafo reflexivo representa a forma do autômato onde nodos e arcos correspondem aos estados e transições, respectivamente, com arcos identidade interpretados como transições sem operações. Uma transição estruturada especifica uma relação de concorrência entre as transições componentes. Um estado estruturado pode ser visto como uma soma de recursos locais a serem consumidos ou produzidos pelas transições, analogamente à noção de "token" para as redes de Petri [12,13,14,15,16,17].

Autômatos não-seqüenciais e seus morfismos constituem uma categoria que é completa e co-completa com produtos isomorfos aos coprodutos [8]. O produto (ou coproduto) é interpretado como a composição paralela. No que se tem conhecimento, os autômatos não-seqüenciais são o primeiro modelo de concorrência a satisfazer a composicionalidade diagonal.

No texto que segue, *CMon* denota a categoria dos monóides comutativos e é omitido que $i \in I$ onde I é um conjunto de índices e $k \in \{0, 1\}$. A operação de composição categorial de morfismos será representada pelo símbolo $\circ$, onde $i \circ j$ é a composição dos morfismos $i: b \rightarrow c$ e $j: a \rightarrow b$.

Definição 2.1 (Autômato Não-Seqüencial) *Um autômato não-seqüencial* $N = (\mathbf{V}, \mathbf{T}, \delta_0, \delta_1, \iota, \mathbf{E}, etiq)$ *é tal que* $\mathbf{T} = (T, \otimes, \tau)$, $\mathbf{V} = (V, \otimes, e)$, $\mathbf{E} = (E, \otimes, e)$ *são CMon-objetos de transições, estados e etiquetas respectivamente,* δ_0, δ_1: $\mathbf{T} \rightarrow \mathbf{V}$ *são CMon-morfismos chamados origem e destino respectivamente,* ι: $\mathbf{V} \rightarrow \mathbf{T}$ *é um CMon-morfismo tal que* $\delta_k \circ \iota = id_V$ *e* $etiq$: $\mathbf{T} \rightarrow \mathbf{E}$ *é um CMon-morfismo tal que* $etiq(t) = \tau$ *quando existe um* $v \in \mathbf{V}$ *onde* $\iota(v) = t$ *com* $t \in \mathbf{T}$.

Portanto, um autômato não-seqüencial $N = (\mathbf{V}, \mathbf{T}, \delta_0, \delta_1, \iota, \mathbf{E}, etiq)$ pode ser visto como $N = (G, \mathbf{E}, etiq)$ onde $G = (V, \mathbf{T}, \delta_0, \delta_1, \iota)$ é um grafo reflexivo interno à *CMon* (i.e., V, $\mathbf{T}$ são *CMon*-objetos e δ_0, δ_1, ι são *CMon*-morfismos) representando a forma do autômato, $\mathbf{E}$ é um monóide comutativo representando as etiquetas e $etiq$ é o morfismo de etiquetação. Em um autômato, a transição etiquetada por τ representa uma transição escondida (encapsulada) e portanto, uma transição que não pode ser referenciada externamente ao autômato. Nota-se que toda transição identidade é escondida. A etiquetação é não-extensional, no sentido em que podem existir dois ou mais arcos paralelos com a mesma etiqueta. A etiquetação não-extensional é importante na semântica de reificação.

Uma transição t tal que $\delta_0(t) = X$, $\delta_1(t) = Y$ é denotada por t: $X \rightarrow Y$. Desde que um estado é um elemento de um monóide, pode ser representado como uma soma de nodos locais $n_1A_1 \oplus ... \oplus n_mA_m$, onde a ordem dos termos é irrelevante, A_i está em V e n_i representa a multiplicidade do correspondente estado, para $i = 1...m$. A representação dos arcos como somas é análogo. Uma transição estruturada é uma composição paralela de transições componentes. Quando o contexto não deixar dúvidas, uma transição estruturada $x \otimes \tau$: $X \oplus A \rightarrow Y \oplus A$ onde t: $X \rightarrow Y$ e ι_A: $A \rightarrow A$ são etiquetados por x e τ, respectivamente, é denotada por x: $X \oplus A \rightarrow Y \oplus A$.

