UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

LUCIANA FOSS

Uma Tradução de Gramáticas de Hipergrafos Baseadas em Objetos para Cálculo-π

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Profa. Dra. Leila Ribeiro
Orientadora

Porto Alegre, maio de 2003

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Foss, Luciana

Uma Tradução de Gramáticas de Hipergrafos Baseadas em Objetos para Cálculo-π / Luciana Foss. – Porto Alegre: PPGC da UFRGS, 2003.

76 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2003. Orientadora: Leila Ribeiro.

1. Gramática de hipergrafos. 2. Sistemas baseados em objetos. 3. Cálculo-π. I. Ribeiro, Leila. II. Título.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

CHGF	Cláusulas de Horn Guardadas *Flat*
EGG	Estruturas de Grafos Generalizadas
GGBO	Gramática de Grafos Baseada em Objetos
GHBO	Gramática de Hipergrafos Baseada em Objetos
HAL	*History Dependant Automata Laboratory*
MBO-π	Modelo Baseado em Objetos descrito em Cálculo-π
SBO	Sistema Baseado em Objetos
STR	Sistema de Transições Rotulado

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

O aumento da escala e funcionalidade dos sistemas de computação e sua crescente complexidade envolvem um aumento significante de custos e exigem recursos humanos altamente qualificados para o desenvolvimento de software. Integrando-se o uso de métodos formais ao desenvolvimento de sistemas complexos, permite-se realizar análises e verificações destes sistemas, garantindo assim sua correção.

Existem diversos formalismos que permitem descrever sistemas, cada qual com diferentes níveis de abstração. Quando consideramos sistemas complexos, surge a necessidade de um modelo que forneça construções abstratas que facilitem o entendimento e a especificação destes sistemas. Um modelo baseado em objetos fornece um nível de abstração que tem sido muito aplicado na prática, onde os dados e os processos que os manipulam são descritos juntos em um objeto.

Gramática de Grafos Baseada em Objetos (GGBO) é um modelo baseado em objetos, que além de ser uma linguagem visual, apresenta a vantagem de as especificações adquirirem um estilo baseado em objetos, que é bastante familiar à maioria dos desenvolvedores.

Porém, as GGBOs não possuem ainda ferramentas para verificação automática de propriedades desejadas nos sistemas modelados. Uma alternativa para resolver isso é definir uma tradução (que preserve a semântica) desta linguagem para outra, para a qual existam verificadores automáticos.

Um formalismo bastante conhecido e estabelecido para descrição de sistemas concorrentes, para o qual existem verificadores automáticos, é o cálculo-π. Porém, sob o aspecto de especificação de sistemas complexos, GGBOs parecem ser mais adequadas como linguagem de especificação que o cálculo-π, pois são visuais, mais intuitivas e possuem um estilo baseado em objetos.

Neste trabalho foi definido um formalismo (baseado nas GGBOs), denominado Gramática de Hipergrafos Baseada em Objetos e uma tradução deste formalismo para o cálculo-π, aliando assim as vantagens desses dois métodos. Além disso, para validar a tradução definida, foram feitas provas de que a semântica das gramáticas de hipergrafos baseadas em objetos é preservada na tradução.

Palavras-chave: Gramática de hipergrafos, sistemas baseados em objetos, cálculo-π.

A Translation from Object-Based Hypergraph Grammars into π-Caculus

ABSTRACT

The increase of size and functionality of computation systems and their growing complexity generates higher development costs and demands highly qualified human resources for the development of software. The integration of formal methods to development process of complex systems allows us to analyze and check them.

There are several formalisms for the specification of computational systems, each one with different abstraction levels. Models that support abstract constructions are desirable to help the understanding and specification of complex systems. Object-based model offer an abstraction level that has been successfully applied in practice, where operations and data are described together within one object.

Object-Based Graph Grammar (OBGG) is an object-based model that, besides the features of object-based languages, offers a visual specification language, what is usually welcome by practitioners.

However, up to now, there are no automatic tools for verification of OBGGs. Instead of construction such tools from scratch, an alternative approach is to define a (semantics preserving) translation from this language (OBGG) into another for which automatic verifications tools already exist.

The π-calculus is a well known and established formalism for the description concurrent systems. There are some automatic checkers for this formalism. But, for the specification of complex systems, OBGGs seem to be more suited that π-calculus as a specification language, because they are visual, more intuitive, and follow an object-based style.

In this work we define a formalism (based on OBGG) called Object-Based Hypergraph Grammars and a translation into π-calculus that allows us to join the advantages of both methods. Moreover, we prove that this translation preserves the semantics of OBGGs.

Keywords: hypergraph grammars, object-based systems, π-calculus.

1 INTRODUÇÃO

Aplicações complexas requerem um processo de desenvolvimento que resulte em um sistema correto e adequado às suas necessidades. Um desenvolvimento rigoroso de software tem como objetivo oferecer garantias de que propriedades fundamentais do sistema não sejam violadas durante o seu funcionamento (BOWEN; HINCHEY, 1999; HERRMANN, 2000; STOREY, 1996; WICHMANN, 1992). Empresas como Intel, IBM, Praxis, órgãos de controle de aviação e tráfego aéreo norte-americanos e ingleses, Tektronix e outras (BOWEN; HINCHEY, 1994; CLARKE; WING, 1996) têm utilizado métodos formais como parte integrante do seu processo de desenvolvimento. Para obter-se um desenvolvimento de software rigoroso, integra-se a este processo o uso de métodos formais. Nesta abordagem, o primeiro passo é construir um modelo da solução (especificação) usando uma linguagem formal.

Existem diversos formalismos que permitem descrever a especificação de sistemas, cada qual com diferentes níveis de abstração. Quando são considerados sistemas complexos, surge a necessidade de um modelo que forneça construções abstratas que facilitem o entendimento e a especificação deste sistema. Um modelo baseado em objetos fornece um nível de abstração que tem sido muito aplicado na prática, onde os dados e os processos que os manipulam são descritos juntos em um objeto. Uma maneira de modelar sistemas baseados em objetos usando gramáticas de grafos foi apresentada em (KORFF, 1995) inspirada no modelo de atores (AGHA, 1986). A idéia básica foi usar gramáticas de grafos como uma linguagem para especificar sistemas de atores. Seguindo as mesmas idéias, foi proposta, em (DOTTI; RIBEIRO, 2000), uma linguagem de especificação formal para especificação de aplicações móveis, chamada Gramática de Grafos Baseadas em Objetos (GGBO), que baseia-se em uma forma restrita de gramáticas de grafos (EHRIG, 1978). Além de ser uma linguagem visual, esta apresenta a vantagem de as especificações adquirirem um estilo baseado em objetos, que é bastante familiar à maioria dos desenvolvedores.

Este trabalho está inserido no projeto IQ-Mobile (IQ-MOBILE, 2000) (cooperação internacional entre Brasil e Itália, envolvendo as universidades UFRGS, PUCRS, UFBA, Universidade di Pisa e CNR). Este projeto tem como principal objetivo fornecer qualidade de software para ambientes abertos, em especial aplicações móveis e distribuídas, através do uso de métodos formais. Dentro do escopo deste projeto, as GGBOs estão sendo utilizadas como linguagem de especificação formal para sistemas de ambientes abertos. Esta linguagem (GGBO) foi mapeada para um subconjunto da linguagem Java: o simulador PLATUS (COPSTEIN; COSTA MÓRA; RIBEIRO, 2000) apresenta uma implementação de GGBO na linguagem Java para criar seus modelos de simulação. As GGBOs, porém, não permitem verificar automaticamente propriedades desejadas nos sistemas modelados, pois não foram desenvolvidos algoritmos para fazer essa verificação. Uma alternativa

para resolver isso, é definir uma tradução (que preserve a semântica) desta linguagem para outra para a qual existam verificadores automáticos.

Um formalismo bastante conhecido para descrição formal de sistemas é o cálculo-π (MILNER; PARROW, 1992). Para este formalismo, existem diversos verificadores de modelos como por exemplo o HAL (FERRARI et al., 1998) e o MWB (Mobility Workbench) (VICTOR; MOLLER, 1994). O HAL (HD-Automata Laboratory) foi desenvolvido por parceiros do projeto IQ-Mobile e está sendo usado neste projeto como ferramenta para especificação, verificação e análise de sistemas concorrentes e distribuídos.

O ambiente HAL inclui módulos que implementam procedimentos de decisão para cálculo de equivalência comportamental, e módulos que suportam verificação de propriedades expressas como fórmulas de uma lógica temporal adequada. Neste momento o HAL trabalha somente com sistemas concorrentes e distribuídos expressos em cálculo-π. Esse ambiente permite traduzir agentes do cálculo-π em autômatos, de modo que os verificadores de equivalência existentes possam ser usados para calcular se os agentes do cálculo-π são bissimilares. O ambiente suporta também a verificação das fórmulas lógicas que expressam propriedades desejadas do comportamento de agentes do cálculo-π. Na implementação atual, o ambiente HAL consiste essencialmente de cinco módulos: três módulos executam as traduções dos agentes do cálculo-π para HD-autômatos (MONTANARI; PISTORE, 1998) (pi-a-hd), dos HD-autômatos para autômatos (hd-a-aut), e das fórmulas da lógica-π para fórmulas ACTL (pl-a-actl). O quarto módulo (hd reduce) fornece rotinas que manipulam os HD-autômatos. O quinto módulo é basicamente o ambiente JACK que trabalha no nível de autômatos e executa as operações padrão sobre eles como a verificação comportamental e de modelos.

Sob o aspecto de especificação de sistemas, as GGBOs parecem ser mais intuitivas e portanto mais fáceis de utilizar do que o cálculo-π, principalmente considerando-se que a maioria dos desenvolvedores de software não possui muita afinidade com métodos formais, e sim com métodos gráficos e com estilo baseado em objetos.

Neste trabalho, foram definidas uma gramática de hipergrafos baseadas em objetos (GHBO) baseada nas GGBOs e uma tradução destas GHBOs para cálculo-π, que permite aliar as vantagens desses dois métodos, permitindo a verificação automática de propriedades de sistemas especificados através de GHBO. Contudo, a validade destas verificações depende do fato de que estas traduções preservam a semântica do sistema modelado. Esta prova também é apresentada neste trabalho.

1.1 Estrutura da Dissertação

No capítulo 2 são apresentados alguns trabalhos relacionados, isto é, traduções de alguns outros tipos de linguagens para o cálculo-π.

No capítulo 3 são apresentados alguns conceitos e características de sistemas baseados em objetos que são consideradas na tradução entre os modelos propostos neste trabalho. É apresentado um exemplo simples de sistema baseado em objetos cuja especificação em GHBO é apresentada no capítulo 4.

No capítulo 4 é dada a definição de gramáticas de hipergrafos baseadas em objetos e sua semântica. Essa definição é dada a partir de diversas definições que também são apresentadas neste capítulo.

No capítulo 5 é apresentada uma breve descrição do cálculo-π e de sua semântica e definido o modelo baseado em objetos descrito em cálculo-π (MBO-π). A semântica deste modelo é também definida aqui, e é dada através de um sistema de transição rotulado.

No capítulo 6 é definida a tradução das gramáticas de hipergrafos baseadas em objetos para o modelo baseado em objetos descrito em cálculo-π. Esta tradução é dada através de funções que geram os elementos do MBO-π. Aqui, é também apresentada a tradução do exemplo especificado no capítulo 4. No final deste capítulo é provado que a tradução preserva a semântica das gramáticas de hipergrafos baseadas em objetos, através da tradução dos STR dos MBO-π para o das GHBO.

Por fim, no capítulo 7 são apresentadas as conclusões deste trabalho, bem como os trabalhos futuros.

2 TRABALHOS RELACIONADOS

Neste capítulo são apresentados alguns trabalhos já realizados de tradução de diferentes tipos linguagens para o cálculo-π. Essas traduções foram realizadas por diferentes razões, mas na maioria dos casos o objetivo é de dar a semântica para a linguagem de origem da tradução, a partir da semântica do cálculo-π e permitir a análise e verificação formal destas linguagens. Em um dos trabalhos a seguir apresentados, não é feita a comparação entre as semânticas das linguagens, isto porque a linguagem de origem não possui ainda uma semântica formal definida.

2.1 Cláusulas de Horn Guardadas "*Flat*"

Em (HIRATA, 1995) é dada uma tradução de uma linguagem lógica concorrente, denominada Cláusulas de Horm Guardadas "*Flat*" (CHGF), para o Cálculo-π Poliádico (MILNER, 1991). Essa linguagem que pode ser vista como uma linguagem de descrição de processos, onde a comunicação entre processo é implementada pela unificação de variáveis lógicas. Um programa é composto de cláusulas e pode ser visto como um conjunto regras condicionais para reescrita de objetivos e a reescrita de objetivos determina a unificação das variáveis lógicas. Essa linguagem possui uma semântica operacional baseada em sistemas de transição definida (UEDA, 1990), onde é dada a semântica das reduções dos objetivos.

Programas CHGF são traduzidos para processos do cálculo-π para fornecer uma semântica para seus comandos a partir da semântica do cálculo-π. Para garantir que os processos traduzidos obedecem as regras da semântica operacional do CHGF, o autor prova a equivalência das semânticas.

O objetivo desta tradução foi dar uma semântica, para a linguagem, com propriedades teoricamente conhecida, que pode ser útil para a especificação de programas nesta linguagem e fornecer fundamentos teóricos para tornar a execução de programas mais eficientes.

2.2 Diagramas UML de Seqüência e de Estados

Em (KORENBLAT; PRIAMI, 2003) é dada uma tradução de dois tipos de diagramas UML para o cálculo-π. UML é uma notação padrão usada para projeto de sistemas de software, que fornece métodos gráficos, semi-formais e estruturados para especificação. Isso, porém, não é suficiente para verificação e validação de sistemas. Assim os autores propõem a tradução desta linguagem para o cálculo-π.

Neste trabalho, é focada a abordagem baseada em diagramas de seqüências, onde os

objetos são considerados como processos do cálculo-π e as mensagens como a comunicação entre estes processos. Um diagrama de estados de um objeto é usado para a escolha de uma seqüência praticável de mensagens que ocorrem no diagrama de seqüência.

O objetivo final desta tradução foi fornecer um ambiente de projeto no qual os usuários interagem somente com o UML para realizar análises formais de suas aplicações.

A linguagem UML não possui semântica formal definida. Assim, os autores obtiveram como resultado desta tradução a definição de uma semântica formal para diagramas de seqüência UML, baseada na semântica operacional do cálculo-π, que depende de uma interpretação própria da descrição informal da semântica da linguagem traduzida.

2.3 Cálculo-ς

Em (HÜTTEL; KLEIST, 1996) é dada a tradução do cálculo-ς para o cálculo-π poliádico assíncrono. Esse cálculo fornece fundamentos teóricos simples para linguagens de programação orientadas a objetos.

A noção central, aqui são os objetos e como eles se relacionam. Os objetos possuem métodos que podem ser invocados e redefinidos. Assim, os termos deste cálculo são os objetos, variáveis, invocação de métodos e redefinição de métodos. Os termos dos objetos do cálculo-ς são traduzidos para processos do cálculo-π. Além da tradução, o autor apresenta a prova de que a sua tradução preserva a semântica dos objetos.

As razões que originaram esse trabalho foram diversas. Com essa tradução é possível mostrar como o cálculo-π é adequado para dar fundamentos à programação orientada a objetos, analisar características fundamentais dessas linguagens e fornecer uma idéia de como implementar uma linguagem orientada a objetos em sistemas distribuídos. Além disso, há o benefício do ponto de vista de verificação. Garantindo que a tradução preserva as propriedades comportamentais dos objetos é possível verificá-las com o cálculo-π. A semântica operacional do cálculo-ς é definida em termos de passos de redução descritos por regras.

2.4 Linguagem Orientada a Objetos

Em (SANGIORGI; WALKER, 2001) foi definida uma tradução de uma linguagem orientada a objetos simples para o cálculo-π. Essa tradução é dada com fins didáticos, para dar uma semântica formal para uma linguagem orientada a objetos genérica. A linguagem considerada possui como categorias sintáticas comandos, expressões e declarações (classes, métodos e variáveis). Cada categoria é traduzida para agentes do cálculo-π. Esses agentes é que definem a semântica de cada elemento traduzido, assim a semântica dos comandos, expressões e declarações pode ser entendida a partir da semântica do cálculo-π.

Entre as linguagens traduzidas, a que pode ser comparada com GHBO pode ser o cálculo-ς, onde são modelados objetos que possuem métodos que podem ser invocados. A tradução desta linguagem é semelhante à proposta para as GHBO. Na tradução do cálculo-ς, a estrutura dos objetos e a invocação de métodos são descritos por agentes, como foi feito na tradução das GHBO. Cada objeto possui uma lista de métodos que possui e são executados quando o agente de invocação sincroniza com o agente do objeto. O agente de invocação tem uma referência para o nome do objeto e do método a ser invocado (destino e tipo da mensagem do MBO-π, respectivamente).

3 SISTEMAS BASEADOS EM OBJETOS - SBO

Um sistema baseado em objetos é organizado como uma coleção de objetos autônomos que se relacionam através de algum mecanismo de comunicação. Esses objetos possuem um conjunto de "operações" e um "estado". Um objeto pode ser contrastado com funções, que não tem memória. O valor de uma função é determinada pelos seus argumentos, sendo o mesmo a cada invocação. Já o resultado de uma operação de um objeto pode depender de seu estado além dos seus argumentos (WEGNER, 1987).

Existe alguma discordância sobre quais são exatamente as características exigidas pela abordagem baseada em objetos, mas geralmente elas incluem quatro aspectos: identidade, classificação, polimorfismo e herança (RUMBAUGH et al., 1994) .

3.1 Características dos Objetos

Identidade significa que os dados são subdivididos em entidades discretas e distintas, denominadas objetos. Os objetos podem ser concretos, como um arquivo em um sistema de arquivos, ou conceituais, como uma norma de escalonamento em um sistema operacional de multiprocessamento ou uma operação em um programa. Cada objeto tem sua própria identidade, que lhe é inerente. Em outras palavras, dois objetos são distintos mesmo que todos os valores de seus atributos (como nome e tamanho) sejam idênticos.

Classificação significa que os objetos com a mesma estrutura de dados (atributos) e o mesmo comportamento (operações) são agrupados em uma classe. Uma classe é uma abstração que descreve propriedades importantes para uma aplicação e ignora o restante.

Uma classe é um "modelo" do qual os objetos podem ser criados. Cada classe descreve um conjunto possivelmente infinito de objetos individuais. Cada objeto é dito ser uma instância de sua classe. Cada instância da classe tem seu próprio valor para cada atributo, mas compartilha os nomes de atributos e operações com as demais instâncias da mesma classe. As classes possuem uma ou mais interfaces que especificam as operações acessíveis aos "clientes".

Polimorfismo significa que a mesma operação pode atuar de modos diversos em classes diferentes. Uma operação é uma ação ou transformação que um objeto executa ou a que ele está sujeito. Uma implementação específica de uma operação por uma determinada classe é chamada de método. Como uma operação baseada em objetos é polimórfica, pode haver mais de um método para a sua implementação.

O usuário de uma operação não precisa saber quantos métodos existem para implementar uma determinada operação polimórfica. Novas classes podem ser adicionadas sem que se modifique o código existente, os métodos fornecidos são fornecidos para cada operação aplicável nas novas classes.

Herança é o compartilhamento de atributos e operações entre classes com base num

relacionamento hierárquico. Uma classe pode ser definida de forma abrangente e depois ser refinada em sucessivas subclasses mais definidas. Cada subclasse incorpora, ou herda, todas as propriedades da sua superclasse e acrescenta suas próprias e exclusivas características. As propriedades da superclasse não precisam ser repetidas em cada subclasse. A capacidade de identificar propriedades comuns a várias classes de uma superclasse comum e de fazê-las herdar as propriedades da superclasse pode reduzir substancialmente as repetições nos projetos e programas e é uma das principais vantagens dos sistemas baseados em objetos.

Além destas características acima citadas, pode-se ainda considerar outro aspecto, a concorrência.

A **concorrência** em um sistema baseado em objetos permite que os objetos executem suas operações concorrentemente. Além disso, um objeto pode ter concorrência interna, isto é, executar suas operações concorrentemente.

Modelos de computação concorrentes baseados em objetos devem especificar como os objetos interagem e diferentes considerações levam a diferentes modelos de comunicação entre os objetos (AGHA, 1990). Tradicionalmente, a interação entre os objetos pode ocorrer de duas maneiras (NIERSTRASZ, 1989):

- objetos ativos comunicam-se indiretamente através de objetos passivos compartilhados. Nesta abordagem, a memória compartilhada é estruturada como uma coleção de objetos passivos e os objetos do sistema são vistos como objetos ativos. As ações em um objeto passivo são realizadas de acordo com sua interface. Para esta abordagem, são necessários mecanismos de sincronização de acesso a estes objetos compartilhados. Isto pode ser feito através de semáforos, bloqueios, monitores ou transações;

- objetos ativos comunicam-se diretamente através de passagem de mensagens. Nesta abordagem, é permitido que qualquer objeto se comunique com outro. O controle do fluxo de execução é determinado implicitamente pela passagem de mensagens. Não há necessidade de sincronização explícita mas deve-se definir o tipo de passagem de mensagem a ser adotada: síncrona ou assíncrona.

3.2 Características da Abordagem

Existem, para conceituar a tecnologia baseada em objetos, várias características, embora essas não sejam exclusivas dos sistemas baseados em objetos. Mas elas são muito bem absorvidas nesses sistemas (RUMBAUGH et al., 1994). São elas:

Abstração

A abstração consiste da concentração nos aspectos essenciais, próprios de uma entidade e em ignorar os demais. No desenvolvimento de sistemas, isso significa concentrar-se no que um objeto é e faz, antes de decidir como será implementado. O uso da abstração permite que o mesmo modelo seja utilizado para análise, projeto de alto nível, estrutura de programas, estrutura de banco de dados e documentação.

Encapsulamento

Também conhecido como "ocultamento" de informações, consiste na separação dos aspectos externos de um objeto, acessíveis por outros objetos, dos detalhes internos da implementação. O encapsulamento impede que um programa se torne tão interdependente que uma pequena modificação possa causar grandes efeitos de propagação. A implementação de um objeto pode ser alterada sem que isso afete as aplicações que o utilizam.

Combinação de dados e o comportamento

Ao se chamar uma operação, não é necessário considerar quantas implementações de uma determinada operação existem. O polimorfismo transfere a responsabilidade da decisão de qual implementação deve ser utilizada da rotina de chamada para a hierarquia de classes. Em um sistema baseado em objetos, a hierarquia da estrutura de dados é idêntica à hierarquia de herança das operações.

Compartilhamento

A herança da estrutura de dados e do seu comportamento permitem que a estrutura comum seja compartilhada por diversas subclasses semelhantes, sem redundâncias. O desenvolvimento baseado em objetos não somente permite que as informações sejam compartilhadas como também oferece a possibilidade da reutilização de modelos e códigos em projetos futuros.

Ênfase na estrutura de objetos

A tecnologia baseada em objetos preocupa-se em especificar o que é um objeto, e não como ele é usado. O uso de um objeto é dependente dos detalhes da aplicação e normalmente mudam durante o desenvolvimento. À medida em que os requisitos evoluem, as características do objeto permanecem mais estáveis do que o modo como é usado, assim os softwares construídos com base na estrutura de objetos são mais estáveis a longo prazo.

Sinergia

Como já visto anteriormente, as principais características dos sistemas baseadas em objetos são: identidade, classificação, polimorfismo e herança. Estes conceitos podem ser utilizados isoladamente, porém, juntos, complementam-se de forma sinérgica. A grande ênfase nas propriedades essenciais de um objeto obriga o desenvolvedor a dar mais atenção a definição do que é e o que faz um objeto, resultando um sistema mais genérico (e limpo) do que se a ênfase se resumisse à utilização dos dados e das operações.

3.3 Descrição do Sistema Baseado em Objetos

Muitas vezes os termos baseados em objetos e orientados a objetos são utilizados com o mesmo sentido, apesar de não terem o mesmo conceito. Um sistema é baseado em objetos se ele suporta o conceito de objetos. A diferença entre sistemas baseados e orientados a objetos é que os últimos estendem as características dos primeiros pela adição dos conceitos de classes, herança e polimorfismo.

Neste trabalho são abordados os sistemas baseados em objetos e por isso não são tratadas as características como classes, herança e polimorfismo.

Um sistema baseado em objetos, aqui será considerado como sendo composto por objetos autônomos que se relacionam uns com os outros através da passagem de mensagens. O comportamento de um objeto é descrito através de suas reações ao receber mensagens. As reações são invocações de execução de operações de um objeto e podem resultar no envio de novas mensagens a outros objetos e/ou na instanciação de novos objetos. Essas reações são descritas, aqui, por "regras" que tratam cada uma das mensagens. Uma operação pode ser não-determinística, isto é, ter diferentes reações para sua execução. Desta forma podem ser necessárias várias "regras" diferentes para tratar uma operação.

Na abordagem deste trabalho, não será considerado o estado interno dos objetos, isto é, os objetos não possuem atributos. Uma generalização deste modelo será discutida nas conclusões.

Um objeto pode apresentar concorrência interna, ou seja, processar várias mensagens em paralelo e conhecer apenas os objetos que são parâmetros das mensagens que recebe.

Um sistema pode possuir diferentes tipos de objetos e diferentes tipos de mensagens (cada operação definida para um tipo de objeto resulta em um tipo de mensagem). Um determinado tipo de objeto pode receber apenas determinados tipos de mensagens.

Um tipo, aqui, pode ser visto como uma espécie de "classe". O estado do sistema é dado por um conjunto de objetos que se relacionam através da troca de mensagens para a invocação de alguma operação. Estes objetos e mensagens que descrevem o estado do sistema, são instâncias das "classes" que representam os tipos dos elementos existentes neste sistema.

