Mecanismos de Interação para um Modelo de Redes de Petri Orientado a Objetos

Ana Karla Alves de Medeiros

Dissertação submetida à Coordenação do Curso de Pós-Gradução em Informática da Universidade Federal da Paraíba - Campus II como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Redes de Petri

Angelo Perkusich
(orientador)

Campina Grande, Paraíba, Brasil

M488M

MEDEIROS, Ana Karla Alves de

Mecanismos de Interação para um Modelo de Redes de Petri Orientado a Objetos.

Dissertação de Mestrado, Universidade Federal da Paraíba, Centro de Ciências e Tecnologia, Coordenação de Pós-Graduação em Informática, Campina Grande - Pb, Setembro de 2000.

92 p. Il.

Orientador: Angelo Perkusich

Palavras-chave: 1. Redes de Petri de Alto Nível 2. Métodos Formais 3. Orientação a objetos 4. Engenharia de Software 5. Mecanismos de Interação

CDU - 519.711

Resumo

Neste trabalho é apresentado o acréscimo de abstrações para mecanismos de interação entre objetos de uma Rede de Petri Orientada a Objetos (RPOO). RPOO é uma notação formal apropriada à especificação de sistemas de *software* paralelos e distribuídos. As abstrações adicionadas referem-se aos mecanismos de *comunicação síncrona* e *sincronização de métodos*. A idéia básica é melhorar o poder de abstração dessa notação e facilitar a tarefa do projetista de *software*. Como complemento, indica-se como proceder à análise da nova notação.

Abstract

This work presents the addition of new abstractions for interaction mechanisms between objects of Rede de Petri Orientada Objetos (RPOO). RPOO is a formal notation well suited for the specification of concurrent and distributed software systems. The added abstractions provide synchronous communication and method synchronization. The main idea is to improve the abstraction power of this notation, facilitating the software designer's task. Additionally, some hints about analyzing the new notation are given.

Agradecimentos

Gostaria de agradecer...

... ao meu orientador, Angelo Perkusich, por todo o apoio, compreensão, incentivo e profissionalismo ao longo desses anos de trabalho.

... ao meus pais, Salomão e Rosilene, e irmãos, Karina e Leonardo, pelo carinho e paciência, sobretudo nos momentos em que eu me irritava.

... ao meu companheiro, Fábio, que sempre me apoiou e cuja ajuda foi fundamental nos últimos meses de trabalho.

... ao meus amigos, em especial a Lívia e Gustavo, cujos espíritos de perseverança foram exemplo para mim (parabéns a vocês também!).

... aos membros do LabPetri, Dalton, Sandro, Érica, Leandro, Jorge, Adriano, Kyller e Márcia, pelas discussões frutíferas e brincadeiras descontraídas.

... aos professores e funcionários do DSC.

... a Deus, por ter sempre me abençoado.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Dentre os objetivos perseguidos pela *Engenharia de Software* está o de proporcionar meios para a construção de sistemas de *software* que operem corretamente, independentemente de sua complexidade. Para tanto, indica como um dos caminhos a seguir o uso de *métodos formais* [CW96].

Os métodos formais são linguagens, técnicas e ferramentas baseadas na matemática e utilizados na *especificação*, *análise* e *verificação* de sistemas de *software*. Por possuírem uma base matemática e, portanto, permitirem automatizar a análise e verificação de especificações, tais métodos tendem a aumentar a confiança no funcionamento dos sistemas que especificam. Veja que, uma análise e verificação automáticas revelam inconsistências, ambigüidades e incompletudes que, de outra forma, seriam indetectáveis e, por conseguinte, aumentam a compreensão dos projetistas de *software*.

Pode-se entender uma especificação como o processo de descrição de um sistema e de suas propriedades. Como a base de uma especificação é a abstração [AP98], uma boa ferramenta (formal) para a especificação de sistemas de *software* deve suportar mecanismos de abstração. Dessa forma, os desenvolvedores preocupam-se apenas com a identificação das características essenciais dos sistemas, obtendo especificações que são simples, gerais e precisas, as quais resultam em um produto completo e correto. Nesse sentido, o presente trabalho trata do enriquecimento da notação de uma ferramenta conceitual formal pelo acréscimo de novos mecanismos de abstração para interação entre agentes computacionais. A adição desses mecanismos visa facilitar a descrição de sistemas complexos (grandes). A ferramenta considerada denomina-se *Rede de Petri*

Orientada a Objetos - RPOO [CGdFP98; dMGdFP98; Gue99].

As *redes de Petri* são uma ferramenta gráfica e matemática, especialmente apropriadas para a descrição e o estudo de sistemas de *software* concorrentes, assíncronos, distribuídos, paralelos, não-determinísticos e/ou estocásticos [Mur89]. RPOO pertence à classe das redes de Petri que visam à especificação de sistemas concorrentes e orientados a objetos. Por combinar redes de Petri (PNs) [Jen92; Mur89] e orientação a objetos (OO) [Boo94; RBP+91], RPOO associa a facilidade de tratamento e estudo de concorrência das PNs com os mecanismos de abstração e decomposição oferecidos em OO. Em seu formalismo básico, um sistema é composto por uma coleção de objetos concorrentes entre si, que se comunicam via troca assíncrona de mensagens. Assume-se garantia de entrega de mensagens, mas sem ordenação. Do ponto de vista da concorrência, esse mecanismo único satisfaz o formalismo básico [Agh86]. Entretanto, se o objetivo é prover uma notação RPOO suficientemente rica para facilitar a descrição de sistemas complexos, novos mecanismos de interação precisam ser incorporados. Dentre eles, podem-se destacar: *transferência síncrona de mensagens* e *sincronização de métodos dos objetos RPOO*.

A transferência síncrona de mensagens baseia-se na comunicação assíncrona do modelo RPOO básico e possibilita aos objetos aguardar/enviar um reconhecimento ou um resultado de processamento para uma comunicação. A sincronização de métodos permite a coordenação de eventos num sistema RPOO. Para utilizá-la, define-se, através de uma linguagem, um sincronizador.

Um sincronizador é uma entidade computacional que implementa os mecanismos para garantir uma ordem na comunicação entre um conjunto de objetos. Ele é definido como um objeto no sistema, e não como parte integrante dos objetos aos quais se aplica, para que esses ainda possam se comunicar com outros, os quais o sincronizador não restringe.

Além de prover abstrações que facilitem a especificação dos sistemas, uma notação formal deve suportar também a análise do sistema especificado. Para o caso de sistemas distribuídos e concorrentes, a análise deve possibilitar o estudo de propriedades comportamentais, tais como, ausência de impasses[1], estados possíveis no sistema, ordem

[1] Tradução para *deadlocks*.

de ocorrência de eventos, entre outros. Então, dentre as técnicas existentes, a análise baseada em *grafos de ocorrência* ou *espaço de estados* [Jen95] destaca-se porque permite desenvolver um algoritmo e automatizar o processo de verificação e validação das propriedades comportamentais de um sistema. Entretanto, como grafos de ocorrência são grafos bipartidos direcionados, os quais possuem um nó para cada estado alcançável no sistema, eles tendem a se tornar grandes, ocasionando o *problema da explosão de estados*.

Nesse sentido, visando amenizar esse problema, neste trabalho mostra-se como utilizar os grafos de ocorrência de modo a omitir estados que não se deseja observar. Para tanto, define-se uma *especificação de equivalência*, que deve ser usada quando da construção do grafo.

1.1 Objetivos da Dissertação

O objetivo principal deste trabalho é adicionar à RPOO novas abstrações para mecanismos de interação entre seus objetos, baseando-se no mecanismo de comunicação assíncrona já existente. Além disso, pretende-se mostrar como utilizar a notação com os novos mecanismos, sem incorrer em aumento no espaço de estados.

1.2 Escopo e Relevância

Os modelos conceituais básicos para computação concorrente devem prover os mecanismos mínimos necessários para a descrição dos sistemas ao quais se aplicam [Agh86]. Para o caso particular de RPOO, o seu formalismo original é suficiente para descrever sistemas distribuídos, concorrentes e orientados a objetos, pois as redes de Petri nele utilizadas tratam os aspectos referentes à concorrência e a comunicação assíncrona existente permite a interação entre os objetos.

Entretanto, uma boa ferramenta para descrição de sistemas de *software* deve prover mecanismos que facilitem a especificação de sistemas complexos [Frø96], permitindo ao projetista preocupar-se apenas em modelar as características essencias do sistema e não, por exemplo, para o caso de RPOO, em como descrever uma interação síncrona

entre objetos.

Assim, é importante construir abstrações, sobre as construções do formalismo básico, que facilitem a especificação de sistemas complexos e, sobretudo, garantam a preservação de suas propriedades. Essa preservação deve ser assegurada sintaticamente, pois é impossível saber, quando da definição das novas abstrações, qual será o estado inicial dos sistemas a serem especificados.

A relevância deste trabalho está no aumento do poder de abstração da ferramenta conceitual RPOO pelo acréscimo de dois dos mecanismos de interação comuns em sistemas distribuídos: comunicação síncrona [AFK+93] e sincronização de métodos [Frø96]. Esses dois mecanismos de interação baseiam-se no mecanismo assíncrono do modelo básico e garantem, estruturalmente, a preservação das propriedades comportamentais dos sistemas em cujas especificações estão presentes.

1.3 Estrutura da Dissertação

O restante da dissertação está organizado como segue:

No capítulo 2, apresenta-se a notação RPOO. A explicação e formalização do mecanismo para *comunicação síncrona* encontram-se no Capítulo 3. No Capítulo 4, discute-se a introdução do mecanismo para *sincronização de métodos* . Considerações sobre análise baseada em grafos de ocorrência estão presentes no Capítulo 5 e, finalmente, no Capítulo 6, apresentam-se as conclusões e trabalhos futuros.

Capítulo 2

RPOO - Rede de Petri Orientada a Objetos

Este capítulo contém uma descrição de redes de Petri suficiente ao bom entendimento de RPOO. Para tanto, está dividido em três seções. Na Seção 2.1 introduzem-se conceitos e definições comuns às redes de Petri. Na Seção 2.2 apresentam-se as definições de Redes de Petri Coloridas, que são base para RPOO, e na Seção 2.3 trata-se do modelo RPOO propriamente dito.

2.1 Redes de Petri

As *redes de Petri* são uma ferramenta matemática, especialmente apropriadas à descrição e ao estudo de sistemas de *software* concorrentes, assíncronos, distribuídos, paralelos, não-determinísticos e/ou estocásticos [Mur89]. Possuem uma representação gráfica associada à sua notação e, por conseguinte, facilitam a interação entre os indivíduos envolvidos na construção de um sistema de *software*.

Uma rede de Petri compõe-se de uma *estrutura de rede*, *inscrições* associadas a essa estrutura e uma *marcação*. A estrutura da rede e as inscrições definem a sintaxe de uma rede de Petri. A evolução de suas marcações, segundo uma *regra de ocorrência*, estabelece a sua semântica.

Como pode-se observar na Figura 2.1, a estrutura de uma rede de Petri pode ser representada por um grafo bipartido direcionado, cujos nós podem ser de dois tipos

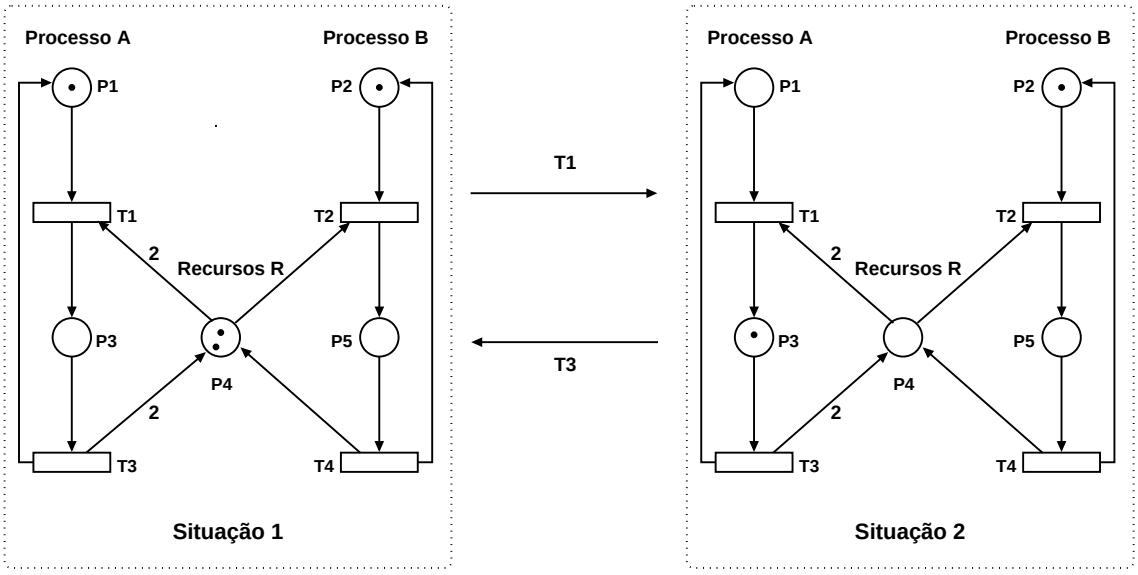

Figura 2.1: Exemplo de rede de Petri Lugar/Transição.

distintos: *lugares* ou *transições*. Os lugares podem representar estados e as transições, ações ou eventos.

Definição 2.1 (Estrutura de Rede de Petri) *Uma estrutura de rede de Petri é uma tripla* $N = \langle P, T, F \rangle$*, na qual:*

- P *é um conjunto finito de lugares;*
- T *é um conjunto finito de transições;*
- $F \subseteq P \times T \cup T \times P$ *é uma relação de fluxo;*
- $P \cap T = \varnothing$, $P \cup T \neq \varnothing$.

Na Definição 2.1 apresenta-se uma formalização para a estrutura de uma rede de Petri. Note que, de acordo com a relação de fluxo F, os arcos da estrutura sempre conectam nós de tipos diferentes. Graficamente, os lugares da estrutura de uma rede de Petri correspondem a círculos, as transições a retângulos e a relação de fluxo a arcos direcionados.

Na rede de Petri apresentada na Figura 2.1, há dois tipos de inscrições: o *peso dos arcos* (por exemplo, o peso 2 associado ao arco direcionado entre $T3$ e $P4$) e a *marcação*, que determina a marcação inicial da rede. A formalização dessas inscrições está na Definição 2.2.

Definição 2.2 (Rede de Petri) *Uma rede de Petri é uma tripla* $PN = \langle N, W, M_0 \rangle$, *em que:*

- N *é uma estrutura de rede de Petri;*
- $W : F \to \mathbb{N}^*$ *é uma função peso;*
- $M_0 : P \to \mathbb{N}$ *é a marcação inicial.*

A marcação de uma rede de Petri estabelece o seu estado. Uma marcação corresponde à associação de elementos, denominados *fichas*, aos lugares. Por exemplo, as fichas, pequenos pontos pretos, nos lugares da rede da Figura 2.1, na *Situação 1*, estabelecem a sua marcação (estado) inicial. Como o estado dos sistemas representados pelas redes de Petri normalmente[1] varia à medida que eventos ocorrem, existe uma regra de ocorrência para determinar a evolução das marcações. Essa regra estabelece as condições para que uma transição esteja *habilitada a ocorrer* e quais as conseqüências da ocorrência.

Para exemplificar o que já foi discutido até o momento, vamos utilizar uma rede de Petri Lugar/Transição [Mur89], que pertence à classe das redes de Petri de Baixo Nível (ver Seção 2.1.2, página 11). Nas redes Lugar/Transição, as fichas representam apenas informação binária, através de sua presença ou ausência nos lugares.

A rede de Petri Lugar/Transição apresentada na Figura 2.1 modela uma situação na qual dois tipos de processos - A e B - utilizam recursos do tipo R. O processo A precisa de *dois* recursos R para executar, enquanto o processo B necessita apenas de *um* recurso R.

De acordo com as Definições 2.1 e 2.2, para o exemplo da Figura 2.1, temos:

- $P = \{P1, P2, P3, P4, P5\}$, em que:
  - *Lugar* $P1$ ($P2$) = processo A (B) aguardando alocação de recurso;
  - *Lugar* $P3$ ($P5$) = processo A (B) executando;
  - *Lugar* $P4$ = recurso R disponível.

[1] Teoricamente, é possível uma situação na qual um evento ocorra e o estado do sistema permaneça inalterado.

- $T = \{T1, T2, T3, T4\}$, em que:
  - *Transição* $T1(T2)$ = processo A(B) aloca recurso;
  - *Transição* $T3(T4)$ = processo A(B) finaliza execução;
- $F = \{(P1, T1), (T1, P3), (P3, T3), (T3, P1), (T3, P4), (P4, T1), (P2, T2), (T2, P5), (P5, T4), (T4, P2), (T4, P4), (P4, T2)\}$;
- $\forall f \in F$,

$$W(f) = \begin{cases} 1 & \text{se } f \in (F - \{(T3, P4), (P4, T1)\}), \\ 2 & \text{se } f \in \{(T3, P4), (P4, T1)\}. \end{cases}$$

- $\forall p \in P$,

$$M_0(p) = \begin{cases} 0 & \text{se } p \in \{P3, P5\} \\ 1 & \text{se } p \in \{P1, P2\}, \\ 2 & \text{se } p \in \{P4\}. \end{cases}$$

Note que o arco $(P4, T1)$, por exemplo, possui peso 2, significando que processos tipo A requerem *dois* recursos tipo R para executar. Os arcos sem inscrição explícita possuem peso 1. Em adição, a marcação inicial (ver *Situação 1*) estabelece um estado inicial em que há um processo de cada tipo (*uma* ficha em cada lugar $P1$ e $P2$) e dois recursos do tipo R (*duas* fichas no lugar P4).

Para definir como a rede de Petri da Figura 2.1 passa da *Situação 1* à *Situação 2*, vamos enunciar a regra de ocorrência para as redes Lugar/Transição. Para tanto, é importante compreender os seguintes conceitos:

Lugares de entrada: lugares de onde partem os arcos direcionados (*arcos de entrada*) que chegam a uma transição, são denotados por ${}^\bullet t$;

Lugares de saída: lugares para onde partem os arcos direcionados (*arcos de saída*) que saem de uma transição, são denotados por $t^\bullet$.

Definição 2.3 (Regra de Ocorrência) *Para uma rede de Petri Lugar/Transição, tem-se que:*

1. *uma transição está habilitada a ocorrer se* cada um *de seus lugares de entrada contém um número de fichas* maior ou igual *ao indicado pelo peso do arco de entrada respectivo;*

2. *uma transição habilitada* pode *ocorrer;*

3. *quando uma transição habilitada ocorre:*

   - *de* cada um *de seus lugares de entrada,* remove-se *um número de fichas igual ao peso do arco de entrada respectivo;*
   - *em* cada um *de seus lugares de saída,* acrescenta-se *um número de fichas igual ao peso do arco de saída respectivo.*

Logo, para a rede de Petri na Figura 2.1, a transição $T1$ tem como seus lugares de entrada $P1$ e $P4$ e, como seu lugar de saída $P3$, ou seja, ${}^\bullet T1 = \{P1, P4\}$ e $T1^\bullet = \{P3\}$. Além disso, na marcação inicial - M_0 - (ver *Situação 1*), existem duas transições - $T1$ e $T2$ - habilitadas. Então, pode-se dizer que, nessa situação, dois *passos* estão habilitados, um para a transição $T1$ e outro para $T2$. Observe que esses passos *não* são concorrentes, pois a ocorrência de um deles desabilita o outro.

A *Situação 2* mostra a marcação - M_1 - alcançada após a ocorrência da transição $T1$. Note que, segundo a regra de ocorrência, *duas* fichas foram removidas do lugar $P4$, *uma* ficha foi removida do lugar $P1$ e *uma* ficha foi adicionada ao lugar $P3$. Diz-se, então, que M_1 é uma *marcação alcançável* a partir de M_0, através da *seqüência de ocorrência* $\sigma = T1$, e denotada por $M_0[\sigma\rangle M1$. O conjunto de *todas* as marcações alcançáveis a partir da marcação inicial é denotado por $[M_0\rangle$.

2.1.1 Propriedades de uma rede de Petri

As propriedades das redes de Petri classificam-se em *estruturais* e *comportamentais*. As propriedades estruturais (comportamentais) caracterizam-se por independerem (dependerem) da marcação inicial associada à rede de Petri.

No contexto deste trabalho, interessam duas propriedades comportamentais em particular: a *vivacidade* e a *limitação*.

Vivacidade

A propriedade de vivacidade de uma rede de Petri relaciona-se ao conceito de ausência de impasses. Antes de explicar o que é uma rede de Petri viva, vamos entender o que é uma transição viva.

Uma transição $t \in T$ é viva se, para qualquer marcação M' alcançável a partir de M_0, existe *sempre* uma marcação M'' alcançável a partir de M', em que t ocorre.

Uma rede de Petri é viva se *todas* as suas transições são vivas. Na Definição 2.4, considere que:

- $L(M_0)$ denota o conjunto de todas as seqüências de ocorrência iniciadas a partir de M_0;
- T^* é o conjunto de todas as seqüências finitas de elementos de T.

Definição 2.4 (Vivacidade de uma rede de Petri) *Uma transição $t \in T$ é viva se, e somente se, para qualquer $\sigma \in L(M_0)$ existe uma seqüência $\sigma' \in T^*$ tal que $\sigma\sigma' t \in L(M_0)$. Uma rede de Petri N é viva se, e somente se, t é viva, $\forall t \in T$.*

Observe que a rede de Petri da Figura 2.1 é viva, pois $T1$, $T2$, $T3$ e $T4$ são vivas para a marcação inicial indicada na *Situação 1*.

Limitação

Uma rede de Petri é considerada *k-limitada*, ou simplesmente *limitada*, se o número de fichas presentes *em cada um* de seus lugares não excede um número finito $k \in \mathbb{N}^+$ para quaisquer das marcações alcançáveis a partir de M_0. Na Definição 2.5, considere que:

- $M(p)$ denota a marcação do lugar p;
- $[M_0\rangle$ denota o conjunto de todas as marcações alcançáveis a partir de M_0.

Definição 2.5 (Limitação de uma rede de Petri) *Um lugar $p \in P$ é k-limitado, $k \in \mathbb{N}^+$ se, e somente se, $M(p) \leq k$, $\forall M \in [M_0\rangle$. Uma rede de Petri N é k-limitada se, e somente se, p for k-limitado, $\forall p \in P$.*

Note que a rede de Petri da Figura 2.1 é 2-limitada.

Outras propriedades podem ser definidas para as redes de Petri, o leitor interessado pode consultar [Mur89].

2.1.2 Classes de Redes de Petri

Atualmente, o termo *redes de Petri* é usado genericamente para referenciar um conjunto de modelos que podem ser agrupados em classes.

Os primeiros modelos de redes de Petri pertencem à classe das *redes de Petri de Baixo Nível*, nas quais as inscrições associadas aos arcos da estrutura da rede definem apenas os pesos desses arcos, as fichas são indistintas e a ocorrência das transições é instantânea. Nessa classe, destacam-se as redes de Petri Elementares [Thi86] e as redes de Petri Lugar/Transição [Mur89].

Entretanto, quando usadas para modelar sistemas complexos (grandes), as redes de Petri de Baixo Nível apresentam algumas restrições. Uma delas é a necessidade de duplicação da estrutura de rede para modelar processos semelhantes ou idênticos. Como exemplo, observe que na rede Lugar/Transição da Figura 2.1 (ver página 6) a sub-rede para o processo A tem estrutura igual àquela do processo B. Essa replicação decorre do fato de ser impossível diferenciar os processos por meio de fichas.

Uma outra limitação das redes de Baixo Nível é a carência de elementos para estudo de desempenho dos sistemas modelados, pois a ocorrência de suas transições é instantânea.

Dessa forma, visando suprir tais limitações e facilitar a modelagem de sistemas complexos, extensões foram propostas para as redes da classe de Baixo Nível. Dentre elas, tem-se a classe das *redes de Petri Temporais* [MBC+95] e a classe das *redes de Petri de Alto Nível*.

As redes de Petri de Alto Nível caracterizam-se sobretudo pela incorporação da teoria de tipos de dados. As fichas para estas redes podem carregar informação complexa, que é manipulada pelo uso de uma linguagem.

O fato das fichas expressarem informação complexa aumenta o poder de descrição dessas redes e, conseqüentemente, modelos mais compactos podem ser obtidos. Essa flexibilidade na manipulação da informação permite ao projetista distribuir a comple-

xidade do modelo de um sistema entre as inscrições e a estrutura da rede.

Dentre as redes de Petri de Alto Nível, as mais notáveis são as redes Predicado/Transição [Gen87] e as redes de Petri Coloridas [Jen92]. Estas últimas são introduzidas na Seção 2.2 e servem de base ao modelo de Rede de Petri Orientada a Objetos - RPOO.

No entanto, as redes de Alto Nível também apresentam limitações no que se refere a mecanismos de modularização e abstração necessários para descrever sistemas complexos. Nesse sentido, objetivando incrementar o poder de expressão dessas redes, foram introduzidas as *redes de Petri Hierárquicas* [Jen92], incorporando modularização, e as *redes de Petri Orientadas a Objetos* [Gue99], integrando estratégias de tratamento de complexidade da orientação a objetos, tais como classificação, herança, agregação e associação.

A classe de redes de Petri Orientadas a Objetos compreende o modelo no qual nosso estudo se baseia.

2.2 Redes de Petri Coloridas

As redes de Petri Coloridas servem de base ao modelo de Rede de Petri Orientada a Objetos. Elas pertencem à classe das redes de Petri de Alto Nível, sendo muito utilizadas na modelagem de aplicações complexas. Uma explicação detalhada sobre esse tipo de rede pode ser encontrada em [Jen92].