Definição 2.2 (Morfismo entre Autômatos Não-Seqüenciais) *Um morfismo de um autômato não-seqüencial* h: $N_1 \rightarrow N_2$ *onde* $N_1 = (\mathbf{V_1}, \mathbf{T_1}, \delta_{01}, \delta_{11}, \iota_1, \mathbf{E_1}, etiq_1)$ *e* $N_2 = (\mathbf{V_2}, \mathbf{T_2}, \delta_{02}, \delta_{12}, \iota_2, \mathbf{E_2}, etiq_2)$ *é uma tripla* $h = (h_V, h_T, h_E)$ *tal que* h_V: $\mathbf{V_1} \rightarrow \mathbf{V_2}$, h_T: $\mathbf{T_1} \rightarrow \mathbf{T_2}$, h_E: $\mathbf{E_1} \rightarrow \mathbf{E_2}$ *são CMon-morfismos,* $h_V \circ \delta_{k1} = \delta_{k2} \circ h_T$, $h_T \circ \iota_1 = \iota_2 \circ h_V$ *e* $h_E \circ etiq_1 = etiq_2 \circ h_T$.

Autômatos não-seqüenciais e seus morfismos constituem a categoria *NAut*. A categoria dos autômatos não-seqüenciais é rica em construções categoriais como restrição, reetiquetação, sincronização, encapsulação e sincronização.

2.1 Restrição e Reetiquetação

Restrição e reetiquetação de transições são operações funtoriais definidas usando as técnicas de fibração e cofibração [15,16]. Ambos funtores são induzidos por morfismos ao

nível de etiquetas. A restrição, restringe um autômato 'apagando' todas as transições que não refletem alguma transição da tabela de restrições:

a) seja N um *NAut*-object onde E é o *CMon*-objeto de etiquetas, seja ***Tabela*** um *CMon*-objeto, chamado tabela de restrições, e seja *restr*: ***Tabela*** $\rightarrow E$ um morfismo de restrição. Seja u: *NAut* $\rightarrow$ *CMon* o funtor esquecimento óbvio, que associa a cada autômato o seu correspondente grafo de etiquetas.

b) o funtor u é uma fibração e as fibras u^{-1} ***Tabela***, $u^{-1} E$ são subcategorias de *NAut*. A fibração u e o morfismo *restr* induzem o funtor ***restr***: $u^{-1} E \rightarrow u^{-1}$ ***Tabela***. O funtor ***restr*** aplicado a N reflete o autômato com a restrição desejada.

Os passos para a reetiquetação são como segue:

a) seja N um *NAut*-objeto onde E_1 é o *CMon*-objeto de etiquetas, seja *reetiq*: $E_1 \rightarrow E_2$ um morfismo de reetiquetação. Seja u o mesmo funtor esquecimento usado para a restrição;

b) o funtor u é uma cofibração (e portanto, uma bifibração) e as fibras $u^{-1} E_1$, $u^{-1} E_2$ são subcategorias de *NAut*. A cofibração u e o morfismo *reetiq* induzem o funtor ***reetiq***: $u^{-1} E_1 \rightarrow u^{-1} E_2$. O funtor ***reetiq*** aplicado a N reflete o autômato com a reetiquetação desejada.

Proposição 2.3 *O funtor esquecimento* u: *NAut* $\rightarrow$ *CMon que associa a cada autômato não-seqüencial o correspondente monóide comutativo de etiquetas é uma fibração e cofibração. Prova: em [9] (proposições 2.6 e 2.8).*

Definição 2.4 (Funtor restr) *Considere a fibração* u: *NAut* $\rightarrow$ *CMon, o autômato* $N = (V, T, \delta_0, \delta_1, \iota, E, etiq)$ *e o morfismo de restrição restr:* ***Tabela*** $\rightarrow E$. *A restrição de N é dada pelo funtor* ***restr***: $u^{-1} E \rightarrow u^{-1}$ ***Tabela*** *induzida por* u *e reetiq aplicada a N.*

Definição 2.5 (Funtor reetiq) *Considere a fibração* u: *NAut* $\rightarrow$ *CMon, o autômato* $N = (V, T, \delta_0, \delta_1, \iota, L_1, lab)$ *e o morfismo de reetiquetação reetiq:* $E_1 \rightarrow E_2$. *A reetiquetação N é dada pelo funtor* ***reetiq***: $u^{-1} E_1 \rightarrow u^{-1} E_2$ *induzida por* u *e reetiq aplicado a N.*

2.2 Sincronização e Encapsulação

Sincronização e encapsulação de autômatos não-seqüenciais são casos especiais de restrição e reetiquetação, respectivamente. Desde que o produto (ou coproduto) em *NAut* é visto como a composição paralela, refletindo todas as possíveis combinações entre as transições componentes, a sincronização pode ser definida usando a operação de restrição e apagando das composição paralela todas transições que não refletem uma transição da tabela de sincronizações [9,10]. Uma visão do autômato é obtida escondendo as transições. Uma transição escondida, isto é, reetiquetada τ, não pode ser usada para sincronização.