Dentro do contexto deste trabalho, os tipos de mensagens e objetos de um SBO serão definidos por um hipergrafo (veja a definição 2, página 22), onde os objetos são representados por vértices e as mensagens por hiperarcos. Este hipergrafo tipo também define quais tipos de mensagem cada tipo de objeto trata.

Exemplo 1 (Sistema Baseado em Objetos) *Neste exemplo será definido um sistema baseado em objetos com três tipos de objetos:* `circulo`, `estrela` *e* `quadrado`. *Cada objeto possui operações que podem executar:* `operacao1` *(*`circulo`*),* `operacao2` *(*`estrela`*),* `operacao3` *e* `operacao4` *(*`quadrado`*). Na figura 3.1 é mostrada a estrutura dos tipos de objetos do sistema e suas operações.*

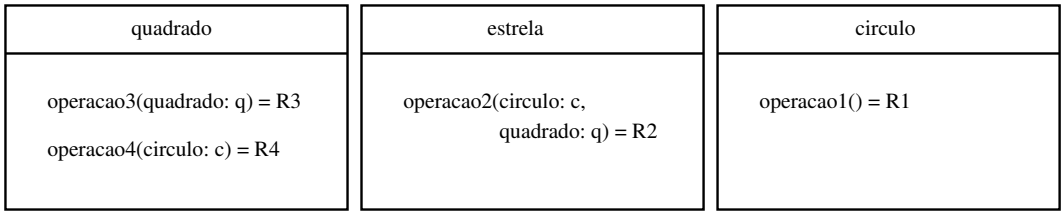


Figura 3.1: Estrutura dos objetos de um sistema baseado em objetos.

Neste esquema, são mostrados os parâmetros de cada operação, que são outros objetos. Por exemplo, a operação `operacao2` *do objeto* `estrela` *possui dois parâmetros:* `c` *e* `q`*, com os tipos* `circulo` *e* `quadrado`*, respectivamente.* $R2$ *representa o procedimento executado pela operação* `operacao2`*.*

O hipergrafo tipo que descreve os tipos deste sistema pode ser visto na figura 3.2.

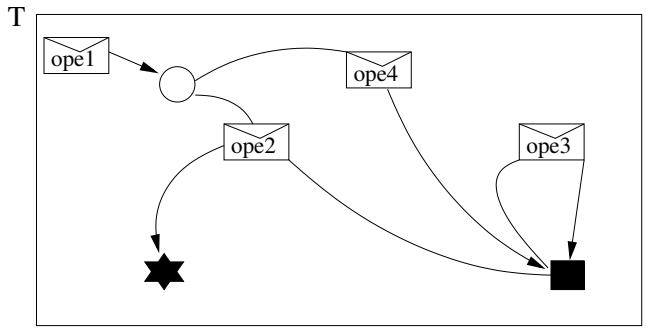


Figura 3.2: Tipo do SBO.

Como já foi dito anteriormente, o estado de um sistema baseado em objetos é dado por um conjunto de objetos (instâncias de alguma "classe") que se relacionam através da troca de mensagens. Na figura 3.3 há um exemplo de um possível estado para o sistema definido neste exemplo.

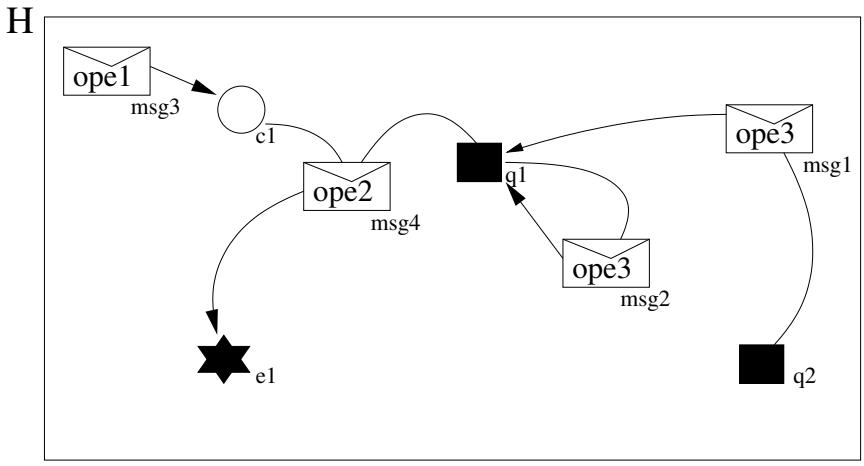


Figura 3.3: Representação de um estado de um SBO.

Neste estado, há quatro objetos e quatro mensagens. Há dois objetos do tipo `quadrado` *(*`q1` *e* `q2`*), um objeto do tipo* `circulo` *(*`c1`*), um objeto do tipo* `estrela` *(*`e1`*), duas mensagens do tipo* `operacao3` *(*`msg1` *e* `msg2`*), uma mensagem do tipo* `operacao1` *(*`msg3`*) e uma mensagem do tipo* `operacao2` *(*`msg4`*). Os parâmetros dessas mensagens (origem dos hiperarcos) são objetos que já existem no sistema.*

As regras que descrevem a reação de cada objeto ao receber cada tipo de mensagem são:

- *R1: descreve a operação* `operacao1` *do objeto* `circulo`*. Essa operação não possui parâmetros e não realiza nenhuma ação.*
- *R2: descreve a operação* `operacao2` *do objeto* `estrela`*. Essa operação possui dois parâmetros: um do tipo* `circulo` *e outro do tipo* `quadrado`*. Suas ações são a criação de uma instância do objeto* `quadrado` *e o envio de uma mensagem para o novo objeto. Essa nova mensagem invoca a* `operacao4` *deste objeto e tem como parâmetro o objeto do tipo* `circulo` *recebido como parâmetro da* `operacao2`*.*

- *R3: descreve a operação* `operacao3` *do objeto* `quadrado`*. Essa operação possui um objeto do tipo* `quadrado` *como parâmetro. Sua ação é a criação de uma instância do objeto* `quadrado`*.*
- *R4: descreve a operação* `operacao4` *do objeto* `quadrado`*. Essa operação possui um objeto do tipo* `circulo` *como parâmetro. Suas ações são a criação de uma instância do objeto* `circulo` *e o envio de uma mensagem para o novo objeto. Essa nova mensagem invoca a* `operacao1` *deste objeto.*

4 GRAMÁTICAS DE HIPERGRAFOS BASEADAS EM OBJETOS - GHBO

As gramáticas de grafos fornecem uma forma bastante natural de expressar situações complexas, onde o estados do sistema são descritos por grafos e os aspectos dinâmicos podem ser capturados pelas regras da gramática. Um tipo especial de grafos são os hipergrafos, onde as arestas, denominadas hiperarcos, podem ter zero ou mais vértices de origem e destino.

Propõem-se, aqui, uma forma de modelar um sistema baseado em objetos, através de gramáticas de hipergrafos tipados seguindo o modelo proposto em (DOTTI; RIBEIRO, 2000). Neste modelo, os objetos, mensagens e atributos são modelados como vértices. Já no modelo proposto, os objetos são representados por vértices, as mensagens por hiperarcos e os atributos não são considerados (como foi dito no capítulo anterior). Cada hiperarco possui um destino (objeto que recebe a mensagem) e várias ou nenhuma origem (objetos que são parâmetros da mensagem).

As reações dos objetos ao receber uma mensagem serão modeladas pelas regras da gramática. Cada regra descreve o tratamento de apenas uma mensagem, que pode resultar na criação de novos vértices (objetos) e/ou novas mensagens (hiperarcos). O hiperarco da mensagem tratada é deletado com a aplicação da regra. Pode-se ter mais de uma regra tratando a mesma mensagem, onde a escolha de qual será aplicada é não-determinística, modelando assim operações não-determinísticas. Uma operação deste tipo é descrita por um conjunto de regras que tratam o mesmo tipo de mensagem.

A concorrência entre os objetos e a concorrência interna, são modeladas aqui pela possibilidade da aplicação das regras da gramática de forma paralela. Isto é, concorrência entre objetos: várias regras podem ser aplicadas ao mesmo tempo; e concorrência interna: pode-se aplicar simultaneamente várias regras cujo destino da mensagem (hiperarco) é o mesmo objeto (vértice).

Os diferentes tipos de objetos e mensagens são descritos pelo hipergrafo tipo da gramática. Neste hipergrafo também são definidas as mensagens que cada tipo de objeto pode receber.

4.1 Sintaxe

Um hipergrafo, aqui, consiste de dois conjuntos, o conjunto de vértices V e o conjunto de hiperarcos E, e duas funções, a de origem sc e a de destino tg. Cada hiperarco possui uma lista de zero ou mais vértices de origem e um vértice de destino.

Notação: Dada uma função parcial $f : A \to B$:

- o domínio e contradomínio da função f são denotados, respectivamente, por $dom(f)$ e $rng(f)$;
- as funções $f^{\blacktriangledown} : dom(f) \to A$ e $f! : dom(f) \to B$ denotam, respectivamente, os domínios inclusão e restrição;
- a função $f^* : A^* \to B^*$ é a extensão da função f para listas.

Definição 2 (Hipergrafo) *Um hipergrafo é uma quádrupla $H = (V_H, E_H, sc^H, tg^H)$, onde V_H é um conjunto de vértices, E_H é um conjunto de hiperarcos, $sc^H : E \to V^*$ é uma função total que atribui a cada hiperarco uma lista de vértices de origem e $tg^H : E \to V$ é uma função total que atribui a cada hiperarco um vértice de destino. $x \in H$ denota um item $x \in V_H \cup E_H$. Um hipergrafo H é dito finito se V_H e E_H são finitos.*

∇

Através de um morfismo de hipergrafos, pode-se relacionar os vértices e hiperarcos de dois hipergrafos. Esta compatibilidade pode ser total ou parcial. Os vértices devem ser mapeados para vértices e os hiperarcos para hiperarcos, respeitando as funções de origem e destino dos hiperarcos.

Definição 3 (Morfismo de Hipergrafos) *Um morfismo parcial de hipergrafos $g : G \to H$ é uma tupla $g = (g_V, g_E)$ consistindo de duas funções parciais $g_V : V_G \to V_H$ e $g_E : A_G \to A_H$ tal que os diagramas abaixo comutem, isto é que respeitam as funções de origem e destino. Um morfismo g é dito total, injetivo ou inclusão se g_V e g_E são totais, injetivas ou inclusões, respectivamente. Um hipergrafo G é um subgrafo do hipergrafo H, escrito $G \subseteq H$, se há uma inclusão $g : G \to H$.*

$$\begin{array}{ccc} E_G & \xleftarrow{g_E^{\blacktriangledown}} dom(g_E) & \xrightarrow{g_E!} E_H \\ {\scriptstyle sc^G}\downarrow & = & \downarrow{\scriptstyle sc^H} \\ V_G^* & \xrightarrow[g_V^*]{} & V_H^* \end{array} \qquad \begin{array}{ccc} E_G & \xleftarrow{g_E^{\blacktriangledown}} dom(g_E) & \xrightarrow{g_E!} E_H \\ {\scriptstyle tg^G}\downarrow & = & \downarrow{\scriptstyle tg^H} \\ V_G & \xrightarrow[g_V]{} & V_H \end{array}$$

*A categoria dos hipergrafos e morfismos de hipergrafos parciais é denotado por **HGrafoP**.*

∇

Todas as categorias de hipergrafos e morfismos parciais (tipados ou não), citadas neste trabalho são bem formadas e possuem *pushouts* (soma amalgamada). Isso se dá pois essas categorias podem ser vistas como instâncias das categorias de Estruturas de Grafos Generalizadas (EGG) e morfismos parciais (KORFF, 1996). Nessas categorias, colimites existem e podem ser construídos a partir dos colimites dos componentes, usando-se no final uma construção livre para "totalizar" o resultado.

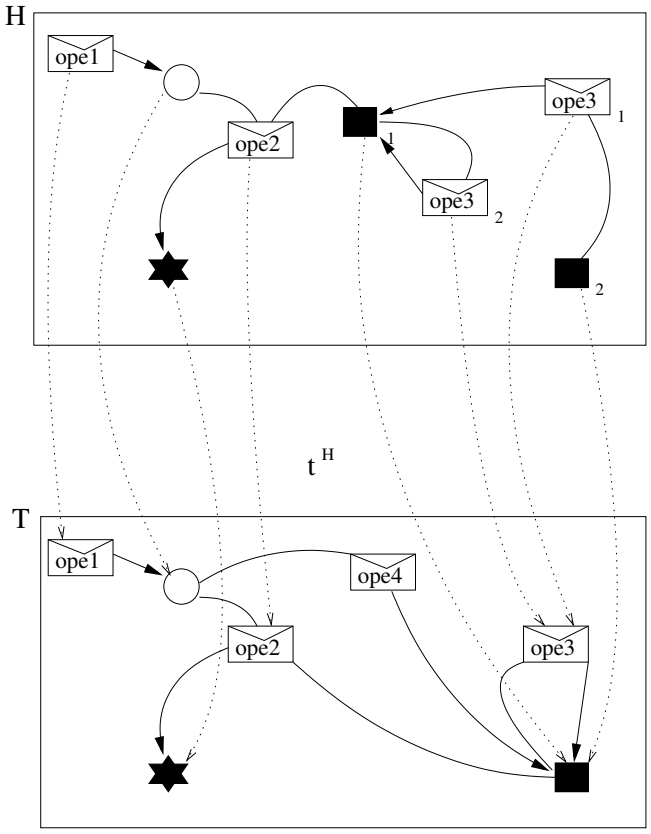


Figura 4.1: Hipergrafos H e T

Exemplo 4 (Hipergrafo, Morfismo de Hipergrafos) *Na figura 4.1 são mostrados dois hipergrafos H e T . O hipergrafo T é definido por $T = (V, E, sc, tg)$, onde $V = \{\bigcirc, \blacksquare, \bigstar\}, E = \{ope1, ope2, op3, ope4\}, sc = \{ope1 \mapsto \langle\rangle, ope2 \mapsto \langle\bigcirc, \blacksquare\rangle, ope3 \mapsto \langle\blacksquare\rangle, ope4 \mapsto \langle\bigcirc\rangle\}$ e $tg = \{ope1 \mapsto \bigcirc, ope2 \mapsto \bigstar, ope3 \mapsto \blacksquare, ope4 \mapsto \blacksquare\}$. De forma análoga é definido o hipergrafo H.*
A figura 4.1 mostra um morfismo total de hipergrafos (t^H). O morfismo t^H é definido pelas setas tracejadas e respeita as funções de origem e destino de todas as mensagens mapeadas. Por exemplo, os vértices de origem da mensagem $ope2$ de H foram mapeados para os vértices que são origem da $ope2$ de T, da mesma forma ocorre com o vértice de destino.

Para especificar um sistema pode ser necessário um grande número de vértices e hiperarcos. Para facilitar o entendimento, pode ser útil fazer distinção entre diferentes tipos de vértices e hiperarcos. Isso pode ser feito de diferentes formas, como por exemplo através de rótulos ou de hipergrafos com diferentes tipos para vértices e hiperarcos. Neste trabalho, será utilizado um hipergrafo, denominado hipergrafo tipo, para especificar o tipo de cada elemento do sistema.

O tipo de cada vértice e hiperarco é dado por um morfismo total de hipergrafos, que relaciona cada elemento do hipergrafo tipado (instância) com um elemento do hipergrafo tipo. Na representação gráfica, usaremos vértices com mesmo *layout* para representar instâncias de um determinado tipo. Instâncias diferentes de um mesmo tipo serão indexadas (ver exemplo na figura 4.1).

Um morfismo entre hipergrafos tipados deve respeitar a função de tipagem, isto é, um elemento e, cuja função de tipagem associa-o ao elemento t, só poderá ser mapeado para

elementos cujo tipo seja t.

Definição 5 (Hipergrafo Tipado) *Um hipergrafo tipado H^T é uma tupla $H^T = (H, t^H, T)$, onde H e T são hipergrafos e $t^H : H \rightarrow T$ é um morfismo total de hipergrafos. Geralmente usa-se $x \in H^T$ significando que x é um vértice ou hiperarco de H.*

Um **morfismo de hipergrafos tipados** *$g^t : H^{T1} \rightarrow G^{T2}$ entre hipergrafos tipados H^{T1} e G^{T2} é um par de morfismos de hipergrafos $g^t = (g, t)$ com $g : H \rightarrow G$ e $t : T1 \rightarrow T2$, tal que o diagrama abaixo comute em **HGrafoP**.*

$$\begin{array}{ccccc} H & \xleftarrow{g^{\blacktriangledown}} & dom(g) & \xrightarrow{g!} & G \\ {\scriptstyle t^H}\downarrow & & = & & \downarrow{\scriptstyle t^G} \\ T1 & & \xrightarrow[t]{} & & T2 \end{array}$$

*A categoria dos hipergrafos tipados e dos morfismos de hipergrafos tipados é denotada por **THGrafoP**. Uma subcategoria de **THGrafoP** é a categoria cujos objetos são todos os hipergrafos sob o tipo T e cujos morfismos são todos os morfismos em **THGrafoP** para os quais o componente tipo é a identidade. Esta categoria é denotada por **THGrafoP(T)**. O conjunto de todos os hipergrafos tipados é denominado por $HiperGTip$.*

∇

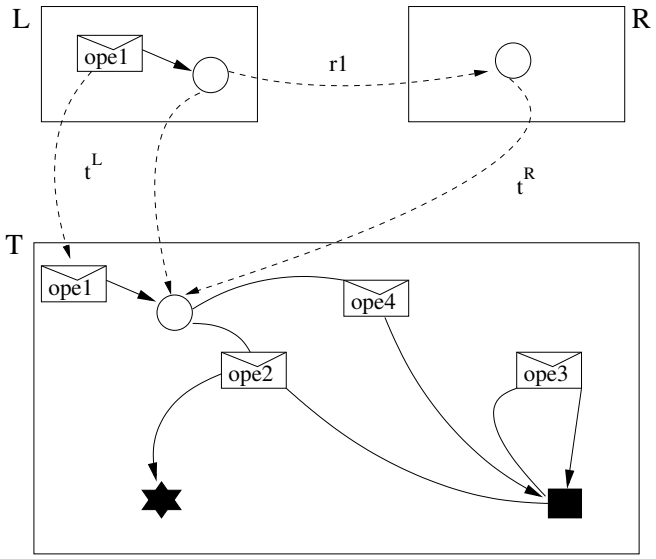


Figura 4.2: $r1$ é um morfismo de hipergrafos tipados.

Exemplo 6 (Hipergrafo Tipado, Morfismo de Hipergrafos Tipados) *A figura 4.1 mostra um exemplo de hipergrafo tipado. Sendo T um hipergrafo tipo, então H é uma instância deste hipergrafo. O relacionamento entre H e T é dado pelo morfismo $t^H : H \rightarrow T$. Este morfismo descreve o tipo de todos os vértices e arcos do hipergrafo H e garante que ele é consistente com relação ao hipergrafo tipo. Um morfismo de hipergrafos tipados deve mapear compativelmente não somente vértices e hiperarcos, mas deve preservar a informação do tipo. A figura*

4.2 mostra o morfismo de hipergrafos tipado $r1$, onde o vértice do hipergrafo L de tipo ○ é mapeado para um vértice do mesmo tipo em R. Note que este morfismo é parcial, pois o hiperarco $ope1$ do hipergrafo L, não é mapeado para nenhum elemento do hipergrafo R.

Em uma gramática de hipergrafos, as mudança de estados de um sistema são descritas por regras. Estas regras podem ser aplicadas a um hipergrafo, que representa o estado atual do sistema, e o alteram. Uma regra é um morfismo (não total e injetivo) de hipergrafos tipados.

Em uma regra há elementos que são preservados, criados e/ou deletados. O domínio ($dom(r)$) de uma regra (r) é o domínio do morfismo, ou seja, os elementos que são mapeados do lado esquerdo para o lado direito da regra. O contradomínio ($rng(r)$) são os elementos do lado direito da regra (r), os quais estão associados a um elemento do domínio da regra. Os elementos preservados são os elementos que pertencem ao lado direito e ao contradomínio da regra. Os elementos criados são os elementos que pertencem ao lado direito da regra e não pertencem ao contradomínio da regra. Os elementos deletados são os elementos que pertencem ao lado esquerdo da regra e não pertencem ao domínio dela.

Definição 7 (Regra) *Dado um hipergrafo T. Então uma regra com relação a T é um morfismo de hipergrafos tipados não total e injetivo $r : L^T \rightarrow R^T$ em **THGrafoP(T)**.*

∇

As regras de uma GHBO são regras que apresentam as seguintes características:

- deve haver apenas um hiperarco do lado esquerdo da regra;
- este hiperarco deve ser consumido pela aplicação da regra;
- é apenas permitida a deleção de hiperarcos;
- somente os vértices de destino e origem do hiperarco podem aparecer no lado esquerdo da regra.

Definição 8 (Regra GHBO) *Uma regra $r : L^T \rightarrow R^T$ com relação a um hipergrafo tipo T é uma regra GHBO se:*

(a) $|E_L| = 1$;

(b) $\forall a \in E_L.a \not\in dom(r)$;

(c) $\forall x \in V_L.x \in dom(r)$;

(d) $\forall x \in V_L.\exists a \in E_L.tg^L(a) = x \vee x \in sc^L(a)$.

A classe de todas as regras GHBO com relação a um grafo tipo T é denotada por $HRegras(T)$.

∇

Proposição 9 *Uma regra GHBO é bem definida.*
Prova. *Uma regra $GHBO$ é bem definida pois é uma regra com algumas restrições, o que não altera sua condição de morfismo não total e injetivo.*

√

Exemplo **10 (Regra GHBO)** *A figura 4.3 mostra alguns exemplos de regras GHBO.*

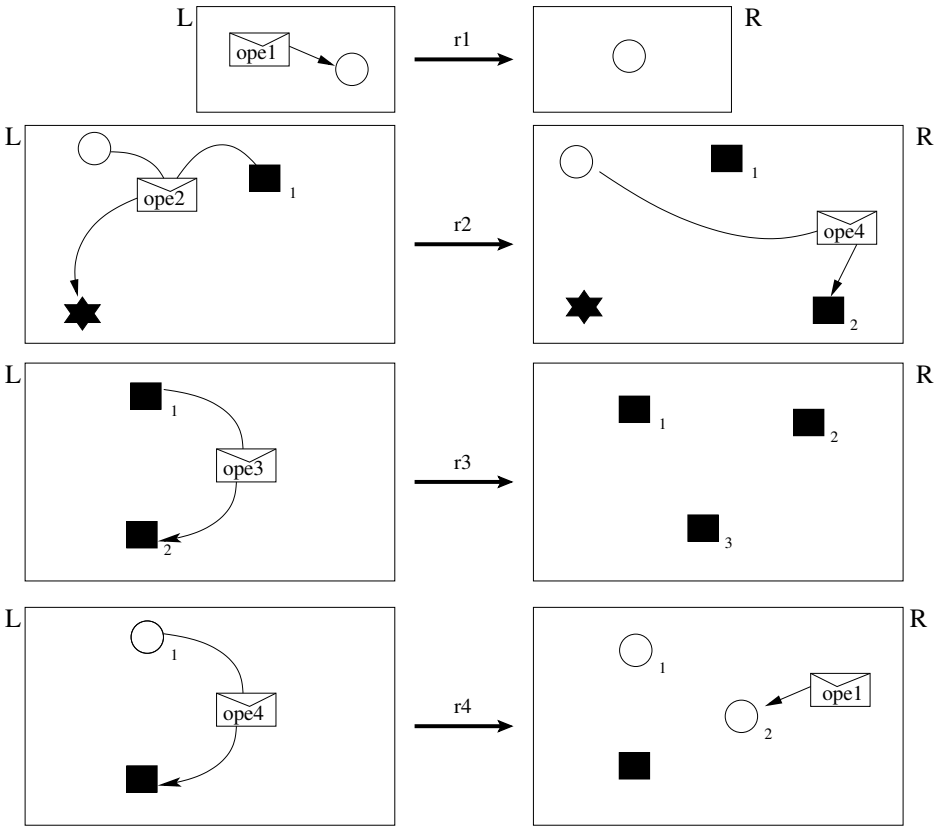


Figura 4.3: Regras GHBO.

Pode-se agora definir uma GHBO.

Definição 11 (Gramática de Hipergrafos Baseada em Objetos) *Uma GHBO tipada é uma tupla GH = (T, I, N, n), onde:*

- *T é um hipergrafo finito, denominado hipergrafo tipo, que será o tipo da gramática;*
- *I é um hipergrafo tipado em **THGrafoP(T)**, que será o hipergrafo inicial da gramática;*
- *N é um conjunto de nomes de regras;*
- *$n : N \rightarrow HRegras(T)$ é uma função total, que atribui a cada nome de regra uma regra GHBO com relação ao tipo T.*

∇

Exemplo **12 (Gramática de Hipergrafos Baseada em Objetos)** *Neste exemplo é feita descrição dos componentes de uma GHBO que especifica o SBO descrito no exemplo 1. O* HIPERGRAFO TIPO*, mostrado na figura 4.4 identifica o tipo dos objetos e mensagens (tipos de operações) apresentados na figura 3.1. O* HIPERGRAFO INICIAL H*, que especifica o estado inicial do SBO do exemplo 1, é mostrado na figura 4.1. Os* NOMES DAS REGRAS $\{r1, r2, r3, r4\}$ *são identificadores para as regras que descrevem as operações dos objetos. E a* FUNÇÃO DE NOMEAÇÃO*, que associa os nomes às regras é mostrada na figura 4.3.*

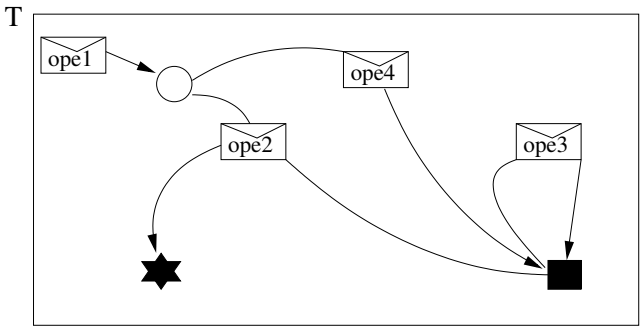


Figura 4.4: Hipergrafo Tipo.