A presente seção trata das definições das redes de Petri Coloridas que são relevantes para a compreensão e o estudo de RPOO.

Uma rede de Petri Colorida compõe-se de três partes distintas: *estrutura*, *declarações* e *inscrições*. A estrutura é formada por lugares, transições e arcos direcionados, sendo definida como na Definição 2.1. As declarações definem conjuntos de cores (domínios), variáveis e operações (funções) usadas nas inscrições. As inscrições, por sua vez, podem ser de quatro tipos:

1. *Cores dos Lugares:* determinam a cor associada ao lugar. Um lugar só pode comportar fichas cujos valores respeitem sua cor;

2. *Guardas:* são expressões booleanas que restringem a ocorrência das transições;

3. *Expressões dos Arcos:* servem para manipular a informação contida nas fichas;

4. *Inicializações:* associadas aos lugares, estabelecem a marcação inicial da rede.

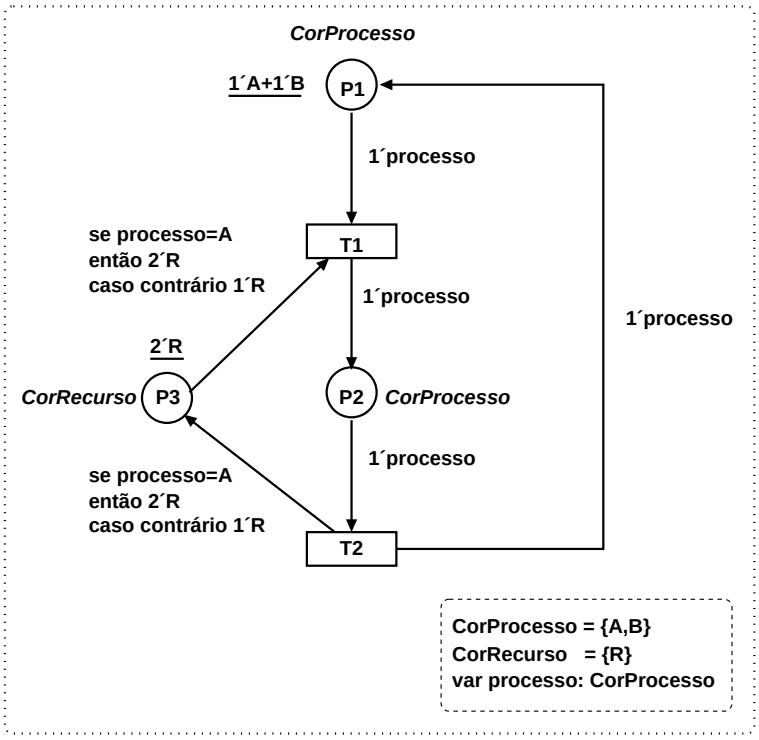

Figura 2.2: Exemplo de rede de Petri Colorida.

Um exemplo de uma rede de Petri Colorida é apresentado na Figura 2.2. As declarações estão expressas na caixa de linhas tracejadas no canto inferior direito. Os textos, em itálico, próximos aos lugares indicam suas cores e as expressões sublinhadas, suas inicializações. As expressões dos arcos localizam-se junto aos arcos direcionados e inexistem guardas associadas a transições. Além disso, perceba que a expressão de inicialização tem a forma 1'A+1'B. Isso ocorre porque, como fichas com valores idênticos podem estar contidas num mesmo lugar, as expressões de inicialização são expressas como *multiconjuntos*. Um multiconjunto é um conjunto em que pode haver *repetição* de elementos. Neste trabalho, denota-se por X^{mc} a família de funções $[X \to \mathbb{N}]$ de multiconjuntos de X.

Nas definições a seguir, considere que:

- $\Sigma = \langle \mathcal{S}, \mathcal{O} \rangle$ é uma assinatura, na qual:

  - $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ é um conjunto finito de nomes de conjuntos;
  - $\mathcal{O} = \{o_1, o_2, \ldots, o_n\}$ é um conjunto finito de símbolos operacionais;

- $\mathcal{V}_\Sigma = V_{S_1} \cup V_{S_2} \cup \cdots \cup V_{S_n}$ é um conjunto de variáveis de Σ, formado pela união disjunta de conjuntos de variáveis indexados por $\mathcal{S}$, de modo a que cada variável de $\mathcal{V}_\Sigma$ esteja associada a um sorte S_i (seja do tipo S_i);
- $T_\Sigma = T_{S_1} \cup T_{S_2} \cup \cdots \cup T_{S_n}$ é o conjunto dos termos de Σ, considerando $\mathcal{V}_\Sigma$, formado pela união disjunta de conjuntos de termos indexados por $\mathcal{S}$, sendo que cada termo de T_Σ está associado a um sorte S_i;
- E_Σ é o conjunto de todas as equações sobre Σ;

Definição 2.6 (Inscrições) *Seja $N = \langle P, T, F\rangle$ uma estrutura de rede. Uma inscrição baseada em Σ para a estrutura é uma quádrupla de funções totais $\langle c, g, e, i\rangle$, em que:*

- $c : P \to \mathcal{S}$ *é uma função de domínios ou cores para os lugares;*
- $g : T \to E_\Sigma$ *é uma função de guardas para as transições*
- $e : F \to T_\Sigma^{mc}$ *é uma função de expressões de arcos, tal que:*
  - *se $f = \langle p, t\rangle \in F$ ou $f = \langle t, p\rangle \in F$ e $c(p) = S_i$ então $e(f) \in T_{S_i}^{mc}$*
- $i : P \to T_\Sigma^{mc}$ *é uma função de inicialização para os lugares, tal que:*
  - *se $p \in P$ e $c(p) = S_i$ então $i(p) \in T_{S_i}^{mc}$*

O par $\langle N, \langle c, g, e, i\rangle\rangle$, ou simplesmente a tupla $\langle N, c, g, e, i\rangle$ é denominada estrutura de rede inscrita.

Definição 2.7 (Rede de Petri Colorida) *Uma rede de Petri colorida é uma tripla $\langle \Sigma, N, I\rangle$, tal que:*

- $\Sigma = \langle \mathcal{S}, \mathcal{O}\rangle$ *é uma assinatura;*
- $N = \langle P, T, F\rangle$ *é uma estrutura de rede de Petri;*

- $I = \langle c, g, e, i \rangle$ *é uma inscrição para* N*, baseada na assinatura* Σ*.*

Para explicar a *regra de ocorrência* para redes de Petri Coloridas, é essencial compreender os conceitos de *variável de transição*, *ligação*[2] e *elemento de ligação*. As variáveis de transição são aquelas presentes nos arcos direcionados. Por exemplo, na Figura 2.2, a transição $T1$ tem *processo* como sua variável de transição. Uma ligação corresponde a associação de uma variável de transição a um valor da sua cor. Por exemplo, para a mesma variável de transição *processo*, são possíveis as ligações b_1=$\langle$*processo*=A$\rangle$ e b_2=$\langle$*processo*=B$\rangle$. Um elemento de ligação é um par (transição, ligação). Como ilustração, considerando os exemplos anteriores, temos be_1=($T1$, b_1=$\langle$*processo*=A$\rangle$) e be_2=($T1$, b_2=$\langle$*processo*=B$\rangle$) como elementos de ligação.

A rede de Petri Colorida da Figura 2.2 modela uma situação idêntica àquela explicada para a rede de Petri Lugar/Transição da Figura 2.1. Entretanto, observe que uma mesma estrutura foi usada para modelar os dois processos - A e B - porque estes podem ser diferenciados por meio das fichas. As inscrições de inicialização dessa rede estabelecem que sua marcação inicial contém um processo do tipo A, um processo do tipo B e dois recursos tipo R. Conseqüentemente, apenas a transição $T1$ está habilitada a ocorrer, pois ela é a única a apresentar algum elemento de ligação satisfeito pela marcação inicial. Nessa situação, pode ocorrer be_1 ou be_2. Se be_1 ocorrer, *uma* ficha A é removida de $P1$, *duas* fichas R de $P3$ e *uma* ficha A é adicionada a $P2$ (pois em be_1 *processo* está ligada a A). Vale ressaltar que um elemento de ligação só está habilitado se satisfizer a guarda da transição. Assim, se para a transição $T1$ fosse associada a guarda [*processo* = A], apenas be_1 estaria habilitado a ocorrer.

2.3 RPOO - Rede de Petri Orientada a Objetos

Rede de Petri Orientada a Objetos (RPOO) é uma ferramenta conceitual formal para especificação de sistemas de *software* concorrentes, distribuídos e orientados a objetos. Seu modelo conceitual de computação foi influenciado pela teoria de Atores [Agh86]. Um sistema modelado em RPOO é formado por uma coleção de objetos concorrentes entre si, que se comunicam *assincronamente*. Os objetos são espacialmente distribuídos

[2]Tradução para *binding*.

e cada qual possui um identificador único, que é utilizado para destinar as mensagens. Assume-se a existência de um ambiente de comunicação que garante a entrega de mensagens, mas sem suporte de mecanismos de ordenação. A coleção de objetos que caracteriza um sistema pode variar dinamicamente por meio da criação ou destruição desses. Portanto, a topologia de interconexão dos objetos é dinâmica.

Os objetos são agrupados em classes e encapsulam *estado*, *comportamento* e *linhas de controle*. A possibilidade de existência de múltiplas linhas de controle proporciona um comportamento interno concorrente.

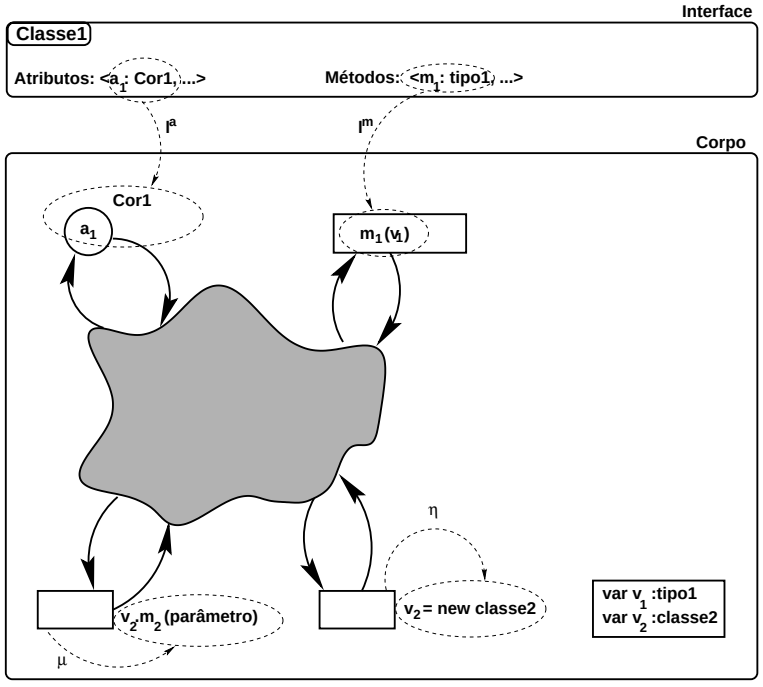

Figura 2.3: Representação gráfica de uma Rede de Petri Orientada a Objetos.

Uma classe em RPOO compõem-se de uma *interface* e de um *corpo* (ver Figura 2.3). A *interface* contém os métodos para *acesso* aos objetos de uma classe, enquanto o corpo possui a descrição do *comportamento* dos objetos de uma classe e é representado por uma rede de Petri Colorida, definida na Seção 2.2. Esses dois conceitos são formalizados nas Definições 2.8 e 2.9. Para melhor compreendê-las, considere que:

- $\mathcal{C} = \{C_1, C_2, \ldots, C_n\}$ é um conjunto finito de nomes de classes;
- $\mathcal{V}_\mathcal{C} = V_{C_1} \cup V_{C_2} \cup \cdots \cup V_{C_n}$ é um conjunto de variáveis indexadas por $\mathcal{C}$;
- $\Sigma' = \langle \mathcal{S}', \mathcal{O} \rangle$ é uma assinatura, na qual:

  - $\mathcal{S}' = \mathcal{S} \cup \mathcal{C}$ é a união de um conjunto finito de nomes de conjuntos com o conjunto de nomes de classes;

- $\vec{\mathcal{O}}_{\mathcal{C}} = \vec{O}_{C_1} \cup \vec{O}_{C_2} \cup \cdots \cup \vec{O}_{C_n}$ é um conjunto de nomes de objetos indexados por $\mathcal{C}$;

- $\mathcal{M}$ é um conjunto de nomes de métodos.

Definição 2.8 (Interface) *Uma interface é uma tripla $\langle A, M, \tau \rangle$, onde:*

- *A é um conjunto finito de nomes de atributos;*

- *$M \subseteq \mathcal{M}$ é um conjunto finito de nomes de métodos;*

- *$\tau : A \cup M \to \mathcal{S}'$ é uma aplicação que a cada nome de atributo e de método faz corresponder um nome de conjunto ou de classe;*

- *A e M são disjuntos, ou seja, $A \cap M = \varnothing$.*

Cada atributo e método possuem um tipo associado, o qual é determinado pela aplicação τ. Dessa forma, para a Figura 2.3, em que se exibe a notação para a representação gráfica de uma classe RPOO, tem-se que $\tau(a_1) = Cor1$ e $\tau(m_1) = tipo1$.

Definição 2.9 (Classe) *Uma sêxtupla $\langle \mathcal{I}, \mathcal{N}, l^a, l^m, \eta, \mu \rangle$ é uma classe baseada em Σ' se, e somente se, as seguintes condições são satisfeitas:*

- *$\mathcal{I} = \langle A, M, \tau \rangle$ é uma interface baseada em Σ';*

- *$\mathcal{N} = \langle \Sigma', N, I \rangle$ é uma rede de Petri Colorida, denominada corpo da classe (considere $N = \langle P, T, F \rangle$ e $I = \langle c, g, e, i \rangle$);*

- *$l^a : A \to P$ é uma aplicação que a cada atributo associa um lugar da estrutura, respeitando os tipos, ou seja, para todo atributo $a \in A$, se $p = l^a(a)$ então $c(p) = \tau(a)$;*

- *$l^m : M \to T \times \mathcal{V}_{\Sigma'}$ é uma aplicação que a cada método associa um conjunto de elementos de ativação que respeita os tipos, ou seja, se $\langle t, v \rangle \in l^m(m)$ e $v \in \mathcal{V}_{S_i}$ então $\tau(m) = S_i$;*

- $\eta : T \rightarrow 2^{\mathcal{V}_{\mathcal{C}} \times \mathcal{C}}$ *é uma aplicação que a cada transição associa um conjunto de inscrições de criação de objetos, tais que, se* $\langle v, C_i \rangle \in \eta(t)$ *então* $v \in V_{C_i}$*;*

- $\mu : T \rightarrow 2^{\mathcal{V}_{\mathcal{C}} \times \mathcal{M} \times T_{\Sigma'}}$ *é uma aplicação que a cada transição associa um conjunto de inscrições de mensagens.*

Segundo a Definição 2.9, há uma associação (ilustrada na Figura 2.3) entre os atributos e métodos da *interface* da classe e os elementos da rede de Petri Colorida no corpo. Mais especificamente, tem-se que:

- Cada atributo na *interface* está relacionado a um lugar no corpo e seu tipo deve coincidir com o conjunto de cores do lugar (aplicação l^a);

- Cada método na *interface* está associado a uma transição (chamada *transição de ativação*) e a uma variável dessa transição (aplicação l^m). O tipo da variável deve coincidir com o tipo do método;

- Inscrições podem ser associadas a transições para indicar a invocação de métodos (aplicação μ) ou para efetuar a criação de novos objetos (aplicação η).

A especificação de um sistema em RPOO é composta da descrição das classes de todos os objetos existentes no sistema, acompanhada da declaração dos objetos iniciais. Esse conjunto de classes deve atender a restrições que o caracterizem como um *conjunto integrado de classes*. Isso significa que: (i) cada inscrição de mensagem deve referenciar um método existente em outra classe, respeitando seu tipo, e (ii) incrições para criação de objetos só devem referenciar classes presentes no sistema, formalmente:

Definição 2.10 (Conjunto Integrado de Classes) *Um conjunto de classes* $K = \{K_{C_1}, K_{C_2}, \ldots, K_{C_n}\}$ *baseado em* Σ' *e indexado por* $\mathcal{C}$ *é dito integrado se, e somente se, para qualquer classe* $k_i \in K$*, toda inscrição de mensagem presente em* k_i *respeita a interface da classe à qual se refere, ou seja*[3]*:*

[3]Sempre que for necessário diferenciar entre os componentes de cada classe utilizar-se-á o nome da classe anteposto ao componente. Por exemplo, a interface de k_1 poderia ser denotada por ${}^{k_1}\mathfrak{I}$, e a estrutura de rede do corpo da classe k_2 por ${}^{k_2}N = \langle {}^{k_2}P, {}^{k_2}T, {}^{k_2}F \rangle$.

- *se* $\langle v, m, te \rangle$ *é uma inscrição de mensagem presente no corpo da classe* $k_i = K_{C_i} \in K$*, onde*
  - $v \in V_{C_j}$*,* C_j *é o nome da classe* k_j *referida, ou seja* $k_j = K_{C_j} \in K$*; e*
  - $te \in T_{S_x}$ *é um termo de tipo* S_x*, para algum* $S_x \in \mathcal{S}$ *da assinatura* Σ'*,*
- *então necessariamente*
  - $m \in {}^{k_j}M$*, ou seja, o nome de método invocado é válido na classe* k_j*; e*
  - ${}^{k_j}\tau(m) = S_x$*, ou seja, o tipo do termo respeita o tipo do método da classe* k_j*.*

Por fim, vale ressaltar que a *regra de ocorrência* para o corpo de uma RPOO é semelhante àquela explicada, na Seção 2.2, para as redes de Petri Coloridas. A diferença decorre da comunicação entre os objetos.

Os objetos de um sistema RPOO comunicam-se através do envio assíncrono de mensagens. Essa interação é unidirecional e envolve dois eventos: o *envio* e a *recepção* da mensagem.

O envio de uma mensagem corresponde à ocorrência de uma transição que tem associada invocação de métodos (aplicação μ) em outros objetos. A recepção de uma mensagem equivale à ativação de tais métodos nos referidos objetos. Assim, quando uma transição com invocação ocorre, o ambiente de comunicação transfere uma mensagem (formalizada na Definição 2.11) de um objeto para outro. Nesse caso, a transição de ativação do método presente na mensagem poderá ficar habilitada a ocorrer. Veja que, embora o ambiente garanta a entrega da mensagem, a sua efetiva recepção dependerá da ocorrência da transição de ativação do método.

Definição 2.11 (Mensagem) *Uma mensagem é uma tripla* $\langle \vec{o}, m, d \rangle$*, onde:*

- $\vec{o} \in \vec{O}_{C_i}$ *é o nome do objeto destinatário;*
- $m \in \mathcal{M}$ *é o nome do método invocado; e*
- $d \in D$ *é um elemento de dado passado como parâmetro,*

tais que se $\vec{o} \in \vec{O}_{C_i}$ *então* $m \in {}^{k_i}M$ *e se* ${}^{k_i}\tau(m) = S_x$, *para algum* $S_x \in \mathbb{S}'$, *então* $d \in D_{S_x}$.

Detalhes sobre o formalismo de RPOO podem ser encontrados em [Gue99].

Capítulo 3

Comunicação Síncrona

O formalismo básico de RPOO suporta apenas a comunicação assíncrona para interação entre objetos. Como ferramenta para construção de modelos distribuídos, concorrentes e orientados a objetos, esse mecanismo é suficiente porque possibilita a construção de soluções que apresentam custos máximos. Qualquer outra solução baseada em mecanismos diversos exibirá custos iguais ou inferiores ao da solução baseada em comunicação assíncrona [BM93].

Entretanto, é interessante que notações para especificação de sistemas distribuídos e concorrentes disponibilizem abstrações de comunicação (interação) que simplifiquem a tarefa do projetista [AFK+93]. Em especial, tais notações devem prover abstrações para os mecanismos de interação normalmente presentes em sistemas distribuídos.

Nesse sentido, neste capítulo apresenta-se a incorporação da abstração para comunicação síncrona ao modelo RPOO, visto que esse tipo de comunicação é comum em sistemas distribuídos [Mul93]. Na Seção 3.1 introduzem-se os conceitos desse tipo de comunicação. Na Seção 3.2 apresenta-se informalmente a notação proposta, sendo que a formalização está na Seção 3.3.

3.1 Conceito

A *Transferência Síncrona de Mensagem* ou *Comunicação Síncrona* é aquela em que o agente computacional remetente da comunicação aguarda que o agente computacional destinatário reconheça o recebimento da (ou de outra forma responda à) comunicação

antes de prosseguir seu processamento [Agh86].

Neste trabalho introduz-se suporte à comunicação síncrona no modelo RPOO tendo por base a comunicação assíncrona já existente. Para facilitar explicações futuras, foram convencionados os seguintes termos:

- *Emissor*: objeto RPOO remetente da mensagem (ou da invocação do método);
- *Receptor*: objeto RPOO destinatário da mensagem (ou que define o método a ser ativado);
- *Comunicação com reconhecimento*: comunicação síncrona *com reconhecimento*;
- *Comunicação com resposta*: comunicação síncrona na qual *a resposta já é o resultado do processamento* do método invocado no receptor.

3.2 Construções do Novo Modelo

Os métodos presentes no formalismo básico de RPOO são assíncronos. Entretanto, para permitir a introdução de comunicação síncrona em RPOO, seu formalismo deve suportar também *métodos síncronos*.

Os métodos síncronos são métodos invocados/ativados via mensagens síncronas. Eles diferem dos assíncronos porque são de um tipo composto $\langle$*tipoSíncrono, tipoMétodo*$\rangle$. O *tipoMétodo* indica o tipo do método propriamente dito e o *tipoSíncrono*, qual o tipo de comunicação síncrona para o método. Observe que, na Definição 2.8 do formalismo básico (ver página 17), os métodos (assíncronos) possuem um tipo simples $\langle tipoMétodo \rangle \in \mathcal{S}'$, que é dado pela aplicação τ.

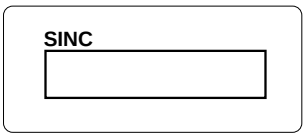

Figura 3.1: Representação gráfica de uma transição para comunicação síncrona.

Uma transição que possui invocação/ativação de método síncrono é chamada de *transição síncrona* e sua representação gráfica está na Figura 3.1.

Transições síncronas têm a regra de ocorrência modificada. Para explicar essa mudança, considere que t_{sinc} e t são transições com métodos síncronos e assíncronos respectivamente associados. A ocorrência de t implica no envio de requisição para ativação de um método em um objeto, enquanto que a ocorrência de t_{sinc} significa que houve o envio de uma requisição de ativação de um método num dado objeto e esse já recebeu tal requisição, enviando o reconhecimento, ou já ativou e executou o referido método, retornando o resultado da computação.

Os métodos síncronos e a nova regra de ocorrência das transições síncronas permitem ao projetista abstrair, sempre que desejar, detalhes da notação RPOO original.

Dessa forma, para especificar uma comunicação síncrona o projetista deve adicionar uma transição síncrona - t_{sinc_E} - na classe do emissor e uma transição síncrona - t_{sinc_R} - correspondente na(em cada uma das) classe(s) do(s) receptor(es). A transição t_{sinc_E} contém as inscrições de mensagem para ativação dos métodos síncronos. Como a comunicação pode ser com reconhecimento ou com resposta, os métodos devem ter como primeiro termo as respectivas palavras-chave *ACK* ou *REPLY* para indicar o tipo de comunicação síncrona a ser usada. A transição t_{sinc_R} corresponde à ativação de um método invocado na t_{sinc_E}.

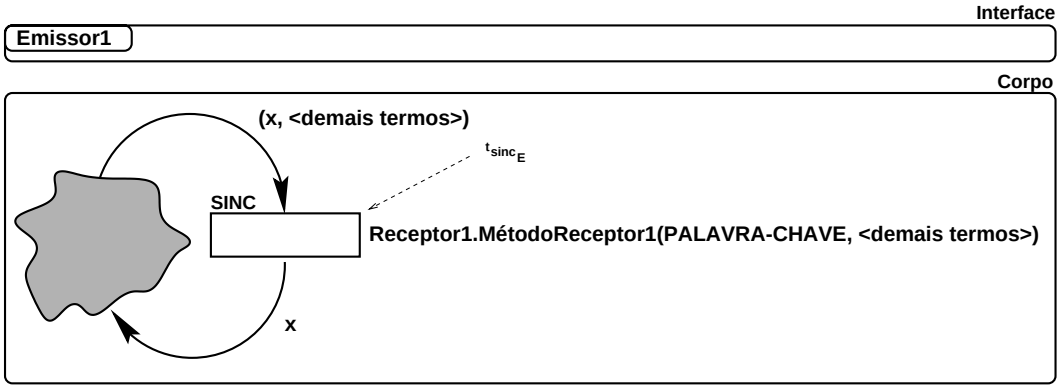

Figura 3.2: Emissor genérico usando comunicação síncrona.