3 Reificação

A reificação é um mecanismo de abstração baseado em transações que consiste em implementar uma ação de alto nível (uma primitiva do sistema operacional, por exemplo), de forma indivisível e em um tempo finito, como composições seqüenciais ou concorrentes (ou ambas) de ações de baixo nível (i.e., várias operações que formam uma ação atômica). A reificação de autômatos não-seqüenciais é um morfismo especial, onde o objeto destino é enriquecido com o seu fecho computacional não-seqüencial, isto é, o autômato destino (mais concreto) é enriquecido com todas as computações possíveis que podem ser obtidas a partir das permutações seqüenciais e não-seqüenciais das transições componentes, respeitando os estados origem e destino.

A categoria das categorias internas a *CMon* é denotada por *Cat(CMon)*. A categoria *ECat(CMon)* pode ser vista como uma generalização da etiquetação de *Cat(CMon)*. O funtor *cn: ECat(CMon)* $\rightarrow$ *NAut*, que esquece a operação de composição e introduz alguns requisitos sobre a concorrência, possui adjunto esquerdo *nc: NAut* $\rightarrow$ *ECat(CMon)* o qual gera livremente a composição nas transições. O fecho computacional não-seqüencial é obtido pela composição dos dois funtores, ou seja, *nfc = cn* $\circ$ *nc: NAut* $\rightarrow$ *NAut*. A composição de reificações de autômatos não-seqüenciais é inspirada pelas categorias Kleisli [6]. Na verdade, a adjunção acima induz uma mônada a qual define uma categoria Kleisli. Então, a reificação se distribui sobre a composição paralela e portanto, a categoria resultante *ReifNAut* de autômatos e reificações satisfaz a composicionalidade diagonal.

Definição 3.1 (Categoria ECat(CMon)) *Seja a categoria Cat(CMon). A categoria* E*Cat(CMon) é a categoria das setas* $id_{Cat(CMon)} \downarrow id_{Cat(CMon)}$ *onde* $id_{Cat(CMon)}$ *é o funtor identidade em Cat(CMon). Portanto, um* E*Cat(CMon)-objeto é uma tripla* N = (G, E, etiq) *onde* G, E *são Cat(CMon)-objetos e* etiq *é um Cat(CMon)-morfismo.*

Proposição 3.2 *A categoria* ECat(CMon) *possui todos coprodutos (pequenos), os quais são isomorfos aos produtos. Prova: em [8] (proposição 6.2).*

Definição 3.3 (Funtor cn) *Seja* N = (G, E, etiq) *um ECat(CMon)-objeto e* h = (h_G, h_E): $N_1 \rightarrow N_2$ *um* ECat(CMon)*-morfismo. O funtor* **cn**: ECat(CMon) $\rightarrow$ NAut *é tal que:*

a) para o Cat(CMon)*-objeto* G = (**V**, **T**, δ_0, δ_1, ι, ;), *cn* G *é o* RGr(CMon)*-objeto* G = (**V**, **T'**, δ_0', δ_1', ι'), *onde* **T'** *é* **T** *sujeito à regra equacional abaixo e* δ_0', δ_1', ι' *são induzidos por* δ_0, δ_1, ι *considerando o monóide* **T'**; *para o* Cat(CMon)*-objeto* E = (**V**, **L**, δ_0, δ_1, ι, ;), *cn* E *é o* CMon*-objeto* **E'**, *onde* **E'** *é* **E** *sujeito as mesmas regras equacionais; para o* ECat(CMon)*-objeto* N = (G, E, etiq), *cn* N *é o* NAut*-objeto N = (G,* **E'**, *etiq) onde etiq é o* RGr(CMon)*-morfismo canonicamente induzido pelo* Cat(CMon)*-morfismo* etiq;

$$\frac{t : A \rightarrow B \in T' \quad u : B \rightarrow C \in T' \quad t' : A' \rightarrow B' \in T' \quad u' : B' \rightarrow C' \in T'}{(t;u) \otimes (t';u') = (t \otimes t');(u \otimes u')}$$

b) para cada ECat(CMon)*-morfismo* $\mathsf{h} = (\mathsf{h_G}, \mathsf{h_E})$: $\mathsf{N_1} \rightarrow \mathsf{N_2}$ *onde* $\mathsf{h_G} = (h_{NV}, h_{NT})$, $\mathsf{h_E} = (h_{EV}, h_{ET})$, *tem-se que cn*h *é o* NAut*-morfismo* $h = (h_{NV}, h_{NT'}, h_{ET'})$: $N_1 \rightarrow N_2$ *onde* $h_{NT'}$ *e* $h_{ET'}$ *são morfismos entre monóides induzidos por* h_{NT} *e* h_{ET}, *respectivamente.*

O funtor ***cn*** independentemente de esquecer sobre a operação de composição introduz o seguinte requisito com respeito à concorrência (t;u)⊗(t';u') = (t⊗t');(u⊗u'). Ou seja, a composição paralela de duas computações independentes t;u e t';u' possui o mesmo efeito que a composição seqüencial das computações paralelas t⊗t' e u⊗u'.