4.2 Semântica

A semântica operacional de uma gramática de hipergrafos é definida em termos de derivações, que são as aplicações das regras da gramática aos seus estados. O resultado da aplicação de uma regra $r : L \rightarrow R$ em um hipergrafo I é obtido da seguinte forma:

1. Adicionar a I tudo que for criado pela regra r (itens que fazem parte do lado direito R da regra, mas não do lado esquerdo L);

2. Remover do grafo resultante do passo 1 tudo que deve ser removido pela regra r (itens que fazem parte do lado esquerdo L e não do lado direito R);

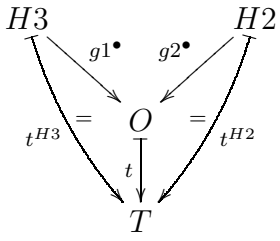
3. Remover hiperarcos pendentes. Este passo é necessário no caso de haver hiperarcos conectados a vértices removidos no passo 2. Como o resultado da aplicação de um regra deve ser um grafo, estes hiperarcos devem ser removidos também.

Formalmente, uma derivação em GHBO é dada por um *pushout* na categoria **THGrafoP(T)**. As definições formais dessas construções podem ser vistas no anexo A. A prova de que estas construções são os *pushouts* em suas categorias vem do fato de que essas categorias são instâncias das categorias de EGG e os *pushouts* nesta categoria são construídos a partir de seus componentes (KORFF, 1996), como segue.

No diagrama a seguir pode-se ver a construção do *pushout* dos morfismos $g1^T$ e $g2^T$ em **THGrafoP(T)**. Essa construção é definida por um objeto O^T e dois morfismos $g1^{\bullet T}$ e $g2^{\bullet T}$.

$$\begin{array}{ccc} H1^T & \xrightarrow{g1^T} & H2^T \\ {\scriptstyle g2^T}\downarrow & = & \downarrow{\scriptstyle g2^{\bullet T}} \\ H3^T & \xrightarrow[g1^{\bullet T}]{} & O^T \end{array}$$

O hipergrafo O^T, resultante desta construção é o hipergrafo (O, t, T), onde O é o objeto e $g1^\bullet$ e $g2^\bullet$ são os morfismos resultantes do *pushout* de $g1$ e $g2$ na categoria **HGrafoP**. t é definido como o morfismo de hipergrafos total único tal que $t \circ g1^\bullet = t^{H3}$ e $t \circ g2^\bullet = t^{H2}$, que pode ser visto no diagrama a seguir:



Os morfismos resultantes desta construção são os morfismos $g1^{\bullet T} = (g1^\bullet, id^T)$ e $g2^{\bullet T} = (g2^\bullet, id^T)$, onde id^T é o morfismo identidade de T.

O *pushout* dos morfismos $g1$ e $g2$ na categoria **HGrafoP** resulta em um hipergrafo $O = (V_O, E_O, sc, tg)$. O conjunto de vértices V_O é construído como o *pushout* dos morfismos $g1_V$ e $g2_V$ na categoria **SetP**. O conjunto de arestas E_O é construído como o *pushout* dos morfismos $g1_E$ e $g2_E$ na categoria **SetP**, excluindo-se os hiperarcos que tiveram algum dos seus vértices de origem e/ou destino excluídos, isto é, não estão no conjunto V_O.

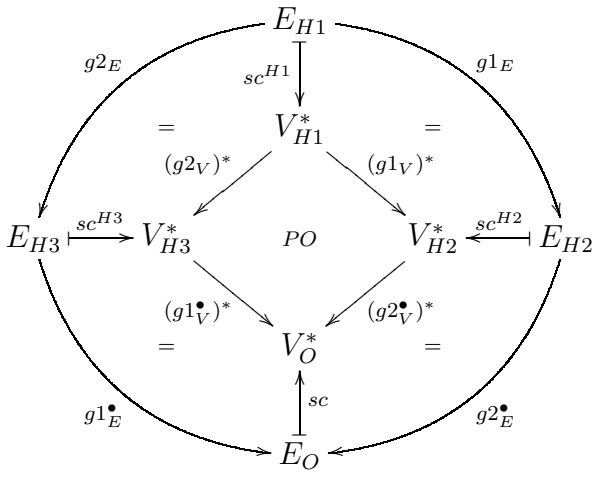
O *pushout* dos morfismos $g1_V$ e $g2_V$ na categoria **SetP** resulta em um conjunto V_O e pode ser visto no diagrama a seguir. Cada elemento do conjunto V_O é uma classe que representa um conjunto de elementos dos conjuntos V_{H2} e V_{H3}. Cada elemento do conjunto V_{H2} que não pertence ao $rng(g1_V)$ é associado a uma classe em V_O. Cada elemento do conjunto V_{H3} que não pertence ao $rng(g2_V)$ é associado a uma classe em V_O. Para todos os elementos $a \in V_{H2}$ e $b \in V_{H3}$ que possuem o mesmo elemento $x \in V_{H1}$ como domínio para as funções $g1_V$ e $g2_V$, isto é, $g1_V(x) = a$ e $g2_V(x) = b$, há uma classe em V_O à qual esses elementos são associados.

$$\begin{array}{ccc} V_{H1} & \xrightarrow{g1_V} & V_{H2} \\ {\scriptstyle g2_V}\downarrow & = & \downarrow{\scriptstyle g2^\bullet_V} \\ V_{H3} & \xrightarrow[g1^\bullet_V]{} & V_O \end{array} \qquad \begin{array}{ccc} E_{H1} & \xrightarrow{g1_E} & E_{H2} \\ {\scriptstyle g2_E}\downarrow & = & \downarrow{\scriptstyle g2^\bullet_E} \\ E_{H3} & \xrightarrow[g1^\bullet_E]{} & E \end{array}$$

O morfismo $g1^\bullet_V$ mapeia cada elemento de V_{H3} para a sua classe em V_O. Da mesma forma, o morfismo $g2^\bullet_V$ mapeia os elementos de V_{H2} para suas classes em V_O.

O *pushout* dos morfismos $g1_E$ e $g2_E$, mostrado no diagrama acima, é construído de forma análoga, resultando em conjunto E. Após esta construção, são retirados de E todos os hiperarcos cujas origens e/ou destino não estão em V_O, gerando então o conjunto E_O.

No diagrama abaixo é mostrada a construção da função de origem sc que é obtida como a função total única tal que $(g2^\bullet_V)^* \circ sc^{H2} = sc \circ g2^\bullet_E$ e $(g1^\bullet_V)^* \circ sc^{H3} = sc \circ g1^\bullet_E$. A função de destino tg é obtida de forma análoga.



Os morfismos resultantes desta construção são os morfismos $g1^\bullet = (g1^\bullet_V, g1^\bullet_E)$ e $g2^\bullet = (g2^\bullet_V, g2^\bullet_E)$.

Dada uma regra $r : L^T \rightarrow R^T$ e um hipergrafo G^T, a regra r pode ser aplicada a G^T se houver uma ocorrência de L^T em G^T, isto é, se L^T for um subgrafo de G. Esta ocorrência é descrita pela existência de um morfismo total de hipergrafos tipados m de L^T para G^T.

Definição 13 ***(Ocorrência)*** *Dada uma regra $r : L^T \rightarrow R^T$ e um hipergrafo tipado G^T. Uma ocorrência é um morfismo total de hipergrafos tipados $m : L^T \rightarrow G^T$.*

∇

Se uma regra r é aplicável a G^T por uma ocorrência m e resulta em um hipergrafo H^T, então obtém-se uma derivação $G^T \overset{r,m}{\Longrightarrow} H^T$. A figura 4.5 mostra uma derivação construída como um *pushout*. Um *pushout* é único a menos de isomorfismo, assim G^T, construído como um *pushout* de m e r, representa um conjunto de objetos isomórficos.

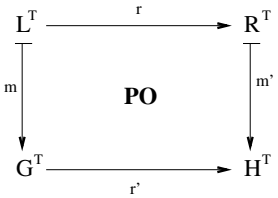
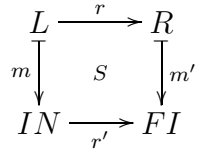


Figura 4.5: Derivação construída como *pushout*

Definição 14 (Derivação) *Dados um hipergrafo IN, uma regra $r : L \rightarrow R$ e uma ocorrência $m : L \rightarrow IN$ com relação ao tipo T. Uma derivação s com nome nr de um grafo IN aplicando r baseado em m, é a tupla $s = (nr, S)$, onde S é o* pushout *de m e r em* ***THGrafoP(T)*** *e $n(nr) = r$. Neste caso, escreve-se $IN \overset{nr,m}{\Longrightarrow} FI$.*

Esta derivação é dada pelo diagrama abaixo, onde IN, FI, r' e m', são chamados de hipergrafo inicial, hipergrafo final, co-regra e co-ocorrência, respectivamente:



A classe de todas as derivações usando regras de um gramática de Hipergrafos GH é denotada por $Passo_{GH}$.

∇

Os aspectos do sistema que nos interessam é que determinam qual o modelo semântico é mais adequado. No caso deste trabalho é importante representar a concorrência e o não-determinismo no tratamento das mensagens do sistema. Pode-se definir a semântica das GHBO de diferentes modos, representando a concorrência e o não-determinismo implícita ou explicitamente.

Como há a necessidade de comparar a semântica das GHBO com a do cálculo-π, que possui uma semântica *interleaving* como padrão (sistema de transição), a semântica definida para GHBO, neste trabalho, será a semântica *interleaving*. Nesse tipo de semântica há representação de não-determinismo e da concorrência de forma implícita. O não-determinismo é descrito por diferentes seqüências de derivações com subseqüências comuns e a concorrência é descrita por diferentes passos de derivação observados em duas ordens diferentes.

A semântica *interleaving* é definida por um conjunto de computações da gramática. Uma derivação descreve um único passo de computação usando uma gramática de hipergrafos. Uma computação pode ser descrita por uma derivação seqüencial, que é uma lista de passos, onde o hipergrafo de saída de um passo é o hipergrafo de entrada do seguinte.

Notação: Se C é um conjunto (ou classe), então o conjunto (ou classe) de todas as seqüências sobre C é denotado por C^{∞}. Para seqüências finitas a notação é C^{*} e uma seqüência vazia é denotada por λ. Se $\sigma \in C^{\infty}$, então $|\sigma| \in \mathbb{N} \cup \{\omega\}$ é o comprimento de σ. O i-ésimo elemento da seqüência σ é denotado por σ_i.

Definição 15 (Derivação Seqüencial) *A classe de* derivações seqüenciais *sobre $GH = (T, I, N, n)$ é definida por:*

$SDer_{GH} = \{\sigma \in Passo_{GH}^{\infty} | \sigma = \lambda \vee IN_{\sigma} = IN_{\sigma_1} = I \wedge FI_{\sigma_i} = IN_{\sigma_{i+1}} \text{ para todo } 1 \leq i < |\sigma|\}$

Diz-se que $s \in SDer_{GH}$ se existe $\sigma \in SDer_{GH}$ e s é um passo da derivação seqüencial σ.

O conjunto de todos os estados IN e FI de todas as derivações seqüencias em $SDer_{GH}$ é denominado $State_{GH}$, ou seja, $State_{GH} = \{G | G \stackrel{r,m}{\Longrightarrow} H \in SDer_{GV} \vee H \stackrel{r,m}{\Longrightarrow} G \in SDer_{GV}\}$

∇

Como o objeto resultante de um *pushout* representa um conjunto de objetos isomórficos, cada passo de uma derivação seqüencial gera várias possibilidades de derivação para o passo seguinte (todos isomórficos). Assim, para cada derivação seqüencial em $SDer_{GH}$ há um conjunto de derivações seqüenciais, isomórficas a ela, que está em $SDer_{GH}$.

Existem outros tipos de semântica para gramáticas de grafos baseadas em classes de equivalências de computações (RIBEIRO, 1996), (CORRADINI; MONTANARI; ROSSI, 1997). Neste trabalho porém, não serão consideradas classes de equivalências de derivações. Assim sendo, no teorema 34 provar-se-á que a derivação seqüencial da semântica de uma GHBO e a derivação seqüencial traduzida de uma seqüência de transições da tradução desta GHBO para o cálculo-π são isomórficas (e não exatamente a mesma).

A semântica operacional de gramáticas de hipergrafos pode ser descrita através de Sistemas de Transição Rotulados (STR). Um STR é definido pelos seguintes componentes: um conjunto de estados do sistema, um conjunto de rótulos para as transições (mudanças de estado), o estado inicial do sistema e uma relação de transição do sistema, que relaciona dois estados e um rótulo.

Definição 16 (Sistema de Transição Rotulado) *Um sistema de transição rotulado é uma tupla $T = (S, R, I, \rightarrow)$ onde S é o conjunto de estados do sistema, R é um conjunto de rótulos, $I \in S$ é o estado inicial do sistema e $\rightarrow \subseteq S \times R \times S$ é a relação de transição do sistema. Uma tripla $(p, \alpha, q) \in \rightarrow$ é chamada de transição e é usualmente escrita $p \xrightarrow{\alpha} q$.*

∇

Para definir o STR de uma GHBO GH, deve-se identificar cada um dos componentes do STR:

- O conjunto de estados S é o conjunto $State_{GH}$, isto é, todos os hipergrafos que são ou o hipergrafo inicial ou o final dos passos de derivação da classe de derivação seqüencial sobre GH;
- O conjunto de rótulos R é composto por um rótulo, do tipo $\overline{nr}.\overline{t}(id)$, para cada hiperarco id que pertence a cada hipergrafo em S, onde t é o tipo deste hiperarco e nr é o nome de uma regra GHBO que contém esse tipo de hiperarco no seu lado esquerdo;
- O estado inicial do STR é o hipergrafo inicial de GH;
- A relação de transição é dada pela seguinte regra: para cada passo de derivação $H \overset{nr,m}{\Longrightarrow} H'$ na classe de derivação sobre GH, há uma transição $H \xrightarrow{\overline{nr}.\overline{t}(id)} H'$, onde id é o contradomínio da função de ocorrência m para o hiperarco do lado esquerdo da regra nr e t é o tipo de id.

Definição 17 (Semântica GHBO) *Dada uma gramática de hipergrafos baseada em objetos tipada $GH = (T, I_{GH}, N, n)$, sua semântica $SemGHBO(GH)$ é dada pelo sistema de transição rotulado $ST = (S, R, I, \rightarrow)$, onde:*

- $S = State_{GH}$;
- $R = \{\overline{nr}.\overline{t}(id) | id \in E_G \wedge G \in S \wedge n(nr) = L^T \rightarrow R^T \wedge t = t^L(e) \wedge e \in E_L\}$, *onde* $t = t^G(id)$
- $I = I_{GH}$;
- $\rightarrow$ *é dada pela seguinte regra:*

$$\frac{H \overset{nr,m}{\Longrightarrow} H' \ \in \ SDer_{HG}}{H \xrightarrow{\overline{nr}.\overline{t}(id)} H'} \qquad t = t^L(id) \ \wedge \ id = m(msg) \ \wedge \ msg \in A_L \wedge n(nr) = L^T \rightarrow R^T$$

∇

Neste capítulo foram apresentadas as definições necessárias para definir gramáticas de hipergrafos baseadas em objetos e sua semântica. A seguir é dado uma breve descrição do cálculo-π e sua semântica, bem como a definição de um modelo baseado em objetos descrito em cálculo-π.

5 CÁLCULO π

O cálculo-π (MILNER; PARROW, 1992) é um modelo matemático de processos cujas interconexões podem mudar quando há interação. O passo computacional básico é a transferência de um canal de comunicação entre dois processos. O processo receptor pode usar o novo canal para futuras interações com outros processos. Os canais, no cálculo-π, são apenas nomes, entidades atômicas sem estrutura interna. O que torna o cálculo-π mais expressivo é a possibilidade da migração de escopo local. Essa migração é feita através do envio de um canal restrito para outro processo, que passará a ter também a possibilidade de usá-lo.

5.1 Sintaxe

A sintaxe básica do cálculo-π pode ser vista na tabela 5.1. Aqui, assume-se um conjunto infinito de *nomes* $\mathcal{N}$, variando sobre $a, b, \ldots$ (que representam os canais, variáveis e valores), e um conjunto de *agentes* variando sobre $P, Q, \ldots$ Um termo do cálculo-π é dado por um agente. Cada agente pode ser da seguinte forma:

- O agente nulo $\mathbf{0}$, que não pode realizar nenhuma ação;
- Um prefixo de saída $\overline{a}x.P$. Esse agente envia o nome x através do canal a e continua agindo como P. Assim, o $\overline{a}$ pode ser visto como um canal de saída e o x como o valor enviado por tal canal;
- Um prefixo de entrada $a(x).P$, que representa o recebimento de um valor pelo canal a que é armazenado em x. Depois de recebido o valor o agente continua agindo como P com o valor recebido substituindo o x;
- Um prefixo silencioso $\tau.P$, que significa que o agente pode passar a agir como P sem interagir com o ambiente;
- Uma escolha $P + Q$, que representa um agente que pode agir ou como P ou como Q;
- Uma composição paralela $P \mid Q$, que representa o comportamento combinado de P e Q executando em paralelo. Os componentes P e Q podem agir independentemente e ainda podem realizar uma comunicação entre si, através de um canal comum.
- Uma restrição $(\nu x)P$. Este agente age como P, mas o nome x é local, isto é, não pode ser usado como canal de comunicação entre P e o ambiente, apenas entre componentes de P;

- Uma condição $[x = y]P$. Se $x = y$, então o agente comporta-se como P.
- Um identificador $A(y_1, \ldots, y_n)$ onde n é a aridade de A. Todos os identificadores têm uma definição $A(x_1, \ldots, x_n) \stackrel{def}{=} P$ onde os x_i são todos distintos. Este agente age como P, com y_i substituindo x_i e $0 < i \leq n$;

Tabela 5.1: Sintaxe do Cálculo-π

Prefixos	$\alpha ::= \overline{a}x$	Saída
	$a(x)$	Entrada
	τ	Silencioso
Agentes	$P ::= \mathbf{0}$	Nulo
	$\alpha.P$	Prefixo
	$P + P$	Escolha
	$P \mid P$	Paralelo
	$(\nu x)P$	Restrição
	$[x = y]P$	Condição
	$A(y_1, \ldots, y_n)$	Identificador
Definições	$A(x_1, \ldots, x_n) \stackrel{def}{=} P$	(onde $i \neq j \Longrightarrow x_i \neq x_j$)

O Prefixo de Entrada e a Restrição são os operadores que ligam nomes. Por exemplo, em $a(x).P$ (a é o sujeito e x é o objeto) e em $(\nu x)P$ as ocorrências x são ditas ligadas em P, isto é, o escopo de x é P. Uma ocorrência de x é dita livre se ela está fora do escopo de uma ocorrência ligada de x. O conjunto de nomes ligados de P é denotado por $bn(P)$ e de nomes livres por $fn(P)$, e de maneira similar $bn(\alpha)$ e $fn(\alpha)$, para o Prefixo α. Pode-se escrever $fn(P, Q)$ para representar $fn(P) \cup fn(Q)$ e apenas α para $fn(\alpha) \cup bn(\alpha)$. Na Definição $A(x_1, \cdots, x_n) \stackrel{def}{=} P$ assume-se que $fn(P) \subseteq \{x_1, \cdots, x_n\}$.

Dois processos P e Q são α-conversíveis se Q pode ser obtido de P por um número finito de substituições dos nomes ligados. Uma substituição de nomes ligados em um processo P é a troca de um subtermo $x(z).Q$ de P por $x(w).Q\{w/z\}$ ou a troca de um subtermo $(\nu z)Q$ de P por $(\nu w)Q\{w/z\}$, onde nos dois casos w não ocorre em Q.

Uma substituição é uma função de nomes para nomes. Escreve-se $\{x/y\}$ para a substituição que mapeia y para x e é a identidade para todos os outros nomes. Usa-se σ para variar sobre as substituições. O agente $P\sigma$ é P onde todos os nomes livres x são substituídos por $\sigma(x)$, com α-conversões sempre que necessário. Com a α-conversão, os nomes ligados são renomeados de tal forma que sempre que x for substituído por $\sigma(x)$, as ocorrências de $\sigma(x)$ são sempre livres. Por exemplo,

$$a(x).(\nu b)\overline{x}b.\overline{c}y.0\{xb/yc\} \quad \text{é} \quad a(z).(\nu d)\overline{z}d.\overline{b}x.0$$

Notação:

- A escolha de diversos agentes $P_1 + \cdots + P_n$ é escrito como $\sum_{i=1}^{n} P_i$. Quando $n = 0$, a escolha é equivalente à 0.

- Uma seqüência de restrições $(\nu x_1) \cdots (\nu x_n)P$ é escrita como $(\nu x_1, \cdots, x_n)P$.
- A composição paralela de vários agentes $P_1 | \cdots | P_n$ é escrito como $\prod_{i=1}^{n} P_i$
- Uma seqüência de ações $a(x_1). \cdots .a(x_n)$ e $\overline{a}x_1. \cdots .\overline{a}x_n$ são escritas como $\bigodot_{i=1}^{n} a(x_i)$ e $\bigodot_{i=1}^{n} \overline{a}x_i$, respectivamente.
- Em um Prefixo pode-se omitir o objeto se ele não tiver importância, por exemplo, $a.P$ representa o agente $a(x).P$ onde x nunca é usado, de forma similar ocorre com o prefixo de saída.
- Pode-se omitir o operador nulo 0 quando isso não causar confusão. Por exemplo, pode-se escrever α para o agente $\alpha.0$.

Um sistema pode ser especificado em cálculo-π. Uma especificação deve descrever o estado inicial do sistema, o que é feito através de um termo do cálculo-π. Além disso, todos os identificadores de agentes deste sistema devem ser definidos.

Definição 18 (Especificação em cálculo-π) *Uma especificação ESP em cálculo-π é dada por uma tupla (T, D), onde T é um termo do cálculo-π que descreve o estado inicial do sistema, denominado termo inicial da especificação, e D é o conjunto de definições dos identificadores dos agentes do sistema especificado. Todas as definições dos agentes de T devem estar em D.*

∇

5.2 Congruência Estrutural

Dois agentes podem ser sintaticamente diferentes mas representar o mesmo comportamento. Assim, é introduzida a congruência estrutural para identificar estes agentes que intuitivamente representam o mesmo comportamento. A congruência estrutural não identifica agentes a partir de seus comportamentos, mas sim a partir da sua estrutura, onde é obvio que são o mesmo. Um exemplo de agentes que intuitivamente representam o mesmo comportamento são os agentes $a(x).\overline{b}x$ e $a(y).\overline{b}y$, apesar de serem diferentes sintaticamente. A congruência estrutural não manipula renomeação de variáveis ligadas. Ao invés, considera-se que α-conversões são realizadas de forma silenciosa sempre que necessárias.

Existem diferentes versões de congruência estrutural na literatura, pois não existe uma definição canônica. A versão adotada aqui, é a dada em (PARROW, 2001) e pode ser vista na tabela 5.2.

A congruência estrutural é mais forte, isto é, identifica muito menos agentes, que qualquer das equivalências comportamentais. A congruência é usada na definição da semântica operacional, que é por sua vez, usada para definir equivalências comportamentais.

5.3 Semântica Operacional

A semântica operacional de álgebras de processos é usualmente dada através de STR. O cálculo-π segue este padrão e a maioria das regras de transição são similares às das outras álgebras.

A semântica de uma especificação $ESP = (T, D)$, denotada por $SemPi(ESP)$ é definida como segue:

Tabela 5.2: Congruência Estrutural

A congruência estrutural $\equiv$ é definida como a menor congruência que satisfaz as seguintes leis:

1. Se P e Q são variantes de α-conversão então $P \equiv Q$.

2. Leis do monóide Abeliano para Composição Paralela (as mesmas leis para a Escolha):
   - comutatividade: $P|Q \equiv Q|P$
   - associatividade: $(P|Q)|R \equiv P|(Q|R)$
   - 0 como identidade: $P|0 \equiv P$

3. Lei *unfolding*: $A(\widetilde{y}) \equiv P\{\widetilde{y}/\widetilde{x}\}$ se $A(\widetilde{x}) \stackrel{def}{=} P$

4. Leis de extensão de escopo:

a. $(\nu x)0$	$\equiv$	0	
b. $(\nu x)(P\|Q)$	$\equiv$	$P\|(\nu x)Q$	se $x \not\in fn(P)$
c. $(\nu x)(P+Q)$	$\equiv$	$P+(\nu x)Q$	se $x \not\in fn(P)$
d. $(\nu x)[u=v]P$	$\equiv$	$[u=v](\nu x)P$	se $x \neq u \wedge x \neq v$
e. $(\nu x)(\nu y)P$	$\equiv$	$(\nu y)(\nu x)P$	

- Os estados do sistema são dados por termos do cálculo-π;
- Os rótulos das transições são dados pelas ações realizadas pelos agentes. Essas ações podem ser:
  - silenciosa (τ): na transição $P \xrightarrow{\tau} Q$, o agente P evolui para Q sem interação com o ambiente. Essa ação pode ter origem de um agente $\tau.P$ ou da comunicação interna de um agente;
  - saída livre ($\overline{x}y$): na transição $P \xrightarrow{\overline{x}y} Q$, o agente P envia um nome livre y pelo canal $\overline{x}$ e passa a comportar-se como Q. Essas ações têm origem de agentes na forma $\overline{x}y.P$;
  - entrada ($x(y)$): na transição $P \xrightarrow{x(y)} Q$, o agente P recebe um nome qualquer w no canal x e evolui para $Q\{w/y\}$. Aqui y é uma referência para o local onde o valor recebido deverá ser colocado. Esse tipo de ação origina-se de agentes na forma $x(y).P$;
  - saída ligada ($\overline{x}(y)$): na transição $P \xrightarrow{\overline{x}(y)} Q$, o agente P envia um nome local no canal $\overline{x}$, e (y) é uma referência para onde esse nome local pode ocorrer. Essa ação tem origem de agentes que enviam nomes para fora de seu escopo, como por exemplo, $(\nu y)\overline{x}y.P$.