Como exemplo genérico, considere a situação ilustrada nas Figuras 3.2 e 3.3, em que o *Emissor1* invoca sincronamente o *MétodoReceptor1* no *Receptor1*. Observe a existência de transições síncronas em ambos os objetos. No emissor, a inscrição de mensagem *Receptor1.MétodoReceptor1*(*PALAVRA-CHAVE, <demais termos>*) invoca sincronamente o *MétodoReceptor1* presente na *interface* de *Receptor1*. No receptor, o

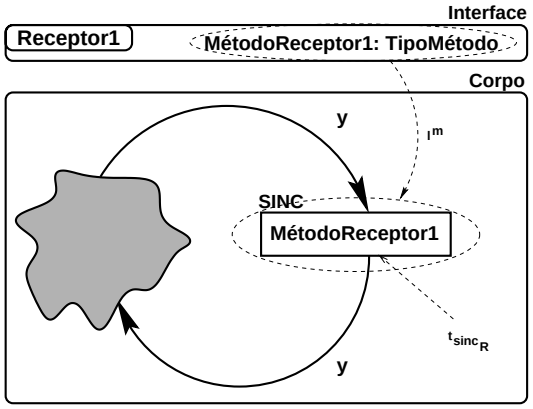

Figura 3.3: Receptor genérico usando comunicação síncrona.

MétodoReceptor1 tem tipo $\langle tipoSíncrono,\ tipoMétodo \rangle$, em que *tipoSíncrono* igual a $TipoACK(TipoREPLY)$ se *PALAVRA-CHAVE* igual a $ACK(REPLY)$, e se relaciona a t_{sinc_R} por meio da aplicação l^m.

Por fim, vale ressaltar que:

1. Cada t_{sinc_E} presente no emissor deve possuir uma ou mais t_{sinc_R} correspondentes no(s) receptor(es), dependendo da quantidade de métodos síncronos diferentes que invoca;

2. Embora transições síncronas possam definir simultâneas invocações de métodos síncronos e assíncronos, apenas um dos tipos de comunicação síncrona pode ser utilizado por transição;

3. Uma transição pode estar relacionada à *ativação* de *apenas um* método síncrono;

4. Cada transição é *unicamente* identificável dentro de uma classe.

Para garantir a compatibilidade entre modelos RPOO contendo transições síncronas e a notação original de RPOO, deve ser possível mapear cada aparição de uma transição síncrona em uma sub-rede equivalente, a qual utilize apenas transições assíncronas. Esse mapeamento deve ser especificado tanto para o caso em que a transição contém invocação de métodos síncronos como para o caso em que contém ativação de tais métodos e deve considerar a ligação das variáveis da t_{sinc} envolvida no mapeamento. Assim, para o exemplo das Figuras 3.2 e 3.3, o mapeamento deve garantir que as

variáveis x e y serão ligadas ao mesmos valores em todos os arcos das respectivas transições.

Como transições síncronas devem pertencer a apenas um tipo de comunicação síncrona, separou-se a explicação para os mapeamentos. Conseqüentemente, o mapeamento para comunicação com reconhecimento está na Seção 3.2.1 e o com resposta, na Seção 3.2.2.

3.2.1 Mapeamento para Comunicação Síncrona com Reconhecimento

O mapeamento de uma transição síncrona envolvida em uma comunicação com reconhecimento difere entre o lado emissor e o receptor. Então, visando facilitar a explanação, considere que t_{sinc_E} (t_{sinc_R}) é uma transição síncrona presente na classe do emissor (receptor) e contendo invocação (ativação) de um método síncrono.

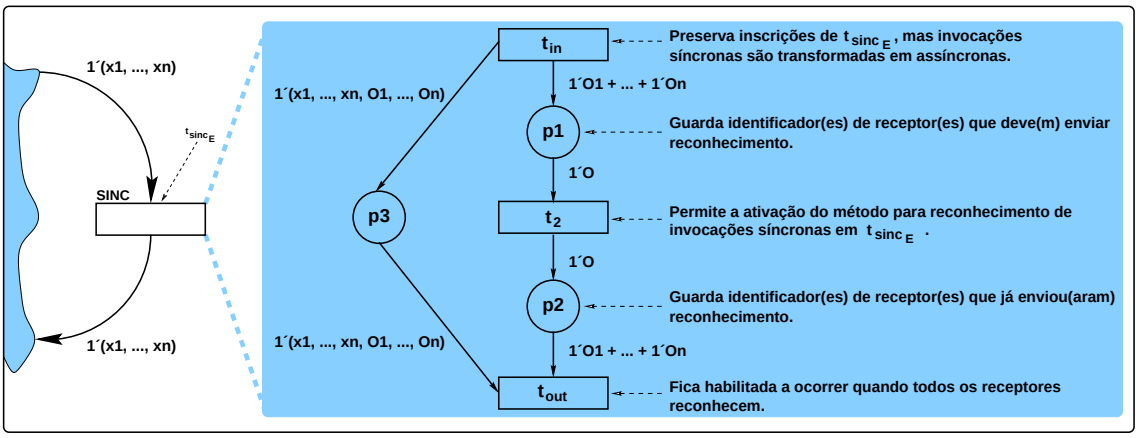

Figura 3.4: Mapeamento de uma transição síncrona usando comunicação com reconhecimento na classe de um emissor.

O mapeamento de uma transição síncrona no formalismo básico acontece da seguinte forma:

- Na classe do *emissor*, a transição t_{sinc_E} desdobra-se na subestrutura de rede de Petri Colorida ilustrada na Figura 3.4.

  A primeira transição (t_{in}) do mapeamento mantém as inscrições associadas a t_{sinc_E}. Contudo, como as redes resultantes do mapeamento devem trocar apenas

mensagens assíncronas, os métodos síncronos são transformados em assíncronos. Para tanto, não apresentam mais a palavra-chave *ACK* como primeiro argumento de dado.

O lugar p_1 adicionado contém informação sobre que objetos ainda não reconheceram o recebimento da mensagem.

A transição t_2 permite que o(s) receptor(es), cujo(s) método(s) síncrono(s) foi(ram) invocado(s) em t_{sinc_E}, envie(m) o *reconhecimento* do recebimento da mensagem síncrona. Como conseqüência, método(s) assíncrono(s) para envio de reconhecimento de método(s) síncrono(s) invocado(s) em uma t_{sinc_E} deve(m) ser adicionado(s) à *interface* do emissor e relacionado(s) à t_2 através da aplicação l^m. O nome de cada método assíncrono adicionado é sempre composto pela concatenação da palavra-chave *ACK* ao nome do método síncrono invocado e seu tipo é *TipoACK*.

O lugar p_2 contém o(s) identificador(es) do(s) receptor(es) que reconheceu(ram) a requisição de ativação do(s) método(s) invocado(s) por t_{in}. A correta ligação para as variáveis da transição t_2 é possível porque cada mensagem contém, implicitamente, o identificador de quem a enviou.

O lugar p_3 armazena fichas com informação sobre o(s) valor(es) da(s) variável(is) de transição do(s) arco(s) de entrada de t_{sinc_E} e os identificadores dos objetos cujos métodos síncronos são invocados nessa transição. O objetivo é garantir a correta ligação da(s) variável(is) da transição síncrona mapeada.

A transição t_{out} só ocorre quando *todos* os receptores reconhecerem o pedido de ativação dos métodos invocados por t_{in}.

- Na classe do *receptor*, há um mapeamento semelhante ao explicado para a do emissor (ver Figura 3.5).

  A transição t_{in} preserva a ativação do método em t_{sinc_R}. Para tanto, tranforma-se esse método em assíncrono convertendo seu tipo composto $\langle$*tipoSícrono, tipoMétodo*$\rangle$ no tipo simples $\langle$*tipoMétodo*$\rangle$. Além disso, t_{in} apresenta uma inscrição de mensagem para envio de reconhecimento de ativação.

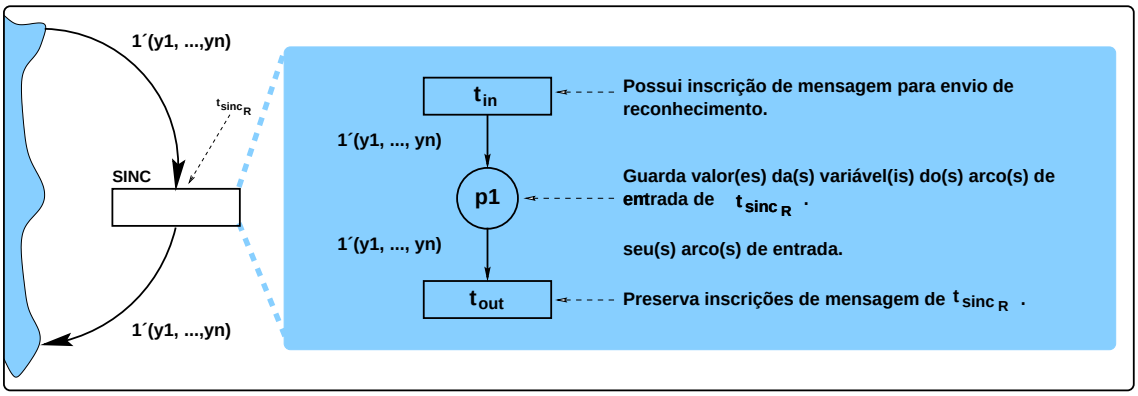

Figura 3.5: Mapeamento de uma transição síncrona usando comunicação com reconhecimento na classe de um receptor.

Vale salientar que tanto a mensagem que o ambiente de comunicação entrega ao receptor, contendo a requisição de ativação do método síncrono presente em t_{in}, como a mensagem que o ambiente entrega ao emissor, contendo o reconhecimento da ativação do método, incluem, em seus dados, o nome único da t_2, presente no emissor que gerou a comunicação. Isso acontece para facilitar o mapeamento numa situação em que duas transições síncronas distintas de um emissor invocam método(s) síncrono(s) de um mesmo receptor.

O lugar p_1 guarda o valor de cada uma das variáveis de transição dos arcos de entrada de t_{sinc_R}. A transição t_{out} preserva todas as inscrições associadas a t_{sinc_R}.

Como exemplo de aplicação do mapeamento até então explicado, considere um sistema RPOO com as classes *C1*, *C2*, *C3* e *C4*. Suponha que o projetista especifica que objetos de *C1* enviam mensagens *síncronas com reconhecimento* a objetos de *C2*, *C3* e *C4*, invocando *métodoX*, *métodoY* e *métodoZ*, respectivamente definidos em suas *interfaces*. Nessa situação (ilustrada nas Figuras 3.6 e 3.7), há uma transição t_{sinc} *em cada uma* das classes. Para a transição t_{sinc_E} em *C1*, associam-se inscrições invocando *métodoX*, *métodoY* e *métodoZ*, as quais têm a palavra-chave ACK e a variável a como argumentos de dado. A transição t_{sinc_R} em *C2* possibilita a ativação de *métodoX*. As figuras para as classes *C3* e *C4* foram omitidas porque são semelhantes à Figura 3.7, diferindo somente pela substituição de cada aparição do nome *métodoX* por *métodoY* e *métodoZ*, respectivamente.

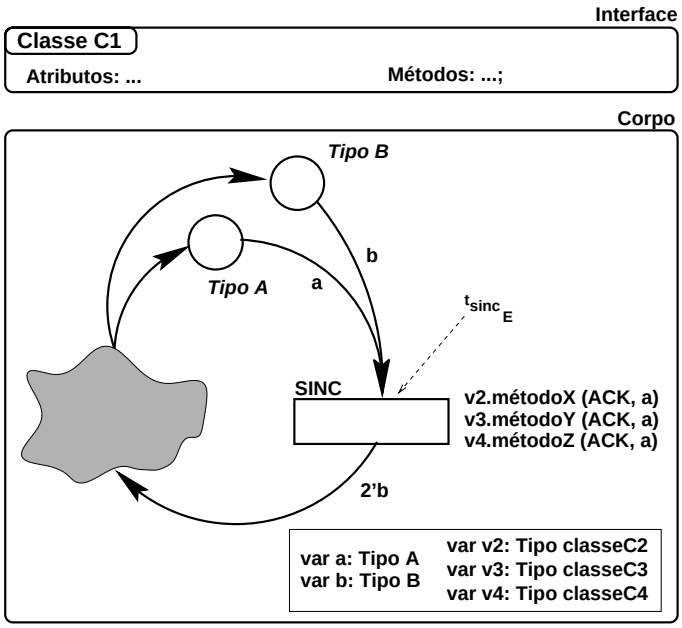

Figura 3.6: Classe de um emissor usando comunicação com reconhecimento.

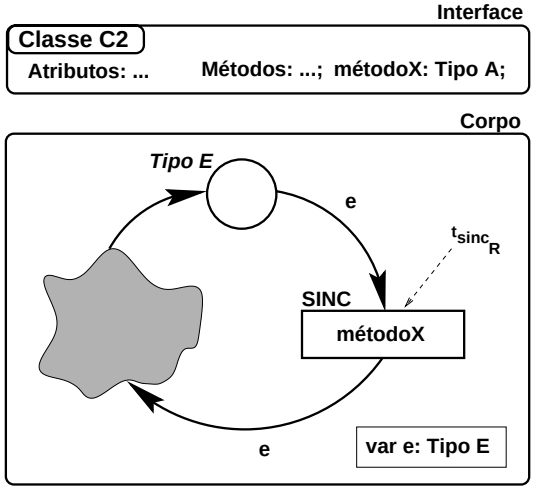

Figura 3.7: Classe de um receptor usando comunicação com reconhecimento.

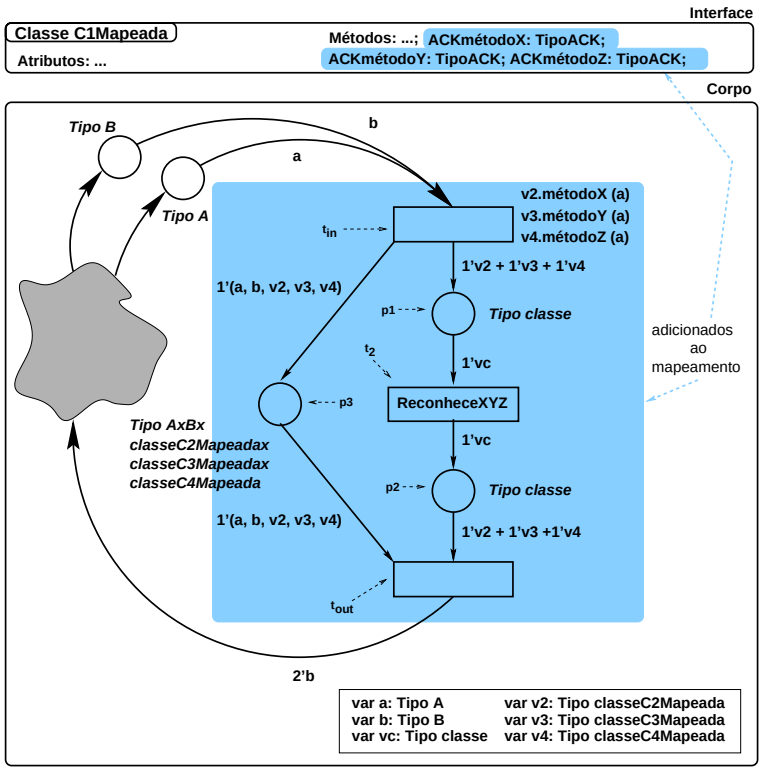

Figura 3.8: Mapeamento de uma classe de emissor usando comunicação com reconhecimento.

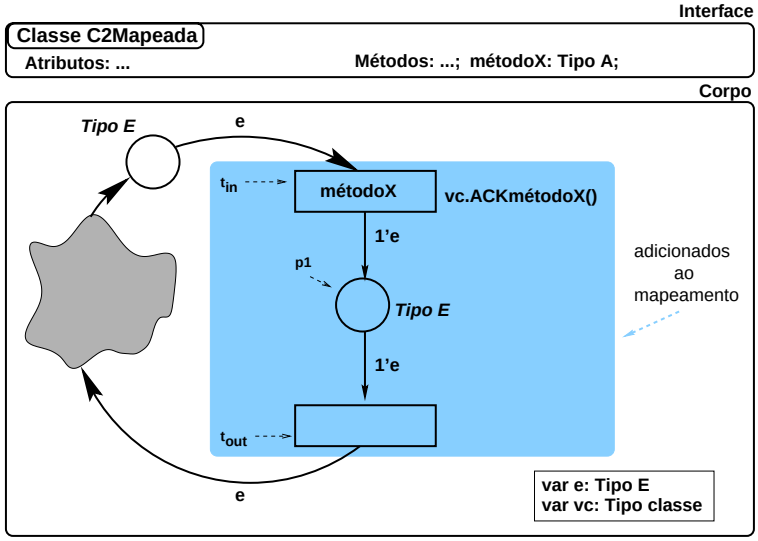

Figura 3.9: Mapeamento de uma classe de receptor usando comunicação com reconhecimento.

O mapeamento dessas classes no formalismo básico está representado graficamente nas Figuras 3.8 e 3.9. Perceba que, para *C1*, classe de um emissor, os métodos $ACKmétodoX$, $ACKmétodoY$ e $ACKmétodoZ$ foram acrescidos à *interface* e relacionados a t_2, isto é, ${}^{C1Mapeada}l^m(ACKmétodoX) = {}^{C1Mapeada}l^m(ACKmétodoY) = {}^{C1Mapeada}l^m(ACKmétodoZ) = \langle ReconheceXYZ, variáveis \rangle$. Para *C2*, classe de um receptor, o método para envio de reconhecimento, $ACKMétodoX$, está como inscrição de mensagem de t_{in}. Além disso, o tipo de $métodoX$ é o tipo simples $\langle TipoA \rangle$, pois ele foi convertido num método assíncrono. De outra forma, $\langle vc,\ ACKmétodoX,\ () \rangle \in {}^{C2Mapeada}\mu(t_{in})$ e ${}^{C2Mapeada}\tau(métodoX) = \langle TipoA \rangle$.

3.2.2 Mapeamento para Comunicação Síncrona com Resposta

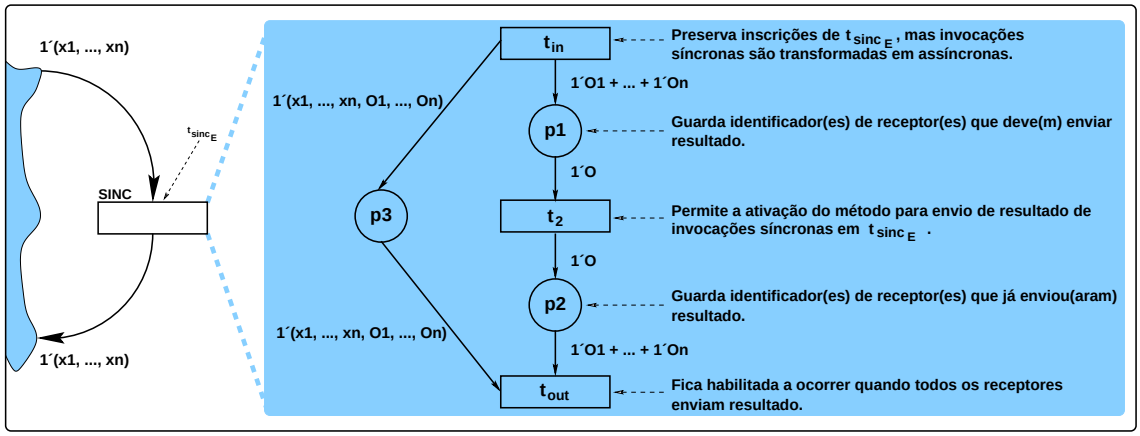

Figura 3.10: Mapeamento de uma transição síncrona usando comunicação com resposta na classe de um emissor.

O mapeamento de uma transição síncrona envolvida em uma comunicação com resposta é semelhante ao explicado, na seção anterior, para a comunicação com reconhecimento. As diferenças ocorrem na palavra-chave, que passa a ser *REPLY*, no uso do *TipoREPLY* ao invés de *TipoACK* e nas inscrições associadas aos elementos da sub-rede que substitui t_{sinc_R}. O mapeamento de uma transição síncrona na classe de um emissor está ilustrado na Figura 3.10 e na de um receptor, na Figura 3.11.

Perceba que, na classe do receptor, o método adicionado para envio de resultado está associado a t_{out} e as inscrições de t_{sinc_R}, a t_{in}. No mapeamento da seção anterior, há uma situação inversa. Essa inversão decorre da própria semântica da comunicação síncrona.

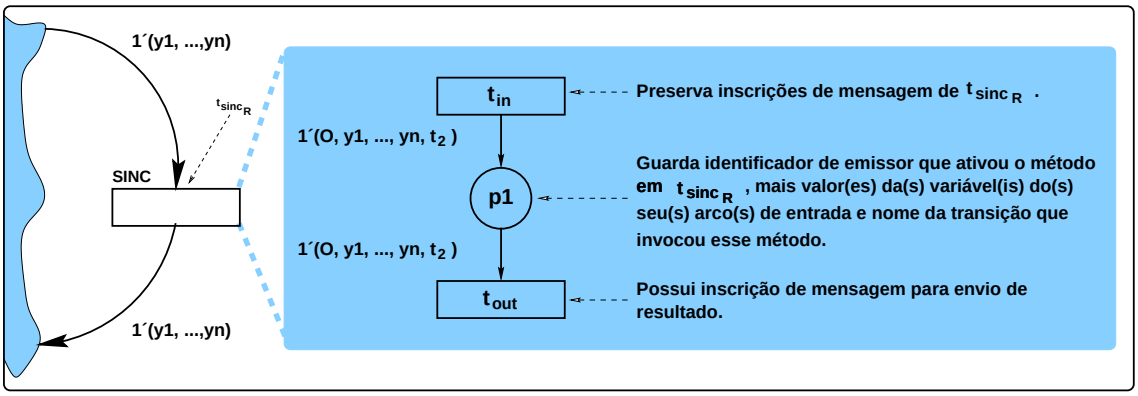

Figura 3.11: Mapeamento de uma transição síncrona usando comunicação com resposta na classe de um receptor.

Na comunicação *com reconhecimento*, o receptor reconhece a entrega (ocorrência de t_{in}) da mensagem, mas o seu efetivo processamento só acontecerá em algum tempo no futuro (ocorrência de t_{out}). Por outro lado, na comunicação *com resposta*, a entrega da mensagem já implica na ativação do método, por isso as inscrições de mensagem são preservadas em t_{in}.

Como última diferença, o lugar p_1 contém uma informação composta de um nome de emissor, o valor de cada uma das variáveis dos arcos de entrada de t_{sinc_R} e o nome da respectiva transição t_2, presente no emissor que invoca o método ativado em t_{in}, no receptor.

A informação sobre o nome da transição t_2 precisa ser guardada porque, conforme visto na Seção 3.2.1 (ver página 27) a mensagem que o ambiente de comunicação entrega ao emissor, contendo o resultado da ativação do método, deve incluir, em seus dados, o nome da t_2 presente no emissor que gerou a comunicação.

Como exemplo, considerando o sistema de classes RPOO da Seção 3.2.1, suponha que o projetista especifica a invocação de *métodoX*, *métodoY* e *métodoZ* com *comunicação síncrona com resposta*. Nesse caso, o projetista realiza um procedimento semelhante ao descrito para comunicação com reconhecimento. A diferença reside no uso da palavra-chave *REPLY*. As Figuras 3.12 e 3.13 ilustram essa nova situação.

O mapeamento dessas classes no formalismo básico também é semelhante ao apresentado na Seção 3.2.1 e está representado graficamente nas Figuras 3.14 e 3.15. Veja que, para *C2*, o método adicionado para envio de resultado

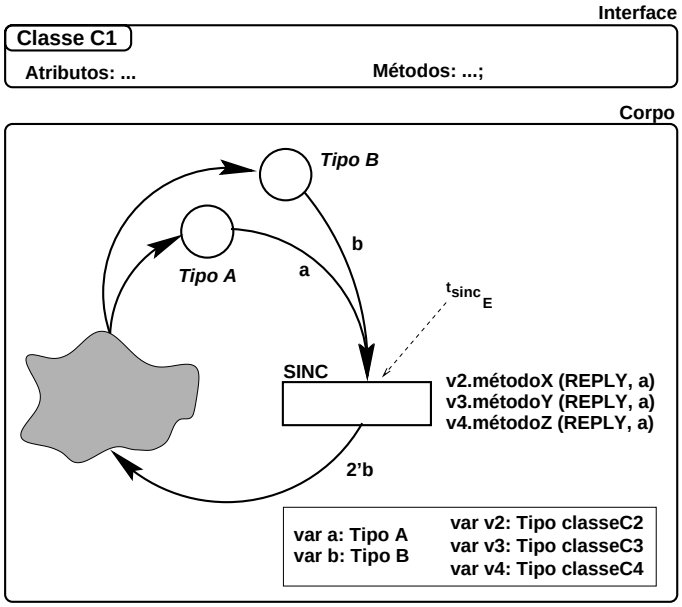

Figura 3.12: Classe de um emissor usando comunicação síncrona com resposta.

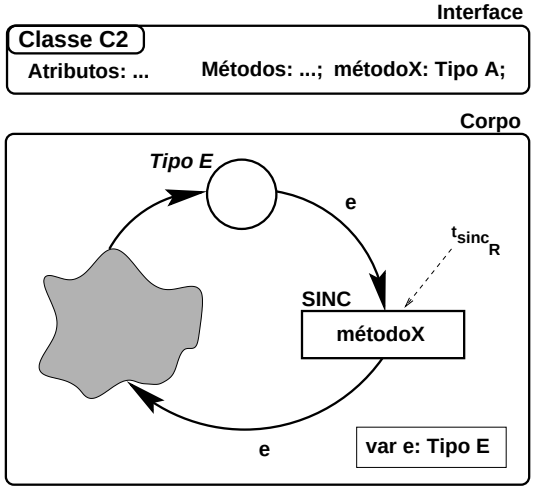

Figura 3.13: Classe de um receptor usando comunicação síncrona com resposta.