Definição 3.4 (Funtor nc) *Seja* $A = (G, \mathbf{L}, etiq)$ *um* NAut*-objeto e* $h = (h_G, h_L)$: $A_1 \rightarrow A_2$ *um* NAut*-morfismo. O funtor* **nc**: NAut $\rightarrow$ ECat(CMon) *é tal que:*

a) para o RGr(CMon)*-objeto* $G = (\mathbf{V}, \mathbf{T}, \delta_0, \delta_1, \iota)$ *com* $\mathbf{V} = (V, \oplus, e)$, $\mathbf{T} = (T, \otimes, \tau)$, *ncG é o* Cat(CMon)*-objeto* $\mathsf{G} = (\mathbf{V}, \mathbf{T^c}, \delta_0{}^c, \delta_1{}^c, \iota, ;)$ *com* $\mathbf{T^c} = (T^c, \otimes, \tau), \delta_0{}^c, \delta_1{}^c, \_;\_: \mathbf{T^c} \times \mathbf{T^c} \rightarrow \mathbf{T^c}$ *indutivamente definidos como segue:*

$$\frac{t : A \rightarrow B \in T}{t : A \rightarrow B \in T^c}$$

$$\frac{t : A \rightarrow B \in T^c \quad u : B \rightarrow C \in T^c}{t;u : A \rightarrow C \in T^c} \qquad \frac{t : A \rightarrow B \in T^c \quad u : C \rightarrow D \in T^c}{t \otimes u : A \oplus C \rightarrow B \oplus D \in T^c}$$

sujeitos às seguintes regras equacionais:

$$\frac{t \in T^c}{\tau;t = t \;\&\; t;\tau = t} \qquad \frac{t : A \rightarrow B \in T^c}{i_A;t = t \;\&\; t;i_B = t}$$

$$\frac{t : A \rightarrow B \in T^c \quad u : B \rightarrow C \in T^c \quad v : C \rightarrow D \in T^c}{t;(u;v) = (t;u);v}$$

$$\frac{t \in T^c \quad u \in T^c}{t \otimes u = u \otimes t} \qquad \frac{t \in T^c}{t \otimes \tau = t} \qquad \frac{i_A \in T^c \quad i_B \in T^c}{i_A \otimes i_B = i_{A \oplus B}}$$

$$\frac{t \in T^c \quad u \in T^c \quad v \in T^c}{t \otimes (u \otimes v) = (t \otimes u) \otimes v}$$

Para o *CMon*-objeto ***E***, ***ncE*** é o *Cat(CMon)*-objeto E = (e, E^c, !, !, !, ;) como acima, onde e denota um monóide com um único elemento e ! denota o morfismo único tendo e como origem e destino; para o *NAut*-objeto *A* = *(G, E, etiq)*, ncA é o E*Cat(CMon)*-objeto A = (G, E, etiq) onde etiq é o morfismo induzido por *etiq;*

b) para o NAut*-morfismo h = (*h_V, h_T, h_E*): $A_1 \rightarrow A_2$, nch é o* Cat(CMon)*-morfismo* h = (hG, hE): A1 → A2 *com* hG = (*h_V, $h_T{}^C$*) e* hE = (!, $h_E{}^C$*), onde $h_T{}^C$ e $h_E{}^C$ são* Cat(CMon)*-morfismos gerados pelos morfismos h_T and h_E, respectivamente.*

Proposição 3.5 O funtor ***nc****: NAut* → E*Cat(CMon) é adjunto esquerdo do funtor* ***cn****:* E*Cat(CMon)* →*NAut. Prova: em [8](proposição 6.5) e [9] (proposição 2.20).*

Definição 3.6 (Fecho Computacional) *O fecho computacional (não-seqüencial) é a operação funtorial* ***nfc*** = ***tc*** = ***cn*** ∘ ***nc***: *NAut* → *NAut.*

A adjunção <***nc***, ***cn***, η, ε>: *NAut* → E*Cat(CMon)* induz uma mônada T = <***tc***, η, μ> em *NAut* onde μ = ***cn*** ε ***nc***: $tc^2 \rightarrow tc$, ***cn***: ***cn*** → ***cn*** e ***nc***: ***nc*** → ***nc*** são as transformações naturais identidade e ***cn*** ε ***nc*** é a composição horizontal das transformações naturais. Para um dado autômato N, ***tc*** N é N enriquecido com suas computações, η_N: N → ***tc*** N inclui N no seu fecho computacional e μ_N: tc^2N → ***tc*** N é a operação de achatamento a qual especifica como duas aplicações sucessivas do fecho computacional ao autômato N é instanciada em uma única aplicação do fecho computacional.