Na tabela 5.3 pode-se ver resumidamente as características de cada ação.

Tabela 5.3: Ações do Cálculo-π

α	Tipo	fn(α)	bn(α)
τ	Silenciosa		
$\overline{x}y$	Saída Livre	$\{x, y\}$	
$x(y)$	Entrada	$\{x\}$	$\{y\}$
$\overline{x}(y)$	Saída Ligada	$\{x\}$	$\{y\}$

- O estado inicial do STR é o termo inicial T da especificação em cálculo-π;
- A relação de transição do cálculo-π é dada pelo conjunto de regras de inferência apresentado na tabela 5.4.

5.4 Bissimilaridade

Em cálculo-π, como na maioria das álgebras de processos, as relações de equivalência de agentes são baseadas em bissimulações. Uma definição genérica de bissimulação é que ela é uma relação binária e simétrica entre agentes que satisfaz:

$$P\mathcal{R}Q \text{ e } P \xrightarrow{\alpha} P' \text{ implica } \exists Q' : Q \xrightarrow{\alpha} Q' \wedge P'\mathcal{R}Q'$$

Dois agentes são ditos bissimilares se estão relacionados por alguma bissimulação. Além disso, para o cálculo-π, tem-se que considerar as ações sem nomes ligados. Basta que um agente simule somente ações ligadas onde o nome ligado não ocorra livre no próprio agente.

Daqui em diante, o uso de "$bn(\alpha)$ é livre" significa que o nome em $bn(\alpha)$ é diferente de qualquer nome livre que ocorre nos agentes envolvidos.

Existem diversos modos de definir equivalência (PARROW, 2001). A mais fundamental equivalência comportamental é a bissimilaridade forte. Porém existe também a definição de bissimilaridade fraca, que é a definição utilizada neste trabalho.

A idéia principal da bissimilaridade fraca é que as transições cuja ação é τ são consideradas como não observáveis. Assim são definidos:

$\Longrightarrow$, que significa $(\xrightarrow{\tau})^*$, isto é, zero ou mais transições τ;

$\overset{\alpha}{\Longrightarrow}$, que significa $\Longrightarrow\xrightarrow{\alpha}\Longrightarrow$;

$\overset{\widehat{\alpha}}{\Longrightarrow}$, que significa:

$\overset{\alpha}{\Longrightarrow}$ se $\alpha \neq \tau$

$\Longrightarrow$ se $\alpha = \tau$

Tabela 5.4: Semântica Operacional do Cálculo-π

TAU: $\dfrac{}{\tau.P \xrightarrow{\tau} P}$ SAÍDA: $\dfrac{}{\overline{x}y.P \xrightarrow{\overline{x}y} P}$

ENTRADA: $\dfrac{}{x(z).P \xrightarrow{x(w)} P\{w/z\}}\ w \not\in fn((\nu z)P)$

ESCOLHA: $\dfrac{P \xrightarrow{\alpha} P'}{P + Q \xrightarrow{\alpha} P'}$ CONDIÇÃO: $\dfrac{P \xrightarrow{\alpha} P'}{[x = x]P \xrightarrow{\alpha} P'}$

PAR: $\dfrac{P \xrightarrow{\alpha} P'}{P|Q \xrightarrow{\alpha} P'|Q}\ bn(\alpha) \cap fn(Q) = \varnothing$

COM: $\dfrac{P \xrightarrow{\overline{x}y} P' \quad Q \xrightarrow{x(z)} Q'}{P|Q \xrightarrow{\tau} P'|Q'\{y/z\}}$ CLOSE: $\dfrac{P \xrightarrow{\overline{x}(w)} P' \quad Q \xrightarrow{x(w)} Q'}{P|Q \xrightarrow{\tau} (\nu w)(P'|Q')}\ w \not\in fn(Q)$

REST: $\dfrac{P \xrightarrow{\alpha} P'}{(\nu y)P \xrightarrow{\alpha} (\nu y)P'}\ y \not\in n(\alpha)$

OPEN: $\dfrac{P \xrightarrow{\overline{x}y} P'}{(\nu y)P \xrightarrow{\overline{x}(w)} P'\{w/y\}}\ \begin{array}{l} y \neq x \\ w \not\in fn((\nu y)P') \end{array}$

ESTRUT: $\dfrac{P' \equiv P \quad P \xrightarrow{\alpha} Q \quad Q \equiv Q'}{P' \xrightarrow{\alpha} Q'}$

Definição 19 (Bissimulação Fraca) *É uma relação binária e simétrica $\mathcal{R}$ entre agentes que satisfaz o seguinte: $P\mathcal{R}Q$ e $P \xrightarrow{\alpha} P'$ onde $bn(\alpha)$ é livre implica que:*

(i) *se* $\alpha = a(x)$ *então* $\exists Q'':$ $Q \Longrightarrow \xrightarrow{a(x)} Q'' \wedge$
$\forall u \exists Q':\ Q''\{u/x\} \Longrightarrow Q' \wedge P'\{u/x\}\mathcal{R}Q'$

(ii) *se* $\alpha \neq a(x)$ *então* $\exists Q' :$ $Q \stackrel{\widehat{\alpha}}{\Longrightarrow} Q' \wedge P'\mathcal{R}Q'$

P e Q são fracamente bissimilares se eles estão relacionados por uma bissimulação fraca, e é escrito $P \dot{\approx} Q$.

∇

5.5 Modelo Baseado em Objetos Descrito em Cálculo-π - MBO-π

A tradução proposta neste trabalho é a tradução de objetos e mensagens e suas relações em agentes do cálculo-π. Estes agentes possuem formas específicas. Nesta seção serão definidos os tipos de termos do cálculo-π que caracterizam as gramáticas de grafos traduzidas. Esta caracterização, chamada de Modelo Baseado em Objetos descrito em Cálculo-π - MBO-π, será útil para a definição e realização das provas nas próximas seções.

Em um MBO-π, os objetos e mensagens são definidos como processos do cálculo-π, denominados agente do objeto e agente da mensagem, respectivamente. Esses agentes se comunicam através de um canal local. O objetos de destino e os parâmetros de cada mensagem são indicados como parâmetros dos agentes das mensagens.

As reações dos objetos ao receber cada mensagem são descritas por agentes de regras, que compõem o agente do objeto. Esses agentes de regras tem como identificação o tipo da mensagem que está sendo tratada e o identificador da "regra" que irá tratar essa mensagem. Uma "regra" descreve os procedimentos a serem executados para o tratamento de uma mensagem. Como pode-se ter diferentes procedimentos para tratar o mesmo tipo de mensagem, pode-se ter diferentes agentes de regra que descrevem o tratamento de um mesmo tipo de mensagem. A escolha de qual procedimento deve ser executado deve ser não-determinística. Isso é representado neste modelo pela composição dos agentes de regras (para um mesmo tipo de mensagem) com o operador de escolha ($+$), sem guarda. Desta forma a escolha do agente a ser processado é não-determinística.

Essas reações podem ser a criação de novos objetos (onde são instanciados novos agentes de objetos de um determinado tipo) e/ou o envio de novas mensagens (onde são instanciados agentes de mensagens de determinados tipos).

A concorrência entre os objetos é modelada pela composição paralela dos agentes dos objetos e das mensagens. Assim cada objeto pode tratar suas mensagens em paralelo. A concorrência interna é modelada pelo operador de replicação utilizado no agente do objeto. Esse operador permite que o objeto possa tratar várias mensagens ao mesmo tempo, disponibilizando várias cópias deste agente.

Este modelo é descrito por uma especificação em cálculo-π (I_π, A_π). I_π é o termo do cálculo-π que representa o estado inicial do sistema. Este termo é formado pela composição paralela de agentes que representam as mensagens e objetos (definidos em A_π). O identificador de cada mensagem e objeto de I_π é tratado como um canal local deste termo.

A_π é o conjunto de definições dos agentes que representam as mensagens e os objetos do sistema.

O tipos dos objetos e mensagens deste modelo também são descritos por um hipergrafo, que pode ser obtido a partir das descrições dos agentes do sistema, isto é, a partir do conjunto A_π.

Notação:

- $|C|$ indica o número de elementos de um conjunto C. Pode-se também utilizar este operador para listas, onde $|l|$ indica o números de elementos de l;
- Se l é uma lista e C é um conjunto, a notação $e \in l$ significa que o elemento e pertence a lista l e $l \in C$ significa que todos os elementos de l estão contidos em C;
- Se l é uma lista o i-ésimo elemento de l é denotado por $l[i]$.
- A notação $\widetilde{p}$ é a abreviatura de $p_1, \cdots, p_n$, onde $n = |p|$;
- A notação $\widetilde{p}_l$, onde l é uma lista e $n = |p|$, é a abreviatura de $p_{1.l[1]}, \cdots, p_{n.l[n]}$;
- Um subtermo $(A_1| \cdots |A_n)$ de um termo P é denominado a_P e usa-se $A_i \in a_P$ para expressar a existência do agente A_i neste subtermo de P.

Definição 20 (Modelo Baseado em Objetos Descrito em Cálculo π) *Dados os conjuntos finitos TO, TM e P, respectivamente, de nomes de tipos de objetos, de nomes de tipos de mensagens e de números de parâmetros de mensagens, RM_{tm}, o conjunto de identificadores de regras que tratam as mensagens do tipo tm, e TM_{to}, subconjunto de TM que contém os nomes das mensagens tratadas pelo tipo to de objeto. Um MBO-π é uma especificação em cálculo-π $MB = (I_\pi, A_\pi)$, onde:*

$$I_\pi = (\nu id_1, \cdots, id_n)\ a_{I_\pi},$$

onde $a_{I_\pi} = (A_1| \cdots |\ A_n)$, id_i é um identificador tal que $id_i = id_j \Rightarrow i = j$ e $A_i = O_t(id_i)$ ou $A_i = @(t, id_i, d, v[1], \cdots, v[m])$, com $v, d \in \{id|O_t(id) \in a_{I_\pi}\}$, $0 < k \leq m$ e $m \geq 0$. Se $m = 0$, então $A_i = @(t, id_i, d)$.

$$A_\pi = Ms \cup Ob \tag{5.1}$$

$$Ms = \{ag\_msg_t|\ t \in TM\}, \text{onde} \tag{5.2}$$

$$ag\_msg_t = @(t, id, d_{td}, \widetilde{p}_{tp}) \stackrel{def}{=} (\nu m)\overline{d}m.\overline{mt}.\bigodot_{i=1}^{|p|} \overline{m}p_{i.tp[i]}.\overline{m}id, \tag{5.3}$$

$$td, tp \in TO \text{ e } |p| \in P$$

$$Ob = \{ag\_obj_{to} \mid to \in TO\}, \text{onde} \tag{5.4}$$

$$ag\_obj_{to} =$$

$$O_{to}(id) \stackrel{def}{=} id(m).(m(t).\sum_{tm \in TM_{to}} [t = tm]M_{tm}(m, id, t) \mid O_{to}(id)), \tag{5.5}$$

$$n, a \in P,\ s, j \in \mathbb{N},$$

$$M_{tm}(m, d, t) \stackrel{def}{=} \sum_{nr \in RM_{tm}} R_{tm.nr}(m, d, t, nr), \tag{5.6}$$

$$R_{tm.nr}(m, d, t, nr) \stackrel{def}{=} \bigodot_{k=1}^{n} m(p_k).((\nu id_1, \cdots, id_s, msg_1, \cdots, msg_j),$$

$$((\prod_{i=1}^{s} CriaObj(id_i, to_i)) \mid (\prod_{i=1}^{s} id_i).m(id).\overline{nr}.\overline{t}id.$$

$$(\prod_{i=1}^{j} @(t_i, msg_i, d_i, v_i[1], \cdots, v_i[a_i])))) \text{ e} \tag{5.7}$$

$$CriaObj(id, t) \stackrel{def}{=} \sum_{c \in TO} ([t = c]\overline{id}.O_c(id)) \tag{5.8}$$

∇

Quando não houver dúvida, os elementos do conjunto A_π serão abreviados por $@(t, id, d_{td}, \widetilde{p}_{tp})$ e $O_{to}(id)$.

Em (5.1) é definido o conjunto A_π, que é formado pela união dos conjuntos Ms e Ob. O conjunto Ms, em (5.2), contém as definições dos agentes das mensagens do sistema. Para cada mensagem com tipo diferente, há um elemento neste conjunto.

Na fórmula (5.3) são descritos os elementos em Ms. Cada agente possui um tipo t, um identificador id, um destino d_{td} e uma lista de parâmetros $\widetilde{p}_{tp}$, sendo que td e tp identificam o tipo do objeto de destino e os tipos dos objetos parâmetros da mensagem, respectivamente.

O número de parâmetros da mensagem é dado pelo comprimento de $\widetilde{p}$. Se $|p| = 0$, o subtermo $\bigodot_{i=1}^{|p|} \overline{m}p_{i.tp[i]}$ é omitido da definição do agente da mensagem.

Cada agente de mensagem possui um canal local m que será utilizado para a comunicação com o objeto d_{td} que irá tratá-la. Esse agente envia o nome do canal local m através do canal d_{td}, que é o identificador do objeto de destino da mensagem. Com isso o canal m se tornará local aos dois agentes (mensagem e objeto de destino).

A seguir o agente comunica-se com o objeto de destino através do canal local, enviando o seu tipo e seus parâmetros. Depois disso, o agente da mensagem envia o seu identificador (id), para o objeto que a está tratando. Este nome é enviado para o agente do objeto, que posteriormente enviará o id para o ambiente através do canal t.

No final destas ações o agente reduz-se a um agente nulo 0, o que indica o final do tratamento desta mensagem.

Em (5.4) é definido o conjunto Ob, que contém as definições dos agentes que representam os objetos. Para cada tipo ($to \in TO$) de objeto há um elemento neste conjunto.

Em (5.5) são descritos os elementos de Ob. Cada objeto possui um identificador id, por onde ele recebe o nome do canal local através do qual ele tratará a mensagem

recebida. Esse agente possui uma composição paralela que permite o tratamento paralelo de diferentes mensagens por um objeto. Isto ocorre porque após a sincronização com a mensagem, o agente apresenta seu comportamento anterior em paralelo com um subtermo que irá tratar a mensagem recebida.

O objeto Ob recebe através do canal m o tipo da mensagem que deverá tratar. Cada objeto to possui, em uma escolha, um agente M_{tm} diferente para cada tm onde $tm \in TM_{to}$ e TM_{to} é o conjunto de tipos de mensagens que o objeto to pode tratar. O agente M_{tm} adequado ao tipo de mensagem recebida é selecionado pela condição $[t = tm]$ que guarda o agente M_{tm}.

Em (5.6), é definido o agente M_{tm}, que trata o tipo tm de mensagem. Esse agente é composto pela escolha de agentes $R_{tm.nr}$ que representam as diferentes ações que podem ser tomadas para tratar aquele tipo de mensagem. Cada $nr \in RM_{tm}$ (onde RM_{tm} é um conjunto de nomes de regras que descrevem o tratamento das mensagens do tipo tm do sistema) representa um tratamento diferente para a mesma mensagem, permitindo assim representar o não-determinismo.

Os agentes $R_{tm.nr}$, em (5.7), é que descrevem o tratamento da mensagem. Inicialmente esses agentes recebem os identificadores dos parâmetros da mensagem. Esses objetos poderão ser referenciados pelas mensagens que serão criadas pelo tratamento da mensagem. Caso a mensagem não possua parâmetros, o subtermo $\bigodot_{k=1}^{n} m(p_k)$ é omitido.

Em seguida, são criados os novos objetos e/ou mensagens. Todos os identificadores dos objetos e mensagens criados são declarados como nomes locais, o que permite declará-los como nomes novos naquele escopo.

Os objetos do sistema são criados a partir do agente $CriaObj$ (5.8). Esse agente contém todos os tipos de objetos $c \in TO$ do sistema, e o objeto a ser criado é selecionado através do seu tipo ($[t = c]$). Cada objeto possui como guarda o seu identificador, no qual deverá ocorrer uma sincronização antes da real criação do objeto.

Antes de começar a criar as mensagens, todos os objetos devem ter sido criados. Isso é garantido pela guarda $(\prod_{i=1}^{s} id_i)$ que permite prosseguir somente após a sincronização com todos os objetos criados. Após essas sincronizações, o objeto recebe, do agente da mensagem que está tratando, o identificador (id) desta, sincroniza com o ambiente através do canal nr que representa a regra que tratou a mensagem e envia ao ambiente o id através do canal t recebido como parâmetro. Essas duas ações indicam que a mensagem id, do tipo t, foi tratada pela regra nr.

Por fim, são criadas as mensagens. Cada mensagem i criada, possui um identificador (msg_i), um tipo (t_i), um destino (d_i) e uma lista de parâmetros (v_i), cujo número de elementos é dado por a_i (caso $a_i = 0$, os parâmetros são omitidos). Os objetos de destino e os parâmetros das mensagens criadas podem ser ou o objeto que está tratando a mensagem (d), ou um dos parâmetros da mensagem que esta sendo tratada (p_i), ou ainda um dos objetos criados (id_i).

Caso a regra que está sendo tratada não crie objetos e/ou mensagens, os subtermos $(\prod_{i=1}^{s} CriaObj(id_i, to_i)) \mid (\prod_{i=1}^{s} id_i)$ e $(\prod_{i=1}^{j} @(t_i, msg_i, d_i, v_i[1], \cdots, v_i[a_i]))$, respectivamente, bem como a declaração de seus identificadores, são omitidos.

No sistema de transição de um MBO-π existem estados que representam o tratamento completo das regras, isto é, todas as regras que começaram a ser tratadas já foram totalmente tratadas. Esses termos são denominados **Termos Completos**.

Definição 21 (Termo Completo) *Dado um MBO-π $MB = (I_\pi, A_\pi)$, um termo completo T_π de MB é um termo do cálculo π tal que:*

i) existe um caminho $I_\pi \xrightarrow{\alpha^*} T_\pi$ *em* $SemPi(MB)$*;*

ii) T_π *tem o formato:*

$$T_\pi = (\nu id_1, \cdots, id_n)\ a_{T_\pi},$$

onde $a_{T_\pi} = (A_1 | \cdots |\ A_n)$ *e* id_i *é um identificador tal que* $A_i = (\nu id_{n+1}, \cdots, id_{n+1+o})\ a'_{T_\pi}$ *ou* $A_i = O_t(id_i)$ *ou* $A_i = @(t, id_i, d, v_1, \cdots, v_m)$*, com* $v_k, d \in \{id | O_t(id) \in a_{T_\pi}\}$*,* $0 < k \leq m$ *e* $m \geq 0$*. Se* $m = 0$*, então* $A_i = @(t, id_i, d)$*, e* a'_{T_π} *é um termo completo.*

O conjunto de todos os termos completos de um MBO-π MB *é denominado de* $TermoCO_{MB}$*.*

∇

Em um termo completo podem haver nomes de variáveis ligadas iguais, porém com diferentes escopos. Um termo estruturalmente congruente a um termo completo que não possui nenhuma variável ligada com mesmo nome é chamado de **Termo BO**.

Definição 22 (Termo BO) *Dado um MBO-*π $MB = (I_\pi, A_\pi)$*, um termo BO* B *de* MB *é um termo completo de MB onde:*

i) $B = (\nu id_1, \cdots, id_n)\ a_B$*,* $a_B = (A_1 | \cdots |\ A_n)$ *e* $A_i = (\nu id_{n+1}, \cdots, id_{n+1+o})\ a'_B$ *ou* $A_i = O_t(id_i)$ *ou* $A_i = @(t, id_i, d, v_1, \cdots, v_m)$*, com* $v_k, d \in \{id | O_t(id) \in a_B\}$*,* $0 < k \leq m$ *e* $m \geq 0$*. Se* $m = 0$*, então* $A_i = @(t, id_i, d)$*,* a'_B *é um termo completo e;*

ii) $\forall id_i \in C.id_i = id_j \Rightarrow i = j$*.*

O conjunto de todos os termos BOs de um MBO-π MB *é denominado de* $TermoBO_{MB}$*.*

∇

Como foi dito no capítulo 3, os tipos dos objetos e mensagens de um SBO podem ser descritos por um hipergrafo. A seguir é dada a definição do hipergrafo tipo de um MBO-π. Esse hipergrafo é construído a partir da descrição dos agentes do sistema, isto é, a partir de A_π. O conjunto de vértices é obtido a partir da definição dos agentes dos objetos, que possuem um índice que indica seu tipo. O conjunto de hiperarcos é obtido a partir da definição dos agentes das mensagens, que armazenam a informação do seu tipo no parâmetro t. As funções de origem e destino de cada hiperarco são definidas a partir dos parâmetros dos agentes da mensagem. O destino de cada hiperarco é identificado pelo índice td do parâmetro d_{td} do agente da mensagem. Já as origens do hiperarco são identificadas pela lista tp, índice dos parâmetros $\widetilde{p}_{tp}$ destes mesmos agentes. Caso a mensagem msg não possua parâmetros, $sc(msg) = \langle\rangle$.

Definição 23 (Tipo de um MBO-π**)** *Dado um MBO-*π $MB = (I_\pi, A_\pi)$*, o tipo deste modelo é um hipergrafo* $HG_{MB} = (V, E, sc, tg)$*, onde*

- $V = \{to \mid O_{to}(id) \in A_\pi\}$*;*

- $E = \{t \mid @(t, id, d_{td}, \widetilde{p}_{tp}) \in A_\pi\}$;
- $\forall a \in E$, $tg(a) = td$, ***tal que*** $@(a, id, d_{td}, \widetilde{p}_{tp}) \in A_\pi$ ***ou*** $@(a, id, d_{td}) \in A_\pi$;
- $\forall a \in E$, $sc(a) = \begin{cases} tp & \text{se } @(a, id, d_{td}, \widetilde{p}_{tp}) \in A_\pi \\ \langle\,\rangle & \text{se } @(a, id, d_{td}) \in A_\pi \end{cases}$

∇

Neste capítulo foram apresentados o cálculo-π e sua semântica, bem como uma caracterização dos agentes resultantes da tradução de uma gramática de grafos em cálculo-π. No próximo capítulo será apresentada a tradução propriamente dita e alguns teoremas que provam que a tradução proposta preserva a semântica das GHBOs. Ainda neste capítulo foram dadas alguma definições utilizadas nas provas do capítulo a seguir.

6 TRADUÇÃO DE GHBO PARA MBO-π

Nos capítulos anteriores, foram mostrados dois modelos para especificação de sistemas baseados em objetos; um baseado em gramáticas de hipergrafos e outro em cálculo-π.

Como foi dito anteriormente, sob o aspecto de especificação de sistemas, as GGBOs parecem ser mais intuitivas e mais fáceis de utilizar do que o cálculo-π, principalmente considerando-se que a maioria dos desenvolvedores de software não possui muita afinidade com métodos formais. Em contrapartida, para as GHBO ainda não existem ferramentas para verificação automática de propriedades desejadas nos sistemas, enquanto que para o cálculo-π sim.

Assim, neste capítulo será apresentada uma tradução do modelo das GHBO para o modelo em cálculo-π (MBO-π). Desta forma, poder-se-á usar a tradução para provar propriedades de sistemas especificados em GHBO. Porém, para que essas verificações sejam válidas, a tradução deve preservar a semântica do sistema modelado e esta prova é demonstrada através da comparação dos sistemas de transição dos dois modelos.

A partir de uma especificação dada em GHBO obtém-se, através da tradução, uma especificação em MBO-π. O termo inicial da especificação MBO-π é obtido a partir do hipergrafo inicial da GHBO. Cada vértice do hipergrafo inicial é traduzido para um identificador de agente de objeto do termo inicial, da mesma forma, cada hiperarco é traduzido para um identificador de agente de mensagem.

As definições dos agentes da especificação MBO-π são obtidas a partir do hipergrafo tipo e das regras da GHBO. As definições dos agentes de mensagens e dos agentes de objetos são obtidas a partir do hipergrafo tipo. Para cada vértice e hiperarco, a tradução gera uma definição de objeto e mensagem, respectivamente. A informação sobre os tipos de mensagens tratadas por cada objeto também é obtida a partir do hipergrafo tipo. As definições dos agentes que descrevem o tratamento de cada mensagem são obtidas a partir das regras da GHBO.

As sincronizações realizadas pelos agentes simulam as aplicações das regras da gramática. Porém, esses agentes realizam diversas sincronizações silenciosas (τ) para simular a aplicação de uma única regra na GHBO (que ocorre em uma única trasição). Além disso, no MBO-π há duas transições para identificar a aplicação da regra (transições diferentes de τ). Essas ações podem aparecer intercaladas, coisa que não ocorre na GHBO, pois a aplicação da regra é atômica. Há seqüências de transições que representam essa aplicação atômica no STR do MBO-π e para comparar os sistemas de transição dos modelos, foram consideradas apenas as transições do STR do MBO-π que identificam a aplicação da regra. A partir disso, provou-se que todas as seqüências de transições do STR de um modelo estavam no do outro, e vice-versa. Assim, provou-se que o comportamento de uma GHBO é preservada e que nenhum comportamento diferente é acrescentado na sua tradução.