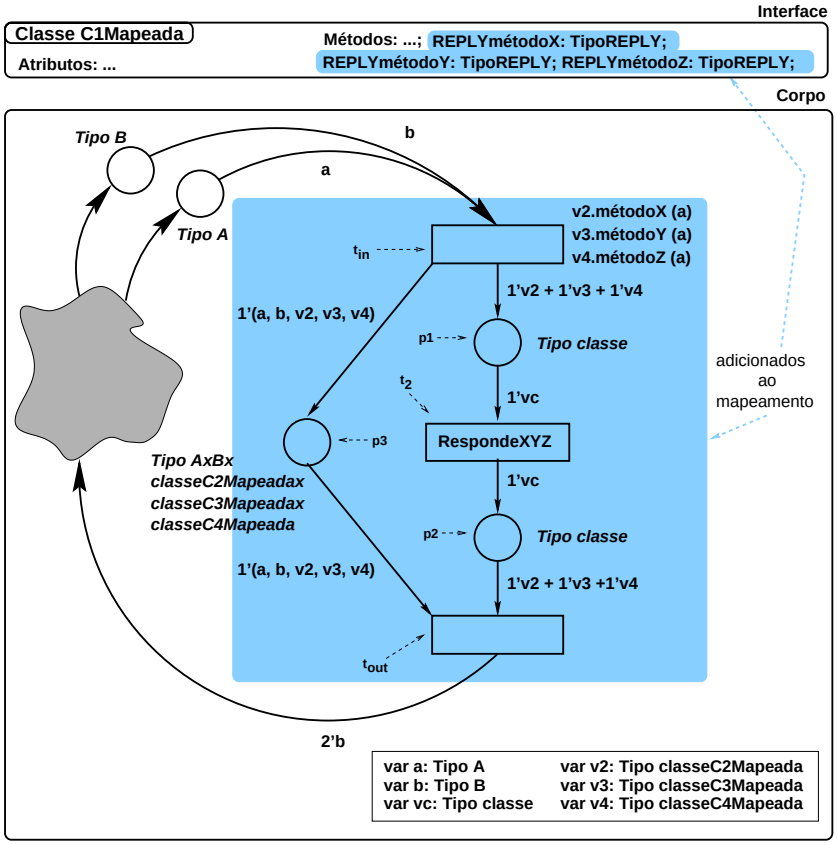

Figura 3.14: Mapeamento de uma classe de emissor usando comunicação síncrona com resposta.

$(REPLYmétodoX)$ agora está associado a t_{out}. Ou seja, $\langle vc,\ REPLYmétodoX,\ ()\rangle \in {}^{C2Mapeada}\mu(t_{out})$. Além disso, na ocorrência de t_{in}, tem-se como ligação $b = \langle vc = um\_nome\_de\_objeto\_de\_C2,\ e = um\_valor\_de\_TipoE,\ t = RespondeXYZ\rangle$.

3.3 Formalização de Comunicação Síncrona

Para que o modelo RPOO suporte a comunicação síncrona, algumas de suas definições devem ser alteradas. Assim sendo, a *interface* deve permitir a ativação de métodos síncronos, o corpo da *classe* precisa incorporar transições para invocação de tais métodos, as *mensagens* podem ser síncronas e o *conjunto integrado de classes* deve garantir, adicionalmente as suas restrições originais, a correta invocação/ativação de métodos síncronos.

Nas Definições 3.1 - 3.4 formalizam-se esses conceitos. A fim de melhor entendê-las,

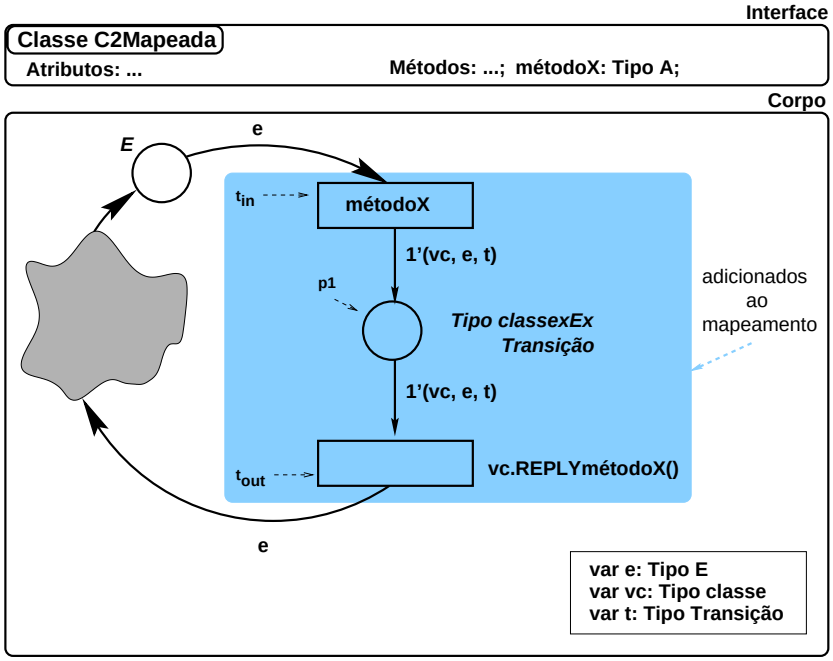

Figura 3.15: Mapeamento de uma classe de receptor usando comunicação síncrona com resposta.

considere que:

1. $SINC = TipoACK \cup TipoREPLY$ é um conjunto finito de nomes de conjuntos de tipos síncronos;

2. $\mathbb{S}_{sinc} = SINC \times \mathbb{S}$ é um conjunto finito de nomes de tipos de métodos síncronos;

3. $\mathbb{S}'' = \mathbb{S} \cup SINC \cup \mathbb{S}_{sinc}$ é um conjunto finito de nomes de conjuntos, proveniente da união de um conjunto finito de nomes com o conjunto de nomes de conjuntos de tipos síncronos e com o conjunto de nomes de tipos de métodos síncronos;

4. $\Sigma''' = \langle \mathbb{S}''', \mathcal{O} \rangle$ é uma assinatura, na qual:

   - $\mathbb{S}''' = \mathbb{S}'' \cup \mathcal{C}$ é a união de um conjunto finito de nomes de conjuntos com o conjunto de nomes de classes;

5. $M_{sinc} \subseteq \mathcal{M}$ é um conjunto finito de nomes de métodos síncronos, cujos tipos pertencem a $\mathbb{S}_{sinc}$;

6. $M_{mapeado} \subseteq \mathcal{M}$ é um conjunto finito de nomes de métodos assíncronos provenientes do mapeamento de uma transição com método(s) síncrono(s) associado(s).

Observação:

Se m' é um nome de método que pertence a $M_{mapeado}$, então ele é formado pela concatenação de uma das palavras-chave ACK ou $REPLY$, vistas na Seção 3.2, com o nome de método síncrono invocado numa dada transição e o seu tipo pertence a $TipoACK$ ou $TipoREPLY$ se for proveniente do mapeamento para comunicação com reconhecimento ou com resposta, respectivamente. Ou seja,

$m' \in M_{mapeado} \iff m' = PC \circ m$, $PC \in \{ACK, REPLY\}$, $m \in M_{sinc}$, e $\tau_{sinc}(m') \in SINC$.

onde, o operador $\circ$ define a concatenação de nomes. A definição da aplicação τ_{sinc} está na Definição 3.1.

Objetivando uma melhor compreensão das novas definições, sugere-se ao leitor compará-las àquelas contidas na Seção 2.3 (ver páginas 17 a 19).

Definição 3.1 (Interfaces) *Uma interface é uma tripla* $\langle A, M'', \tau_{sinc} \rangle$*, onde:*

1. A *é um conjunto finito de nomes de atributos;*
2. $M'' = M \cup M_{sinc} \cup M_{mapeado}$ *é um conjunto finito de nomes de métodos;*
3. $\tau_{sinc} : A \cup M'' \rightarrow \mathcal{S}'''$ *é uma aplicação que a cada nome de atributo e de método faz corresponder um nome de conjunto ou de classe;*
4. A *e* M'' *são disjuntos, ou seja,* $A \cap M'' = \varnothing$.

Observações:

(1) Essa assertiva permanece inalterada em relação à definição original (ver Definição 2.8) porque a introdução do mecanismo síncrono não interfere nos nomes de atributos;

(2) O conjunto de nomes de métodos deve ser redefinido para englobar também nomes de métodos síncronos e de métodos provenientes de mapeamento;

(3) A nova aplicação τ_{sinc} relaciona também os métodos englobados a seus tipos. Exemplificando, perceba que, para a classe da Figura 3.7 (ver página 28), $\tau_{sinc}(métodoX) = \langle TipoACK,\ TipoA \rangle$.

Definição 3.2 (Classe) *Uma sêxtupla* $\langle \mathfrak{I}, \mathfrak{N}, l^a, l^m, \eta, \mu \rangle$ *é uma classe baseada em* Σ''' *se, e somente se, as seguintes condições são satisfeitas:*

1. $\mathfrak{I} = \langle A, M'', \tau_{sinc} \rangle$ *é uma interface baseada em* Σ'''*;*

2. $\mathfrak{N} = \langle \Sigma''', N, I \rangle$ *é uma rede de Petri Colorida, denominada corpo da classe (considere* $N = \langle P, T, F \rangle$ *e* $I = \langle c, g, e, i \rangle$*);*

3. $l^a : A \to P$ *é uma aplicação que a cada atributo associa um lugar da estrutura, respeitando os tipos, ou seja, para todo atributo* $a \in A$*, se* $p = l^a(a)$ *então* $c(p) = \tau_{sinc}(a)$*;*

4. $l^m : M'' \to T \times \mathcal{V}_{\Sigma'''}$ *é uma aplicação que a cada método associa um conjunto de elementos de ativação que respeita os tipos, ou seja, se* $\langle t, v \rangle \in l^m(m), m \in M''$ *e* $v \in \mathcal{V}_{S_i''}$ *então* $\tau_{sinc}(m) = S_i''$*;*

5. $\eta : T \to 2^{\mathcal{V}_{\mathfrak{C}} \times \mathfrak{C}}$ *é uma aplicação que a cada transição associa um conjunto de inscrições de criação de objetos, tais que se* $\langle v, C_i \rangle \in \eta(t)$ *então* $v \in V_{C_i}$*;*

6. $\mu : T \to 2^{\mathcal{V}_{\mathfrak{C}} \times \mathcal{M} \times T_{\Sigma'''}}$ *é uma aplicação que a cada transição associa um conjunto de inscrições de mensagens.*

Observações:

(1) e (2) Para incorporar os métodos síncronos e seus mapeamentos, essas definições agora baseiam-se na assinatura Σ''';

(4) e (6) Redefine-se l^m e μ para que seja possível a respectiva ativação e invocação de métodos síncronos.

Definição 3.3 (Conjunto Integrado de Classes) *Um conjunto de classes* $K = \{K_{C_1}, K_{C_2}, \ldots, K_{C_n}\}$ *baseado em* Σ''' *e indexado por* $\mathcal{C}$ *é dito integrado se, e somente se, para qualquer classe* $k_i \in K$*, toda inscrição de mensagem presente em* k_i *respeita a interface da classe à qual se refere, ou seja:*

1. *se* $\langle v, m, te \rangle$ *é uma inscrição de mensagem presente no corpo da classe* $k_i = K_{C_i} \in K$*, em que*

(a) $v \in V_{C_j}$, C_j *é o nome da classe* k_j *referida, ou seja* $k_j = K_{C_j} \in K$; *e*

(b) $te \in T_{S''_x}$ *é um termo de tipo* S''_x, *para algum* $S''_x \in \mathbb{S}''$ *da assinatura* Σ''',

2. *então necessariamente*

(a) $m \in {}^{k_j}M''$, *ou seja, o nome de método invocado é válido na classe* k_j; *e*

(b) ${}^{k_j}\tau_{sinc}(m) = S''_x$, *ou seja, o tipo do termo respeita o tipo do método da classe* k_j.

Definição 3.4 (Mensagem) *Uma mensagem é uma tripla* $\langle \vec{o}, m, d \rangle$, *onde:*

1. $\vec{o} \in \vec{O}_{C_i}$ *é o nome do objeto destinatário;*

2. $m \in \mathfrak{M}$ *é o nome do método invocado; e*

3. $d \in D$ *é um elemento de dado passado como parâmetro,*

tais que se $\vec{o} \in \vec{O}_{C_i}$ *então* $m \in {}^{k_i}M''$ *e se* ${}^{k_i}\tau_{sinc}(m) = S''_x$, *para algum* $S''_x \in \mathbb{S}'''$, *então* $d \in D_{S''_x}$.

Formalização do Mapeamento para os Mecanismos de Comunicação Síncrona

Esta seção contém a formalização dos mapeamentos informalmente explicados nas Seções 3.2.1 e 3.2.2. A fim de facilitar a explanação da formalização, esta foi dividida em três etapas:

Etapa I *Mapeamento para Bloco de Rede de Petri Colorida*: contém as definições para o desdobramento de uma transição com método(s) síncrono(s) - uma t_{sinc} - num bloco (sub-rede) de rede de Petri Colorida, ou simplesmente rede Colorida;

Etapa II *Incorporação do Bloco de Rede Colorida ao Corpo de uma Classe RPOO*: mostra as definições para a ligação do bloco obtido na Etapa I ao corpo da classe RPOO que continha t_{sinc};

Etapa III *Associação entre Interface e Corpo da Classe RPOO*: apresenta as definições para associação da *interface* ao corpo da classe RPOO resultante após Etapa II.

Adicionalmente, cada etapa subdivide-se de modo a explicitar quais definições aplicam-se ao lado emissor e quais ao receptor.

Antes de prosseguir às etapas, é importante ter em mente as seguintes definições:

Definição 3.5 (Transição Síncrona) *Uma transição $t \in T$ é denotada por t_{sinc} se possuir alguma invocação ou ativação de método síncrono. Ou seja,*

$t \in T \wedge (\exists m \in M_{sinc} : \langle t, v \rangle \in l^m(m) \vee m \in \mu(t))$.

Denota-se $T_{sinc} \subseteq T$ o conjunto de *todas* as transições síncronas de um sistema RPOO. Analogamente, usa-se T_E (T_R) para denotar o conjunto de *todas* as transições síncronas com *invocação* (*ativação*) de métodos síncronos. Ou seja:

$$T_E = \{t \in T,\ \exists m \in M_{sinc} : m \in \mu(t)\}$$

$$T_R = \{t \in T,\ \exists m \in M_{sinc} : m \in l^m(t)\}$$

$$T_{sinc} = T_E \cup T_R \text{ e } T_E \cap T_R = \varnothing$$

Como um sistema RPOO deve constituir um conjunto integrado de classes (ver Definição 3.3), cada transição síncrona pertencente a T_E deve corresponder a pelo menos uma transição síncrona pertencente a T_R. Dessa forma, pode-se definir duas funções de correspondência para transições síncronas:

1. $F_{ER} : T_E \rightarrow 2^{T_R}$ é uma função que, dada uma t_{sinc} com *invocação* de métodos síncronos, retorna um conjunto contendo transições síncronas correspondentes com *ativação* desses métodos;

2. $F_{RE} : T_R \rightarrow 2^{T_E}$ é uma função que, dada uma t'_{sinc} com *ativação* de método síncrono, retorna um conjunto contendo transições síncronas correspondentes com *invocação* desse método.

Definição 3.6 (Correspondência para Transições Síncronas) *Duas transições $t_1 \in T_E$ e $t_2 \in T_R$ são correspondentes se, e somente se, t_1 invoca um método síncrono de uma classe a qual t_2 pertence e t_2 possui a ativação desse método. Ou seja,*

$(t_2 \in F_{ER}(t_1)) \wedge (t_1 \in F_{RE}(t_2)) \iff \exists \langle v, m, te \rangle \in \mu(t_1),\ v \in \mathcal{V}_{C_j},\ m \in {}^{k_j}M_{sinc} :$ ${}^{k_j}l^m(m) = \langle t_2, v' \rangle$.

Exemplificando, suponha a situação ilustrada nas Figuras 3.6 e 3.7 (ver página 28). Considere a transição síncrona $t_1(t_2)$ presente no corpo da classe *C1* (*C2*). Essas duas transições são correspondentes, pois:

- t_1 contém inscrição de invocação $\langle v$=*um_ nome_ de_ objeto_ de_ C2*, m=*métodoX*, te=*(ACK, a)*$\rangle$,
- *métodoX* $\in {}^{C2}M_{sinc}$ e
- na classe *C2* existe a *ativação* de *métodoX* em t_2, isto é, ${}^{C2}l^m(\textit{métodoX}) = \langle t_2 = \textit{métodoX}, v' = e \rangle$.

<u>ETAPA I</u> - Mapeamento para Bloco de Rede de Petri Colorida

Para as definições seguintes, considere que:

- B_E = Bloco Emissor;
- B_R = Bloco Receptor;
- $OBJ : T_E \rightarrow \mathcal{V}_{\mathcal{C}}$ é uma função que retorna um conjunto contendo variáveis cujas classes têm métodos síncronos invocados numa transição $t_{sinc} \in T_E$. Ou seja, $OBJ(t_{sinc}) = \bigcup_{\forall \langle v,m,te \rangle \in \mu(t_{sinc}),\ m \in M_{sinc}} v$;
- $MC\_OBJ : T_E \rightarrow \mathcal{V}_{\mathcal{C}}{}^{mc}$ é uma função que retorna um multiconjunto formado por variáveis cujas classes têm métodos síncronos invocados numa transição $t_{sinc} \in T_E$. Ou seja, $MC\_OBJ(t_{sinc}) = \sum_{\forall \langle v,m,te \rangle \in \mu(t_{sinc}),\ m \in M_{sinc}} 1'v$;
- $VAR : T_E \rightarrow 2^{\mathcal{V}_{\Sigma'''}}$ é uma função que retorna um conjunto formado pelo produto cartesiano dos conjuntos de variáveis existentes nas expressões de cada um

dos arcos de entrada de uma transição $t_{sinc} \in T_E$. Ou seja, $VAR(t_{sinc}) = \bigtimes_{\forall p \in {}^\bullet t_{sinc},\, \forall v \in e(\langle p, t_{sinc}\rangle)} v$;

Lado Emissor

Definição 3.7 (Mapeamento Bloco Lado Emissor) *Seja $t_{sinc} \in {}^{k_i}T_E$ uma transição no corpo de uma classe RPOO, k_i. Então, o mapeamento de t_{sinc} para um bloco de rede Colorida $B_E = \langle \Sigma_{B_E}, N_{B_E}, I_{B_E}\rangle$ é tal que:*

1. $\Sigma_{B_E} = \Sigma'''$;
2. $N_{B_E} = \langle P_{B_E}, T_{B_E}, F_{B_E}\rangle$, *em que :*
   (a) $P_{B_E} = \{p1, p2, p3\}$, *onde* ${}^{k_i}P \cap P_{B_E} = \varnothing$;
   (b) $T_{B_E} = \{t_{in}, t2, t_{out}\}$, *onde* ${}^{k_i}T \cap T_{B_E} = \varnothing$;
   (c) $F_{B_E} = \{(t_{in}, p1), (p1, t2), (t2, p2), (p2, t_{out}), (t_{in}, p_3), (p_3, t_{out})\}$.
3. $I_{B_E} = \langle c_{B_E}, g_{B_E}, e_{B_E}, i_{B_E}\rangle$,*em que:*
   (a) $\forall p \in P_{B_E}$,

$$c_{B_E}(p) = \begin{cases} \mathcal{C} & se\ p \neq p_3, \\ (\bigtimes_{\forall p' \in {}^\bullet t_{sinc}} c(p')) \times (\bigtimes_{\forall v \in OBJ(t_{sinc}),\, v \in V_{C_i}} C_i) & se\ p \neq p_3. \end{cases}$$

   (b) $\forall t \in T_{B_E}$,

$$g_{B_E}(t) = \begin{cases} g(t_{sinc}) & se\ t = t_{in}, \\ \varnothing & se\ t \neq t_{in}. \end{cases}$$

   (c) $\forall f \in F_{BE}$,

$$e_{B_E}(f) = \begin{cases} MC\_OBJ(t_{sinc}) & se\ f \in \{(t_{in}, p1), (p2, t_{out})\}, \\ 1'v',\ v' \in \mathcal{V}_{\mathcal{C}} & se\ f \in \{(p1, t2), (t2, p2)\} \\ 1'\langle 1'VAR(t_{sinc}), OBJ(t_{sinc})\rangle & se\ f \in \{(t_{in}, p_3), (p_3, t_{out})\}. \end{cases}$$

   (d) $\forall p \in P_{B_E}, i_{B_E}(p) = \varnothing$.

Observações:

(1) A rede Colorida do bloco baseia-se na assinatura Σ''' para que seja possível, nas próximas etapas, a incorporação dos métodos pertencentes a $M_{mapeado}$;

(2) Tanto o conjunto de lugares do bloco como o de transições são disjuntos dos respectivos conjuntos de lugares e transições do corpo da classe que contém a transição síncrona a ser substituída. Essa restrição é necessária para assegurar que todas as transições e lugares são únicos dentro de uma classe;

(3) Para as inscrições do bloco, tem-se que:

(a) A cor associada aos lugares $p1$ e $p2$ permite-lhes armazenar fichas cujos valores são nomes de objetos. O lugar $p3$ tem uma cor formada pelo produto cartesiano do (i) produto cartesiano das cores dos lugares de entrada da transição síncrona mapeada com o (ii) produto cartesiano das cores das variáveis cujas classes têm métodos síncronos invocados na transição mapeada.

(b) A guarda presente em t_{sinc} deve ser preservada em t_{in} e as demais transições não devem possuir guardas associadas;

(c) Considerando o exemplo das Figuras 3.6 e 3.8 (ver páginas 28 e 29), tem-se que $e_{B_E}((t_{in}, p_1)) = MC\_OBJ(t_{sinc}) = 1'v2+1'v3+1'v4$ e $e_{B_E}((p_1, t_2)) = 1'vc$, onde essa variável vc pode ser ligada a qualquer nome de objeto. Mais especificamente, as ligações possíveis para vc nesse caso são:

- $b_1 = \langle vc{=}um\_nome\_de\_objeto\_de\_C2,... \rangle$,
- $b_2 = \langle vc{=}um\_nome\_de\_objeto\_de\_C3,... \rangle$ e
- $b_3 = \langle vc{=}um\_nome\_de\_objeto\_de\_C4,... \rangle$;

(d) A marcação inicial do bloco é vazia.

Lado Receptor

Definição 3.8 (Mapeamento Bloco Lado Receptor) *Seja $t_{sinc} \in {}^{k_i}T_R$ uma transição síncrona presente no corpo de uma classe RPOO, k_i. Então, o mapeamento de t_{sinc} para um bloco de rede Colorida $B_R = \langle \Sigma_{B_R}, N_{B_R}, I_{B_R} \rangle$ é tal que:*

1. $\Sigma_{B_R} = \Sigma'''$;

2. $N_{B_R} = \langle P_{B_R}, T_{B_R}, F_{B_R} \rangle$ *em que:*

   (a) $P_{B_R} = \{p1\}$, *onde* ${}^{k_i}P \cap P_{B_R} = \varnothing$;

   (b) $T_{B_R} = \{t_{in}, t_{out}\}$, *onde* ${}^{k_i}T \cap T_{B_R} = \varnothing$;

   (c) $F_{B_R} = \{(t_{in}, p1), (p1, t_{out})\}$.

3. $I_{B_R} = \langle c_{B_R}, g_{B_R}, e_{B_R}, i_{B_R} \rangle$, *em que:*

   (a) $\forall p \in P_{B_R}$,:

$$c_{B_R}(p) = \begin{cases} \bigtimes\limits_{\forall p' \in {}^\bullet t_{sinc}} c(p') & se\ \exists m \in {}^{k_i}M_{sinc} : {}^{k_i}l^m(m) = \langle t_{sinc}, var \rangle \wedge \\ & {}^{k_i}\tau_{sinc}(m) = \langle TipoACK, x \rangle,\ x \in \mathbb{S}, \\ \mathbb{C} \times \bigtimes\limits_{\forall p' \in {}^\bullet t_{sinc}} c(p') \times T & se\ \exists m \in {}^{k_i}M_{sinc} : {}^{k_i}l^m(m) = \langle t_{sinc}, var \rangle \wedge \\ & {}^{k_i}\tau_{sinc}(m) = \langle TipoREPLY, x \rangle,\ x \in \mathbb{S}. \end{cases}$$

   (b) $\forall t \in T_{B_R}$,

$$g_{B_R}(t) = \begin{cases} g(t_{sinc}) & se\ t = t_{in}, \\ \varnothing & se\ t \neq t_{in}. \end{cases}$$

   (c) $\forall f = \langle p, t \rangle \in F_{B_R}$ *ou* $\forall f = \langle t, p \rangle \in F_{B_R}$:

$$e_{B_R}(f) = \begin{cases} 1'VAR(t_{sinc}) & se\ c_{B_R}(p) = \bigtimes\limits_{\forall p' \in {}^\bullet t_{sinc}} c(p'), \\ 1'\langle v, VAR(t_{sinc}), t \rangle,\ v \in \mathcal{V}_{\mathbb{C}},\ t \in T & se\ c_{B_R}(p) = \mathbb{C} \times \bigtimes\limits_{\forall p' \in {}^\bullet t_{sinc}} c(p') \times T. \end{cases}$$

   (d) $\forall p \in P_{B_R}$, $i_{B_R}(p) = \varnothing$.

Observações:

(3) Para as inscrições do bloco, tem-se que:

   (a) Se o mapeamento é proveniente de uma comunicação com reconhecimento, o lugar $p1$ tem uma cor formada pelo produto cartesiano das cores dos lugares

de entrada da transição síncrona em questão. Caso contrário, a cor associada a $p1$ compõe-se do produto cartesiano do (i) conjunto finito de nomes de classes, com o (ii) produto cartesiano das cores dos lugares de entrada da transição e com (iii) o conjunto de transições. Assim, para o exemplo das Figuras 3.7 e 3.9 (ver páginas 28 e 29), $c_{B_R}(p_1) = TipoE$ pois $métodoX \in {}^{C2}M_{sinc}$, ${}^{C2}l^m(métodoX) = \langle t_{sinc_R}, variáveis\rangle$ e ${}^{C2}\tau_{sinc}(métodoX) = \langle TipoACK, TipoA\rangle$.