Um morfismo de reificação φ de A em computações de B pode ser definido como um *NAut*-morfismo φ: A → ***tc*** B e a composição de reificações é definido como a composição nas categorias Kleisli (cada mônada define uma categoria Kleisli).

Definição 3.7 (Categoria ReifNAut) *Seja t =* <**tc**, *η, μ> onde η =* <η_G, η_E>, *μ =* <μ_G, μ_E> *seja a mônada induzida pela adjunção* <**nc**, **cn**, *η, ε>: NAut* → E*Cat(CMon). A categoria dos autômatos não-seqüencias e das reificações, denotada por ReifNAut, é tal que (suponha os NAut-objetos* N_k = <G_k, E_k, $etiq_k$>, *para k em {1, 2, 3}):*

a) ReifNAut-objetos são os NAut-objetos;

b) φ = φ_G: $N_1 \rightarrow N_2$ *é um ReifNAut-morfismo onde* φ_G: $G_1 \rightarrow$ **tc** G_2 *é um* RGr(CMon)-morfismo e para cada *NAut-objeto N,* φ = η_G: $N \rightarrow N$ *é o morfismo identidade de N em ReifNAut;*

c) sejam φ: $N_1 \rightarrow N_2$, ψ: $N_2 \rightarrow N_3$ *ReifNAut-morfismos. A composição* ψ ∘ φ *é o morfismo* $\psi_G \circ_K \varphi_G$: $N_1 \rightarrow N_3$ *onde* $\psi_G \circ_K \varphi_G$ *é determinado como ilustrado na Figura 1.*

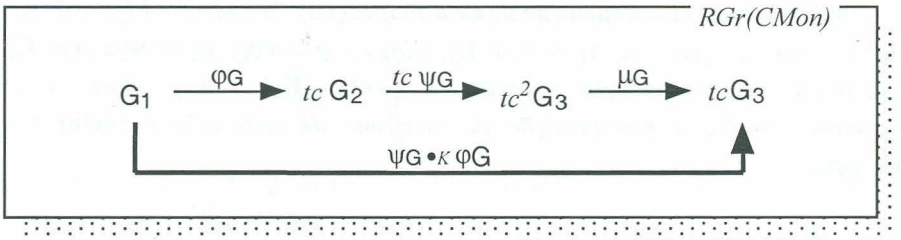

Figura 1. Composição de reificações

No que segue, um autômato $<G, \boldsymbol{E}, etiq>$ pode ser denotado como um morfismo $etiq: G \rightarrow inc\ \boldsymbol{E}$ ou na sua forma abreviada $etiq: G \rightarrow \boldsymbol{E}$.

Proposição 3.8 *O endofuntor nfc:NAut → NAut preserva produtos (ou coprodutos), isto é,* $nfc \times N_i = \times (nfc N_i)$. *Prova: em [9] (proposição 2.25).*

Proposição 3.9 *Seja* $\{\varphi_i: N1_i \rightarrow \mathbf{tc}\ N2_i\}$ *uma família indexada de reificações. Então* $\times_{i \in I} \varphi_i: \times_{i \in I} N1_i \rightarrow \times_{i \in I}\ \mathbf{tc}\ N2_i$ *é uma reificação. Prova: em [8] (proposição 6.14).*

3.1 Restrição e Reetiquetação de Reificações

A restrição de uma reificação é a restrição do autômato origem. A restrição de uma família de reificações (a composição paralela de autômatos reificados) é a restrição da composição paralela do autômato origem cuja reificação é induzida pelas reificações componentes. Para esta construção, a composição horizontal é um resultado necessário. O endofuntor ***tc*** preserva produtos e todo morfismo de restrição (ao nível de etiquetas) possui uma elevação cartesiana, ao nível de autômatos. A reetiquetação de uma reificação é induzida pela reetiquetação do autômato origem.

Definição 3.10 (Restrição de uma Reificação) *Seja* $\varphi: N1 \rightarrow \mathbf{tc}\ N2$ *uma reificação e seja* $restr_E$: ***Tabela*** $\rightarrow \mathbf{E}1$ *um morfismo de restrição e seja* $restr_N$: ***restr*** $N1 \rightarrow N1$ *sua elevação cartesiana. A reificação de um autômato restringido* ***restr*** $N1$ *é* ***restr*** φ: ***restr*** $N1 \rightarrow$ ***tc*** $N2$ *tal que* ***restr*** $\varphi = \varphi \circ restr_N$.