Na seção 6.1 são definidas algumas funções auxiliares que nos permitem simplificar a definição da tradução de uma GHBO para o cálculo-π. Na seção 6.2 são definidos: a tradução de GHBO para um MBO-π, caminhos completos e caminhos BOs de um MBO-π, a tradução de caminhos completos do MBO-π para caminhos da $SemGHBO$ e a tradução de termos BO para hipergrafos tipados. E na seção 6.3 é demonstrada a compatibilidade das semânticas dos dois modelos.

Notação:

- Se C é um conjunto, $P(C)$ é o conjunto das partes de C;
- Se $l1$ e $l2$ são duas listas, a concatenação delas é denotada por $l1.l2$;
- Se l é uma lista a notação $(\nu l[1], \cdots, l[|l|])$ é abreviada para (νl).

6.1 Funções Auxiliares

Algumas funções são utilizadas ao longo da definição da tradução. O domínio e contradomínio destas funções são dados pelos conjuntos a seguir:

- Set é o conjunto de todos os conjuntos;
- $List$ é o conjunto de listas;
- $Arestas$ é um conjunto de identificadores de arestas;
- $Vertices$ é um conjunto de identificadores de vértices;
- $Hiper$ é o conjunto de hipergrafos tipados;
- $GramHiper$ é o conjunto de gramáticas de hipergrafos baseada em objetos;
- $Nomes$ é um conjunto de nomes de regras;
- $Regras$ é o conjunto de regras GHBO.

A funções são definidas a seguir:

- $Lista : Set \rightarrow List$

  $$Lista(C) = \begin{cases} \langle\,\rangle & \text{se } |C| = 0 \\ Lista(C - \{c\}).\langle c\rangle & \text{se } |C| > 1 \text{ e } c \in C \end{cases}$$

  Esta função transforma um conjunto C em uma lista com o seguinte formato: $\langle a, b, c, \cdots\rangle$, onde $a, b, c, \cdots \in C$. É uma função não-derterminística, isto é, a ordem dos elementos na lista é aleatória.

- $DestMsg : Vertices, Hiper \rightarrow P(Arestas)$

  $DestMsg(v, G) = \{x | tg^G(x) = v \wedge x \in E_G\}$

  É uma função parcial, definida somente para os pares (v, G), tal que $v \in G$. Essa função retorna o conjunto de arestas de G que chegam no vértice v, isto é, as arestas que tem como destino este vértice.

- $Par : Arestas, Hiper \rightarrow P(Vertices)$

  $Par(a, G) = \{x | x \in sc^G(a) \wedge a \in E_G\}$

  A função Par retorna o conjunto de vértices de origem a partir da aresta a do hipergrafo G. É também uma função parcial definida somente para os pares (a, G), tal que $a \in G$.

- $RegrasMsg : Arestas, GramHiper \rightarrow Nomes$

  $RegrasMsg(m, (T, I^T, N, n)) = \{nr | n(nr) = L \rightarrow R \wedge nr \in N \wedge t^L(a) = m \wedge a \in E_L\}$

  A função $RegraMsg$ retorna o conjunto de regras que tratam o tipo de mensagem m, isto é, as regras que possuem uma aresta do tipo m no seu lado esquerdo. Esta função é parcial, sendo definida para os pares (m, G) onde m é o tipo de um hiperarco que pertence ao lado esquerdo de alguma regra de G.

- $NewObj : Regras \rightarrow P(Vertices)$

  $NewObj(r : L \rightarrow R) = \{x |\ x \in V_R\ \wedge\ x \notin rng(r)\}$

  A função total $NewObj$ retorna o conjunto de vértices criados pela regra indicada.

- $NewMsg : Regras \rightarrow P(Arestas)$

  $NewMsg(r : L \rightarrow R) = \{x |\ x \in E_R\}$

  A função $NewMsg$ retorna o conjunto de arestas criadas pela regra indicada. Esta função é total.

- $trigger : Regras \rightarrow Arestas$

  $trigger(r : L \rightarrow R) = a$, onde $E_L = \{a\}$

  A função total $trigger$ retorna a mensagem que dispara a regra r, isto é, a aresta que está no lado esquerdo da regra.

- $Pos : Arestas \cup Vertices, List \rightarrow \mathbb{N}$

  $Pos(e, l) = PosAux(e, l, 1)$

  $PosAux : Arestas \cup Vertices, List, \mathbb{N} \rightarrow \mathbb{N}$

  $$PosAux(e, l, n) = \begin{cases} n & \text{se } l = \langle e \rangle.L \vee l = \langle e \rangle \\ PosAux(e, L, n+1) & \text{se } l = \langle a \rangle.L \wedge a \neq e \end{cases}$$

  A função Pos retorna a posição do elemento e na lista l. Ela utiliza a função auxiliar $PosAux$ para calcular esta posição. Essa função é parcial e é definida somente para os pares (e, l) onde $e \in l$.

6.2 Tradução

A seguir, é definida uma função através da qual obtém-se uma especificação em um modelo MBO-π a partir de uma gramática de hipergrafos baseada em objetos. Os termos definidos não são termos do cálculo-π em si, mas sim fórmulas para obtê-los. Os termos são instâncias destas fórmulas. Cada fórmula possui variáveis que assumem valores diferentes para traduções diferentes. Essas variáveis (apresentam-se com fonte em negrito) devem ser instanciadas no momento da tradução. Em alguns casos, ao invés de variáveis há funções, que são avaliadas no momento da tradução.

Definição 24 (Tradução de GHBO para MBO-π) *Dada uma GHBO* $HG = (T, I^T, N, n)$*, com* $T = (V_T, E_T, sc^T, tg^T)$ *e* $I = ((V_I, E_I, sc^I, tg^I), t^I, T)$*, uma tradução de* HG *para um MBO-*π *é definida por* $Trad_{GH}(HG) = (I_\pi, A_\pi)$*, onde:* $I_\pi = Trad_H(I)$*,* $A_\pi = Trad_R(T, N, n)$ *e*

$$Trad_H(I) = (\nu \mathbf{lv}, \mathbf{la})(\prod_{i=1}^{|\mathbf{lv}|} O_{\mathbf{t^I(lv[i])}}(\mathbf{lv[i]}) \mid \prod_{j=1}^{|\mathbf{la}|} @(\mathbf{t^I(la[j])}, \mathbf{la[j]}, \mathbf{tg^I(la[j])}, \mathbf{p_j[1]}, \cdots, \mathbf{p_j[n_j]})), \quad (6.1)$$

$$\text{onde } \mathbf{lv} = Lista(V_I), \mathbf{la} = Lista(E_I), \mathbf{p_j} = \mathbf{sc^I(la[j])}, \mathbf{n_j} = |\mathbf{p_j}|$$

$$Trad_R(T, N, n) = \{ag |\ ag \in Ms \vee ag \in Ob\} \quad (6.2)$$

$$Ms = \{@(\mathbf{t}, id, d_{\mathbf{td}}, \widetilde{p}_{\mathbf{tp}}) \stackrel{def}{=} (\nu m)\overline{d_{\mathbf{td}}}m.\overline{m}\mathbf{t}. \bigodot_{i=1}^{\mathbf{n}} \overline{m}p_{i.\mathbf{tp[i]}}.\overline{m}id\} \quad (6.3)$$

$$\text{onde } \mathbf{t} \in E_T, \mathbf{td} = \mathbf{tg^T(t)}, \mathbf{tp} = \mathbf{sc^T(t)}, \mathbf{n} = |\mathbf{tp}|\ e\ |p| = \mathbf{n}$$

$$Ob = \{O_{\mathbf{v}}(id) \stackrel{def}{=}\ id(m).(m(t). \sum_{\mathbf{a} \in DestMsg(\mathbf{v},T)} [t = \mathbf{a}]M_{\mathbf{a}}(m, id, t) \mid O_{\mathbf{v}}(id))\} \quad (6.4)$$

$$\text{onde } M_{\mathbf{a}}(m, d, t) \stackrel{def}{=} \sum_{\mathbf{nr} \in RegrasMsg(\mathbf{a},HG)} R_{\mathbf{a.nr}}(m, d, t, \mathbf{nr}) \ \wedge \quad (6.5)$$

$$R_{\mathbf{a.nr}}(m, d, t, \mathbf{nr}) \stackrel{def}{=} \bigodot_{i=1}^{|\mathbf{sc^T(a)}|} m(p_i).((\nu \mathbf{lv}, \mathbf{lm})(\prod_{i=1}^{|\mathbf{lv}|}(CriaObj(\mathbf{lv[i]}, \mathbf{t^R(lv[i])}))| \prod_{i=1}^{|\mathbf{lv}|} \mathbf{lv[i]}.m(id).\overline{\mathbf{nr}}.\bar{t}id. \prod_{i=1}^{|\mathbf{lm}|}(@(\mathbf{t^R(lm[i])}, \mathbf{lm[i]}, \mathbf{dm_i}, \mathbf{pm_i[1]}, \cdots, \mathbf{pm_i[|lpR_i|]}))))) \ \wedge \quad (6.6)$$

$$CriaObj(id, t) \stackrel{def}{=} \sum_{\mathbf{b} \in V_T} ([t = \mathbf{b}]\overline{id}.O_{\mathbf{b}}(id)) \wedge \quad (6.7)$$

$$\mathbf{dm_i} = \begin{cases} d & \text{se } tg^R(lm[i]) \in rng(r) \text{ e} \\ & r^{-1}(tg^R(lm[i])) = tg^L(trigger(r)) \\ p_{\mathbf{x}} & \text{se } tg^R(lm[i]) \in rng(r) \text{ e} \\ & r^{-1}(tg^R(lm[i])) \in sc^L(trigger(r)), \text{onde} \\ & \mathbf{x} = Pos(r^{-1}(tg^R(lm[i])), sc^L(trigger(r))) \\ \mathbf{tg^R(lm[i])} & \text{se } tg^R(lm[i]) \notin rng(r) \end{cases} \wedge \quad (6.8)$$

$$\mathbf{pm_i}[\mathbf{j_i}] = \begin{cases} d & \text{se } lpR_i[j_i] \in rng(r) \text{ e} \\ & r^{-1}(lpR_i[j_i]) = tg^L(trigger(r)) \\ p_{\mathbf{x}} & \text{se } lpR_i[j_i] \in rng(r) \text{ e} \\ & r^{-1}(lpR_i[j_i]) \in sc^L(trigger(r)), \text{onde} \\ & \mathbf{x} = Pos(r^{-1}(lpR_i[j_i]), sc^L(trigger(r))) \\ \mathbf{lpR_i}[\mathbf{j_i}] & \text{se } lpR_i[j_i] \notin rng(r) \end{cases} \quad \wedge \tag{6.9}$$

$$\mathbf{v} \in V_T \ \wedge \ r = \mathbf{n}(\mathbf{nr}) \ \wedge \ \mathbf{lv} = Lista(NewObj(r)) \ \wedge$$
$$\mathbf{lm} = Lista(NewMsg(r)) \ \wedge \ \mathbf{lpR_i} = \mathbf{sc^R}(\mathbf{lm}[\mathbf{i}]) \ \wedge \ 0 < \mathbf{j_i} \leq |\mathbf{lpR_i}| \tag{6.10}$$

∇

A fórmula (6.1) descreve como obter o termo que representa o estado inicial do sistema. Nesta fórmula, são utilizadas algumas variáveis (listas) auxiliares. As listas $\mathbf{la}$ e $\mathbf{lv}$ são, respectivamente, listas de identificadores de mensagens (arestas) e de objetos (vértices) do hipergrafo inicial de HG. Cada um destes identificadores é declarado como canal local ao termo inicial, e isso é descrito por $(\nu\mathbf{lv}, \mathbf{la})$. Os agentes que compõem este termo são todos colocados em uma composição paralela. O primeiro produtório de (6.1) representa a composição paralela de todos objetos (todos os elementos de $\mathbf{lv}$) do estado inicial do sistema. Há uma definição de agente $O_{\mathbf{t^I}(\mathbf{lv}[\mathbf{i}])}$ diferente para cada tipo de objeto, onde o tipo do objeto é representado por $\mathbf{t^I}(\mathbf{lv}[\mathbf{i}])$. Assim podemos ter, no termo inicial, diferentes instâncias do mesmo agente, onde o que os diferencia são os seus identificadores $\mathbf{lv}[\mathbf{i}]$. O mesmo ocorre com as mensagens no segundo produtório. As definições dos agentes das mensagens são diferenciadas pelo tipo de cada mensagem. Para cada mensagem presente no hipergrafo inicial, existe uma instância dos agentes das mensagens, cujos parâmetros são: o tipo da mensagem $\mathbf{t^I}(\mathbf{la}[\mathbf{j}])$, onde $\mathbf{t^I}$ é a função de tipagem do hipergrafo inicial; o seu identificador $\mathbf{la}[\mathbf{j}]$; o objeto de destino da mensagem $\mathbf{tg^I}(\mathbf{la}[\mathbf{j}])$, onde $\mathbf{tg^I}$ é a função de destino das arestas do hipergrafo inicial; e uma lista de parâmetros da mensagem $\mathbf{p_j}$. O comprimento da lista $\mathbf{p_j}$ é igual ao número de parâmetros da mensagem.

As listas $\mathbf{la}$ e $\mathbf{lv}$ são geradas de forma não-determinística, a partir da função $Lista$ (que transforma o conjunto de arestas/vértices em lista), o que pode resultar em termos sintaticamente diferentes para duas traduções do mesmo hipergrafo inicial. Porém estes termos são estruturalmente congruentes e, portanto, apresentam o mesmo comportamento.

Na fórmula (6.2) é definido o conjunto com as definições dos agentes do sistema. Os elementos deste conjunto podem ser definições dos agentes dos objetos (elementos que pertencem a Ob) ou das mensagens (elementos que pertencem a Ms).

Em (6.3), é definido o conjunto de definições dos agentes de mensagens do sistema. Há uma definição de agente diferente para cada aresta do grafo tipo T, isto é, há uma definição para cada tipo de mensagem do sistema ($\mathbf{t} \in E_T$). O identificador do tipo de cada agente de mensagem ($\mathbf{t}$) será o identificador de cada aresta de T. O vértice $\mathbf{td}$ de destino de cada aresta de T é identificado no parâmetro $d_{\mathbf{td}}$, de forma análoga ocorre com a lista de vértices de origem $\mathbf{tp}$, que é identificada em $p_{i.\mathbf{tp}[\mathbf{i}]}$.

Em (6.4), é definido o conjunto de definições dos agentes dos objetos do sistema. Há uma definição diferente para cada vértice do hipergrafo tipo T, isto é, uma definição para cada tipo de objeto. A definição dos agentes $O_{\mathbf{v}}(id)$ são diferenciados pelo seu tipo $\mathbf{v}$. Esta definição é instanciada substituindo-se todos os nomes $\mathbf{a}$ pelo identificador de cada

uma das arestas que chegam no vértice $\mathbf{v} \in T$ (isso é representado pelo somatório desta fórmula). Assim, haverá tantas definições $M_{\mathbf{a}}$ quantas forem as arestas em T.

Em (6.5) é descrita a fórmula para gerar a definição dos agentes $M_{\mathbf{a}}$ do sistema. Este agente é instanciado alterando-se cada nome $\mathbf{nr}$, pelo nome de uma regra que pertence a N e que possui do seu lado esquerdo uma aresta do tipo $\mathbf{a}$. Assim, haverá no somatório desta fórmula, tantos agentes $R_{\mathbf{a.nr}}$ quantos forem as regras em N.

Na fórmula (6.6), é descrita a definição dos agentes que tratam as regras. Para instanciar essa fórmula, deve-se substituir todas as ocorrências da variável $\mathbf{lv}$ pela lista de vértices criados pela $\mathbf{nr}$. Da mesma forma ocorre com a variável $\mathbf{lm}$, que deve ser substituída pela lista de arestas criadas por $\mathbf{nr}$. Além disso, há algumas funções que devem ser calculadas: $\mathbf{t^R(lv[i])}$, que é o tipo do i-ésimo vértice criado por $\mathbf{nr}$ e $\mathbf{t^R(lm[i])}$, que é o tipo da i-ésima aresta criada. O vértice ($\mathbf{dm_i}$) de destino das mensagens criadas pela regra é calculado em (6.8). A partir da regra, seleciona-se este vértice, que pode ser o vértice de destino da aresta deletada (d) pela regra, ou um dos vértices de origem desta aresta ($p_{\mathbf{x}}$), ou ainda um dos vértices criados ($\mathbf{tg^R(lm[i])}$) pela regra. Os vértices de origem das arestas criadas são calculados em (6.9), da mesma forma que é calculado o vértice de destino.

Em (6.7) é descrita a definição do agente responsável pela criação dos objetos. Para instanciar esta fórmula, devem ser substituídas todas as ocorrências de $\mathbf{b}$ pelos identificadores dos vértices do hipergrafo tipo T.

Na fórmula (6.10) são definidas as variáveis utilizadas da fórmula (6.4) à (6.9), onde a variável $\mathbf{lpR_i}$ representa a lista de vértices de origem da i-ésima aresta criada pela mensagem nr.

Exemplo 25 (Tradução de GHBO para MBO-π) *Neste exemplo é apresentado a tradução de uma gramática de hipergrafos baseada em objetos do exemplo 12 (página 27) para um modelo baseados em objetos descrito em cálculo-π. Dada a GHBO $GH = (T, H, N, n)$, a tradução de GH para o seu MBO-π correspondente é dado pela tupla (I_π, A_π), onde*

$$I_\pi = (\nu c1, q1, q2, e1, m1, m2, m3, m4)(O_{circ}(c1)|O_{quad}(q1)|O_{quad}(q2)|O_{estr}(e1)| \\ @(ope1, m3, c1)|@(ope2, m4, e1, c1, q1)| \\ @(ope3, m1, q1, q2)|@(ope3, m2, q1, q1))$$

$A_\pi = Ms \cup Ob$, *onde*

$$Ms = \{@(ope1, id, d_{circ}) \stackrel{def}{=} (\nu m)\overline{d_{circ}}m.\overline{m}ope1.\overline{m}id, \\ @(ope2, id, d_{estr}, p_{1.circ}, p_{2.quad}) \stackrel{def}{=} (\nu m)\overline{d_{estr}}m.\overline{m}ope2.\overline{m}p_{1.circ}. \\ \overline{m}p_{2.quad}.\overline{m}id, \\ @(ope3, id, d_{quad}, p_{1.quad}) \stackrel{def}{=} (\nu m)\overline{d_{quad}}m.\overline{m}ope3.\overline{m}p_{1.quad}.\overline{m}id, \\ @(ope4, id, d_{quad}, p_{1.circ}) \stackrel{def}{=} (\nu m)\overline{d_{quad}}m.\overline{m}ope4.\overline{m}p_{1.circ}.\overline{m}id\}$$

$$Ob = \{O_{circ}(id) \stackrel{def}{=} id(m).(m(t).([t = ope1]M_{ope1}(m, id, t)) \mid O_{circ}(id)), \\ O_{estr}(id) \stackrel{def}{=} id(m).(m(t).([t = ope2]M_{ope2}(m, id, t)) \mid O_{estr}(id)), \\ O_{quad}(id) \stackrel{def}{=} id(m).(m(t).([t = ope3]M_{ope3}(m, id, t)+ \\ [t = ope4]M_{ope4}(m, id, t)) \mid O_{quad}(id))\}$$, *onde*

$$M_{ope1}(m,d,t) \stackrel{def}{=} R_{ope1.r1}(m,d,t,r1)$$
$$M_{ope2}(m,d,t) \stackrel{def}{=} R_{ope2.r2}(m,d,t,r2)$$
$$M_{ope3}(m,d,t) \stackrel{def}{=} R_{ope3.r3}(m,d,t,r3)$$
$$M_{ope4}(m,d,t) \stackrel{def}{=} R_{ope4.r4}(m,d,t,r4)$$

$$R_{ope1.r1}(m,d,t,nr) \stackrel{def}{=} m(id).\overline{nr}.\overline{t}id$$
$$R_{ope2.r2}(m,d,t,nr) \stackrel{def}{=} m(p_1).m(p_2).((\nu q2,m1)(CriaObj(q2,quad)|\ q2.m(id).\overline{nr}.\overline{t}id.@(ope4,m1,q2,p_1)))$$
$$R_{ope3.r3}(m,d,t,nr) \stackrel{def}{=} m(p_1).((\nu q3)(CriaObj(q3,quad)|\ q3.m(id).\overline{nr}.\overline{t}id))$$
$$R_{ope4.r4}(m,d,t,nr) \stackrel{def}{=} m(p_1).((\nu c2,m1)(CriaObj(c2,circ)|\ c2.m(id).\overline{nr}.\overline{t}id.@(ope1,m1,c2)))$$

$$CriaObj(id,t) \stackrel{def}{=} ([t=circ]\overline{id}.O_{circ}(id)+ [t=estr]\overline{id}.O_{estr}(id)+ [t=quad]\overline{id}.O_{quad}(id))$$

6.3 Compatibilidade Semântica

A comparação da semântica de uma GHBO e de um MBO-π é feita a partir da comparação dos caminhos gerados pelos respectivos STR. Um caminho é uma lista de transições onde o estado final de uma é o estado inicial da outra.

Definição 26 (Caminho) *Dado um STR $ST = (S, R, I, \rightarrow)$, o conjunto C dos caminhos de ST denotado por Caminho(ST) é definido por:*

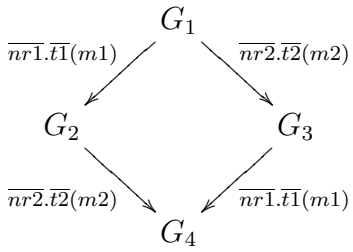
$$C = \{l | l \in (S \times R \times S)^* \wedge \forall t_j = (i_j, r_j, p_j), t_{j+1} = (i_{j+1}, r_{j+1}, p_{j+1}) \in l.p_j = i_{j+1} \wedge$$
$$j \in \{1..|l|\} \wedge i_1 = I\}$$

∇

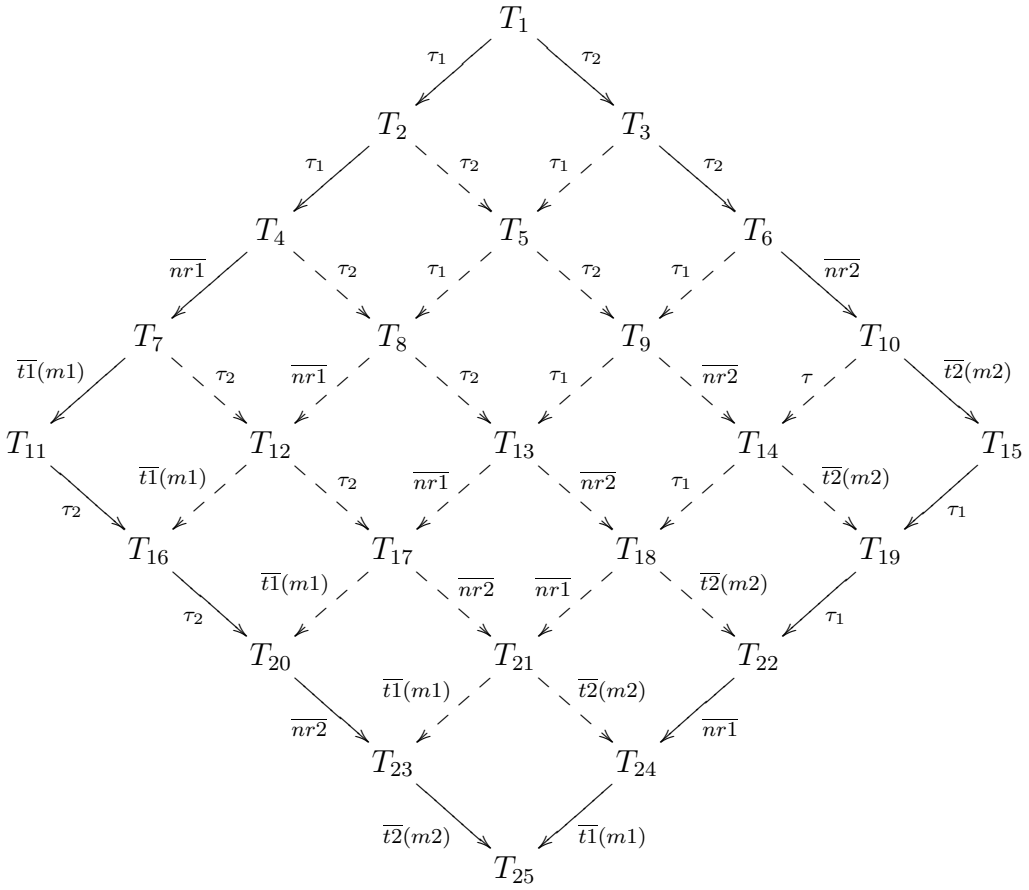
O tratamento de uma mensagem em uma GHBO é realizado com a aplicação de apenas uma regra, assim o sistema de transição de uma GHBO apresenta somente uma transição para cada mensagem tratada. Essas transições possuem rótulos do tipo $\overline{nr}.\overline{t}(id)$, que indica qual a mensagem (id) que foi tratada e por qual regra (nr).