(c) As inscrições dos arcos baseiam-se nas cores dos lugares. Para o exemplo da Figura 3.15 (ver página 34), tem-se a ligação $b_1 = \langle vc = um\_nome\_de\_objeto\_de\_C1Mapeada, e = um\_valor\_de\_TipoE, t = RespondeXYZ\rangle$.

A explicação para as inscrições dos itens *(b)* e *(d)* é semelhante àquela apresentada na Definição 3.7.

ETAPA II - Incorporação do Bloco de Rede Colorida ao Corpo de uma Classe RPOO

Lado Emissor

Definição 3.9 (Ligação de Bloco a Rede Colorida do Corpo - Lado Emissor) *Seja* ${}^{k_i}\mathcal{N} = \langle {}^{k_i}\Sigma''', {}^{k_i}N, {}^{k_i}I\rangle$ *a rede Colorida que compõe o corpo de uma classe RPOO -* k_i *- e* $B_E = \langle \Sigma_{B_E}, N_{B_E}, I_{B_E}\rangle$ *um bloco de rede Colorida proveniente do mapeamento de uma transição síncrona -* $t_{sinc} \in {}^{k_i}T_E$ *- segundo a Definição 3.7. A ligação do bloco* B_E *à rede Colorida* ${}^{k_i}\mathcal{N}$ *resultará numa nova rede Colorida* ${}^{k_i}\mathcal{N}_{map} = \langle {}^{k_i}\Sigma_{map}, {}^{k_i}N_{map}, {}^{k_i}I_{map}\rangle$*, em que:*

1. ${}^{k_i}\Sigma_{map} = {}^{k_i}\Sigma''' \cup \Sigma_{B_E}$;
2. ${}^{k_i}N_{map} = \langle {}^{k_i}P_{map}, {}^{k_i}T_{map}, {}^{k_i}F_{map}\rangle$*, em que:*
   - *(a)* ${}^{k_i}P_{map} = {}^{k_i}P \cup P_{B_E}$;
   - *(b)* ${}^{k_i}T_{map} = ({}^{k_i}T \cup T_{B_E}) - \{t_{sinc}\}$;

(c) $\forall(p,t),\ (t,p) \in {}^{k_i}F_{map}$,

$$(p,t) = \begin{cases} \varnothing & se\ (t \in ({}^{k_i}T - \{t_{sinc}\}) \wedge p \in P_{B_E}) \vee \\ & (t \in (T_{B_E} - \{t_{in}\}) \wedge p \in {}^{k_i}P), \\ {}^{k_i}(p,t) & se\ p \in {}^{k_i}P \wedge t \in ({}^{k_i}T - \{t_{sinc}\}), \\ (p,t)_{B_E} & se\ p \in P_{B_E} \wedge t \in T_{B_E}, \\ {}^{k_i}(p,t_{sinc}) & se\ p \in {}^{k_i}P \wedge t = t_{in}. \end{cases}$$

$$(t,p) = \begin{cases} \varnothing & se\ (t \in ({}^{k_i}T - \{t_{sinc}\}) \wedge p \in P_{B_E}) \vee \\ & (t \in (T_{B_E} - \{t_{out}\}) \wedge p \in {}^{k_i}P), \\ {}^{k_i}(t,p) & se\ p \in {}^{k_i}P \wedge t \in ({}^{k_i}T - \{t_{sinc}\}), \\ (t,p)_{B_E} & se\ p \in P_{B_E} \wedge t \in T_{B_E}, \\ {}^{k_i}(t_{sinc},p) & se\ p \in {}^{k_i}P \wedge t = t_{out}. \end{cases}$$

3. ${}^{k_i}I_{map} = \langle {}^{k_i}c_{map},\ {}^{k_i}g_{map},\ {}^{k_i}e_{map},\ {}^{k_i}i_{map} \rangle$, *em que:*

(a) $\forall p \in {}^{k_i}P_{map}$,

$${}^{k_i}c_{map}(p) = \begin{cases} {}^{k_i}c(p) & se\ p \in {}^{k_i}P, \\ c_{B_E}(p) & se\ p \in P_{B_E}. \end{cases}$$

(b) $\forall t \in {}^{k_i}T_{map}$,

$${}^{k_i}g_{map}(t) = \begin{cases} {}^{k_i}g(t) & se\ t \in ({}^{k_i}T - \{t_{sinc}\}), \\ g_{B_E}(t) & se\ t \in T_{B_E}. \end{cases}$$

(c) $\forall(p,t),\ (t,p) \in {}^{k_i}F_{map}$,

$${}^{k_i}e_{map}(p,t) = \begin{cases} \varnothing & se\ (t \in ({}^{k_i}T - \{t_{sinc}\}) \wedge p \in P_{B_E}) \vee \\ & (t \in (T_{B_E} - \{t_{in}\}) \wedge p \in {}^{k_i}P), \\ {}^{k_i}e(p,t) & se\ p \in {}^{k_i}P \wedge t \in ({}^{k_i}T - \{t_{sinc}\}), \\ e_{B_E}(p,t) & se\ p \in P_{B_E} \wedge t \in T_{B_E}, \\ {}^{k_i}e(p,t_{sinc}) & se\ p \in {}^{k_i}P \wedge t = t_{in}. \end{cases}$$

$$
{}^{k_i}e_{map}(t,p) = \begin{cases} \varnothing & se\ (t \in ({}^{k_i}T - \{t_{sinc}\}) \wedge p \in P_{B_E}) \vee \\ & (t \in (T_{B_E} - \{t_{out}\}) \wedge p \in {}^{k_i}P), \\ {}^{k_i}e(t,p) & se\ p \in {}^{k_i}P \wedge t \in ({}^{k_i}T - \{t_{sinc}\}), \\ e_{B_E}(t,p) & se\ p \in P_{B_E} \wedge t \in T_{B_E}, \\ {}^{k_i}e(t_{sinc},p) & se\ p \in {}^{k_i}P \wedge t = t_{out}. \end{cases}
$$

(d) $\forall p \in {}^{k_i}P_{map}$,

$$
{}^{k_i}i_{map}(p) = \begin{cases} {}^{k_i}i(p) & se\ p \in {}^{k_i}P, \\ i_{B_E}(p) & se\ p \in P_{B_E}. \end{cases}
$$

Observações:

Define-se a nova assinatura da classe em *(1)*, a nova estrutura de rede do corpo da classe em *(2)* e as novas inscrições em *(3)*.

Lado Receptor

Definição 3.10 (Ligação de Bloco a Rede Colorida do Corpo - Lado Receptor) *Seja* ${}^{k_i}\mathcal{N} = \langle {}^{k_i}\Sigma''', {}^{k_i}N, {}^{k_i}I\rangle$ *a rede Colorida que compõe o corpo de uma classe RPOO,* k_i, *e* $B_R = \langle \Sigma_{B_R}, N_{B_R}, I_{B_R}\rangle$ *um bloco de rede Colorida proveniente do mapeamento de uma transição síncrona,* $t_{sinc} \in {}^{k_i}T_R$, *segundo a Definição 3.8. A ligação do bloco* B_R *à rede Colorida* ${}^{k_i}\mathcal{N}$ *resultará numa nova rede Colorida* ${}^{k_i}\mathcal{N}_{map} = \langle {}^{k_i}\Sigma_{map}, {}^{k_i}N_{map}, {}^{k_i}I_{map}\rangle$, *em que:*

1. ${}^{k_i}\Sigma_{map} = {}^{k_i}\Sigma''' \cup \Sigma_{B_R}$;
2. ${}^{k_i}N_{map} = \langle {}^{k_i}P_{map}, {}^{k_i}T_{map}, {}^{k_i}F_{map}\rangle$, *em que:*
   - *(a)* ${}^{k_i}P_{map} = {}^{k_i}P \cup P_{B_R}$;
   - *(b)* ${}^{k_i}T_{map} = ({}^{k_i}T \cup T_{B_R}) - \{t_{sinc}\}$;

(c) $\forall(p,t),\ (t,p) \in {}^{k_i}F_{map}$,

$$(p,t) = \begin{cases} \varnothing & se\ (t \in ({}^{k_i}T - \{t_{sinc}\}) \wedge p \in P_{B_R}) \vee \\ & (t \in (T_{B_R} - \{t_{in}\}) \wedge p \in {}^{k_i}P), \\ {}^{k_i}(p,t) & se\ p \in {}^{k_i}P \wedge t \in ({}^{k_i}T - \{t_{sinc}\}), \\ (p,t)_{B_R} & se\ p \in P_{B_R} \wedge t \in T_{B_R}, \\ {}^{k_i}(p,t_{sinc}) & se\ p \in {}^{k_i}P \wedge t = t_{in}. \end{cases}$$

$$(t,p) = \begin{cases} \varnothing & se\ (t \in ({}^{k_i}T - \{t_{sinc}\}) \wedge p \in P_{B_R}) \vee \\ & (t \in (T_{B_R} - \{t_{out}\}) \wedge p \in {}^{k_i}P), \\ {}^{k_i}(t,p) & se\ p \in {}^{k_i}P \wedge t \in ({}^{k_i}T - \{t_{sinc}\}), \\ (t,p)_{B_R} & se\ p \in P_{B_R} \wedge t \in T_{B_R}, \\ {}^{k_i}(t_{sinc},p) & se\ p \in {}^{k_i}P \wedge t = t_{out}. \end{cases}$$

3. ${}^{k_i}I_{map} = \langle {}^{k_i}c_{map},\ {}^{k_i}g_{map},\ {}^{k_i}e_{map},\ {}^{k_i}i_{map} \rangle$, *em que:*

(a) $\forall p \in {}^{k_i}P_{map}$,

$${}^{k_i}c_{map}(p) = \begin{cases} {}^{k_i}c(p) & se\ p \in {}^{k_i}P, \\ c_{B_R}(p) & se\ p \in P_{B_R}. \end{cases}$$

(b) $\forall t \in {}^{k_i}T_{map}$,

$${}^{k_i}g_{map}(t) = \begin{cases} {}^{k_i}g(t) & se\ t \in ({}^{k_i}T - \{t_{sinc}\}), \\ g_{B_R}(t) & se\ t \in T_{B_R}. \end{cases}$$

(c) $\forall(p,t),\ (t,p) \in {}^{k_i}F_{map}$,

$${}^{k_i}e_{map}(p,t) = \begin{cases} \varnothing & se\ (t \in ({}^{k_i}T - \{t_{sinc}\}) \wedge p \in P_{B_R}) \vee \\ & (t \in (T_{B_R} - \{t_{in}\}) \wedge p \in {}^{k_i}P), \\ {}^{k_i}e(p,t) & se\ p \in {}^{k_i}P \wedge t \in ({}^{k_i}T - \{t_{sinc}\}), \\ e_{B_R}(p,t) & se\ p \in P_{B_R} \wedge t \in T_{B_R}, \\ {}^{k_i}e(p,t_{sinc}) & se\ p \in {}^{k_i}P \wedge t = t_{in}. \end{cases}$$

$$
{}^{k_i}e_{map}(t,p) = \begin{cases} \varnothing & se\ (t \in ({}^{k_i}T - \{t_{sinc}\}) \wedge p \in P_{B_R}) \vee \\ & (t \in (T_{B_R} - \{t_{out}\}) \wedge p \in {}^{k_i}P), \\ {}^{k_i}e(t,p) & se\ p \in {}^{k_i}P \wedge t \in ({}^{k_i}T - \{t_{sinc}\}), \\ e_{B_R}(t,p) & se\ p \in P_{B_R} \wedge t \in T_{B_R}, \\ {}^{k_i}e(t_{sinc},p) & se\ p \in {}^{k_i}P \wedge t = t_{out}. \end{cases}
$$

(d) $\forall p \in {}^{k_i}P_{map}$,

$$
{}^{k_i}i_{map}(p) = \begin{cases} {}^{k_i}i(p) & se\ p \in {}^{k_i}P, \\ i_{B_R}(p) & se\ p \in P_{B_R}. \end{cases}
$$

Observações:

De forma semelhante à da Definição 3.9, define-se a nova assinatura da classe em *(1)*, a nova estrutura de rede do corpo da classe em *(2)* e as novas inscrições em *(3)*.

ETAPA III - Associação entre *Interface* e Corpo da Classe RPOO

Nesta seção apresentam-se as modificações que se devem efetuar na *interface* e nas aplicações l^a, l^m, η e μ para mapear uma classe RPOO com transição síncrona. Para tanto, considere que:

- *MétodosInterface*: $T_E \rightarrow 2^{M_{mapeado}}$ é uma função que, dada uma transição síncrona, retorna um conjunto contendo nomes de métodos a serem adicionados à *interface* de uma classe para que seus objetos possam receber o reconhecimento/resultado de um método invocado sincronamente em $t_{sinc} \in T_E$. Ou seja:

  $MétodosInterface(t_{sinc}) = \{t_{e_1} \circ m, \forall \langle v, m, t_{e_1} ... t_{e_n} \rangle \in \mu(t_{sinc}),\ n \in \mathbb{N}^+ : m \in M_{sinc}\}$.

  Observação:

  Uma transição síncrona possui inscrições de mensagens para apenas um dos tipos de comunicação síncrona, por conseguinte, *todos* os métodos resultantes na

aplicação da função *MétodosInterface* terão a forma $ACK \circ m$ ou $REPLY \circ m$. Perceba que se extraiu t_{e_1} para a concatenação porque a palavra-chave, ACK ou $REPLY$, deve vir sempre como *primeiro termo* de uma inscrição de mensagem.

- *EliminaTermo1*: $T_E \rightarrow 2^{\mathcal{V}_{\mathbb{e}} \times M \times T_{\Sigma'''}}$ é uma função que transforma um conjunto de inscrições de mensagens síncronas, associadas a uma dada transição síncrona, em um conjunto de inscrições de mensagens assíncronas. Ou seja, para $t_{sinc} \in T_E$,

  $EliminaTermo1\,(t_{sinc}) = \{\langle v, m, t_{e_2} ... t_{e_n} \rangle,\ \forall \langle v, m, t_{e_1} ... t_{e_m} \rangle \in \mu(t_{sinc}),\ n \geq 2,\ n, m \in \mathbb{N}^+ :\ m \in M_{sinc}\}$.

- *MétodosAssíncronos*: $T_E \rightarrow 2^{\mathcal{V}_{\mathbb{e}} \times M \times T_{\Sigma'''}}$ é uma função que retorna o conjunto de inscrições de mensagens assíncronas associadas a uma dada transição síncrona. Ou seja, para $t_{sinc} \in T_E$,

  $MétodosAssíncronos(t_{sinc}) = \{\langle v, m, te \rangle, \forall \langle v, m, te \rangle \in \mu(t_{sinc}) :\ m \in M\}$.

- *MétodoAtivado*: $T_R \rightarrow M_{sinc}$ é uma função que retorna o nome do método síncrono ativado em uma dada transição. Ou seja, para $t_{sinc} \in T_R$,

  $MétodoAtivado(t_{sinc}) = m,\ m \in M_{sinc} : l^m(m) = \langle t_{sinc}, var \rangle$.

- *EmTipoAssinc*: $\mathcal{S}_{sinc} \rightarrow \mathcal{S}$ é uma função que, dado um tipo síncrono, retorna um tipo assíncrono equivalente. Ou seja, para $\langle y, x \rangle \in \mathcal{S}_{sinc}$,

  $EmTipoAssinc(\langle y, x \rangle) = \langle x \rangle$.

- *ÉComRec*: $T_R \rightarrow booleano$ é uma função que retorna *verdadeiro* (*falso*) se uma dada transição síncrona está envolvida numa comunicação com reconhecimento (resposta). Ou seja, para $t_{sinc} \in T_R$,

$$\acute{E}ComRec(t_{sinc}) = \begin{cases} verdadeiro & \text{se } \exists m \in M_{sinc} :\ l^m(m) = \langle t_{sinc}, var \rangle \wedge \\ & \tau_{sinc}(m) = \langle TipoACK, x \rangle, x \in \mathcal{S}, \\ falso & \text{se } \exists m \in M_{sinc} :\ l^m(m) = \langle t_{sinc}, var \rangle \wedge \\ & \tau_{sinc}(m) = \langle TipoREPLY, x \rangle, x \in \mathcal{S}. \end{cases}$$

Lado Emissor

Para a Definição 3.11, considere que:

- O conjunto de nomes de conjuntos:

$$ {}^{k'_i}\mathcal{S}'' = \begin{cases} {}^{k_i}\mathcal{S}'' \cup TipoACK & \text{se } ACK \circ m \in MétodosInterface(t_{sinc}), \\ {}^{k_i}\mathcal{S}'' \cup TipoREPLY & \text{se } REPLY \circ m \in MétodosInterface(t_{sinc}). \end{cases} $$

- A assinatura ${}^{k'_i}\Sigma''' = {}^{k'_i}\mathcal{S}'' \cup \mathcal{C}$.

Definição 3.11 (Ligação *Interface*/Corpo de Classe RPOO - Lado Emissor) *Seja* $k_i = \langle {}^{k_i}\mathcal{I}, {}^{k_i}\mathcal{N}, {}^{k_i}l^a, {}^{k_i}l^m, {}^{k_i}\eta, {}^{k_i}\mu \rangle$ *uma classe RPOO e* ${}^{k_i}\mathcal{N}_{map} = \langle {}^{k_i}\Sigma_{map}, {}^{k_i}N_{map}, {}^{k_i}I_{map} \rangle$ *a rede Colorida resultante após ligação, segundo a Definição 3.9, de* ${}^{k_i}\mathcal{N}$ *ao bloco* $B_E = \langle \Sigma_{B_E}, N_{B_E}, I_{B_E} \rangle$*, proveniente de uma transição síncrona,* $t_{sinc} \in {}^{k_i}T_E$*. A ligação da rede* ${}^{k_i}\mathcal{N}_{map}$ *à* interface *de* k_i *resultará numa nova classe* $k'_i = \langle {}^{k'_i}\mathcal{I}, {}^{k'_i}\mathcal{N}, {}^{k'_i}l^a, {}^{k'_i}l^m, {}^{k'_i}\eta, {}^{k'_i}\mu \rangle$*, baseada em* ${}^{k'_i}\Sigma'''$*, onde:*

1. ${}^{k'_i}\mathcal{I} = \langle {}^{k'_i}A, {}^{k'_i}M'', {}^{k'_i}\tau_{sinc} \rangle$*, em que:*
   - *(a)* ${}^{k'_i}A = {}^{k_i}A$*;*
   - *(b)* ${}^{k'_i}M'' = {}^{k_i}M'' \cup MétodosInterface(t_{sinc})$*;*
   - *(c)* ${}^{k'_i}\tau_{sinc} : {}^{k'_i}A \cup {}^{k'_i}M'' \rightarrow {}^{k'_i}\mathcal{S}''$*, onde:*
     - *i.* $\forall a \in {}^{k'_i}A$, ${}^{k'_i}\tau_{sinc}(a) = {}^{k_i}\tau_{sinc}(a)$*;*
     - *ii.* $\forall m \in {}^{k_i}M''$, ${}^{k'_i}\tau_{sinc}(m) = {}^{k_i}\tau_{sinc}(m)$*;*
     - *iii.* $\forall m \in MétodosInterface(t_{sinc})$,

$$ {}^{k'_i}\tau_{sinc}(m) = \begin{cases} TipoACK & se\ m = ACK \circ m',\ m' \in {}^{k_i}M_{sinc}, \\ TipoREPLY & se\ m = REPLY \circ m',\ m' \in {}^{k_i}M_{sinc}. \end{cases} $$

2. ${}^{k'_i}\mathcal{N} = {}^{k_i}\mathcal{N}_{map}$*;*
3. ${}^{k'_i}l^a = {}^{k_i}l^a$*;*

4. $\forall m \in {}^{k'_i}M''$,

$$ {}^{k'_i}l^m(m) = \begin{cases} {}^{k_i}l^m(m) & se\ m \in {}^{k_i}M'', \\ \langle t_2, var \rangle & se\ m \in \mathit{MétodosInterface}(t_{sinc}). \end{cases} $$

5. $\forall t \in {}^{k'_i}T$,

$$ {}^{k'_i}\eta(t) = \begin{cases} {}^{k_i}\eta(t) & se\ t \in ({}^{k_i}T - \{t_{sinc}\}), \\ {}^{k_i}\eta(t_{sinc}) & se\ t = t_{in}, \\ \varnothing & se\ t \in (T_{B_E} - \{t_{in}\}). \end{cases} $$

6. $\forall t \in {}^{k'_i}T$,

$$ {}^{k'_i}\mu(t) = \begin{cases} {}^{k_i}\mu(t) & se\ t \in ({}^{k_i}T - \{t_{sinc}\}), \\ \mathit{EliminaTermo1}(t_{sinc}) \cup \mathit{MétodosAssíncronos}(t_{sinc}) & se\ t = t_{in}, \\ \varnothing & se\ t \in (T_{B_E} - \{t_{in}\}). \end{cases} $$

Observações:

(1) A *interface* da classe resultante *(a)* mantém os mesmos atributos da classe original, *(b)* deve incorporar os nomes dos métodos adicionados para envio de reconhecimento/resposta e *(c)* deve relacionar corretamente os métodos e atributos a nomes de tipos. Para o exemplo das Figuras 3.6 e 3.8 (ver páginas 28 e 29):

- $\mathit{métodosInterface}({}^{C1}t_{sinc_E}) = \{\mathit{ACKmétodoX},\ \mathit{ACKmétodoY},\ \mathit{ACKmétodoZ}\}$ e
- ${}^{C1Mapeada}\tau_{sinc}(\mathit{ACKmétodoX}) = {}^{C1Mapeada}\tau_{sinc}(\mathit{ACKmétodoY}) = {}^{C1Mapeada}\tau_{sinc}(\mathit{ACKmétodoZ}) = \mathit{TipoACK}$, pois $\mathit{ACKmétodoX} = \mathit{ACK} \circ \mathit{métodoX}$, $\mathit{ACKmétodoY} = \mathit{ACK} \circ \mathit{métodoY}$ e $\mathit{ACKmétodoZ} = \mathit{ACK} \circ \mathit{métodoZ}$;

(3) Pois não há mudanças nos atributos presentes na classe original, isto é, ${}^{k'_i}A = {}^{k_i}A$;

(4) A aplicação l^m deve relacionar a ativação dos métodos adicionados para envio de reconhecimento/resposta com a transição t_2. Assim, para o exemplo das Figuras 3.12 e 3.14 (ver páginas 32 e 33),

${}^{C1\,Mapeada}l^m(REPLY método X) = {}^{C1\,Mapeada}l^m(REPLY método Y) = {}^{C1\,Mapeada}l^m(REPLY método Z) = \langle RespondeXYZ, variáveis\rangle$;

(5) As inscrições de criação de objetos presentes na transição síncrona mapeada são preservadas em t_{in} e as demais transições do bloco não contêm esse tipo de inscrição;

(6) As inscrições de mensagem presentes na transição síncrona mapeada são associadas a t_{in}. Entretanto, inscrições síncronas são transformadas em assíncronas através do método *EliminaTermo1*(t_{sinc}). Por exemplo, para a situação ilustrada nas Figuras 3.12 e 3.14, $Métodos Assíncronos({}^{C1}t_{sinc_E}) = \{$*v2.métodoX(a)*, *v3.métodoY(a)*, *v4.métodoZ(a)*$\} = {}^{C1\,Mapeada}\mu(t_{in})$.

Inexistem inscrições de mensagem nas outras transições do bloco.

Lado Receptor

Para a Definição 3.12, considere que:

- O conjunto de nomes de conjuntos ${}^{k_i'}\mathbb{S}'' = ({}^{k_i}\mathbb{S}'' - \{{}^{k_i}\tau_{sinc}(MétodoAtivado(t_{sinc}))\}) \cup EmTipoAssinc({}^{k_i}\tau_{sinc}(MétodoAtivado(t_{sinc})))$;
- A assinatura ${}^{k_i'}\Sigma''' = {}^{k_i'}\mathbb{S}'' \cup \mathbb{C}$;
- $\neg$ é o operador de negação.