Proposição 3.11 *Seja* $\{\varphi_i: N1_i \rightarrow \mathbf{tc}\ N2_i\}$ *uma família indexada de reificações onde* $N_{ki} = <G_{ki}, \boldsymbol{E}_{ki}, etiq_{ki}>$. *Seja* $restr_E$: ***Tabela*** $\rightarrow \times_i \boldsymbol{E1}_i$ *um morfismo de restrição e seja* $restr_N$: ***restr*** $N1_i \rightarrow \times_i N1_i$ *sua elevação cartesiana. A restrição da composição paralela das reificações componentes é* ***restr*** φ_i: ***restr*** $N1_i \rightarrow$ ***tc*** $(\times_i N2_i)$ *tal que* ***restr*** $\varphi_i = \times_i \varphi_i \circ restr_N$ *onde* $\times_i \varphi_i$ *é induzido unicamente pela construção produto. Prova: em [9] (proposição 2.27).*

Definição 3.12 (Reetiquetação de uma Reificação) *Considere a Figura 2. Seja* φ: $N_1 \rightarrow$ **tc** N_2 *uma reificação onde* $N_k = \langle G_k, \mathbf{E}_k, etiq_k \rangle$ *e* $\varphi = \langle \varphi_G, \varphi_E \rangle$. *Seja etiq:* $\mathbf{E}_1 \rightarrow \mathbf{E}_1'$ um morfismo de reetiquetação *e* **reetiq** $N_1 = \langle G_1, \mathbf{E}_1', reetiq \circ etiq_1 \rangle$ *o autômato reetiquetado. Então, a reetiquetação do morfismo de reificação é* **reetiq** $\varphi = \langle \varphi_G,$ **reetiq** $\varphi_E \rangle$.

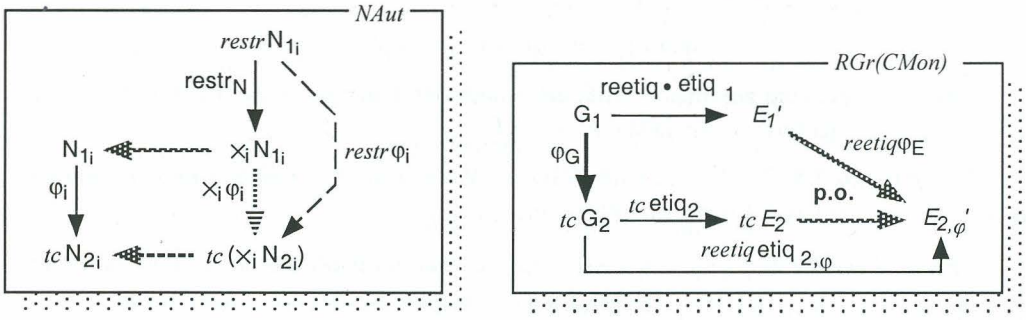

Figura 2. Restrição e reetiquetação de uma reificação

4 Semântica da Reificação

Nesta seção é descrita a semântica de composições de primitivas, possibilitando a definição de novas primitivas a partir de outras primitivas. A semântica de uma primitiva é especificada por um funtor que mapeia uma reificação em uma computação. Uma computação de autômatos não-seqüenciais implementa as computações de um autômato sobre as computações de um outro autômato. Dessa maneira, pode ser definida como um morfismo de autômatos onde os objetos origem e destino são enriquecidos com o fecho computacional não-seqüencial. Autômatos e computações constituem a categoria do domínio semântico *CReifNAut,* onde objetos e morfismos são da categoria *NAut*. A semântica da reificação é dada pelo funtor ***sr:*** *ReifNAut* $\rightarrow$ *CReifNAut* , o qual especifica como uma reificação é instanciada em uma computação.

Definição 4.1 (Categoria dos Autômatos Não-Seqüenciais e Computações CReifNAut) *Seja* $T = \langle \mathbf{tc}, \eta, \mu \rangle$ *onde* $\eta = \langle \eta_M, \eta_E \rangle$ *e* $\mu = \langle \mu_M, \mu_E \rangle$ *a mônada determinada pela adjunção* $\langle \mathbf{nc}, \mathbf{cn}, \eta, \varepsilon \rangle$*:NAut*$\rightarrow$ *ECat(CMon). A categoria dos autômatos não-seqüenciais e computações, denotada por CReifNAut, é tal que (supõe-se os NAut-objetos* $N_k = \langle G_k, E_k, etiq_k \rangle$*, para k com valores em {1, 2, 3}):*