Já em um MBO-π, o tratamento de uma mensagem requer uma série de sincronizações entre os agentes da mensagem e do objeto. Com isso, o sistema de transição de um MBO-π possui uma seqüência de transições para cada mensagem, sendo que os rótulos são uma série de τ, seguido pelo rótulo que identifica a regra aplicada ($\overline{nr}$, ação de saída livre) e pelo rótulo que identifica a mensagem tratada ($\overline{t}(id)$, ação de saída ligada). Em um MBO-π, as sincronizações resultantes do tratamento de duas mensagens em paralelo geram caminhos no STR onde as transições (do tratamento de cada mensagem) aparecem intercaladas e transições onde todas as transições de cada mensagem estão em seqüência finalizadas pelas transições com o identificador da regra e o da mensagem. O diagrama

abaixo mostra o STR para uma GHBO com duas mensagens $m1$ e $m2$ sendo tratadas em paralelo pelas regras $nr1$ e $nr2$, respectivamente.



O STR para um termo T_1 para as mesmas duas mensagens (e suas respectivas regras) pode ser visto no diagrama a seguir. Para simplificar, reduziu-se o número de τ para o tratamento de cada mensagem.



O paralelismo, nos dois STR é representado pela existência de dois caminhos onde o tratamento das mensagens $\overline{nr1}.\overline{t1}(m1)$ e $\overline{nr2}.\overline{t2}(m2)$ aparecem em ordens diferentes.

Em uma GHBO, o tratamento das mensagens é atômico, e no STR da sua tradução existem caminhos que representam esse tratamento atômico. No diagrama do STR do MBO-π, esses caminhos estão representados pelas setas contínuas. O que caracteriza estes caminhos é que todas as suas transições com rótulos do tipo $\overline{t}(id)$ (ação de saída ligada), têm como termo final um termo completo. Esses caminhos são ditos **caminhos completos**. Os demais caminhos (representados por setas tracejadas) não representam comportamentos existentes no STR da GHBO. Assim, a compatibilidade das semânticas é dada somente para os caminhos do STR do MBO-π que representam o tratamento atômico das mensagens, isto é, para os caminhos completos. Essa escolha gera questões que

devem ser levadas em conta na verificação de propriedades desejadas nas GHBOs através de verificadores para o cálculo-π. Essas questões serão discutidas nas conclusões.

Definição 27 (Caminho Completo) *Dado um MBO-π MB. Um caminho $c \in Caminho(SemPi(MB))$ é dito completo se $\forall t = (i, l, p) \in c$, onde l é uma ação de saída ligada, $p \in TermoCO_{MB}$. O conjunto de todos os caminhos completos de $Caminho(SemPi(MB))$ é denominado $CaminhoCO(MB)$.*

∇

Em um MBO-π há a criação de variáveis novas através do operador de restrição. Os nomes são novos no escopo em que são declarados e isso faz com que possam existir, no mesmo termo, mensagens e objetos com o mesmo nome. Desta forma, o mesmo identificador de mensagem pode aparecer em rótulos de diferentes transições de um caminho.

A figura 6.1 mostra a representação de um caminho do STR da GHBO do exemplo 12 (página 27) 6.1(a) e um caminho completo do STR de sua tradução para um MBO-π (exemplo 25, página 50) 6.1(b), onde aparece essa carcterística.

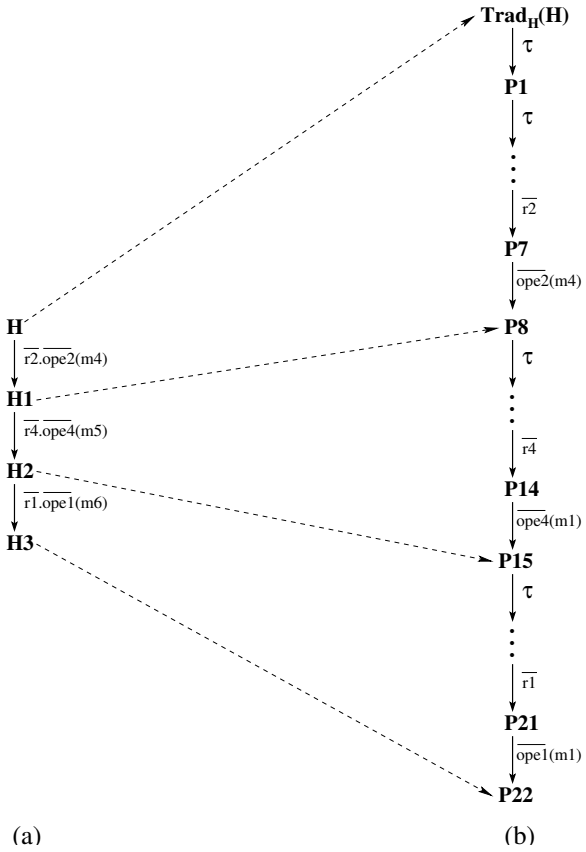


Figura 6.1: Exemplo de caminhos do STR de uma GHBO (a) e do STR de sua tradução para um MBO-π (b).

Como na tradução para uma GHBO cada nome representa um vértice ou hiperarco diferente, todos os nomes novos devem ser diferentes para ter uma representação na GHBO. Desta forma, os caminhos a serem considerados para a comparação entre as semânticas são caminhos estruturalmente congruentes aos caminhos completos do STR da tradução de GHBO, porém com nomes diferentes para todos os nomes novos. Este caminhos são denominados **caminhos BOs**.

Definição 28 (Caminho BO) *Dados um modelo MBO-π MB e um $c \in CaminhoCO(MB)$ completo. Um caminho BO equivalente a c é um caminho b onde:*

i) $|b| = |c|$*;*

ii) $\forall t = (i, l, p)$*, onde l é uma ação de saída ligada,* $p \in TermoBO_{MB}$*;*

iii) $\forall s_j = (i^s_j, l^s_j, p^s_j) \in c, t_j = (i^t_j, l^t_j, p^t_j) \in b . i^t_j \equiv i^s_j \wedge p^t_j \equiv p^s_j$*, com* $j = 1..|c|$ *e* $i^t_1 = i^s_1$*;*

iv) $\forall s = (i^s, l^s, p^s), t = (i^t, l^t, p^t) \in b$*, onde* $l^s = \overline{t^s}(id^s)$ *e* $l^t = \overline{t^t}(id^t)$ *são ações de saídas ligadas,* $id^s \neq id^t$*;*

O conjunto de todos os caminhos BOs de $Caminho(SemPi(MB))$ é denominado $CaminhoBO(MB)$.

∇

Teorema 29 *Dada uma GHBO GH e sua tradução $MB = (I_\pi, A_\pi)$, qualquer caminho completo de MB pode ser traduzido para um caminho BO de MB equivalente.*

Prova

Propriedade : $c \in CaminhoCO(MB) \rightarrow \exists b \in CaminhoBO(MB) \wedge c \approx b$

Pela definição de $CaminhoCO$ e $CaminhoBO$, temos:

$$c = I_\pi \xrightarrow{\tau} C_0 \xrightarrow{\overline{nr}} C_1 \xrightarrow{\overline{t}(id^c)} C_2 \xrightarrow{\tau} \cdots$$

e

$$b = I_\pi \xrightarrow{\tau} B_0 \xrightarrow{\overline{nr}} B_1 \xrightarrow{\overline{t}(id^b)} B_2 \xrightarrow{\tau} \cdots ,$$

onde $c \in CaminhoCO(MB)$, $b \in CaminhoBO(MB)$, $B_i \equiv C_i$, com $i = 0..|c| - 1$ e todos os nomes ligados de B_i são diferentes.

Isso se dá pois, pelas regras da semântica do cálculo-π e, pelo termo inicial de c e b ser o mesmo (I_π) e a primeira transição ser sempre τ, $C_0 \equiv B_0$, por ESTRUT *(tabela 5.4). Para todo B_i e C_i com $i = 0..|c| - 1$:*

Se $B_i \equiv C_i$ *e* $C_i \xrightarrow{l^c_i} C_{i+1}$ *e* $B_i \xrightarrow{l^b_i} B_{i+1}$ *então*

se $l^b_i = l^c_i$*, então* $C_{i+1} \equiv B_{i+1}$ *por* ESTRUT*;*

se $l^b_i \neq l^c_i$*, então* $l^b_i = \overline{t}(id^b_i)$ *e* $l^c_i = \overline{t}(id^c_i)$ *e* $id^b_i \neq id^c_i$*, assim:*

$B_{i+1} \equiv C_{i+1}$ *por* ESTRUT*, pois* $id^b_i \in bn(B_i)$ *e* $B_i \equiv C_i$ *são estruturalmente congruentes por α-conversão.*

$\surd$

Dados uma GHBO e MB (sua tradução), em $CaminhoBO(MB)$ há transições cujos rótulos são τ ou ações de saída livre, onde os estados finais destas transições não são termos BOs. Desta forma, estes termos não podem ser obtidos a partir da tradução de algum hipergrafo. Porém, os termos válidos (resultantes de transições com rótulos de ação de saída ligada) são termos BOs e a tradução dos hipergrafos do STR da GHBO resultam em termos equivalentes a eles. Assim, os estados equivalentes dos dois STRs da figura 6.1 são $Trad_H(H1) \equiv P8$, $Trad_H(H2) \equiv P15$, $Trad_H(H3) \equiv P22$. Esses termos são apenas equivalentes (e não os mesmos) devido a escolha dos nomes novos que é automática, assim os identificadores criados pela GHBO podem ser diferentes dos criados pela tradução de GHBO.

A comparação entre os dois STRs é feita baseada no conceito de bissimilaridade fraca. Aqui, são desconsideradas as transições τ. Note que não é realmente uma relação de bissimilaridade, pois esta é definida para dois agentes do cálculo-π. Além disso, as transições com rótulos de ação livre também são desconsideradas, embora a informação do seu rótulo seja mantida.

Para podermos comparar a semântica dos dois modelos, necessitamos então eliminar as transações τ dos caminhos do STR do MBO-π e as demais transições devem ser traduzidos para transições do STR das GHBO.

A função TC traduz um caminho do STR do MBO-π para um caminho STR das GHBO eliminando as transições τ e traduzindo os termos válidos para hipergrafos tipados.

Definição 30 (Tradução de um termo BO para um hipergrafo tipado) *Dados MBO-π MB, um termo BO B_π de MB e o hipergrafo tipo T_π de MB, onde*

$$B_\pi = (\nu ob_1, \cdots, ob_n, msg_1, \cdots, msg_m)(\prod_{i=1}^{n} O_{to_i}(ob_i) \mid \prod_{j=1}^{m} @(t_j, msg_j, d_j, p_j[1], \cdots, p_j[k])),$$

a tradução de B_π para um hipergrafo H^T é dada pela função $Trad_\pi^{T_\pi} : TermoBO_{MB} \rightarrow HiperGTip$ definida por:

$$Trad_\pi^{T_\pi}(B_\pi) = (H, t^H, T),$$

onde $H = (V_H, E_H, sc^H, tg^H)$, $V_H = \{ob_i | 0 < i \leq n\}$, $E_H = \{msg_j | 0 < j \leq m\}$,
$sc^H(msg_j) = \begin{cases} p_j & \text{se } k > 0 \\ \langle\,\rangle & \text{se } k = 0 \end{cases}$,
$tg^H(msg_j) = d_j$, $t^H(ob_i) = to_i$, $t^H(msg_j) = t_j$ *e* $T = T_\pi$

∇

Definição 31 (Tradução de Caminhos) *Dados uma GHBO GH e sua tradução $MB = Trad_{GH}(GH)$ e G sendo o tipo de MB, a função total $TC^G : CaminhoBO(MB) \rightarrow Caminho(SemGHBO(GH))$ é definida por:*

$$TC^G(c) = \begin{cases} \lambda & \text{se } c = \lambda \vee\ c = \langle (i,l,p) \rangle \vee \\ & \quad c = \langle (i,\tau,p1),(p1,\tau,p2) \rangle \\ \langle (Trad^G_\pi(i), l1.l2, Trad^G_\pi(p2)) \rangle & \text{se } c = \langle (i,l1,p1),(p1,l2,p2) \rangle \\ & \quad \wedge\, l1 \neq \tau\ \wedge l2 \neq \tau \\ TC^G(\langle (i,l2,p2) \rangle .R) & \text{se } c = \langle (i,\tau,p1),(p1,l2,p2) \rangle .R \\ TC^G(\langle (i,l1,p1),(p1,l2,p2) \rangle). & \text{se } c = \langle (i,l1,p1),(p1,l2,p2) \rangle .R \\ TC^G(R) & \quad \wedge\, l1 \neq \tau \wedge\ l2 \neq \tau \end{cases}$$

∇

Se uma GHBO é traduzida para um MBO-π o hipergrafo tipo desta gramática deve ser preservado, isto é, o hipergrafo tipo da tradução deve ser mesmo que o da GHBO. No lema 32 prova-se que isso é verdade, mostrando que o tipo obtido a partir das definições dos agentes do MBO-π é igual ao tipo da GHBO.

Lema 32 *Dada uma GHBO $GH = (T, I_{GH}, N, n)$, se $Trad_{GH}(GH) = MB$, então o tipo de MB é igual a T.*

Prova

Pela definição de $Trad_R(T, N, n)$, temos que $A_\pi = Ms \cup Ob$ e para cada aresta $t \in T$ há um elemento $@(t, id, d_{td}, \widetilde{p}_{tp}) \in Ms$ e para cada vértice $v \in T$ há um elemento $O_v(id) \in Ob$. As funções de origem e destino são representadas na definição de $@(t, id, d_{td}, \widetilde{p}_{tp})$ onde $tg^T(t) = td$ e $sc^T(t) = tp$.

Assim, a partir de A_π construído por $Trad_R(T, N, n)$, obtemos como tipo T_π de $Trad_{GH}(GH)$ o próprio T, pois:

- *V_{T_π} é composto por todos os to, tal que $O_{to}(id) \in A_\pi$. Pela definição de $Trad_R$, A_π possui um elemento para cada $v \in V_T$, assim $V_{T_\pi} = V_T$;*
- *De forma análoga, $E_{T_\pi} = E_T$;*
- *A função de destino $tg^{T_\pi}(t) = td$ é definida para todo $t \in E_{T_\pi}$, a partir dos elementos $@(t, id, d_{td}, \widetilde{p}_{tp}) \in A_\pi$. Pela definição de $Trad_R$, cada elemento $@(t, id, d_{td}, \widetilde{p}_{tp})$ identifica o vértice de destino (td) de t. Assim, para todo $t \in E_T$, $tg^{T_\pi}(t) = tg^T(t)$;*
- *De forma análoga, $sc^{T_\pi}(t) = sc^T(t)$, para todo $t \in E_T$.*

$\surd$

Se uma GHBO é traduzida para um MBO-π, a tradução do termo inicial de MBO-π para um hipergrafo tipado deve ser igual ao hipergrafo inicial da GHBO, pois a tradução do estado inicial um modelo para o outro deve preservar estado inicial original.

Lema 33 *Dada uma GHBO $GH = (T, I_{GH}, N, n)$, se $Trad_{GH}(GH) = (I_\pi, A_\pi)$, então $Trad^T_\pi(I_\pi) = I_{GH}$.*

Prova

Pelo lema 32, T é o hipergrafo tipo do MBO-π de I_π.

Pela definição de $Trad_H(I_{GH})$, obtém-se um termo BO I_π com um agente $O_t(id)$ para cada $id \in V_{I_{GH}}$, com $t^{I_{GH}}(id) = t$, e um agente $@(t, id, d, p[1], \cdots, p[n])$ (se $n > 0$) ou $@(t, id, d)$ (se $n = 0$) para cada $id \in E_{I_{GH}}$, com $t^{I_{GH}}(id) = t$, $tg^{I_{GH}}(id) = d$ e $sc^{I_{GH}}(id) = p$, se $n > 0$, ou $sc^{I_{GH}}(id) = \langle\,\rangle$, se $n = 0$.

Pela definição de $Trad^T_\pi(I_\pi)$, obtém-se um hipergrafo (H, t^H, T) que é igual a I_{GH}, pois:

- *V_H contém todos os nomes de objetos de I_π, que por $Trad_H$, possui um objeto para cada vértice de I_{GH}. Assim $V_H = V_{I_{GH}}$;*
- *De forma análoga, $E_H = E_{I_{GH}}$;*
- *A definição da função de destino $tg^H(m) = d$ é obtida a partir de cada agente $@(t, m, d, p[1], \cdots, p[n])$ ou $@(t, m, d)$ em I_π. Por $Trad_H$, d representa o vértice de destino de m em I_{GH}, assim $tg^H(m) = tg^{I_{GH}}(m)$;*
- *De forma análoga, $sc^H(m) = sc^{I_{GH}}(m)$;*
- *A função $t^H(o) = to$ para os vértices, é obtida a partir de cada agente $O_{to}(o)$ em I_π. Por $Trad_H$, to representa o tipo do vértice o em I_{GH}, Assim $t^H(o) = t^{I_{GH}}(o)$;*
- *De forma análoga, $t^H(m) = t^{I_{GH}}(m)$, para as arestas;*

$\surd$

Para provar que a tradução $Trad_{GH}$ preserva a semântica das GHBO, demonstrou-se que todos os caminhos que estão no STR da GHBO também estão no conjunto de caminhos completos da sua tradução e vice-versa.

Como foi dito na seção 4.2, existem em $Caminho(SemGHBO(GH))$ subconjuntos de caminhos isomórficos, que representam o mesmo comportamento. Assim, no teorema 34, é demonstrado que para cada caminho que está em $Caminho(SemGHBO(GH))$, existe um caminho em $CaminhoCO(Trad_{GH}(GH))$, que traduzido é isomórfico ao primeiro.

Notação: Dados dois objetos A e B. Se A e B são isomórficos, pode-se escrever $A \approx B$.

Teorema 34 *Seja $GH = (T, I_{GH}, N, n)$ e $Trad_{GH}(GH) = (I_\pi, A_\pi) = MB$. Se $c \in Caminho(SemGHBO(GH))$, então $\exists t \in CaminhoCO(MB)$ tal que $TC^T(t) \approx c$.*

Prova

Como, pelo teorema 29, todo caminho em $CaminhoCO(MB)$ possui um caminho equivalente em $CaminhoBO(MB)$, a prova é feita para todo caminho em $CaminhoBO(MB)$.

Propriedade : $c \in Caminho(SemGHBO(GH)) \longrightarrow$
$\exists t \in CaminhoBO(MB) \wedge TC^T(t) \approx c$

A prova será por indução no tamanho do caminho c.

Base

a) Para $|c| = 0$:

$c = \lambda$

$TC^T(\lambda) = \lambda$ *e* $\lambda \in CaminhoBO(MB)$, *assim* $TC^T(\lambda) = c$;

b) Para $|c| = 1$:

Seja $c = \langle (I_{GH}, lab_1, G_1) \rangle$.

Como $(I_{GH}, lab_1, G_1) \in Caminho(SemGHBO(GH))$ *então existe a seguinte derivação em* $SDer_{GH}$:

$$\begin{array}{ccc} L^T & \xrightarrow{r} & R^T \\ {\scriptstyle m}\downarrow & PO & \downarrow{\scriptstyle m'} \\ I_{GH} & \xrightarrow[r']{} & G_1 \end{array}$$

onde $lab_1 = nr.\overline{t}(id)$, $n(nr) = r : L^T \longrightarrow R^T$, $tg^{I_{GH}}(id) = v$, $sc^{I_{GH}}(id) = p$, $t = t^L(id)$, $tv = t^{I_{GH}}(v)$, $e \in E_L$, $tg^L(e) = ve$, $sc^L(e) = pe$, $m(e) = id$, $m(ve) = v$, $m(pe[i]) = p[i]$, $i \in \{1..|pe|\}$ *e* G_1 *é o* pushout *de* m *e* r *em* ***THGrafoP(T)***.

Por definição, $Trad_H(I_{GH}) = I_\pi$ *e* $\exists x, y \in a_{I_\pi}$ *tal que* $x = O_{tv}(v)$ *e* $y = @(t, id, v, p[1], \cdots, p[|p|])$

Pela definição de $Trad_R(T, N, n)$:

- $@(t, id, v, p[1], \cdots, p[|p|]) = (\nu m)\overline{v}m.\overline{mt}.\bigodot_{i=1}^{|p|} \overline{m}p[i].\overline{m}id$
- $O_{tv}(v) = v(m).(m(t).\sum_{tm \in TM}[t = tm]M_{tm}(m, v, t)|O_{tv}(v))$, *onde* TM *é o conjunto de tipos de mensagens do sistema que o objeto tipo* tv *trata.*
- $M_t(m, v, t) = \sum_{rm \in RM_t} R_{t.rm}(m, v, t, rm)$, *onde* RM_t *é o conjunto de regras que tratam mensagens do tipo* t, *no qual* $nr \in RM_t$.

  Observação: *somente o agente* $M_t(m, v, t)$ *foi descrito pois é o agente selecionado para o tratamento da mensagem do tipo* t. *A descrição dos demais agentes (agentes que tratam os outros tipos de mensagens para o objeto* v*) foi omitida.*
- $R_{t.nr}(m, v, t, nr) = \bigodot_{i=1}^{|p|} m(p_i).((\nu id_1, \cdots, id_s, msg_1, \cdots, msg_j)$
  $(\prod_{i=1}^{s} CriaObj(id_i, to_i)|\prod_{i=1}^{s} id_i.m(id).\overline{nr}.\overline{t}id.$
  $\prod_{i=1}^{j} @(t_i, msg_i, d_i, p_i[1], \cdots, p_i[b_i])))$, *onde* s *e* j *são, respectivamente, os números de objetos e mensagens criadas pelo tratamento da mensagem do tipo* t *pela regra* nr.

  Observação: *de forma análoga ao agente anterior, a descrição dos agentes das demais regras (exceto* nr*) foi omitida.*
- $CriaObj(id, t) = \sum_{b \in V_T}([t = b]\overline{id}.O_b(id))$, *onde* V_T *é o conjunto de tipos de objetos do sistema.*

Seja R *é um termo BO que contém os agentes dos objetos e mensagens que não são afetados pelo tratamento da mensagem* id. *Pelas regras da semântica operacional do cálculo* π *e pela definição de* $CaminhoBO$, *um caminho de* $CaminhoBO(MB)$ *é*

$$t = I_\pi \xrightarrow{\tau} T_0 \xrightarrow{\tau} T_1 \xrightarrow{\tau^*} T_{1+|p|} \xrightarrow{\tau^*} T_{1+|p|+|s|} \xrightarrow{\tau} T_{2+|p|+|s|}$$

$$\xrightarrow{\overline{nr}} T_{3+|p|+|s|} \xrightarrow{\overline{t}(id)} T_{4+|p|+|s|} = P,$$

onde:

a) $I_\pi = R \mid (\nu v)(\mathbf{v}(\mathbf{m}).(m(t).\sum_{tm \in TM}[t = tm]M_{tm}(m, v, t) \mid O_{tv}(v)) \mid$
$(\nu id)((\nu m)\overline{\mathbf{v}}\mathbf{m}.\overline{m}t.\bigodot_{i=1}^{|p|}\overline{m}p[i].\overline{m}id))$

Os canais dos agentes do objeto e da mensagem que sincronizam estão destacados em negrito. Essa sincronização gera em uma transição τ, resultando no termo T_0. Essa sincronização, ainda, resulta na expansão do escopo do canal m que passará a ser visto também pelo agente do objeto.

b) $T_0 = R \mid (\nu v, m)(O_{tv}(v) \mid \mathbf{m}(\mathbf{t}).\sum_{tm \in TM}[t = tm]M_{tm}(m, v, t) \mid$
$(\nu id)(\overline{\mathbf{m}}\mathbf{t}.\bigodot_{i=1}^{|p|}\overline{m}p[i].\overline{m}id))$

Neste termo a sincronização que ocorre resulta no envio do tipo t da mensagem para o agente do objeto, gerando o termo T_1.

c) $T_1 = R \mid (\nu v, m)$
$(O_{tv}(v) \mid$
$\bigodot_{\mathbf{i=1}}^{\mathbf{|p|}}\mathbf{m}(\mathbf{p_i}).((\nu id_1, \cdots, id_s, msg_1, \cdots, msg_j)$
$(\prod_{i=1}^{s} CriaObj(id_i, to_i)|$
$\prod_{i=1}^{s} id_i.m(id).\overline{nr}.\overline{t}id.$
$\prod_{i=1}^{j} @(t_i, msg_i, d_i, p_i[1], \cdots, p_i[b_i]))) \mid$
$(\nu id)(\bigodot_{\mathbf{i=1}}^{\mathbf{|p|}}\overline{\mathbf{m}}\mathbf{p}[\mathbf{i}].\overline{m}id))$

Este termo deriva de uma série de reescritas do termo resultante da transição anterior. O termo resultante apresenta a seguinte forma:

$$R \mid (\nu v, m)(O_{tv}(v) \mid \sum_{tm \in TM}[t = tm]M_{tm}(m, v, t) \mid (\nu id)(\bigodot_{i=1}^{|p|}\overline{m}p[i].\overline{m}id))$$

Nesse termo, o agente do objeto recebeu o tipo t da mensagem que irá tratar. Como o conjunto TM possui todos os tipos de mensagens que esse objeto pode tratar, então $t \in TM$. Desta forma, uma das condições do somatório ($\sum_{tm \in TM}[t = tm]M_{tm}(m, v, t)$) é satisfeita, isto é, $[t = t]$. Assim, o agente do objeto passa a comportar-se como $M_t(m, v, t)$. O termo, portanto, pode ser reescrito da seguinte forma:

$$R \mid (\nu v, m)(O_{tv}(v) \mid \sum_{rm \in RM_t} R_{t.rm}(m, v, t, rm) \mid (\nu id)(\bigodot_{i=1}^{|p|}\overline{m}p[i].\overline{m}id))$$

O termo, agora, apresenta uma escolha não-determinística ($\sum_{rm \in RM_t} R_{t.rm}(m, v)$), porém como o caminho do STR de MB que está sendo considerado é aquele em que a regra que descreve o tratamento da mensagem é a nr, o agente selecionado foi o $R_{t.nr}(m, v, t, nr)$ resultando no termo T_1. No termo T_1 ocorrem $|p|$ sincronizações, gerando uma lista de τ, denotada por τ^*, resultando no termo $T_{1+|p|}$.

d) $T_{1+|p|} = R \mid (\nu v, m)(O_{tv}(v) \mid$
$((\nu id_1, \cdots, id_s, msg_1, \cdots, msg_j)$
$(\prod_{i=1}^{s}(\overline{\mathbf{id_i}}.O_{to_i}(id_i))|$
$\prod_{\mathbf{i=1}}^{\mathbf{s}} \mathbf{id_i}.m(id).\overline{nr}.\overline{t}id.$
$\prod_{i=1}^{j} @(t_i, msg_i, d_i, p_i[1], \cdots, p_i[b_i]))) \mid$
$(\nu id)(\overline{m}id))$

O termo $T_{1+|p|}$ é derivado de uma série de reescritas do termo resultante das sincronizações anteriores. Esse termo resultante tem a seguinte forma:

$R \mid (\nu v, m)(O_{tv}(v) \mid$
$((\nu id_1, \cdots, id_s, msg_1, \cdots, msg_j)$
$(\prod_{i=1}^{s} CriaObj(id_i, to_i)|$
$\prod_{i=1}^{s} id_i.m(id).\overline{nr}.\overline{t}id.$
$\prod_{i=1}^{j} @(t_i, msg_i, d_i, p_i[1], \cdots, p_i[b_i]))) \mid$
$(\nu id)(\overline{m}id))$

Este termo pode ser reescrito da forma que segue:

$R \mid (\nu v, m)(O_{tv}(v) \mid$
$((\nu id_1, \cdots, id_s, msg_1, \cdots, msg_j)$
$(\prod_{i=1}^{s}(\sum_{b \in V_T}([t = b]\overline{id_i}.O_b(id_i)))|$
$\prod_{i=1}^{s} id_i.m(id).\overline{nr}.\overline{t}id.$
$\prod_{i=1}^{j} @(t_i, msg_i, d_i, p_i[1], \cdots, p_i[b_i]))) \mid$
$(\nu id)(\overline{m}id))$

Neste termo há uma escolha que é feita através do tipo to_i passado como parâmetro em $CriaObjto(id_i, to_i)$. Como to_i é um tipo de objeto do sistema, $to_i \in TO$. Desta forma, uma das condições da escolha é satisfeita ($[to_i = to_i]$) e o agente selecionado é o agente $\overline{id_i}.O_{to_i}(id_i)$. O termo resultante da reescrita deste termo é $T_{1+|p|}$.