Definição 3.12 (Ligação *Interface*/Corpo de Classe RPOO - Lado Receptor) *Seja* $k_i = \langle {}^{k_i}\mathfrak{I}, {}^{k_i}\mathcal{N}, {}^{k_i}l^a, {}^{k_i}l^m, {}^{k_i}\eta, {}^{k_i}\mu\rangle$ *uma classe RPOO e* ${}^{k_i}\mathcal{N}_{map} = \langle {}^{k_i}\Sigma_{map}, {}^{k_i}N_{map}, {}^{k_i}I_{map}\rangle$ *a rede Colorida resultante após ligação, segundo a Definição 3.10, de* ${}^{k_i}\mathcal{N}$ *ao bloco* $B_R = \langle \Sigma_{B_R}, N_{B_R}, I_{B_R}\rangle$*, proveniente de uma transição síncrona,* $t_{sinc} \in {}^{k_i}T_R$*. A ligação da rede* ${}^{k_i}\mathcal{N}_{map}$ *à* interface *de* k_i *resultará numa nova classe* $k_i' = \langle {}^{k_i'}\mathfrak{I}, {}^{k_i'}\mathcal{N}, {}^{k_i'}l^a, {}^{k_i'}l^m, {}^{k_i'}\eta, {}^{k_i'}\mu\rangle$*, baseada em* ${}^{k_i'}\Sigma'''$*, onde:*

1. ${}^{k_i'}\mathfrak{I} = \langle {}^{k_i'}A, {}^{k_i'}M'', {}^{k_i'}\tau_{sinc}\rangle$*, em que:*
   (a) ${}^{k_i'}A = {}^{k_i}A$;
   (b) ${}^{k_i'}M'' = {}^{k_i}M''$;

(c) ${}^{k'_i}\tau_{sinc} : {}^{k'_i}A \cup {}^{k'_i}M'' \rightarrow {}^{k'_i}\mathcal{S}''$, *onde:*

i. $\forall a \in {}^{k'_i}A$, ${}^{k'_i}\tau_{sinc}(a) = {}^{k_i}\tau_{sinc}(a)$;

ii. $\forall m \in {}^{k'_i}M''$,

$$ {}^{k'_i}\tau_{sinc}(m) = \begin{cases} {}^{k_i}\tau_{sinc}(m) & se\ m \neq \mathit{MétodoAtivado}(t_{sinc}), \\ \mathit{EmTipoAssinc}({}^{k_i}\tau_{sinc}(m)) & se\ m = \mathit{MétodoAtivado}(t_{sinc}). \end{cases} $$

2. ${}^{k'_i}\mathcal{N} = {}^{k_i}\mathcal{N}_{map}$;

3. ${}^{k'_i}l^a = {}^{k_i}l^a$;

4. $\forall m \in {}^{k'_i}M''$,

$$ {}^{k'_i}l^m(m) = \begin{cases} {}^{k_i}l^m(m) & se\ m \neq \mathit{MétodoAtivado}(t_{sinc}), \\ \langle t_{in}, var \rangle & se\ m = \mathit{MétodoAtivado}(t_{sinc}). \end{cases} $$

5. $\forall t \in {}^{k'_i}T$,

$$ {}^{k'_i}\eta(t) = \begin{cases} {}^{k_i}\eta(t) & se\ t \in ({}^{k_i}T - \{t_{sinc}\}), \\ {}^{k_i}\eta(t_{sinc}) & se\ t = t_{in} \ \wedge\ \neg \mathit{ÉComRec}(t_{sinc}), \\ {}^{k_i}\eta(t_{sinc}) & se\ t = t_{out} \wedge \mathit{ÉComRec}(t_{sinc}), \\ \varnothing & caso\ contrário. \end{cases} $$

6. $\forall t \in {}^{k'_i}T$,

$$ {}^{k'_i}\mu(t) = \begin{cases} {}^{k_i}\mu(t) & se\ t \in ({}^{k_i}T - \{t_{sinc}\}), \\ \mathit{ACK} \circ \mathit{MétodoAtivado}(t_{sinc}) & se\ t = t_{in} \wedge \mathit{ÉComRec}(t_{sinc}), \\ {}^{k_i}\mu(t_{sinc}) & se\ t = t_{in} \ \wedge \neg \mathit{ÉComRec}(t_{sinc}), \\ \mathit{REPLY} \circ \mathit{MétodoAtivado}(t_{sinc}) & se\ t = t_{out} \ \wedge \neg \mathit{ÉComRec}(t_{sinc}), \\ {}^{k_i}\mu(t_{sinc}) & se\ t = t_{out} \wedge \mathit{ÉComRec}(t_{sinc}). \end{cases} $$

Observações:

(1) A *interface* da classe resultante *(a)* mantém os mesmos atributos da classe original, *(b)* o mesmo conjunto de nomes de métodos, pois a conversão de síncrono para assíncrono refere-se aos seus tipos e *(c)* deve relacionar corretamente os métodos e atributos a nomes de tipos, estabelecendo a conversão quando necessário. Assim, para o exemplo na Figura 3.7 (ver página 28), ${}^{C2}\tau_{sinc}(métodoX) = \langle TipoACK,\ TipoA\rangle$. Mas, após mapeamento (ver Figura 3.9, página 29), ${}^{C2Mapeada}\tau_{sinc}(métodoX) = EmTipoAssinc({}^{C2}\tau_{sinc}(métodoX)) = EmTipoAssinc(\langle TipoACK,\ TipoA\rangle) = \langle TipoA\rangle$, pois $métodoX = MétodoAtivado({}^{C2}t_{sinc_R})$;

(4) Após mapeamento, a aplicação l^m deve relacionar a ativação do método ativado em t_{sinc} à transição t_{in}. Por exemplo, para a situação ilustrada nas Figuras 3.13 e 3.15 (ver páginas 32 e 34) ${}^{C2Mapeada}l^m(métodoX) = \langle t_{in}, variáveis\rangle$;

(5) As inscrições de criação de objetos presentes na transição síncrona mapeada são preservadas em t_{in} se a comunicação é com resposta e em t_{out} se é com reconhecimento;

(6) Se a comunicação é com reconhecimento, a transição t_{in} contém somente inscrição de mensagem para envio de reconhecimento e a transição t_{out} preserva as inscrições de mensagem da transição síncrona mapeada. Na comunicação com resposta há uma situação inversa. Então, considerando o exemplo das Figuras 3.13 e 3.15, em que há comunicação com resposta, ${}^{C2Mapeada}\mu(t_{in}) = {}^{C2}\mu(t_{sinc_R}) = \varnothing$, pois $\neg ÉcomRec({}^{C2}t_{sinc_R}) = verdadeiro$, e ${}^{C2Mapeada}\mu(t_{out}) = REPLY \circ MétodoAtivado({}^{C2}t_{sinc_R}) = REPLYmétodoX$, pois $\neg ÉcomRec({}^{C2}t_{sinc_R}) = verdadeiro$.

Por fim, convém enfatizar que as definições presentes nesta seção aplicam-se ao mapeamento de *uma* transição síncrona por vez.

Considerações sobre a análise das redes mapeadas são apresentadas no Capítulo 5.

Capítulo 4

Sincronização de Métodos

A idéia central em sistemas distribuídos é a execução paralela de processos. Entretanto, em muitas aplicações distribuídas pode ser necessário determinar uma ordem particular para a ocorrência de eventos [Frø96], por exemplo, o encadeamento de transações para bancos de dados distribuídos. Nesse sentido, uma ferramenta para especificação de tais sistemas deve prover mecanismos que permitam ao projetista descrever a coordenação das atividades dos processos.

Em RPOO, um processo pode ser formado por um ou mais objetos. Quando se compõe de um único objeto, o projetista pode descrever a coordenação da ativação de seus métodos diretamente, por meio da estrutura do corpo da classe RPOO que o representa. Entretanto, quando o processo se compõe de muitos objetos, fica trabalhoso para o projetista estabelecer sincronização entre as mensagens enviadas entre esses objetos, pois o ambiente de comunicação não assegura que as mensagens serão entregues aos objetos na mesma ordem em que foram enviadas. Assim, por exemplo, fica difícil especificar uma *ordem total* na invocação de certos métodos.

Neste capítulo introduz-se a abstração do mecanismo para *sincronização de métodos* no modelo RPOO. Um *sincronizador* é uma entidade computacional que implementa os mecanismos para garantir uma ordem na ativação e/ou execução de métodos pertencentes a um conjunto de classes. Ele é definido *textualmente* pelo projetista, que determina o *conjunto de classes* sobre o qual o sincronizador se aplica e as *relações* de ordem a serem obedecidas. Escolheu-se definir o sincronizador como uma entidade externa às classes porque, dessa forma, os métodos dessas classes podem ser ativados

por outras classes, as quais o sincronizador não se aplica.

O restante do capítulo está organizado de modo que, na Seção 4.1, conceitua-se o mecanismo de sincronização de métodos, na Seção 4.2, apresentam-se as novas construções adicionadas ao modelo básico de RPOO, na Seção 4.3, explicam-se os mapeamentos necessários para que um sincronizador seja incorporado ao conjunto de classes ao qual se aplica e, na Seção 4.4, encontra-se a formalização dessa nova abstração.

4.1 Conceito

Num sentido mais restrito, sincronizar métodos significa executá-los simultaneamente. Num sentido mais amplo, sincronizar métodos implica em determinar uma ordem para suas execuções, isto é, coordená-los [Frø96]. Neste trabalho, usa-se o conceito mais amplo para sincronização de métodos que pertencem a um *conjunto de classes*.

A abstração para sincronização de métodos, explicada nas seções posteriores, baseia-se em construções do modelo básico e na abstração para comunicação síncrona, vista no Capítulo 3.

Como a formalização original do modelo RPOO já possibilita o tratamento direto da sincronização de métodos intra-objeto, por meio da estrutura de rede do corpo de uma classe RPOO, a abstração introduzida refere-se à *sincronização de métodos interobjetos*. Por meio dela torna-se possível garantir um aspecto fundamental de coordenação em sistemas distribuídos, a correta ordem no recebimento de mensagens.

4.2 Construções do Sincronizador

Para definição do sincronizador, utiliza-se uma linguagem que descreve o *conjunto de classes*, as quais o sincronizador relaciona-se, e as possíveis *relações* de ordem para ativação dos métodos dessas classes. Optou-se pelo uso de uma linguagem porque, como toda descrição textual de um sincronizador é transformada numa classe RPOO e incorporada ao sistema ao qual se aplica, fica mais fácil automatizar esse processo.

As relações que o projetista pode utilizar obedecem às regras de formação da gramática definida na Tabela 4.1. Assim, o projetista pode determinar que:

Tabela 4.1: Gramática das relações de um sincronizador.

(1)	⟨Relações⟩	::=	⟨relação⟩ | ⟨relação⟩ ⟨Relações⟩
(2)	⟨relação⟩	::=	ORDENE TUDO (⟨relação2⟩); | ⟨método⟩ ⟨ação⟩ ⟨resto⟩;
(3)	⟨relação2⟩	::=	⟨método⟩ ACONTECE ANTES ⟨resto2⟩
(4)	⟨resto⟩	::=	métodoSimples | ⟨método⟩ ⟨ação⟩ ⟨resto⟩
(5)	⟨resto2⟩	::=	métodoSimples | ⟨método⟩ ACONTECE ANTES ⟨resto2⟩
(6)	⟨método⟩	::=	⟨número⟩.⟨classe⟩.⟨nomeMétodo⟩
(7)	⟨métodoSimples⟩	::=	1.⟨classe⟩.⟨nomeMétodo⟩
(8)	⟨classe⟩	::=	⟨nome⟩
(9)	⟨nomeMétodo⟩	::=	⟨nome⟩
(10)	⟨nome⟩	::=	⟨letra⟩ ⟨nome⟩ | ⟨letra⟩ | ⟨número⟩ ⟨nome⟩ | ⟨número⟩
(11)	⟨ação⟩	::=	ACONTECE ANTES | DESABILITA
(12)	⟨letra⟩	::=	todas as letras do alfabeto, maiúsculas ou minúsculas
(13)	⟨número⟩	::=	1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

1. O método m, da classe k, deve sempre ser ativado e/ou executado antes do método m', pertencente à classe k'. Em termos gramaticais: $C.m$[1] ACONTECE ANTES $C'.m'$;

2. A ativação do método m, na classe k, desabilita a ativação do método m', na classe k'. De outra forma: $C.m$ DESABILITA $C'.m'$;

3. Deve haver uma ordenação total na ativação e/ou execução de certos métodos. Por exemplo, se para a relação do item 1 o projetista desejar que a segunda ativação de m só aconteça após a ativação e/ou execução de m', então a relação será: ORDENE TUDO ($C.m$ ACONTECE ANTES $C'.m'$).

Observe que a relação para ordenação total não permite o uso de construções para desabilitação de métodos, pois poderia resultar em impasses no sistema. Todas as regras da gramática na Tabela 4.1 são transitivas. Exemplificando:

se $C.m$ ACONTECE ANTES $C'.m'$ e $C'.m'$ ACONTECE ANTES $C''.m'' \Rightarrow$

$C.m$ ACONTECE ANTES $C''.m''$

[1] Onde houver $C.m$ leia-se $1.C.m$. Além disso, considere que C é o nome de uma classe k.

Além disso, ao especificar relações, o projetista não pode defini-las de modo que a transitividade seja quebrada e ciclos sejam originados. Por exemplo, não é permitida a definição de relações tipo:

$$C.m \text{ ACONTECE ANTES } C'.m' \text{ e } C'.m' \text{ ACONTECE ANTES } C.m$$

Como dito no início desta seção, embora o projetista especifique o sincronizador utilizando uma linguagem, cujas relações estão definidas na gramática da Tabela 4.1, na verdade ele é uma classe RPOO interligada ao conjunto de classes que coordena. Então, para facilitar o entendimento, explica-se a seguir como cada relação da especificação textual é mapeada numa estrutura de rede de Petri Colorida e se ilustra como acontece a associação de relações.

Mapeamento das Relações em Estruturas de Redes de Petri

O mapeamento da descrição textual do sincronizador para uma classe RPOO acontece da seguinte maneira:

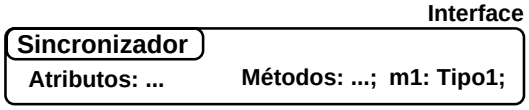

Figura 4.1: *Interface* de um sincronizador.

1. Cada método que aparece nas relações pertencerá à *interface* da classe do sincronizador. Por exemplo, se $m1$ é um método assíncrono de $Tipo1$, tem-se a situação ilustrada na Figura 4.1. Como será visto na Seção 4.3, todos os métodos da *interface* de um sincronizador devem ser *assíncronos*;

2. Para cada relação "$C1.m1$ ACONTECE ANTES $C2.m2$", tem-se uma subestrutura de rede de Petri Colorida, composta de duas transições e de um lugar. Para facilitar o mapeamento na subestrutura, imagine que os nomes de métodos e classes são individualmente associados a um número que representa a posição em que aparecem pela *primeira vez* nas relações. Assim, para identificar que transição está associada a que método, basta compor a posição da classe com a

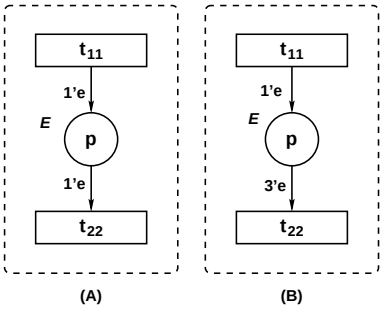

Figura 4.2: Subestrutura de rede de Petri para a relação ACONTECE ANTES.

do método. Por exemplo, para a relação acima, ao método $C1.m1$ associa-se a transição t_{11} (ver Figura 4.2.(A)), pois $C1$ e m_1 são o *primeiro* nome de classe e método que aparecem, respectivamente, no conjunto de relações. Além disso, o peso do arco de entrada do lugar de entrada da transição correspondente ao segundo método da relação (no caso, $C2.m2$) será sempre $1'e$, enquanto o peso de seu arco de entrada será igual ao número associado ao primeiro membro (no caso, $C1.m1$). Conseqüentemente, se a relação especificasse que o método $m1$ deveria ser ativado e/ou executado *três vezes* para que $m2$ pudesse ser ativado e/ou executado, ou seja, "$3.C1.m1$ ACONTECE ANTES $C2.m2$", tem-se a situação ilustrada na Figura 4.2.(B), onde o peso do arco que liga o lugar p à transição t_{22} é 3;

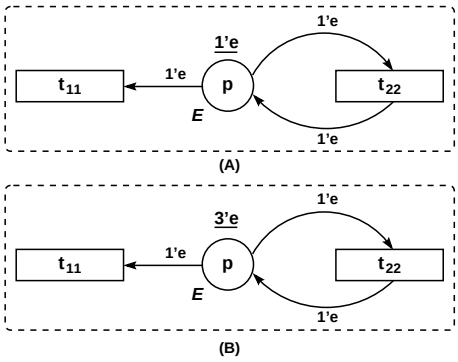

Figura 4.3: Subestrutura de rede de Petri para a relação DESABILITA.

3. Para cada relação "$C1.m1$ DESABILITA $C2.m2$", tem-se a subestrutura de rede Colorida mostrada na Figura 4.3.(A). De modo semelhante, se a relação for "$3.C1.m1$ DESABILITA $C2.m2$", tem-se a subestrutura na Figura 4.3.(B). Perceba que o multiconjunto para a marcação inicial do lugar p tem coeficiente igual

ao valor do número associado ao primeiro método da relação;

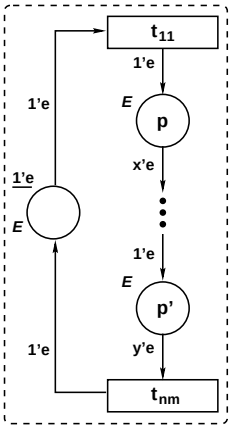

Figura 4.4: Subestrutura de rede de Petri para a relação ORDENE TUDO.

4. Para cada relação "ORDENE TUDO ($Ci.mi$ ACONTECE ANTES ... ACONTECE ANTES $Cj.mj$)", tem-se a subestrutura ilustrada na Figura 4.4.

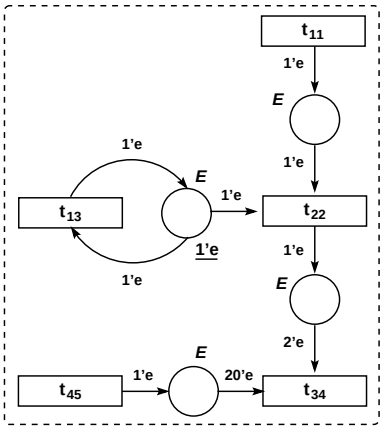

Figura 4.5: Subestrutura de rede de Petri para composição de relações.

Por fim, vale salientar que cada uma das subestruturas de rede Colorida ilustradas nas Figuras 4.2-4.4 pode ser combinada, resultando numa subestrutura maior. Exemplificando, para as relações:

$C1.m1$ ACONTECE ANTES $C2.m2$ DESABILITA $C1.m5$; $2.C2.m2$ ACONTECE ANTES $C3.m4$; $20.C6.m20$ ACONTECE ANTES $C3.m4$;

tem-se a subestrutura ilustrada na Figura 4.5. Observe que, nessa figura, as posições associadas aos nomes de classes e métodos são:

$$C1 = 1 \quad C2 = 2 \quad C3 = 3 \quad C6 = 4$$
$$m1 = 1 \quad m2 = 2 \quad m5 = 3 \quad m4 = 4 \quad m20 = 5$$

A formalização para sincronização de métodos já realiza diretamente essa composição e será vista na Seção 4.4.

4.3 Integração do Sincronizador ao Conjunto de Classes

Dada a descrição textual do sincronizador, é necessário criar a classe que o representará e realizar as devidas modificações nas classes dos emissores e receptores envolvidos. Essas modificações podem diferir de acordo com o tipo de comunicação utilizada nas invocações/ativações dos métodos. Nesta seção, a princípio explicam-se genericamente as modificações, para, em seguida, aplicá-las a um exemplo. A formalização para sincronização de métodos é apresentada na Seção 4.4.

Modificações Lado Emissor

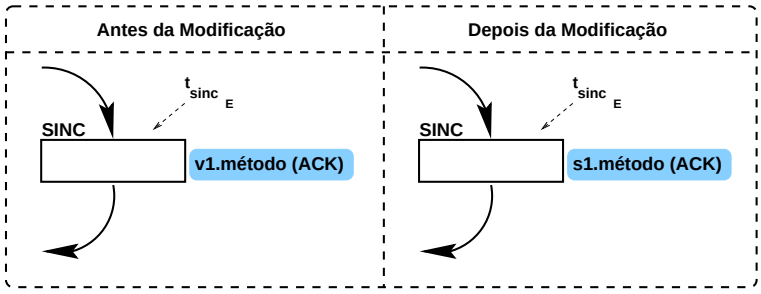

Figura 4.6: Modificação na inscrição de uma transição com invocação de método síncrono pertencente a uma relação de um sincronizador.

Para o emissor, as modificações são idênticas, independentemente da comunicação utilizada ser síncrona ou assíncrona. Basicamente, para cada transição que tem associada invocação de método presente nas relações do sincronizador, substitui-se a variável de classe por outra, associada ao valor do sincronizador que restringe a classe do emissor. O objetivo é fazer com que a invocação que era destinada diretamente à classe com o método, seja intermediada pelo sincronizador. Na Figura 4.6 estão ilustradas

as modificações necessárias para uma transição, na classe de um emissor, que possui invocação de método pertencente a uma ou mais relações do sincronizador.

Modificações Lado Receptor

Para a classe de um receptor, as mudanças diferem de acordo com a comunicação usada. Assim, se o método a ser ativado é *assíncrono*, a transição ao qual se relaciona terá, além das inscrições já associadas a ela, a seguinte inscrição:

se $objetoRemetente = umSincronizador$
então $objetoRemetente.ACKMétodo()$;

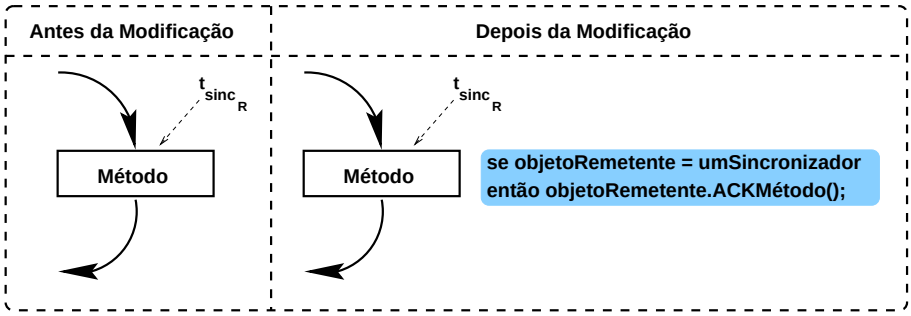

Figura 4.7: Modificação na inscrição de uma transição com ativação de método assíncrono pertencente a uma relação de um sincronizador.

Essa inscrição permitirá à transição, que é *assíncrona*, enviar reconhecimento ao sincronizador. É possível saber se o objeto que requisitou a ativação desse método (*objetoRemetente*) é um sincronizador porque, como vimos no Capítulo 3 (ver página 26), toda mensagem carrega, implicitamente, o identificador de quem a enviou. A adição dessa inscrição ocorre porque, como será explicado adiante, todas as transições de um sincronizador contêm invocações síncronas. Não faz sentido modificar o tipo do método no receptor pois, caso contrário, ele não poderá ser ativado por outros emissores que não pertençam a nenhuma classe do conjunto de classes do sincronizador. As mudanças para um receptor com ativação assíncrona são mostradas na Figura 4.7. Observe que, na inscrição adicionada, o nome do método ativado para envio de reconhecimento é sempre composto pela concatenação da palavra-chave ACK ao nome do método ativado pelo sincronizador e está, portanto, coerente com o mapeamento visto para comunicação síncrona no Capítulo 3.

Se o método ativado é *síncrono*, não há necessidade de mudanças, visto que o retorno é sempre enviado ao objeto que invocou o método.

Modificações no Sincronizador

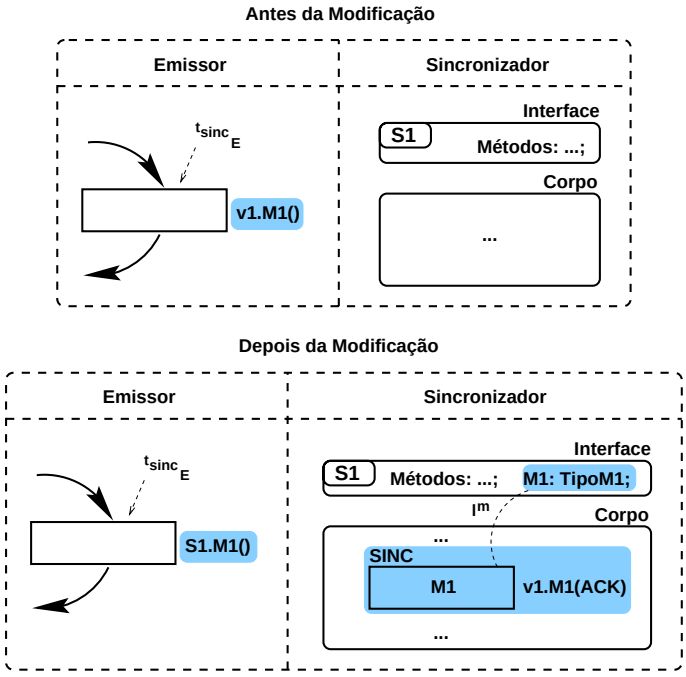

Figura 4.8: Estrutura do sincronizador para intermediação de invocação assíncrona.

Se o método a ser sincronizado é *assíncrono*, então ele será adicionado diretamente à *interface* do sincronizador, sem alterações em seu tipo, e ligado a uma transição, no corpo do sincronizador, através da aplicação l^m. Essa transição terá inscrição de mensagem semelhante à do emissor, com a diferença de que ela será transformada numa invocação *síncrona com reconhecimento*, como pode ser observado na Figura 4.8. A invocação no sincronizador deve ser *síncrona* para que a coordenação seja viável. É por essa razão, também, que se adiciona, na transição de um receptor com ativação de método assíncrono, a inscrição em destaque na Figura 4.7.