*a) os CReifNAut-objetos são os NAut-*objetos $M =$ **tc***N, tal que N é um NAut-objeto.*

b) os CReifNAut -morfismos são construídos como segue:

b1) uma computação $\varphi = \varphi_G$: $tcN_1 \rightarrow tcN_2$ *é um NAut-morfismo onde* φ_G: $tcG_1 \rightarrow tcG_2$ *é um RGr(CMon)-morfismo.*

b2) para cada NAut objeto N, φ: $tcN \rightarrow tcN$ *é o CReifNAut-morfismo identidade de N.*

c) sejam φ: $tcN_1 \rightarrow tcN_2$, ψ: $tcN_2 \rightarrow tcN_3$ *CReifNAut-morfismos. A composição* $\psi_G \circ \varphi_G$: $tcN_1 \rightarrow tcN_3$ *é um RGr(CMon)-morfismo ilustrado na Figura 3.*

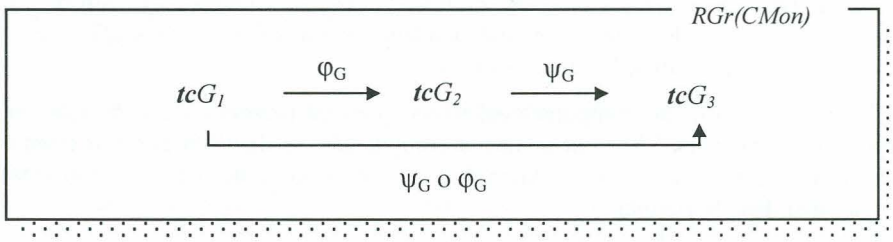

Figura 3. Composição de computações

CReifNAut (Figura 4) é uma subcategoria plena de *NAut*, sendo importante notar que a categoria possui todos os morfismos entre computações, ou seja, inclui morfismos que não serão considerados como semântica válida para a reificação.

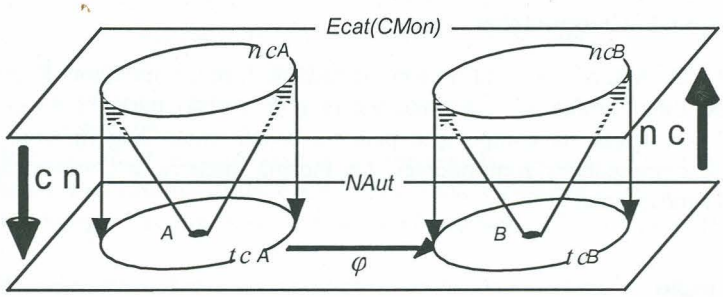

Figura 4. Categoria *CreifNAut*

Definição 4.2 (Semântica da Reificação) *Seja N = <G, E, etiq> um ReifNAut-objeto onde G=(V, T,* δ_0, δ_1, ι*) e* φ: $N_1 \rightarrow N_2$ *um ReifNAut-morfismo. A semântica de uma reificação é dada pelo funtor* **sr**: *ReifNAut* $\rightarrow$ *CReifNAut , tal que:*

a) para o ReifNAut-objeto N, **srN** *é o CReifNAut-objeto N' =* **tc***N;*

b) para o ReifNAut-morfismo $\varphi:N_1 \rightarrow N_2$, **sr**$\varphi$ *é o CReifNAut-morfismo* $\varphi':N_1' \rightarrow N_2'$ *indutivamente definido como segue (onde* N_1' *e* N_2' *são definidos no item anterior e t, w são transições em* N_1'*):*

$$b.1)\ \varphi'(<t>) = \varphi(<t>);$$

$$b.2)\ \varphi'(<t;w>) = \varphi(<t>);\varphi'(<w>).$$

É importante notar que <t;w> está representando uma classe de transições, uma vez que o funtor ***cn*** independentemente de esquecer sobre a composição, introduz que a execução concorrente de duas transições independentes é equivalente à execução seqüencial, em qualquer ordem, das transições componentes.

Uma vez que o fecho computacional não-seqüencial preserva a composição paralela de autômatos (proposição 3.8), tem-se que a computação se distribui sobre a composição paralela dos autômatos componentes. Além disso, a categoria do domínio semântico satisfaz a composicionalidade horizontal.