As sincronizações que ocorrem no termo $T_{1+|p|}$, são entre os subtermos do agente do objeto. Essas sincronizações resultam no termo $T_{1+|p|+|s|}$, onde s é o número de objetos criados e, conseqüentemente, o número de sincronizações realizadas.

e) $T_{1+|p|+|s|} =$

$R \mid$
$(\nu v, m)(O_{tv}(v) \mid$
$((\nu id_1, \cdots, id_s, msg_1, \cdots, msg_j)$
$(O_{to_1}(id_1) \mid \cdots \mid$
$O_{to_s}(id_s)|$
$\mathbf{m}(\mathbf{id}).\overline{nr}.\overline{t}id. \prod_{i=1}^{j} @(t_i, msg_i, d_i, p_i[1], \cdots, p_i[b_i]))) \mid$
$(\nu id)(\overline{\mathbf{mid}}))$

Neste termo há uma sincronização entre o agente da mensagem e do objeto, onde o objeto recebe o identificador id da mensagem. O termo resultante desta sincronização é o termo $T_{2+|p|+|s|}$. Em $T_{2+|p|+|s|}$, o canal local m não é mais utilizado, assim ele pode ser eliminado e o canal local id, que é enviado para o objeto, tem seu escopo estendido e passa a ser visto pelo objeto também.

f) $T_{2+|p|+|s|} =$

$$
\begin{aligned}
&R \mid \\
&(\nu v)(O_{tv}(v) \mid ((\nu id_1, \cdots, id_s, msg_1, \cdots, msg_j) \\
&\qquad\qquad (O_{to_1}(id_1) \mid \cdots \mid \\
&\qquad\qquad O_{to_s}(id_s)\mid \\
&\qquad\qquad (\nu id)(\overline{\mathbf{nr}}.\overline{t}id.\textstyle\prod_{i=1}^{j} @(t_i, msg_i, d_i, p_i[1], \cdots, p_i[b_i])))))
\end{aligned}
$$

Este termo realiza a ação $\overline{nr}$, sincronizando com o ambiente. O termo resultante desta transição é o termo $T_{3+|p|+|s|}$.

g) $T_{3+|p|+|s|} =$

$$
\begin{aligned}
&R \mid \\
&(\nu v)(O_{tv}(v) \mid ((\nu id_1, \cdots, id_s, msg_1, \cdots, msg_j) \\
&\qquad\qquad (O_{to_1}(id_1) \mid \cdots \mid \\
&\qquad\qquad O_{to_s}(id_s)\mid \\
&\qquad\qquad (\nu id)(\overline{\mathbf{t}}\mathbf{id}.\textstyle\prod_{i=1}^{j} @(t_i, msg_i, d_i, p_i[1], \cdots, p_i[b_i])))))
\end{aligned}
$$

Este termo pode realizar a ação $\overline{t}(id)$, onde o identificador id é enviado para o ambiente. Desta forma o escopo de id é estendido para todo o ambiente, deixando de ser local. Essa ação gera uma transição o rótulo $\overline{t}(id)$ e resulta no termo $T_{4+|p|+|s|}$.

h) $T_{4+|p|+|s|} = R \mid (\nu v)(O_{tv}(v) \mid ((\nu id_1, \cdots, id_s, msg_1, \cdots, msg_j)$

$$
\begin{aligned}
&\qquad\qquad (O_{to_1}(id_1) \mid \cdots \mid O_{to_s}(id_s)\mid \\
&\qquad\qquad @(t_1, msg_1, d_1, p_1[1], \cdots, p_1[b_1]) \mid \cdots \mid \\
&\qquad\qquad @(t_j, msg_j, d_j, p_j[1], \cdots, p_j[b_j])))))
\end{aligned}
$$

Como $I_{GH} = (V, E, sc, tg)$, onde $V = \{v, v_1, \cdots, v_n\}$, $E = \{id, a_1, \cdots, a_m\}$, $n = |V| - 1$, $m = |E| - 1$, $tg(a_i) = d_{a_i}$, $sc(a_i) = p_{a_i}$ e $t^{I_{GH}}(a_i) = t_{a_i}$, com $0 < i \leq m$, e $t^{I_{GH}}(v_k) = to_{v_k}$, com $0 < k \leq n$, então $G1$, por sua construção, possui um elemento para cada elementos criado por r (vértices: $\{id_1, \ldots, id_s\}$ e hiperarcos: $\{msg_1, \ldots, msg_j\}$), um elemento para cada elemento de V e um elemento para cada elemento de $\{a_1, \ldots, a_m\}$, além disso, os elementos identificados por m estão identificados em $G1$. Como $G1$ é o pushout *de r e m em **THGrafoP(T)** as funções de tipo, origem e destino são respeitadas. Desta forma, tem-se:*

$$
\begin{aligned}
Trad_H(G1) = (\nu v, v_1, \ldots, v_n, a_1, \ldots, a_m, id_1, \ldots, id_s, msg_1, \ldots, msg_j) \\
O_{tv}(v) \mid O_{to_1}(id_1) \mid \cdots \mid O_{to_s}(id_s) \mid O_{to_{v_1}}(v_1) \mid \cdots \mid O_{to_{v_n}}(v_n) \mid \\
@(t_{a_1}, a_1, d_{a_1}, p_{a_1}[1], \ldots, p_{a_1}[|p_{a_1}|]) \mid \cdots \mid \\
@(t_{a_m}, a_m, d_{a_m}, p_{a_m}[1], \ldots, p_{a_m}[|p_{a_m}|]) \mid \\
@(t_1, msg_1, d_1, p_1[1], \ldots, p_1[|p_1|]) \mid \cdots \mid \\
@(t_j, msg_j, d_j, p_j[1], \ldots, p_j[|p_j|]) \equiv P, \textit{ com}
\end{aligned}
$$

$t^{G1}(v_i) = to_{v_i}$, para $v_i \in \{v_1, \ldots, v_n\}$; $t^{G1}(id_i) = to_i$, para $id_i \in \{id_1, \ldots, id_s\}$; $t^{G1}(a_i) = t_{a_i}$, $tg^{G1}(a_i) = d_{a_i}$ e $sc^{G1}(a_i) = p_{a_i}$, para $a_i \in \{a_1, \ldots, a_m\}$; $t^{G1}(msg_i) = t_i$, $tg^{G1}(msg_i) = d_i$ e $sc^{G1}(msg_i) = p_i$, para $msg_i \in \{msg_1, \ldots, msg_j\}$;

$TC^T(t) = \langle (Trad^T_\pi(I_\pi), \overline{nr}.\overline{t}(id), Trad^T_\pi(Trad_H(G_1)))\rangle$, pois

$TC^T(t) = TC^T(\langle(I_\pi, \tau, T_1)\rangle.t_{R1})$

$\ldots = TC^T(\langle(I_\pi, \tau, T_{1+|p|})\rangle.t_{R2})$

$\ldots = TC^T(\langle(I_\pi, \tau, T_{2+|p|+|s|})\rangle.t_{R3})$

$= TC^T(\langle(I_\pi, \overline{nr}, T_{3+|p|+|s|}), (T_{3+|p|+|s|}, \overline{t}(id), T_{4+|p|+|s|})\rangle)$

$= \langle(Trad^T_\pi(I_\pi), \overline{nr}.\overline{t}(id), Trad^T_\pi(T_{4+|p|+|s|}))\rangle$, *onde*

$t = \langle(I_\pi, \tau, T_0), (T_0, \tau, T_1)\rangle.t_{R1}$

$t_{R1} = \langle(T_1, \tau, T_2), \ldots, (T_{|p|}, \tau, T_{1+|p|})\rangle.t_{R2}$

$t_{R2} = \langle(T_{1+|p|}, \tau, T_{2+|p|}), \ldots, (T_{1+|p|+|s|}, \tau, T_{2+|p|+|s|})\rangle.t_{R3}$

$t_{R3} = \langle(T_{2+|p|+|s|}, \overline{nr}, T_{3+|p|+|s|}), (T_{3+|p|+|s|}, \overline{t}(id), T_{4+|p|+|s|})\rangle$

Como $Trad_H(G1) \equiv T_{4+|p|+|s|}$, *então* $Trad_\pi(Trad_H(G1)) \approx Trad^T_\pi(T_{4+|p|+|s|})$; *e pelo lema 33,* $Trad^T_\pi(Trad_H(I_{GH})) = I_{GH}$ *e* $Trad^T_\pi(Trad_H(G_1)) = G_1$ *temos* $TC^T(t) = \langle(I_{GH}, \overline{nr}.\overline{t}(id), G_1)\rangle \approx c$

Hipótese

Para $|c| = n$ *e* $n > 1$:
$c \in Caminho(SemGHBO(GH)) \longrightarrow$
$\exists t \in CaminhoBO(MB) \wedge TC^T(t) = c$,
onde $c = \langle(I_{GH}, lab_1, G_1), \ldots, (G_{n-1}, lab_n, G_n)\rangle$

Passo

Sejam $|c| = n + 1$ *e*
$c+ = \langle(I_{GH}, lab_1, G_1), \ldots, (G_{n-1}, lab_n, G_n), (G_n, \overline{nr}.\overline{t}(id), G_{n+1})\rangle$

Pela hipótese, se há um caminho c *do estado* I_{GH} *até* G_n, *existe um caminho* t *de* $Trad_H(I_{GH})$ *até* $Trad_H(G_n)$, *tal que* $TC^T(t) = c$.

Se for possível em G_n *mais uma derivação através de uma regra* nr *e uma ocorrência* m, *resultando em um grafo* G_{n+1}, *há em* $Trad_H(G_n)$ *um agente da mensagem que será tratada por* nr *e um objeto que receberá essa mensagem. Assim esses agentes sincronizam gerando um caminho partindo de* $Trad_H(G_n)$ *com uma seqüência de* τ, *seguido de* $\overline{nr}$ *e* $\overline{t}(id)$, *resultando em um termo* $P \equiv Trad_H(G_{n+1})$. *A prova segue de forma análoga à prova da base com* $|c| = 1$.

$\surd$

No teorema 34 provou-se que todos os caminhos no STR das GHBO estão no conjunto de caminhos completos de sua tradução. A seguir será provado o inverso. No teorema 35, é demonstrado que para todos os caminhos em $CaminhoCO(Trad_{GH}(GH))$, há a tradução deste caminho em $Caminho(SemGHBO(GH))$.

Teorema 35 *Seja* $GH = (T, I_{GH}, N, n)$, $Trad_{GH}(GH) = (I_\pi, A_\pi) = MB$ *e* T *o hipergrafo tipo de* MB. *Se* $t \in CaminhoCO(MB)$, *então* $TC^T(t) \in Caminho(SemGHBO(GH))$.

Prova

Como, pelo teorema 29, todo caminho em $CaminhoCO(MB)$ possui um caminho equivalente em $CaminhoBO(MB)$, a prova é feita para todo caminho em $CaminhoBO(MB)$.

Propriedade : $t \in CaminhoBO(MB) \longrightarrow TC^T(t) \in Caminho(SemGHBO(GH))$

Cada caminho não vazio de $CaminhoBO(MB)$ tem o formato

$$I_\pi \xrightarrow{\tau^*} T_1 \xrightarrow{\overline{nr_1}} T_2 \xrightarrow{\overline{t_1}(id_1)} P_1 \xrightarrow{\tau^*} T_3 \xrightarrow{\overline{nr_2}} T_4 \xrightarrow{\overline{t_2}(id_2)} P_2 \xrightarrow{\tau^*} \cdots T_m \xrightarrow{\overline{nr_n}} T_{m+1} \xrightarrow{\overline{t_n}(id_n)} P_n$$

Se o número de transições de I_π até P_1 é $n1$, temos que a tradução TC de todos subcaminhos de t de tamanho menor que $n1$ é λ (pois todos os rótulos são diferentes de saída ligada). Assim, o relevante na tradução do caminho t para um caminho de $Caminho(SemGHBO(GH))$ são os rótulos de saída ligada, e portanto, a prova será por indução no número destes rótulos no caminho t, denominado de $n_\tau(t)$.

Base

a) Para $n_\tau(t) = 0$:

Neste caso, o caminho t; é uma seqüência de transições rotuladas por τ ou $\overline{nr}$. De acordo com a definição de TC (definição 31), a tradução será $TC^T(t) = \lambda$ e $\lambda \in Caminho(SemGHBO(GH))$.

b) Para $n_\tau(t) = 1$:

Neste caso o caminho t tem o formato

$$t = \langle (I_\pi, \tau, T_0), (T_0, \tau, T_1), \ldots, (T_{n-1}, \overline{nr}, T_{n-1}), (T_n, lab_1, P) \rangle \text{ ou}$$

$$t = \langle (I_\pi, \tau, T_0), (T_0, \tau, T_1), \ldots, (T_{n-1}, \overline{nr}, T_{n-1}), (T_n, lab_1, P) \rangle.R,$$

onde R é um subcaminho que contém apenas transições com rótulos τ ou $\overline{nr}$.

De acordo com a definição de TC, obtem-se a seguinte tradução (para os dois casos):

$TC^T(t) = \langle (Trad^T_\pi(I_\pi), \overline{nr}.lab_1, Trad^T_\pi(P)) \rangle$*;*

Se existe t então existem em I_π os agentes $O_{tv}(v)$ e $@(t, id, v, p[1], \ldots, p[k])$, onde $lab_1 = \overline{t}(id)$ e k é o número de parâmetros de mensagens do tipo t, que é igual à $|sc^T(t)|$ (pela definição de $Trad_H$). Pelo conjunto de definições A_π e a definição de $Trad_R(T, n, N)$:

- $@(t, id, v, p[1], \cdots, p[k]) = (\nu m)\overline{v}m.\overline{mt}.\bigodot_{i=1}^{k} \overline{m}p[i].\overline{m}id$
- $O_{tv}(v) = v(m).(m(t).\sum_{tm \in TM}[t = tm]M_{tm}(m, v, t) \mid O_{tv}(v))$*, onde TM é o conjunto de tipos de mensagens do sistema que o objeto tipo tv trata.*
- $M_t(m, v, t) = \sum_{rm \in RM_t} R_{t.rm}(m, v, t, rm)$*, onde RM_t é o conjunto de regras que tratam mensagens do tipo t.*

- $R_{t.nr}(m,v,t,nr) = \bigodot_{i=1}^{k} m(p_i).((\nu id_1,\cdots,id_s,msg_1,\cdots,msg_j)$
$(\prod_{i=1}^{s} CriaObj(id_i,to_i)|\prod_{i=1}^{s} id_i.m(id).\overline{nr}.\overline{t}id.$
$\prod_{i=1}^{j} @(t_i,msg_i,d_i,p_i[1],\cdots,p_i[b_i])))$, *onde* s *e* j *são, respectivamente, os números de objetos e mensagens criadas pelo tratamento da mensagem do tipo* t *pela regra* nr.
- $CriaObj(id,t) = \sum_{b\in V_T}([t=b]\overline{id}.O_b(id))$, *onde* V_T *é o conjunto de tipos de objetos do sistema.*

Seja R *é um termo BO que contém os agentes dos objetos e mensagens que não são afetados pelo tratamento da mensagem* id, *e* $k = |sc^T(t)|$. *Pelas regras da semântica operacional do cálculo* π *e pela definição de* $CaminhoBO$, *um caminho de* $CaminhoBO(MB)$ *é*

$$t = I_\pi \xrightarrow{\tau} T_0 \xrightarrow{\tau} T_1 \xrightarrow{\tau^*} T_{1+|k|} \xrightarrow{\tau^*} T_{1+|k|+|s|} \xrightarrow{\tau} T_{2+|k|+|s|}$$
$$\xrightarrow{\overline{nr}} T_{3+|k|+|s|} \xrightarrow{\overline{t}(id)} T_{4+|k|+|s|} = P,$$

onde:

a) $I_\pi = R \mid (\nu v)(\mathbf{v}(\mathbf{m}).(m(t).\sum_{tm\in TM}[t=tm]M_{tm}(m,v,t) \mid O_{tv}(v)) \mid$
$(\nu id)((\nu m)\overline{\mathbf{v}}\mathbf{m}.\overline{m}t.\bigodot_{i=1}^{|k|}\overline{m}p[i].\overline{m}id))$

b) $T_0 = R \mid (\nu v,m)(O_{tv}(v) \mid \mathbf{m}(\mathbf{t}).\sum_{tm\in TM}[t=tm]M_{tm}(m,v,t) \mid$
$(\nu id)(\overline{\mathbf{m}}\mathbf{t}.\bigodot_{i=1}^{|k|}\overline{m}p[i].\overline{m}id))$

c) $T_1 = R \mid (\nu v,m)$
$(O_{tv}(v) \mid$
$\bigodot_{\mathbf{i=1}}^{\mathbf{|k|}}\mathbf{m}(\mathbf{p_i}).((\nu id_1,\cdots,id_s,msg_1,\cdots,msg_j)$
$(\prod_{i=1}^{s} CriaObj(id_i,to_i)|$
$\prod_{i=1}^{s} id_i.m(id).\overline{nr}.\overline{t}id.$
$\prod_{i=1}^{j} @(t_i,msg_i,d_i,p_i[1],\cdots,p_i[b_i]))) \mid$
$(\nu id)(\bigodot_{\mathbf{i=1}}^{\mathbf{|k|}}\overline{\mathbf{m}}\mathbf{p}[\mathbf{i}].\overline{m}id))$

d) $T_{1+|k|} = R \mid (\nu v,m)(O_{tv}(v) \mid$
$((\nu id_1,\cdots,id_s,msg_1,\cdots,msg_j)$
$(\prod_{i=1}^{s}(\overline{\mathbf{id_i}}.O_{to_i}(id_i))|$
$\prod_{\mathbf{i=1}}^{\mathbf{s}}\mathbf{id_i}.m(id).\overline{nr}.\overline{t}id.$
$\prod_{i=1}^{j} @(t_i,msg_i,d_i,p_i[1],\cdots,p_i[b_i]))) \mid$
$(\nu id)(\overline{m}id))$

e) $T_{1+|k|+|s|} =$
$R \mid$
$(\nu v,m)(O_{tv}(v) \mid$
$((\nu id_1,\cdots,id_s,msg_1,\cdots,msg_j)$
$(O_{to_1}(id_1) \mid \cdots \mid$
$O_{to_s}(id_s)|$
$\mathbf{m}(\mathbf{id}).\overline{nr}.\overline{t}id.$
$\prod_{i=1}^{j} @(t_i,msg_i,d_i,p_i[1],\cdots,p_i[b_i]))) \mid$
$(\nu id)(\overline{\mathbf{m}}\mathbf{id}))$

f) $T_{2+|k|+|s|} =$

$$
\begin{array}{l}
R \mid \\
(\nu v)(O_{tv}(v) \mid \\
\qquad ((\nu id_1, \cdots, id_s, msg_1, \cdots, msg_j) \\
\qquad\quad (O_{to_1}(id_1) \mid \cdots \mid \\
\qquad\quad O_{to_s}(id_s)\mid \\
\qquad\quad (\nu id)(\overline{\mathbf{nr}}.\overline{t}id.\prod_{i=1}^{j} @(t_i, msg_i, d_i, p_i[1], \cdots, p_i[b_i])))))
\end{array}
$$

g) $T_{3+|k|+|s|} =$

$$
\begin{array}{l}
R \mid \\
(\nu v)(O_{tv}(v) \mid \\
\qquad ((\nu id_1, \cdots, id_s, msg_1, \cdots, msg_j) \\
\qquad\quad (O_{to_1}(id_1) \mid \cdots \mid \\
\qquad\quad O_{to_s}(id_s)\mid \\
\qquad\quad (\nu id)(\overline{\mathbf{t}}\mathbf{id}.\prod_{i=1}^{j} @(t_i, msg_i, d_i, p_i[1], \cdots, p_i[b_i])))))
\end{array}
$$

h) $T_{4+|k|+|s|} = R \mid (\nu v)(O_{tv}(v) \mid ((\nu id_1, \cdots, id_s, msg_1, \cdots, msg_j)$
$(O_{to_1}(id_1) \mid \cdots \mid O_{to_s}(id_s)\mid$
$@(t_1, msg_1, d_1, p_1[1], \cdots, p_1[b_1] \mid \cdots \mid$
$@(t_j, msg_j, d_j, p_j[1], \cdots, p_j[b_j])))$

Para $TC^T(t) = \langle (Trad^T_\pi(I_\pi), \overline{nr}.\overline{t}id, Trad^T_\pi(P)) \rangle \in Caminho(\ SemGHBO(\ GH))$ *deve existir a derivação a seguir em* $SDer_{GH}$*:*

$$
\begin{array}{ccc}
L^T & \xrightarrow{\quad r \quad} & R^T \\
{\scriptstyle m}\downarrow & PO & \downarrow{\scriptstyle m'} \\
Trad^T_\pi(I_\pi) & \xrightarrow[\quad r' \quad]{} & Trad^T_\pi(P) = G
\end{array}
$$

Pela definição de $Trad_R$*, o nome* nr *da regra aplicada no modelo MBO-*π *pertence a* N*. Assim tem-se o nome da regra da GHBO correspondente a execução destes agentes.*

Assim temos $r = n(nr)$ *e o* m *pode ser construído pela associação dos identificadores dos objetos e mensagem presentes nos agentes* $O_{tv}(v)$ *e* $@(t, id, v, p[1], \ldots .p[k])$ *com os identificadores dos objetos e mensagem de* L^T*. Desta forma a definição de* m *é:*

$m(msg) = id$*, onde* $msg \in A_L$

$m(ob) = v$*, onde* $d^L(msg) = ob \ \wedge \ msg \in A_L$

$m(v[i]) = p[i]$*, onde* $o^L(msg) = v \ \wedge \ msg \in A_L$

e m *é um morfismo total, pois:*

i) m é um morfismo, isto é, respeita as funções de origem e destino do hiperarco msg *em* L^T*. Como* m *mapeia* msg *para* id *em* $Trad^T_\pi(I_\pi)$*, as funções de origem e destino são respeitadas, pois* m *mapeia o destino (e origens) de* msg *para o destino (e origens) de* id*.*

ii) m é total. Como em L^T há apenas um hiperarco e os vértices de origem e destino deste hiperarco, m é definido para todos os elementos de L^T.

Se G é construído como o pushout *de r e m, pela definição 39 e pelas características de r e m temos que G contém todos os elementos de $Trad^T_\pi(I_\pi)$ que não estão em $rng(m)$, contém todos os elementos criados por r e contém um elemento para cada grupo de elementos que são mapeados por m para um mesmo elemento.*

Pela definição de $Trad^T_\pi$ e P, temos que $Trad^T_\pi(P)$ é um hipergrafo com todos os elementos de R (agentes de I_π que não estão envolvidos no tratamento da mensagem id, isto é, o agente da mensagem id e o agente do objeto que irá tratá-la), todos os elementos criados pelo tratamento da mensagem id e o objeto que a trata.