Em caso do método ser *síncrono*, ele também será acrescido à *interface* do sincronizador, mas seu tipo será transformado em *assíncrono*. Essa mudança no tipo é necessária porque, a fim de tornar possível a coordenação, toda transição presente no corpo de um sincronizador é *síncrona*. E, como vimos na formalização da comunicação síncrona, uma transição não pode conter ativação e invocação de métodos síncronos

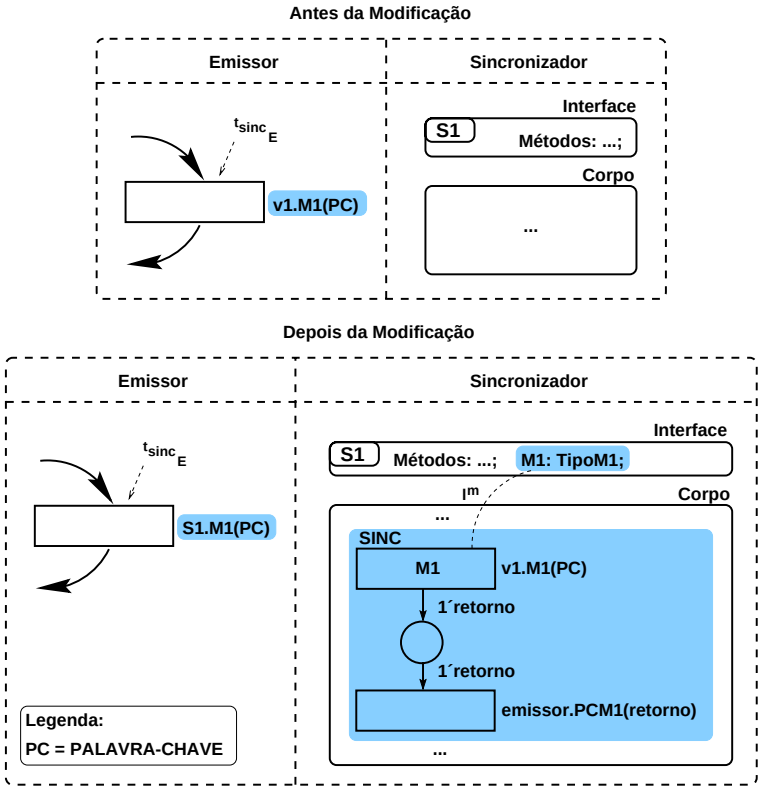

Figura 4.9: Estrutura do sincronizador para intermediação de invocação síncrona.

($T_E \cap T_R = \varnothing$). Então, para resolver esse impasse, o método síncrono é adicionado como assíncrono na *interface* do sincronizador e uma *subestrutura de rede Colorida* é adicionada a seu corpo para que ainda seja possível ao sincronizador enviar o retorno/resultado da ativação do método ao emissor. Todas essas transformações estão ilustradas na Figura 4.9. Observe que a inscrição de mensagem, associada à primeira transição da subestrutura adicionada ao sincronizador, será a mesma que havia no emissor.

Como exemplo, considere o sistema de classes explicado na Seção 3.2.1 (ver página 27), com a modificação da classe $C1$ conter invocação *assíncrona* do *métodoZ* na transição t_{sinc_E}. Isto é, as inscrições de mensagem de t_{sinc_E} são: $v2.métodoX(ACK,a)$, $v3.métodoY(ACK,a)$ e $v4.métodoZ(a)$

Agora, suponha que o projetista especifique o seguinte sincronizador para esse conjunto de classes:

Sincronizador = *S1*

Conjunto de Classes = *C1, C2, C3, C4*

Relações = *C2.métodoX* ACONTECE ANTES *C3.métodoY* ACONTECE ANTES *C4.métodoZ*;

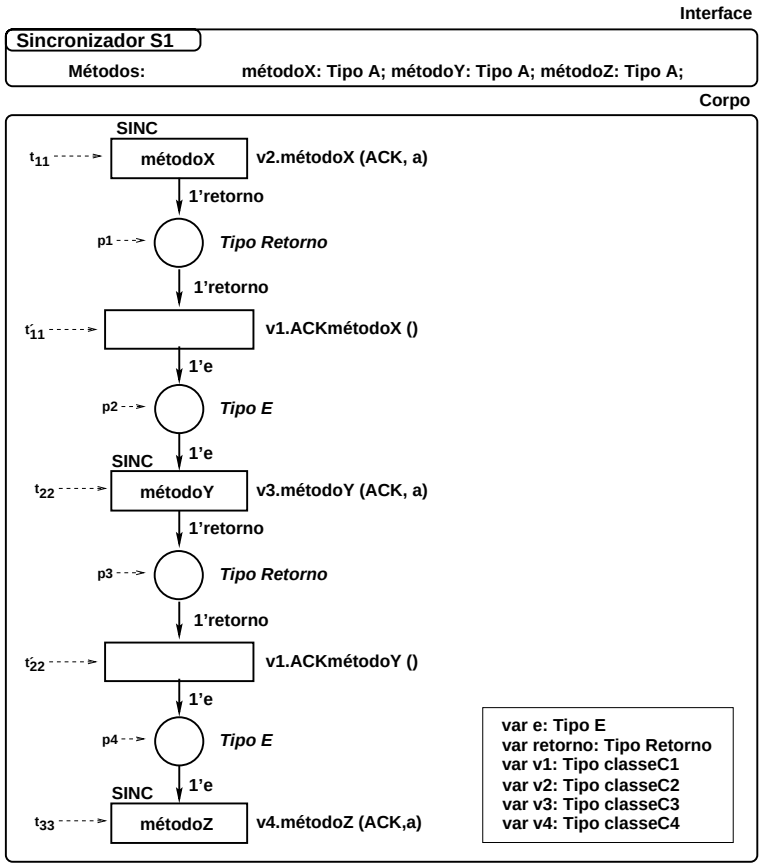

Figura 4.10: Classe do sincronizador S1.

De acordo com as explicações vistas até então, o sincronizador terá a classe ilustrada na Figura 4.10. Perceba que um sincronizador não possui atributos em sua *interface*.

Após modificações, a especificação da classe C1 será como ilustrada na Figura 4.11. As mudanças podem ser observadas mediante comparação com a Figura 3.6 (ver página 28), lembrando que *métodoZ* agora é invocado assincronamente.

Da mesma forma, a classe *C4* sofre as modificações destacadas na Figura 4.12. Não há mudanças nas classes *C2* e *C3* porque elas já utilizam comunicação síncrona.

Aparentemente, pode parecer que a introdução do sincronizador fere a restrição de um sistema RPOO constituir um conjunto integrado de classes. Entretanto, essa impressão é falsa por vários motivos. Primeiro, exige-se um conjunto integrado quando da especificação inicial de um sistema RPOO. Como o sincronizador é especificado textualmente, se o sistema ao qual se aplica já for integrado, continuará a sê-lo, pois

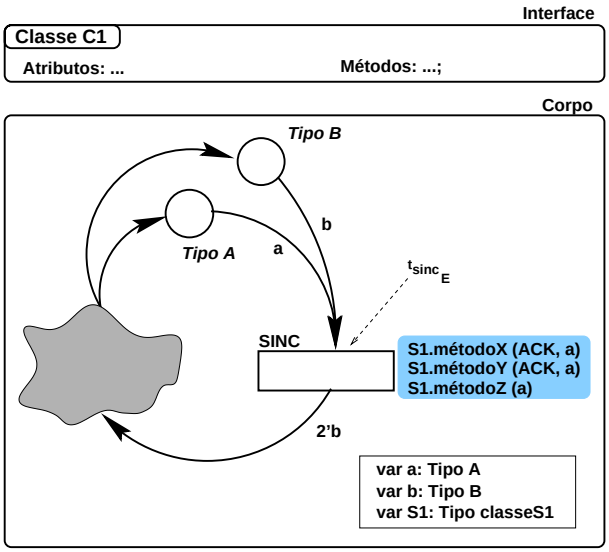

Figura 4.11: Ligação do sincronizador S1 à classe C1.

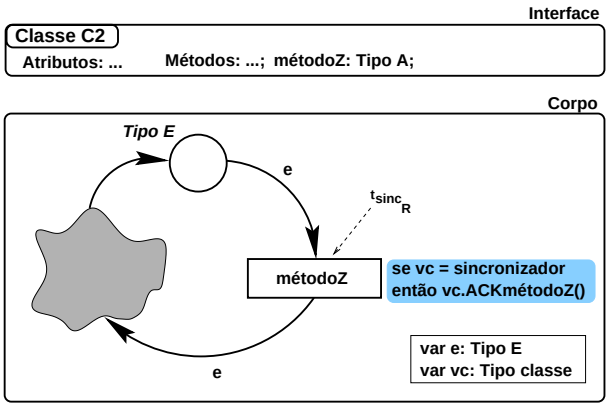

Figura 4.12: Ligação do sincronizador S1 à classe C4.

nenhuma classe é alterada inicialmente. Segundo, mesmo após transformação da descrição textual do sincronizador numa classe RPOO, para o sistema ser estudado, todas as comunicações síncronas devem ser mapeadas nas estruturas assíncronas equivalentes. Logo, embora possa parecer que a introdução do sincronizador não satisfaz a restrição do "Conjunto Integrado de Classes", isso não acontece.

4.4 Formalização de Sincronização de Métodos

Nesta seção apresenta-se a definição de um sincronizador, a formalização de seu mapeamento numa classe RPOO e a formalização das alterações nas classes ao qual se relaciona.

Definição 4.1 (Sincronizador) *Um sincronizador (S) é uma dupla composta de classes do sincronizador (CS) e relações de ordem do sincronizador (RS),* $S = \langle CS, RS \rangle$*, em que:*

- $CS \subseteq \mathcal{C}$ *é um conjunto finito de nomes de classes, tal que* CS *é uma partição de* $\mathcal{C}$*;*
- RS *é um conjunto finito de relações formadas segundo a gramática* G*, definida na Tabela 4.1, tal que:*
  1. *Se* $C_i.m \circledS r \in RS \Rightarrow m \in {}^{k_i}M$*;*
  2. *Se* $C_i.m \circledcirc C_j.m' \wedge C_j.m' \circledcirc C_l.m'' \Rightarrow C_i.m \circledcirc C_l.m'' \wedge \nexists r \in RS : C_l.m'' \circledcirc C_i.m$*.*

Onde $\circledS$ *denota "está presente numa relação" e* $\circledcirc$*, uma das ações definidas em* G*.*

Observações:

A assertiva *(1)* assegura que não há relacionamentos entre métodos inexistentes. A assertiva *(2)* garante a transitividade e a ausência de situações que possam acarretar impasses.

O mapeamento de S numa classe RPOO está formalizado na Definição 4.2. Para melhor compreendê-la, considere que:

- $C_i, C_j \in \mathcal{C}$ são os respectivos nomes das classes $k_i, k_j \in K$;
- m_i e m_j são nomes de métodos, tais que, $m_i \in {}^{k_i}M''$ e $m_j \in {}^{k_j}M''$;
- x, y, r, w são as posições de C_i, C_j, m_i, m_j, respectivamente;
- $n \in \mathbb{N}^*$ é o número associado a um nome de método de uma classe numa dada relação.

Definição 4.2 (Mapeamento de um Sincronizador numa Classe RPOO) *Para um dado sincronizador $S = \langle CS, RS\rangle$ existe a classe $S = \langle {}^S\mathfrak{I}, {}^S\mathfrak{N}, {}^S l^a, {}^S l^m, {}^S\eta, {}^S\mu\rangle$, baseada em Σ''', em que:*

1. ${}^S\mathfrak{I} = \langle {}^SA, {}^SM'', {}^S\tau_{sinc}\rangle$, *tal que:*
   (a) ${}^SA = \varnothing$;
   (b) ${}^SM'' = \{m_i : C_i.m_i \circledS\, r, r \in RS\}$ *é um conjunto finito de nomes de métodos;*
   (c) ${}^S\tau_{sinc} : M'' \to \mathbb{S}$, *onde:*
   - $\forall m_i \in {}^SM''$,

$$ {}^S\tau_{sinc}(m_i) = \begin{cases} {}^{k_i}\tau_{sinc}(m_i) & \text{se } m_i \notin {}^{k_i}M_{sinc}, \\ EmTipoAssinc({}^{k_i}\tau_{sinc}(m_i)) & \text{se } m_i \in {}^{k_i}M_{sinc}. \end{cases} $$

2. ${}^S\mathfrak{N} = \langle {}^SN, {}^Sc, {}^Sg, {}^Se, {}^Si\rangle$, *em que:*
   (a) ${}^SN = \langle {}^SP, {}^ST, {}^SF\rangle$ *está definido como na Definição 2.1;*
   (b) *As funções ${}^Sc, {}^Sg, {}^Se, {}^Si$ são tais que:*
   - $\forall r \in RS : r = n.C_i.m_i$ *ACONTECE ANTES* $C_j.m_j$:
     i. *Se* $m_i \notin {}^{k_i}M_{sinc} \wedge m_j \notin {}^{k_j}M_{sinc}$, *então:*
     - ${}^Sc(p) = E, p \in {}^SP : \{p\} = {}^\bullet t_{y,w} = t^\bullet_{x,r}$;
     - ${}^Sg(t) = \varnothing, t \in \{t_{x,r}, t_{y,w}\} \subseteq {}^ST$;
     - *Para* $f \in {}^SF$, *tem-se:*

$$ {}^Se(f) = \begin{cases} 1'e, e \in E & \text{se } f = (t_{x,r}, p), \\ n'e, e \in E & \text{se } f = (p, t_{y,w}). \end{cases} $$

– ${}^{S}i(p) = \varnothing, \forall p \in {}^{S}P : \{p\} = {}^{\bullet}t_{y,w} = t^{\bullet}_{x,r}$.

ii. Se $m_i \in {}^{k_i}M_{sinc} \wedge m_j \notin {}^{k_j}M_{sinc}$, *então:*

– *Para* $p \in {}^{S}P$, *tem-se:*

$$ {}^{S}c(p) = \begin{cases} E & se\ \{p\} = {}^{\bullet}t_{y,w} = t'^{\bullet}_{x,r}, \\ \S & se\ \{p\} = t^{\bullet}_{x,r} = {}^{\bullet}t'_{x,r}. \end{cases} $$

– ${}^{S}g(t) = \varnothing, t \in \{t_{x,r}, t'_{x,r}, t_{y,w}\} \subseteq {}^{S}T$;

– *Para* $f \in {}^{S}F$, $p, p' \in {}^{S}P$ *tem-se:*

$$ {}^{S}e(f) = \begin{cases} 1'e, e \in E & se\ f \in \{(t'_{x,r}, p')\} : \{p'\} = t'^{\bullet}_{x,r}, \\ n'e, e \in E & se\ f \in \{(p', t_{y,w})\} : \{p'\} = {}^{\bullet}t_{y,w}, \\ 1'retorno, retorno \in \S & se\ f \in \{(p, t'_{x,r}), (t_{x,r}, p)\} : \\ & \{p\} = t^{\bullet}_{x,r} = {}^{\bullet}t'_{x,r}. \end{cases} $$

– ${}^{S}i(p) = \varnothing, \forall p \in {}^{S}P : \{p\} = {}^{\bullet}t_{y,w} = t'^{\bullet}_{x,r} \vee \{p\} = t^{\bullet}_{x,r} = {}^{\bullet}t'_{x,r}$.

iii. Se $m_i \notin {}^{k_i}M_{sinc} \wedge m_j \in {}^{k_j}M_{sinc}$, *então:*

– *Para* $p \in {}^{S}P$, *tem-se:*

$$ {}^{S}c(p) = \begin{cases} E & se\ \{p\} = {}^{\bullet}t_{y,w} = t^{\bullet}_{x,r}, \\ \S & se\ \{p\} = t^{\bullet}_{y,w} = {}^{\bullet}t'_{y,w}. \end{cases} $$

– ${}^{S}g(t) = \varnothing, t \in \{t_{x,r}, t_{y,w}, t'_{y,w}\} \subseteq {}^{S}T$;

– *Para* $f \in {}^{S}F$, $p, p' \in {}^{S}P$ *tem-se:*

$$ {}^{S}e(f) = \begin{cases} 1'e, e \in E & se\ f \in \{(t_{x,r}, p)\} : \{p\} = t^{\bullet}_{x,r}, \\ n'e, e \in E & se\ f \in \{(p, t_{y,w})\} : \{p\} = {}^{\bullet}t_{y,w}, \\ 1'retorno, retorno \in \S & se\ f \in \{(t_{y,w}, p'), (p', t'_{y,w})\} : \\ & \{p'\} = t^{\bullet}_{y,w} = {}^{\bullet}t'_{y,w}. \end{cases} $$

– ${}^{S}i(p) = \varnothing, \forall p \in {}^{S}P : \{p\} = {}^{\bullet}t_{y,w} = t^{\bullet}_{x,r} \vee \{p\} = t^{\bullet}_{y,w} = {}^{\bullet}t'_{y,w}$.

iv. Se $m_i \in {}^{k_i}M_{sinc} \wedge m_j \in {}^{k_j}M_{sinc}$, *então:*

– *Para* $p \in {}^{S}P$, *tem-se:*

$$ {}^{S}c(p) = \begin{cases} E & se\ \{p\} = {}^{\bullet}t_{y,w} = t'^{\bullet}_{x,r}, \\ \mathbb{S} & se\ \{p\} = t^{\bullet}_{y,w} = {}^{\bullet}t'_{y,w} \vee \{p\} = t^{\bullet}_{x,r} = {}^{\bullet}t'_{x,r}. \end{cases} $$

– ${}^{S}g(t) = \varnothing, t \in \{t_{x,r}, t'_{x,r}, t_{y,w}, t'_{y,w}\} \subseteq {}^{S}T$;

– *Para* $f \in {}^{S}F$, $p, p', p'' \in {}^{S}P$ *tem-se:*

$$ {}^{S}e(f) = \begin{cases} 1'e, e \in E & se\ f \in \{(t'_{x,r}, p')\} : \{p'\} = t'^{\bullet}_{x,r}, \\ n'e, e \in E & se\ f \in \{(p', t_{y,w})\} : \{p'\} = {}^{\bullet}t_{y,w}, \\ 1'retorno, retorno \in \mathbb{S} & se\ f \in \{(t_{x,r}, p), (p, t'_{x,r}), (t_{y,w}, p''), \\ & (p'', t'_{y,w})\} : \{p\} = t^{\bullet}_{x,r} = {}^{\bullet}t'_{x,r} \wedge \\ & \{p''\} = t^{\bullet}_{y,w} = {}^{\bullet}t'_{y,w}. \end{cases} $$

– ${}^{S}i(p) = \varnothing, \forall p \in {}^{S}P : \{p\} = t^{\bullet}_{x,r} = {}^{\bullet}t'_{x,r} \vee \{p\} = t'^{\bullet}_{x,r} = {}^{\bullet}t_{y,w} \vee \{p\} = t^{\bullet}_{y,w} = {}^{\bullet}t'_{y,w}$.

- $\forall r \in RS : r = n.C_i.m_i$ *DESABILITA* $C_j.m_j$*:*

  i. Se $m_i \notin {}^{k_i}M_{sinc} \wedge m_j \notin {}^{k_j}M_{sinc}$, *então:*

  – ${}^{S}c(p) = E, p \in {}^{S}P : \{p\} = {}^{\bullet}t_{y,w} = t^{\bullet}_{y,w} = {}^{\bullet}t_{x,r}$;

  – ${}^{S}g(t) = \varnothing, t \in \{t_{x,r}, t_{y,w}\} \subseteq {}^{S}T$;

  – ${}^{S}e(f) = 1'e, e \in E, f \in \{(t_{x,r}, p), (t_{y,w}, p), (p, t_{y,w})\} : \{p\} = {}^{\bullet}t_{y,w} = t^{\bullet}_{y,w} = {}^{\bullet}t_{x,r}$;

  – ${}^{S}i(p) = n'e, e \in E, \forall p \in {}^{S}P : \{p\} = {}^{\bullet}t_{x,r} = {}^{\bullet}t_{y,w} = t^{\bullet}_{y,w}$.

  ii. Se $m_i \in {}^{k_i}M_{sinc} \wedge m_j \notin {}^{k_j}M_{sinc}$, *então:*

  – *Para* $p \in {}^{S}P$, *tem-se:*

$$ {}^{S}c(p) = \begin{cases} E & se\ \{p\} = {}^{\bullet}t_{x,r} = t^{\bullet}_{y,w} = {}^{\bullet}t_{y,w}, \\ \mathbb{S} & se\ \{p\} = t^{\bullet}_{x,r} = {}^{\bullet}t'_{x,r}. \end{cases} $$

  – $g(t) = \varnothing, t \in \{t_{x,r}, t'_{x,r}, t_{y,w}\} \subseteq {}^{S}T$;

– *Para* $f \in {}^{S}F$, $p, p' \in {}^{S}P$ *tem-se:*

$$ {}^{S}e(f) = \begin{cases} 1'e, e \in E & se\ f \in \{(p, t_{x,r}), (p, t_{y,w}), (t_{y,w}, p)\}: \\ & \{p\} = {}^{\bullet}t_{x,r} = {}^{\bullet}t_{y,w} = t^{\bullet}_{y,w} \\ 1'retorno, retorno \in \mathbb{S} & se\ f \in \{(t_{x,r}, p'), (p', t'_{x,r})\}: \\ & \{p'\} = t^{\bullet}_{x,r} = {}^{\bullet}t'_{x,r}. \end{cases} $$

– *Para* $p \in {}^{S}P$ *tem-se:*

$$ {}^{S}i(p) = \begin{cases} \varnothing & se\ \{p\} = t^{\bullet}_{x,r} = {}^{\bullet}t'_{x,r} \\ n'e, e \in E & se\ \{p\} = {}^{\bullet}t_{x,r} = {}^{\bullet}t_{y,w} = t^{\bullet}_{y,w}. \end{cases} $$

iii. Se $m_i \notin {}^{k_i}M_{sinc} \wedge m_j \in {}^{k_j}M_{sinc}$, *então:*

– *Para* $p \in {}^{S}P$, *tem-se:*

$$ {}^{S}c(p) = \begin{cases} E & se\ \{p\} = {}^{\bullet}t_{x,r} = t^{\bullet}_{y,w} \cap {}^{\bullet}t_{y,w}, \\ \mathbb{S} & se\ \{p\} = t^{\bullet}_{y,w} \cap {}^{\bullet}t'_{y,w}. \end{cases} $$

– $g(t) = \varnothing, t \in \{t_{x,r}, t_{y,w}, t'_{y,w}\} \subseteq {}^{S}T$;

– *Para* $f \in {}^{S}F$, $p, p' \in {}^{S}P$ *tem-se:*

$$ {}^{S}e(f) = \begin{cases} 1'e, e \in E & se\ f \in \{(p, t_{x,r}), (p, t_{y,w}), (t_{y,w}, p)\}: \\ & \{p\} = {}^{\bullet}t_{x,r} = {}^{\bullet}t_{y,w} \cap t^{\bullet}_{y,w} \\ 1'retorno, retorno \in \mathbb{S} & se\ f \in \{(t_{y,w}, p'), (p', t'_{y,w})\}: \\ & \{p'\} = t^{\bullet}_{y,w} \cap {}^{\bullet}t'_{y,w}. \end{cases} $$

– *Para* $p \in {}^{S}P$ *tem-se:*

$$ {}^{S}i(p) = \begin{cases} \varnothing & se\ \{p\} = t^{\bullet}_{y,w} \cap {}^{\bullet}t'_{y,w} \\ n'e, e \in E & se\ \{p\} = {}^{\bullet}t_{x,r} = {}^{\bullet}t_{y,w} \cap t^{\bullet}_{y,w}. \end{cases} $$

iv. Se $m_i \in {}^{k_i}M_{sinc} \wedge m_j \in {}^{k_j}M_{sinc}$, *então:*

– *Para* $p \in {}^{S}P$, *tem-se:*

$$ {}^{S}c(p) = \begin{cases} E & se\ \{p\} = {}^{\bullet}t_{x,r} = t^{\bullet}_{y,w} \cap {}^{\bullet}t_{y,w}, \\ \mathbb{S} & se\ \{p\} = t^{\bullet}_{y,w} \cap {}^{\bullet}t'_{y,w} \vee \{p\} = t^{\bullet}_{x,r} = {}^{\bullet}t'_{x,r}. \end{cases} $$

- $g(t) = \varnothing, t \in \{t_{x,r}, t'_{x,r}, t_{y,w}, t'_{y,w}\} \subseteq {}^{S}T$;
- *Para* $f \in {}^{S}F$, $p, p', p'' \in {}^{S}P$ *tem-se:*

$$ {}^{S}e(f) = \begin{cases} 1'e, e \in E & se\ f \in \{(p, t_{x,r}), (p, t_{y,w}), (t_{y,w}, p)\} : \\ & \{p\} = {}^{\bullet}t_{x,r} = {}^{\bullet}t_{y,w} \cap t_{y,w}^{\bullet} \\ 1'retorno, retorno \in \mathcal{S} & se\ f \in \{(t_{x,r}, p'), (p', t'_{x,r}), (t_{y,w}, p''), \\ & (p'', t'_{y,w})\} : \{p'\} = t_{x,r}^{\bullet} = {}^{\bullet}t'_{x,r} \wedge \\ & \{p''\} = t_{y,w}^{\bullet} \cap {}^{\bullet}t'_{y,w}. \end{cases} $$

- *Para* $p \in {}^{S}P$ *tem-se:*

$$ {}^{S}i(p) = \begin{cases} \varnothing & se\ \{p\} = t_{x,r}^{\bullet} = {}^{\bullet}t'_{x,r} \vee \{p\} = t_{y,w}^{\bullet} \cap {}^{\bullet}t'_{y,w}, \\ e, e \in E & se\ \{p\} = {}^{\bullet}t_{x,r} = {}^{\bullet}t_{y,w} \cap t_{y,w}^{\bullet}. \end{cases} $$

- $\forall r \in RS : r =$ *ORDENE TUDO (* $n.C_i.m_i$ *ACONTECE ANTES ... ACONTECE ANTES* $C_j.m_j$*):*
  1. *Quaisquer que sejam os tipos dos métodos* m_i *e* m_j*, ou seja,* $\forall m_i \in {}^{k_i}M'' \wedge \forall m_j \in {}^{k_j}M''$*, tem-se:*
     - $c(p) = E, p \in {}^{S}P : p \in {}^{\bullet}t_{x,r} = t_{y,w}^{\bullet}$;
     - $g(t) = \varnothing, t \in \{t_{x,r}, t_{y,w}\} \subseteq {}^{S}T$;
     - $e(f) = 1'e, e \in E, f \in \{(p, t_{x,r}), (t_{y,w}, p)\}$.
     - $i(p) = 1'e, e \in E, p \in {}^{S}P : p \in {}^{\bullet}t_{x,r} = t_{y,w}^{\bullet}$.