Proposição 4.3 *Seja* $\{\varphi_i: \mathbf{tc}N_{1i} \rightarrow \mathbf{tc}N_{2i}\}$ *uma família indexada de computações. Então* $\times_{i\in I}\varphi_i: \times_{i\in I}\mathbf{tc}N_{1i} \rightarrow \times_{i\in I}\mathbf{tc}N_{2i}$ *é uma computação.*

Prova: Desde que **nc** *é adjunto esquerdo de* **cn** *então* **nc** *preserva colimites e* **cn** *preserva limites. Desde que produtos e coprodutos são isomórficos em Ecat(CMon),* **tc** *preserva produtos. Seguindo esta aproximação, prova-se que* $\times_{i\in I}\varphi_i$ *é uma computação.*

4.1 Restrição de Computações

A restrição de uma computação é construída de forma semelhante à reificação. Isto é, a restrição de uma família de computações (a composição paralela de computações de autômatos) é a restrição da composição paralela do autômato origem, com a computação induzida pelas computações componentes. Da mesma maneira, a composição horizontal é um resultado fundamental.

5 Conclusão

Autômatos não-seqüenciais constituem um domínio semântico categorial baseado em um sistema de transições rotuladas com concorrência plena. Este modelo satisfaz a composicionalidade diagonal, ou seja, onde as reificações compõem (verticalmente) e distribuem-se sobre a composição paralela (horizontalmente).

Uma reificação mapeia transições em transações refletindo a implementação de um autômato sobre outro autômato. Sob a ótica do domínio semântico apresentado, uma

primitiva (ação do autômato origem) é composta por várias operações formando uma transação atômica (seqüência de ações do objeto destino).

Autômatos e computações constituem a categoria do domínio semântico. Uma computação de autômatos não-seqüenciais implementa transações em transações. Dessa maneira, a semântica de uma primitiva é dada por um funtor o qual especifica como mapear uma reificação em uma computação.

Através da interpretação semântica desenvolvida, foi possível estender o conceito de transação típico de sistemas distribuídos para primitivas do sistema operacional em si, tornando possível a composição de primitivas complexas sobre primitivas mais simples, ou seja, a definição de primitivas a partir de primitivas.

Referências

[1] Tanenbaum, A. S. *Sistemas Operacionais Modernos*. Prentice-Hall do Brasil, 1995.
[2] Stallings, W. *Operating Systems: internals and design principles*. 4 ed. Prentice-Hall, 2001.
[3] Silberschatz, A; Galvin, P.; Gagne, G. *Sistemas Operacionais*. Campus, 2000.
[4] Mac Lane, S. *Categories for the Working Mathematician*. New York: Springer-Verlag, 1971.
[5] Barr, M.; Wells, C. *Category Theory for Computing Science*. London: Prentice Hall, 1990
[6] Asperti, A.; Longo, G. *Categories, Types and Structures - An Introduction to the Working Computer Science*. Foundations of Computing (Garey, M.; Meyer, A. Eds.), MIT Press, 1991.
[7] Adámek, J.; Herrlich, H.; Strecker, G. *Abstract and Concrete Categories*. Wiley, 1990.
[8] Menezes, P. B.; Costa, J. F. *Compositional Reification of Concurrent Systems*. Journal of the Brazilian Computer Society, v.2, n.1, Rio de Janeiro, Brazil, p.50-67,1995.
[9] Menezes, P. B.; Costa, J. F.; Sernadas, A. *Nonsequencial Automata Semantics for a Concurrent, Object Based Language*. Electronic Notes in Theoretical Computer Science 14, Elsevier Science, pp. 29, 1998.
[10] Menezes P. B.; Costa, J. F.; Sernadas, A. *Refinement Mapping for (Discrete event) System Theory*. Proceedings of the Fifth International Conference on Computer Aided System Technology, Lecture Notes in Computer Science 1030, Springer-Verlag, p.103-116, 1996.
[11] Gorrieri, R. *Refinement, Atomicity and Transactions for Process Description Language*. Tese de doutoramento, Università di Pisa, 1990.
[12] Reisig, W. *Petri Nets: An Introduction*. EATCS Monographs on Theoretical Computer Science 4, Springer- Verlag, 1985.
[13] Bednarczyk, M. A. *Categories of Asynchronous Systems*. Ph.D. thesis, technical report 1/88, University of Sussex, 1988.
[14] Mazurkiewicz, A. *Basic Notion of Trace Theory*. REX 88: Linear Time, Branching Time and Partial Orders in Logic and Models for Concurrency (J. W. de Bakker, W. -P. de Roever Eds.), LNCS 354, Springer-Verlag, pp.285-363, 1988.

[15] Menezes, P. B.; Costa, J. F. *Synchronization in Petri Nets*. Fundamenta Informaticae v.26.1, p.11-22, Annales Societatis Mathematicae Polonae, IOS Press, 1996.
[16] Winskel, G. *Petri Nets, Algebras, Morphisms and Compositionality*. Information and Computation 72, Academic Press, p.197-238, 1987.
[17] Meseguer, J.; Montanari, U. *Petri Nets are Monoids*. Information and Computation 88, Academic Press, p.105-155, 1990.