Como $Trad^T_\pi(P) = G$, temos que $Trad^T_\pi(P)$ é o pushout *de r e m em* ***THGrafoP(T)*** *e conseqüentemente $TC^T(t) \in Caminho(SemGHBO(GH))$*

Hipótese

Para $n_\tau(t) = n$ e $n > 1$:
$t \in CaminhoBO(MB) \longrightarrow TC^T(t) \in Caminho(SemGHBO(GH))$,
onde $t = \langle (I_\pi, \tau, T_0), \ldots, (T_{j-1}, \overline{nr_1}, T_j), (T_j, lab_1, P_1), (P_1, \tau, T_{j+1}), \ldots,$
$(T_{j+k-1}, \overline{nr_2}, T_{j+k}), (T_{j+k}, lab_2, P_2), \ldots, (T_{j+k+l-1}, \overline{nr_n}, T_{j+k+l}),$
$(T_{j+k+l}, lab_n, P_n)\rangle$ *e*
$TC^T(t) = \langle (Trad^T_\pi(I_\pi), \overline{nr_1}.lab_1, Trad^T_\pi(P_1)),$
$(Trad^T_\pi(P_1), \overline{nr_2}.lab_2, Trad^T_\pi(P_2)), \ldots,$
$(Trad^T_\pi(P_{n-1}), \overline{nr_n}.lab_n, Trad^T_\pi(P_n))\rangle$

Passo

Sejam $n_\tau(t) = n + 1$ e
$t+ = \langle (I_\pi, \tau, T_0), \ldots, (T_{j-1}, \overline{nr_1}, T_j), (T_j, lab_1, P_1), (P_1, \tau, T_{j+1}), \ldots,$
$(T_{j+k-1}, \overline{nr_2}, T_{j+k}), (T_{j+k}, lab_2, P_2), \ldots, (T_{j+k+l-1}, \overline{nr_n}, T_{j+k+l}),$
$(T_{j+k+l}, lab_n, P_n), \ldots, (T_{j+k+l+m-1}, \overline{nr_{n+1}}, T_{j+k+l+m}),$
$(T_{j+k+l+m}, \overline{t}(id), P_{n+1})\rangle.$

De acordo com a definição de TC^T,
$TC^T(t+) = \langle (Trad^T_\pi(I_\pi), \overline{nr_1}.lab_1, Trad^T_\pi(P_1)),$
$(Trad^T_\pi(P_1), \overline{nr_2}.lab_2, Trad^T_\pi(P_2)), \ldots,$
$(Trad^T_\pi(P_{n-1}), \overline{nr_n}.lab_n, Trad^T_\pi(P_n)),$
$(Trad^T_\pi(P_n), \overline{nr}.\overline{t}(id), Trad^T_\pi(P_{n+1}))\rangle$

Seja t o subcaminho de $t+$ de I_π até P_n. Como $t \in CaminhoBO(MB))$, pela hipótese há um caminho c de $Trad^T_\pi(I_\pi)$ até $Trad^T_\pi(P_n)$ tal que $TC^T(t) = c$.

Se existe um caminho de P_n até P_{n+1}, existe um agente do objeto v e um agente da mensagem id em P_n que sincronizam, gerando um caminho com uma seqüência de τ, seguido por $\overline{nr}$ e $\overline{t}(id)$, resultando no termo P_{n+1}. Em P_n há ainda os agentes dos objetos de origem e destino de id e os demais agentes não envolvidos no tratamento desta mensagem. P_{n+1} termo contém todos os objetos e mensagens criados pela regra nr e os elementos de P_n com exceção da mensagem id.

Se em P_n há os dois agentes que sincronizam, em $Trad^T_\pi(P_n)$ há uma aresta id e um vértice v, bem como os vértices de origem e destino de id e os demais elementos que representam os agentes de P_n que não estão envolvidos no tratamento de id. A partir de $nr \in N$, constrói-se m. O hipergrafo G resultante do pushout *de $n(nr)$ e m contém todos os vértices e arestas que não pertencem ao $rng(m)$, todos os vértices e arestas criados por nr, e os objetos identificados por m estão identificados em G. Assim $G = Trad_\pi(P_{n+1})$.*

$\surd$

Com as provas dos teoremas 34 e 35, prova-se então que a semântica das GHBO é preservada pela tradução $Trad_{GH}$.

7 CONCLUSÕES

O aumento da escala e funcionalidade dos sistemas de computação e sua crescente complexidade envolvem um aumento significante de custos e exigem recursos humanos altamente qualificados para o desenvolvimento de software. Além disso torna-se difícil garantir que o sistema desenvolvido é exatamente o sistema desejado.

Para permitir a verificação de propriedades desejadas nos sistemas desenvolvidos integra-se ao processo de desenvolvimento o uso de métodos formais. Quando são considerados sistemas complexos, surge a necessidade de um modelo que forneça construções abstratas que facilitem o entendimento e a especificação deste sistema. Um modelo baseado em objetos fornece um nível de abstração que tem sido muito aplicado na prática.

Existem alguns formalismos que permitem descrição de sistemas baseados em objetos, entre eles gramáticas de grafos baseadas em objetos. Este formalismo possui uma linguagem visual, o que o torna bastante intuitivo, porém não possui ainda ferramentas que permitam a verificação automática de propriedades.

Neste trabalho, propôs-se um modelo de gramáticas de hipergrafos baseadas em objetos (baseada nas gramáticas de grafos baseadas em objetos) para especificação de sistemas baseados em objetos e apresentou-se uma tradução para um modelo em cálculo-π. Desta forma, torna-se possível a verificação de propriedades de sistemas especificados em gramáticas de hipergrafos baseados em objetos (GHBO) usando-se verificadores para o cálculo-π.

Os modelos propostos descrevem modelos baseados em objetos, onde as entidades principais são os objetos que se comunicam através de mensagens. No modelo das GHBO, os objetos são descritos por vértices e as mensagens por hiperarcos. No modelo do cálculo-π (MBO-π), os objetos e mensagens são descritos por agentes.

Não foi considerado, neste trabalho, o estado interno dos objetos, isto é, não são tratados atributos. Essa escolha foi feita pois desejava-se definir um modelo finito, e a inclusão de atributos pode gerar modelos infinitos (devido à possibilidade de conjuntos de valores infinitos para os atributos), que não permitem a verificação automática através de verificadores de modelos (LORETO; RIBEIRO; TOSCANI, 2002). A partir deste modelo pode-se construir um modelo com atributos finitos. Essa extensão pode ser feita sem alterar o modelo proposto drasticamente. Para o modelo da GHBO, inclui-se atributos algébricos para os vértices e arestas. Isso pode ser feito estendendo a definição de GHBO para gramáticas de hipergrafos com atributos, onde os atributos pertencem a uma álgebra finita. Para o modelo MBO-π, incluem-se novos agentes que representam os atributos de cada objeto e mensagem.

A função de tradução definida para os modelos, toma como entrada uma GHBO e constrói uma especificação MBO-π correspondente. Para provar que esta tradução preserva o comportamento das GHBO, foram comparados os sistemas de transição dos dois

modelos. Porém, há uma diferença de granularidade entre os STR dos dois modelos. No STR da tradução de uma GHBO há diversas transições que representam a aplicação de uma regra, enquanto no da GHBO há apenas uma. Desta forma, existem seqüências de transições no STR da tradução que não existem na da GHBO. Essas seqüências são aquelas em que as diversas ações que representam a aplicação de uma regra estão intercaladas, isto é, ocorrem em paralelo. Desta forma, existem estados no STR da tradução em que uma regra não foi totalmente aplicada, o que não ocorre no STR da GHBO. Existem, porém, no STR da tradução, seqüências que representam a aplicação atômica de uma regra: são aquelas em que todas as ações da aplicação da regra ocorrem seqüencialmente, sem outras ações em paralelo. Assim, as seqüências de transições do STR da tradução consideradas, para efeitos de comparação, são apenas aquelas que representam a aplicação atômica de uma regra.

Como as provas de preservação da semântica foram feitas para apenas para alguns caminhos do STR da tradução, alguns cuidados devem ser tomados na verificação de propriedades das GHBO através de verificadores para o cálculo-π. O que ocorre é que se uma propriedade é verdadeira para um modelo traduzido, será verdade para a GHBO, pois se a propriedade for verdadeira para todos os caminhos no STR do MBO-π será verdadeira para todos os caminhos da GHBO. Mas, caso uma propriedade seja falsa para o modelo traduzido, ela pode ser verdadeira para a GHBO, pois a propriedade pode ter falhado para um dos caminhos que não tem correspondência no STR da GHBO. Assim, a propriedade enunciada deve levar em conta essa questão. Porém, como os caminhos considerados da tradução são os caminhos que representam o comportamento das GHBO (as ações que representam o tratamento das mensagens aparecem na mesma ordem nos dois STRs), as propriedades relevantes para este modelo provavelmente não terão esse tipo de problema.

No decorrer do desenvolvimento deste trabalho, algumas afirmações que pareciam óbvias revelaram-se, nas provas dos teoremas e lemas, não serem corretas. Isso porque, durante essas provas surgiram detalhes que escaparam à primeira vista e foram percebidos neste momento. Ficou, assim, evidenciada a importância da realização destas provas.

7.1 Trabalhos Futuros

O sistema proposto possibilita a modelagem de problemas que não necessitem guardar informações sobre o estado interno dos objetos. Uma extensão para incluir atributos nos modelos pode ser feita para tornar o modelo mais abrangente. Nesta extensão devem ser levadas em conta questões de bloqueio aos atributos acessados concorrentemente. Esta extensão já foi esboçada, restando agora formalizá-la e provar que a semântica é preservada.

O número de sincronizações necessárias para representar a aplicação de uma regra poderia ter sido reduzida se a tradução fosse feita para o cálculo-π poliádico (MILNER, 1991), onde podem ser passados diversos nomes de uma só vez através de um canal. A escolha da versão monádica foi feita para que se possa realizar verificações no HAL, que utiliza esta versão. Pode-se porém com poucas alterações alterar a versão do cálculo-π, diminuindo assim o número de transições para a aplicação de cada regra.

As propriedades de sistemas baseados em objetos especificados em GHBO que se deseja verificar através de verificadores do cálculo-π devem ser descritas em lógica-π (FERRARI et al., 1997). Assim, para que alguém possa realizar a verificação de alguma propriedade, deve ter prévio conhecimento deste formalismo. Para desenvolvedores que

têm maior afinidade com métodos gráficos de especificação, torna-se mais atraente a utilização da tradução proposta se ele puder descrever suas propriedades através de um formalismo gráfico. Assim, um trabalho a ser desenvolvido é o estudo de maneiras de descrever as propriedades desejadas através de gramáticas de grafos.

Para permitir a verificação automática das propriedades de sistemas modelados em GHBO a tradução proposta deve ser implementada, permitindo assim a avaliação dos resultados utilizando algum verificador de modelos. Para avaliar os resultados deste trabalho pretende-se então modelar sistemas utilizando GHBO e verificar propriedades desejadas através da tradução e da ferramenta HAL.

ANEXO A CONSTRUÇÕES CATEGORIAIS

A.1 Alguns Conceitos de Teoria das Categorias

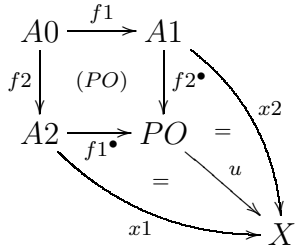
Categoria: Uma categoria **Cat** consiste de um classe $Obj_{\mathbf{Cat}}$ de objetos, para cada par $A, B \in Obj_{\mathbf{Cat}}$ um conjunto $Mor_{\mathbf{Cat}}(A, B)$ de morfismos, escrito $f : A \to B$ para cada $f \in Mor_{\mathbf{Cat}}(A, B)$, e um composição $g \circ f : A \to C$ para cada par de morfismos $f : A\ toB$ e $g : B \to C$ tal que se tenha:

1. $(h \circ g) \circ f = h \circ (g \circ f)$ para todos os morfismos f, g e h se pelo menos um lado for definido, e

2. para cada objeto A de **Cat** há um morfismo identidade distinto id_A com $f \circ id_A = f$ e $id_A \circ g = g$ sempre que o lado esquerdo for definido.

Isomorfismo: Um morfismo $f : A \to B$ na categoria **Cat** é chamado de isomorfismo se existe um morfismo $g : B \to A$ tal que $g \circ f = id_A$ e $f \circ g = id_B$.

Pushout: O *pushout* de um par de morfismos $f1 : A0 \to A1$ e $f2 : A0 \to A2$ em **Cat** é um objeto PO juntamente com os morfismos $f2^{\bullet} : A1 \to PO$ e $f1^{\bullet} : A2 \to PO$ tal que para todos os objetos X e todos os morfismos $xi : Ai \to X$ em **Cat** com $x1 \circ f1 = x2 \circ f2$ há um único $u : PO \to X$ com $u \circ fi^{\bullet} = xi$, para $i = 1, 2$.



A.2 Categoria dos Conjuntos e Funções Parciais - SetP

Definição 36 (Categoria SetP) *A categoria **SetP** possui todos os conjuntos como objetos e as funções parciais como morfismos. As identidades são as funções identidade e a composição é a composição de funções.*

∇

Construção 37 (Pushouts em SetP) *Considere os conjuntos Ai, com $i = 1..3$, as funções parciais fj, com $j = 1..2$ e $\perp$ que é um elemento que não pertence a nenhum destes conjuntos.*

$$\begin{array}{ccc} A1 & \xrightarrow{f1} & A2 \\ {\scriptstyle f2}\downarrow & = & \downarrow{\scriptstyle f2^\bullet} \\ A3 & \xrightarrow[f1^\bullet]{} & O \end{array}$$

Então um pushout *na categoria **SetP** é uma tupla $(O, \rightarrow^O)$, onde:*

$$\begin{aligned} O = PO^{\mathbf{SetP}}(A3 \overset{f2}{\leftarrow} A1 \overset{f1}{\rightarrow} A2) = \{Q \subseteq (A3 \times \{\perp\}) \cup (\{\perp\} \times A2) | \\ (a, \perp) \in Q, a = f2(x) \text{ implica } (\perp, f1(x)) \in Q \text{ e} \\ (\perp, b) \in Q, b = f1(x) \text{ implica } (f2(x), \perp) \in Q\} \end{aligned}$$

$$\rightarrow^O = (f2^\bullet, f1^\bullet), \text{onde } f1^\bullet(a) = \begin{cases} [(a, \perp)] & \text{se } [(a, \perp)] \in O \\ undef & \text{em outro caso} \end{cases} \text{ e}$$

$$f2^\bullet(b) = \begin{cases} [(\perp, b)] & \text{se } [(\perp, b)] \in O \\ undef & \text{em outro caso} \end{cases}$$

∇

A.3 Categorias dos Hipergrafos (Tipados) e Morfismos Parciais (Tipados) - HGrafoP e THGrafoP(T)

Notação: A operação de pertinência ($\in$) é utilizada, aqui, entre listas e conjuntos. Dada uma lista l e um conjunto C, $l \in C$ significa que todos os elementos da lista l pertencem ao conjunto C.

Construção 38 (Pushouts em HGrafoP) *Considere os hipergrafos Hi, com $i = 1..3$, os morfismos gj, com $j = 1..2$ e $\perp$ que é um elemento que não pertence a nenhum dos conjuntos envolvidos.*

$$\begin{array}{ccc} H1 & \xrightarrow{g1} & H2 \\ {\scriptstyle g2}\downarrow & = & \downarrow{\scriptstyle g2^\bullet} \\ H3 & \xrightarrow[g1^\bullet]{} & O \end{array}$$

Então um pushout *na categoria **HGrafoP** é uma tupla $(O, \rightarrow^O)$, onde :*

$O = PO^{\mathbf{HGrafoP}}(H3 \overset{g2}{\leftarrow} H1 \overset{g1}{\rightarrow} H2) = (V, E, sc, tg)$, *onde:*

$V = PO^{\mathbf{SetP}}(V_{H3} \overset{g2_V}{\leftarrow} V_{H1} \overset{g1_V}{\rightarrow} V_{H2})$,

$E = PO^{\mathbf{SetP}}(E_{H3} \overset{g2_E}{\leftarrow} E_{H1} \overset{g1_E}{\rightarrow} E_{H2}) - \{[(e3, e2)] | (sc^{H3}(e3) \not\in V$ ou $tg^{H3}(e3) \not\in V$ ou $e3 = \perp)$ e $(sc^{H2}(e2) \not\in V$ ou $tg^{H2}(e2) \not\in V$ ou $e2 = \perp)\}$,

$sc : E_{PO} \rightarrow V^*_{PO}$ *é obtida como uma função total única tal que* $g2^\bullet_{V^*} \circ sc^{H2} = sc \circ g2^\bullet_E$ *e* $g1^\bullet_{V^*} \circ sc^{H3} = sc \circ g1^\bullet_E$, *e*

$tg : E_{PO} \rightarrow V_{PO}$ *é obtida de forma análoga (tal que a função existe pela construção de E e é única devido às propriedades do* pushout *de V_{PO}).*

$\rightarrow^O = (g2^\bullet, g1^\bullet)$, *onde* $g1^\bullet = (g1^\bullet_V, g1^\bullet_E)$ *e* $g2^\bullet = (g2^\bullet_V, g2^\bullet_E)$.

∇

Definição 39 (Pushouts em THGrafoP(T)) *Considere os hipergrafos tipados* Hi^T*, com* $i = 1..3$ *e os morfismos* gj^T*, com* $j = 1..2$*.*

$$\begin{array}{ccc} H1^T & \xrightarrow{g1^T} & H2^T \\ {\scriptstyle g2^T}\downarrow & = & \downarrow{\scriptstyle g2^{\bullet T}} \\ H3^T & \xrightarrow[g1^{\bullet T}]{} & PO^T \end{array}$$

Então definimos um pushout *na categoria* ***THGrafoP(T)*** *como:*

$O = PO^{\mathbf{THGrafoP(T)}}(H3^T \overset{g2^T}{\leftarrow} H1^T \overset{g1^T}{\rightarrow} H2^T) = (PO^{\mathbf{HGrafoP}}(H3 \overset{g2}{\leftarrow} H1 \overset{g1}{\rightarrow} H2), t, T)$*, onde*

$t : PO \rightarrow T$ *é o morfismo de hipergrafos total único tal que* $t \circ g1^\bullet = t^{H3}$ *and* $t \circ g2^\bullet = t^{H2}$*.*

$\rightarrow^O = (g2^{\bullet T}, g1^{\bullet T})$*, onde* $g1^{\bullet T} = (g1^\bullet, id_T)$ *e* $g2^{\bullet T} = (g2^\bullet, id_T)$*.*

∇

REFERÊNCIAS

AGHA, G. A. **ACTORS**: a Model of Concurrent Computation in Distributed Systems. Cambridge, MA, USA: MIT Press, 1986. 144p.

AGHA, G. Concurrent Object-Oriented Programming. **Communications of the ACM**, New York, v.33, n.9, p.125–141, 1990.

BOWEN, J.; HINCHEY, M. **Seven more myths on formal methods**. [S.l.]: Oxford University, 1994. Technical Report. (TR-9-94).

BOWEN, J.; HINCHEY, M. **High-Integrity System Specification and Design**. London: Springer-Verlag, 1999. (Formal Approaches to Computing and Information Technology (FACIT)).

CLARKE, E.; WING, J. Formal Methods: State of the Art and Future Directions. **ACM Computing Surveys**, New York, v.28, n.4, 1996.

COPSTEIN, B.; COSTA MÓRA, M. da; RIBEIRO, L. An Environment for Formal Modeling and Simulation of Control Systems. In: ANNUAL SIMULATION SYMPOSIUM, 33., 2000, Washington. **Proceedings...** [S.l.]: Institute of Electrical and Electronics Engineers, 2000. p.74–82.

CORRADINI, A.; MONTANARI, U.; ROSSI, F. Algebraic Approaches to Graph Transformation. In: ROZENBERG, G. (Ed.). **Handbook of Graph Grammars and Computing by Graph Transformation**. Singapore: World Scientific, 1997. v.1, p.163–245.

DOTTI, L. F.; RIBEIRO, L. Specification of Mobile Code Systems Using Graph Grammars. In: IFIF TC6 WG6.1 INTERNATIONAL CONFERENCE ON FORMAL METHODS FOR OPEN OBJECT-BASED DISTRIBUTED SYSTEMS, 4., 2000, Stanford. **Formal Methods for Open Object-Based Distributed Systems IV.** Boston: Kluwer Academic, 2000. p.45–63.

EHRIG, H. Introduction to the Algebraic Theory of Graph Grammars. In: INTERNATIONAL WORKSHOP ON GRAPH-GRAMMARS AND THEIR APPLICATION TO COMPUTER SCIENCE AND BIOLOGY, 1978, Bad Honnef. **Proceedings...** Berlin: Springer Verlag, 1978. p.1–69. (Lecture Notes in Computer Science, v.73).

FERRARI, G. et al. An Automata Based Verification Environment for Mobile Processes. In: INTERNATIONAL WORKSHOP ON TOOLS AND ALGORITHMS FOR THE CONSTRUCTION AND ANALYSIS OF SYSTEMS, 3., 1997, Enschede. **Tools and Algorithms for the Construction and Analysis of Systems:** proceedings. Berlin: Springer Verlag, 1997. p.275–289. (Lecture Notes in Computer Science, v.1217).

FERRARI, G. et al. Verifying Mobile Processes in the HAL Environment. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED VERIFICATION, 1998, Vancouver, CA. **Computer Aided Verification:** proceedings. Berlin: Springer Verlag, 1998. p.511–515. (Lecture Notes in Computer Science, v.1427).

HERRMANN, D. **Software Safety and Reliability**: Techniques, Approaches, and Standards of Key Industrial Sectors. [S.l.]: IEEE Computer Society Press, 2000.

HIRATA, K. **π-Calculus Semantics of Moded Flat HGC**. Kanagawa, Japan: NTT Basic Research Laboratories, 1995. Technical Report. (ISRL-95-3).

HÜTTEL, H.; KLEIST, J. **Objects as mobile processes**. Aalborg, Denmark: Aalborg University, 1996. BRICS Technical Report. (RR-96-38).

IQ-MOBILE. **Improving the Quality of Open Systems with Code Mobility through Rigorous Development**. [S.l.: s.n.], 2000. Brazilian/Italian Bilateral Cooperation Project Proposal.

KORENBLAT, K.; PRIAMI, C. **Extraction of π-Calculus Specifications From UML Sequence and State Diagrams**. Povo-Trento, Italy: University of Trento, Department of Information and comminication Technology, 2003. Technical Report. (DIT-03-007).

KORFF, M. True Concurrency Semantics for Single Pushout Graph Transformations with Applications to Actor Systems. In: WORKING PAPERS OF THE INTERNATIONAL WORKSHOP ON INFORMAL INFORMATION SYSTEMS - CORRECTNESS AND REUSABILITY, 1995. **Proceedings...** [S.l.: s.n.], 1995. p.33–50. (World Scientic).

KORFF, M. **Generalized Graph Structure Grammars with Applications to Concurrent Object-Oriented Systems**. 1996. PhD thesis — Technische Universität Berlin, Berlin.

LORETO, A. B.; RIBEIRO, L.; TOSCANI, L. V. Decidability and Tractability of Problems in Object-Based Graph Grammars. In: IFIP TCS, 2002. **Proceedings...** Boston: Kluwer Academic, 2002. p.396–408. (IFIP Conference Proceedings, v.223).

MILNER, R. **The Polyadic π-Calculus**: a Tutorial. Edinburgh, UK: University of Edinburgh, Computer Science Department, 1991. LFCS Report. (ECS-LFCS-91-180).

MILNER, R.; PARROW, J. A Calculus for Mobile Processes I e II. **Information and Computation**, [S.l.], v.100, p.1–77, 1992.

MONTANARI, U.; PISTORE, M. **History-Dependent Automata**. Pisa, Italy: University of Pisa, Computer Science Department, 1998. Technical Report. (TR-98-11).

NIERSTRASZ, O. A Survey of Object-Oriented Concepts. In: KIM, W.; LOCHOVSKY, F. (Ed.). **Object-Oriented Concepts, Databases and Applications**. Reading, MA: ACM Press: Addison Wesley, 1989. p.3–21.

PARROW, J. **An Introduction to the π-Calculus**. [S.l.]: Elsevier, 2001. p.476–543. (Handbook of Process Algebra).

RIBEIRO, L. **Parallel Composition and Unfolding Semantics of Graph Grammars**. 1996. PhD thesis — Technische Universität Berlin, Berlin.

RUMBAUGH, J.; BLAHA, M.; PREMERLANI, W.; EDDY, F.; LORENSEN, W. **Modelagem e Projetos Baseados em Objetos**. Rio de Janeiro: Campus, 1994.

SANGIORGI, D.; WALKER, D. **The π-calculus**: a theory of Mobile Processes. Cambridge, UK: Cambridge University Press, 2001. p.517–532.

STOREY, N. **Safety-Critical Computer Systems**. [S.l.]: Addison-Wesley, 1996.

UEDA, K. Designing A Concurrent Programming Language. In: INFORMATION PROCESSING SOCIETY OF JAPAN, INFOJAPAN, 1990. **Proceedings...** [S.l.: s.n.], 1990. p.97–94.

VICTOR, B.; MOLLER, F. The Mobility Workbench - A Tool for the π-Calculus. In: INTERNATIONAL CONFERENCE ON COMPUTER AIDED VERIFICATION, CAV, 1994. **Proceedings...** Berlin: Springer-Verlag, 1994. p.428–440. (Lecture Notes in Computer Science, v.818).

WEGNER, P. Dimensions of Object-based Language Design. **SIGPLAN Notices**, New York, v.22, n.12, p.168–182, Dec. 1987. Trabalho Apresentado na Object-Oriented Programming Systems Languages and Applications Conference, 1987, Orlando, USA.

WICHMANN, B. **Software in Safety Related Systems**. [S.l.]: John Wiley and Sons, 1992.