3. $\nexists\, {}^{S}l^{a} : {}^{S}A \rightarrow {}^{S}P$;
4. $\nexists\, {}^{S}\eta : {}^{S}T \rightarrow 2^{\mathcal{V}_{\mathfrak{C}} \times \mathfrak{C}}$;
5. ${}^{S}l^{m} : {}^{S}M'' \rightarrow {}^{S}T \times \mathcal{V}_{\Sigma'''}$*, tal que:*

$$\forall n.C_i.m_i \circledS\, r, r \in RS,\ {}^{S}l^{m}(m_i) = (t_{x,r}, var), var \in \mathcal{V}_{\Sigma'''}, t_{x,r} \in {}^{S}T;$$

6. $\forall n.C_i.m_i \circledS\, r, r \in RS, t_{x,r}, t'_{x,r} \in {}^{S}T$*, tem-se:*
   (a) *Se* $m_i \notin {}^{k_i}M_{sinc}$*,*

$$ {}^{S}\mu(t_{x,r}) = v_i.m_i(ACK, te), v_i \in \mathcal{V}_{C_i}, te \in T_{\Sigma'''}. $$

(b) Se $m_i \in {}^{k_i}M_{sinc}$,

$$
{}^{S}\mu(t_{x,r}) = \begin{cases} v_i.m_i(ACK, te), v_i \in \mathcal{V}_{C_i}, te \in T_{\Sigma'''} & se\ {}^{k_i}\tau_{sinc}(m_i) = \\ & \langle TipoACK, x\rangle,\ x \in \mathcal{S}, \\ v_i.m_i(REPLY, te), v_i \in \mathcal{V}_{C_i}, te \in T_{\Sigma'''} & se\ {}^{k_i}\tau_{sinc}(m_i) = \\ & \langle TipoREPLY, x\rangle,\ x \in \mathcal{S}. \end{cases}
$$

$$
{}^{S}\mu(t'_{x,r}) = \begin{cases} v.ACKm_i(), v \in \mathcal{V}_{\mathcal{C}} & se\ {}^{k_i}\tau_{sinc}(m_i) = \\ & \langle TipoACK, x\rangle,\ x \in \mathcal{S}, \\ v.REPLYm_i(te), v \in \mathcal{V}_{\mathcal{C}}, te \in T_{\Sigma'''} & se\ {}^{k_i}\tau_{sinc}(m_i) = \\ & \langle TipoREPLY, x\rangle,\ x \in \mathcal{S}. \end{cases}
$$

Observações:

(1) A *interface* de um sincronizador *(a)* não possui atributos, *(b)* contém apenas os nomes de métodos presentes em suas relações e *(c)* todos esses métodos são assíncronos;

(2) Para cada nome de classe e de método, $C_i.m_i$, presente numa relação de um sincronizador, existe uma transição correspondente na rede Colorida que constitui o corpo do sincronizador, a qual não tem guarda associada. A quantidade de lugares presentes no corpo, bem como suas cores e marcações iniciais dependem do tipo dos métodos das transições para as quais constituem lugar de entrada e/ou saída. As expressões dos arcos são determinadas pelas relações e pelos tipos de métodos;

(3) Inexiste uma aplicação para relacionar atributos da *interface* de um sincronizador a lugares de seu corpo. Veja que, por causa de *(1.a)*, o conjunto de atributos é vazio;

(4) Nenhuma transição presente no corpo de um sincronizador contém inscrição para criação de objetos;

(5) Todo método da *interface* de um sincronizador está relacionado a uma transição em seu corpo, indexada pela posição, no conjunto de relações, do nome da classe mais a do nome do método;

(6) Se o método da relação é assíncrono, a transição a qual ele se relaciona contém inscrição de mensagem com invocação síncrona *com reconhecimento*. Se o método é síncrono, a inscrição será do mesmo tipo do método. Além disso, quando o método é síncrono, há uma transição a mais usada para enviar o reconhecimento/resposta ao emissor, cujo identificador está associado à variável v.

A formalização para as mudanças nas classes dos emissores e receptores coordenados por um sincronizador S estão nas Definições 4.3 e 4.4. Para melhor compreendê-las, considere que:

- $\mathcal{V}_S \subset \mathcal{V}_{\mathcal{C}}$ é um conjunto de variáveis cujos valores representam identificadores de sincronizadores.

Definição 4.3 (Mudanças Lado Emissor) *Seja* $k_i = \langle {}^{k_i}\mathfrak{I}, {}^{k_i}\mathcal{N}, {}^{k_i}l^a, {}^{k_i}l^m, {}^{k_i}\eta, {}^{k_i}\mu\rangle$ *a classe de um emissor cujas transições contêm invocação de método presente numa das relações de um sincronizador* $S = \langle CS, RS\rangle$*, ou seja,* $n.C_j.m_j \circledS\ r, r \in RS \wedge \exists t \in {}^{k_i}T : v_j.m_j(te) \in {}^{k_i}\mu(t)$*. A classe* $k_i' = \langle {}^{k_i}\mathfrak{I}, {}^{k_i}\mathcal{N}, {}^{k_i}l^a, {}^{k_i}l^m, {}^{k_i}\eta, {}^{k_i'}\mu\rangle$*, originada após incorporação de* S *a* k_i*, é tal que:*

$$\forall t \in {}^{k_i}T : v_j.m_j(te) \in {}^{k_i}\mu(t), v_j \in \mathcal{V}_{C_j}, te \in T_{\Sigma'''},$$

$${}^{k_i'}\mu(t) = ({}^{k_i}\mu(t) - \{v_j.m_j(te)\}) \cup \{s.m_j(te)\}, s \in \mathcal{V}_S.$$

Definição 4.4 (Mudanças Lado Receptor) *Seja* $k_i = \langle {}^{k_i}\mathfrak{I}, {}^{k_i}\mathcal{N}, {}^{k_i}l^a, {}^{k_i}l^m, {}^{k_i}\eta, {}^{k_i}\mu\rangle$ *a classe de um receptor cuja transição contém ativação de método presente numa das relações de um sincronizador* $S = \langle CS, RS\rangle$*, ou seja,* $n.C_i.m_i \circledS\ r, r \in RS \wedge \exists t \in {}^{k_i}T : m_i \in {}^{k_i}l^m(t)$*. A classe* $k_i' = \langle {}^{k_i}\mathfrak{I}, {}^{k_i}\mathcal{N}, {}^{k_i}l^a, {}^{k_i}l^m, {}^{k_i}\eta, {}^{k_i'}\mu\rangle$*, originada após incorporação de* S *a* k_i*, é tal que:*

$$\forall t \in {}^{k_i}T : {}^{k_i}l^m(m_i) = \langle t, v\rangle, v \in \mathcal{V}_{\Sigma'''},$$

$$ {}^{k'_i}\mu(t) = \begin{cases} {}^{k_i}\mu(t) \cup \{s.ACKm_i()\}, s \in \mathcal{V}_S & se\ m_i \in {}^{k_i}M, \\ {}^{k_i}\mu(t) & se\ m_i \in {}^{k_i}M_{sinc}. \end{cases} $$

Observe que a inscrição de mensagem adicionada na Definição 4.4, $s.ACKm_i()$, estará sempre dentro da expressão condicional destacada na Figura 4.7 (ver página 61).

Capítulo 5

Análise para Mecanismo de Interação Acrescido a RPOO

Os métodos de análise para redes de Petri permitem o estudo de propriedades comportamentais e estruturais. Para as redes de Petri Coloridas, propriedades comportamentais podem ser estudadas por meio de *grafos de ocorrência* (*espaços de estado*) e propriedades estruturais, por meio de *invariantes*.

Neste trabalho, deseja-se observar o comportamento de sistemas RPOO, estudando a interação entre os objetos, os estados pelos quais passam, entre outros. Conseqüentemente, interessa o estudo das propriedades baseado em grafos de ocorrência.

Os grafos de ocorrência são grafos bipartidos direcionados, em que os nós representam marcações alcançáveis e os arcos, elementos de ligação que ocorrem.

Nos Capítulos 3 e 4 foram apresentadas as abstrações para introdução de novos mecanismos de interação para o formalismo original de RPOO. Na comunicação síncrona, por exemplo, uma transição síncrona é mapeada numa sub-rede de Petri Colorida, a qual é incorporada ao corpo da respectiva classe RPOO. Perceba que essa introdução de novos lugares e transições, no corpo da classe resultante do mapeamento, pode implicar num considerável aumento no tamanho do grafo de ocorrência a ser gerado, pois mais estados tornam-se possíveis. Então, faz-se necessário o uso de algum artifício que permita a construção de grafos de ocorrência em que sejam omitidos os estados que não se deseja observar.

Nesse sentido, o presente capítulo mostra a definição de uma *especificação de equi-*

valência, que possibilita o uso do mapeamento para o mecanismo de comunicação síncrona, porém sem interferir no tamanho do grafo de ocorrência das classes RPOO envolvidas. Para facilitar o entendimento, na Seção 5.1 introduzem-se os conceitos relevantes à análise baseada em grafos de ocorrência, enquanto que na Seção 5.2 encontra-se a respectiva especificação de equivalência para o mecanismo de comunicação síncrona.

5.1 Análise Baseada em Grafos de Ocorrência

Os métodos de análise das redes de Petri visam ao estudo de suas propriedades dinâmicas. Para redes pequenas, uma análise por simulação pode ser suficiente. Entretanto, para modelos complexos, uma análise mais formal se faz necessária.

O estudo das propriedades comportamentais das redes de Petri Coloridas pode basear-se na construção de um grafo de ocorrência e no uso de regras de prova. Os grafos de ocorrência são grafos bipartidos gerados a partir de uma dada marcação inicial, M_0, da rede de Petri Colorida em questão. Cada regra de prova estabelece uma relação entre uma propriedade dinâmica da rede Petri Colorida e uma propriedade do grafo de ocorrência correspondente. Tanto o processo de construção do grafo de ocorrência como o de verificação das regras de prova podem ser automatizados.

Os grafos de ocorrência classificam-se em *grafos de ocorrência completos*, *grafos de ocorrência com classes de equivalência* e *grafos de ocorrência com simetrias* [Jen95].

Os grafos de ocorrência completos são o tipo mais básico de grafo de ocorrência. Eles apresentam um nó para cada marcação alcançável e um arco para cada elemento de ligação que ocorre. Assim, se $M_0[be_1\rangle M_1$, então há um nó para M_0, um para M_1 e um arco direcionado rotulado com be_1, ligando o nó de M_0 ao de M_1.

Formalmente, tem-se:

Definição 5.1 (Grafo de Ocorrência Completo) *O* Grafo de Ocorrência Completo *(GO) de uma rede de Petri Colorida é o grafo direcionado* $GO = (V, A, N)$*, em que:*

- $V = [M_0\rangle$ *é o conjunto de vértices;*
- $A = \{(M_1, be, M_2) \in V \times \mathbb{BE} \times V \mid M_1[be\rangle M_2\}$ *é o conjunto de arcos direcionados, tal que* $V \cap A = \varnothing$ *e* $\mathbb{BE}$ *denota o conjunto de todos os elementos de ligação;*

- $\forall a = (M_1, be, M_2) \in A : N(a) = (M_1, M_2)$, *onde* $N : A \rightarrow V \times V$ *é a função de nó.*

Entretanto, como os grafos de ocorrência completos mostram-se inadequados ao estudo de sistemas complexos por causa do *problema da explosão de estados*, há os outros dois tipos de grafos, em que cada nó (arco) representa uma classe de equivalência de marcações (elementos de ligação).

Nesse trabalho, interessam os grafos de ocorrência com classes de equivalência. Uma explicação detalhada sobre os métodos de análise para as redes de Petri Coloridas pode ser encontrada em [Jen95].

Grafo de Ocorrência com Classes de Equivalência

A construção de um grafo de ocorrência com classes de equivalência baseia-se numa *especificação de equivalência*, que estabelece quando duas (dois) marcações (elementos de ligação) devem ser consideradas(os) equivalentes. Formalmente:

Definição 5.2 (Especificação de Equivalência) *Uma especificação de equivalência para uma rede de Petri Colorida é um par* $(\approx_M, \approx_{BE})$*, em que* $\approx_M$ $(\approx_{BE})$ *é uma relação de equivalência em* $\mathbb{M}(\mathbb{BE})$*.* $\mathbb{M}(\mathbb{BE})$ *denota o conjunto de todas(os) as(os) marcações (elementos de ligação).*

Uma especificação de equivalência pode ser *consistente* ou *compatível*. Numa especificação consistente, se duas marcações pertencem a uma mesma classe de equivalência, então suas marcações *diretamente* alcançáveis também devem pertencer a uma mesma classe de equivalência. Numa especificação compatível, os elementos de ligação, cujas ocorrências levam a uma marcação final pertencente a mesma classe de equivalência de uma marcação inicial, são ignorados. Conseqüentemente, a ocorrência de tais elementos de ligação torna-se inobservável, pois as marcações permanecem dentro de uma mesma classe de equivalência. Esses dois conceitos são formalizados na Definição 5.3. Para melhor compreendê-la, considere que:

- Para duas marcações (ou dois elementos de ligação) x e y, se x é equivalente a y, escreve-se $x \approx y$, e a classe de equivalência é $[x]$;

- Para um conjunto X, $[X]$ denota o conjunto de elementos equivalentes a algum elemento de X. Por exemplo, $[[M_0\rangle]$ denota o conjunto de marcações equivalentes a uma marcação alcançável;

- $Next(M_1) = \{(be, M) \in \mathbb{BE} \times \mathbb{M} : M_1[be\rangle M\}$, é o conjunto de todos os pares (elemento de ligação, marcação final) que efetivamente estão habilitados a ocorrer para uma dada marcação inicial $M_1 \in \mathbb{M}$;

- $Next_\tau(M_1) = \{(be, M) \in \mathbb{BE} \times \mathbb{M} : M_1[\tau be\rangle M\}$,
  onde $M_1[\tau be\rangle M$ denota que existe uma sequência de ocorrência finita $M_1[be_1\rangle M_2[be_2\rangle M_3 ... M_n[be_n\rangle M_{n+1}[be\rangle M$, tal que $n \in \mathbb{N}$, $be_j \in \mathbb{BE}$, para todo $j \in 1..n$ e $M_i \approx_M M_1$, para todo $i \in 1..n+1$, <u>mas</u> $M \not\approx_M M_1$;

- Dois pares $(be, M), (be^*, M^*) \in \mathbb{BE} \times \mathbb{M}$ são equivalentes entre si - $(be, M) \approx (be^*, M^*)$ - se, e somente se, $be \approx_{BE} be^*$ e $M \approx_M M^*$.

Definição 5.3 (Especificação de Equivalência Consistente/Compatível) *Uma especificação de equivalência é* consistente *se, e somente se, a seguinte propriedade é satisfeita,* $\forall M_1, M_2 \in [[M_0\rangle]$*:*

$M_1 \approx_M M_2 \Rightarrow [Next(M_1)] = [Next(M_2)]$.

Uma especificação de equivalência é compatível *se, e somente se, a seguinte propriedade é satisfeita,* $\forall M_1, M_2 \in [[M_0\rangle]$*:*

$M_1 \approx_M M_2 \Rightarrow [Next_\tau(M_1)] = [Next_\tau(M_2)]$.

A Definição 5.4 formaliza um grafo de ocorrência com classes de equivalência. Para melhor entendê-la, considere que $M_\approx(BE_\approx)$ denota o conjunto de *todas* as classes de equivalência para uma relação $\approx_M$ ($\approx_{BE}$).

Definição 5.4 (Grafo de Ocorrência com Classes de Equivalência) *Dada uma rede de Petri Colorida e uma especificação de equivalência* consistente ($\approx_M$, $\approx_{BE}$)*, o* grafo de ocorrência com classes de equivalência *(GOE) é o grafo direcionado* $GOE = (V, A, N)$*, em que:*

- $V = \{C \in M_\approx \mid C \cap [M_0\rangle \neq \varnothing\}$;

- $A = \{(C_1, BE, C_2) \in V \times BE_{\approx} \times V \mid \exists (M_1, be, M_2) \in C_1 \times BE \times C_2 : M_1[be\rangle M_2\}$;
- $\forall a = (C_1, BE, C_2) \in A : N(a) = (C_1, C_2)$.

Para uma especificação de equivalência compatível, *substitui-se* $M_1[be\rangle M_2$ *por* $M_1[\tau be\rangle M_2$.

Por fim, vale ressaltar que é importante ter cuidado ao escolher uma especificação de equivalência pois ela tem um impacto significativo sobre o conjunto de propriedades que se pode investigar no respectivo grafo de ocorrência. Note que uma especificação muito permissiva pode tornar tudo equivalente, enquanto outra muito restritiva pode gerar um grafo com apenas uma marcação por classe de equivalência, isto é, um grafo com o mesmo tamanho do grafo de ocorrência completo.

5.2 Especificação de Equivalência para Comunicação Síncrona

No Capítulo 3, viu-se que a abstração para comunicação síncrona mapeia uma transição síncrona num bloco que é uma sub-rede de Petri Colorida. Na Seção 5.1 viu-se também que uma especificação de equivalência deve ser escolhida de modo que ainda seja possível verificar as propriedades comportamentais desejadas, que, no caso deste trabalho, são vivacidade e limitação.

O objetivo desta seção é apresentar a definição e a prova de uma especificação de equivalência que torne o mapeamento das abstrações de comunicação síncrona transparente à análise. Para tanto, define-se uma especificação de equivalência *compatível* que ignora as ocorrências dos elementos de ligação cujas transições pertencem ao(s) bloco(s) adicionado(s).

O restante da seção organiza-se da seguinte forma: na Seção 5.2.1 apresentam-se e provam-se propriedades comportamentais dos blocos de rede de Petri Colorida usados nos mapeamentos, na Seção 5.2.2 define-se a especificação de equivalência compatível desejada, cuja prova é mostrada na Seção 5.2.3.

5.2.1 Propriedades do Bloco de Rede de Petri Colorida

Esta seção apresenta a prova de que os blocos de rede de Petri Coloridas usados no mapeamento estabelecem um fluxo que caracteriza um *invariante de lugar* [Jen95] e, portanto, podem ser ignorados na construção dos grafos de ocorrência porque não alteram as propriedades de vivacidade e limitação do corpo das classes RPOO, na qual são inseridos.

Antes de prosseguir à prova, apresenta-se uma explanação breve sobre invariante de lugar.

Invariante de Lugar

Um invariante de lugar estabelece um conjunto de equações que é sempre satisfeito para toda marcação alcançável no sistema. Essas equações referem-se a propriedades estruturais das redes. A Definição 5.5 e o Teorema 5.1 a seguir foram extraídos de [Jen95] (ver páginas 109 e 110, respectivamente). De acordo com essas formalizações, basta provar a existência de fluxo de lugar nos blocos usados no mapeamento, para garantir que eles definem um invariante de lugar. Um fluxo de lugar garante que todas as fichas que chegam aos lugares de uma determinada (sub-)rede de Petri também saem deles e que outras não são criadas.

Definição 5.5 (Fluxo e Invariante de Lugar) *Para uma rede de Petri Colorida não-hierárquica, um* conjunto ponderado de lugares[1] *com imagem $A \in \Sigma$ é o conjunto de funções $W = \{W_p\}_{p \in P}$ tal que $W_p \in [C(p)^{cp} \rightarrow A^{cp}]_L$, $\forall p \in P$.*

1. *W é um* fluxo de lugar[2] *se, e somente se:*
   $$\forall (t,b) \in BE : \sum_{p \in P} W_p(e(p,t)\langle b \rangle) = \sum_{p \in P} W_p(e(t,p)\langle b \rangle).$$
2. *W detemina um* invariante de lugar[3] *se, e somente se:*
   $$\forall M \in [M_0\rangle : \sum_{p \in P} W_p(M(p)) = \sum_{p \in P} W_p(M_0(p)).$$

Onde:

[1] Tradução para *set of place weights*.
[2] Tradução para *place flow*.
[3] Tradução para *place invariant*.

- *Cada peso W_p é uma função que mapeia $C(p)$ em algum conjunto de cor $A \in \Sigma$ compartilhados por todos os pesos;*

- *X^{cp} denota o conjunto de todos os* conjuntos ponderados[4] *de X. Um conjunto ponderado é um multiconjunto em que os coeficientes dos elementos podem ser negativos;*

- *$[X \to Y]_L$ denota o conjunto de todas as funções lineares de X em Y.*

Teorema 5.1 (Correspondência entre Fluxo e Invariante de Lugar) *W é um fluxo de lugar se, e somente se, W determina um invariante de lugar*

Para facilitar o entendimento, as provas para os blocos foram separadas para o lado emissor e o receptor.

Prova de fluxo de lugar para bloco lado emissor

Seja B_E um bloco para mapeamento de uma transição síncrona $t_{sinc} \in T_E$ com invocação de método síncrono. De acordo com a Definição 3.7, item 3(c) (ver página 40) determinar-se que:

- $\forall p \in P_{B_E}$ e $\forall t \in T_{B_E}$:

$$W_p(e(t_{in}, p_1)\langle b\rangle) = W_p(e(p_2, t_{out})\langle b\rangle) \tag{5.1}$$

$$W_p(e(t_2, p_2)\langle b\rangle) = W_p(e(p_1, t_2)\langle b\rangle) \tag{5.2}$$

$$W_p(e(t_{in}, p_3)\langle b\rangle) = W_p(e(p_3, t_{out})\langle b\rangle) \tag{5.3}$$

Logo, adicionando $W_p(e(t_2, p_2)\langle b\rangle)$ a ambos os lados da equação 5.1, tem-se:

$$W_p(e(t_{in}, p_1)\langle b\rangle) + W_p(e(t_2, p_2)\langle b\rangle) = W_p(e(p_2, t_{out})\langle b\rangle) + W_p(e(t_2, p_2)\langle b\rangle) \tag{5.4}$$

Mas, de acordo com a equação 5.2, $W_p(e(t_2, p_2)\langle b\rangle) = W_p(e(p_1, t_2)\langle b\rangle)$. Então, aplicando essa igualdade ao lado direito da equação 5.4, tem-se:

$$W_p(e(t_{in}, p_1)\langle b\rangle) + W_p(e(t_2, p_2)\langle b\rangle) = W_p(e(p_2, t_{out})\langle b\rangle) + W_p(e(p_1, t_2)\langle b\rangle) \tag{5.5}$$

[4] Tradução para *weighted-sets*.

De forma semelhante, pode-se adicionar $W_p(e(t_{in}, p_3)\langle b\rangle)$ a ambos os lados da equação 5.5 e depois substituí-lo, no lado direito da nova equação gerada após adição, pela igualdade em 5.3, resultando na seguinte equação:

$$\begin{aligned} W_p(e(t_{in}, p_1)\langle b\rangle) + W_p(e(t_2, p_2)\langle b\rangle) + W_p(e(t_{in}, p_3)\langle b\rangle) &= W_p(e(p_2, t_{out})\langle b\rangle) \\ &\quad + W_p(e(p_1, t_2)\langle b\rangle) \\ &\quad + W_p(e(p_3, t_{out})\langle b\rangle) \\ \Rightarrow \forall (t,b) \in BE : \sum_{p \in P_{B_E}} W_p(e(t,p)\langle b\rangle) &= \sum_{p \in P_{B_E}} W_p(e(p,t)\langle b\rangle) \end{aligned} \tag{5.6}$$

Portanto, conclui-se que todo bloco B_E apresenta um fluxo de lugar e, segundo o Teorema 5.1, um invariante de lugar. □

Observe que o número de fichas nesse bloco é sempre múltiplo do número de invocações síncronas da transição síncrona mais um, ou seja,

$$\sum_{p \in P_{B_E}} |W_p(e(t,p)\langle b\rangle)| + \sum_{p \in P_{B_E}} |W_p(e(p,t)\langle b\rangle)| = (|MC\_OBJ(t_{sinc})| + 1) \cdot (x - y) \tag{5.7}$$

onde:

- $|cp|$ denota o número de elementos de um conjunto ponderado cp;
- $x \in \mathbb{N}$ denota o número de ocorrências da transição $t_{in} \in T_{B_E}$;
- $y \in \mathbb{N}$ denota o número de ocorrências da transição $t_{out} \in T_{B_E}$.

Exemplificando, note que o número de fichas presentes no bloco mostrado na Figura 3.8 (ver página 29) é sempre múltiplo de 4, pois há 3 invocações de métodos síncronos na t_{sinc_E} da Figura 3.6 (ver página 28).

Pelo fluxo de lugar e devido ao fato do número de invocações síncronas em uma transição síncrona $t_{sinc} \in T_E$ ser sempre finito, conclui-se que um bloco B_E é limitado.

Além disso, B_E é sempre vivo pois, uma vez que $t_{in_{B_E}}$ ocorre, $t_{2_{B_E}}$ seguramente ocorrerá em algum momento no futuro, quando as classes, cujos métodos síncronos foram invocados em $t_{in_{B_E}}$, forem vivas, porque o ambiente de comunicação garante a entrega da mensagem. Após $t_{2_{B_E}}$ ocorrer, a habilitação de $t_{out_{B_E}}$ é garantida, pois essa transição não sofre interferência externa.

Por conseguinte, conclui-se que o bloco B_E pode ser ignorado na análise das classes RPOO.

Prova de fluxo de lugar para bloco lado receptor

A prova de que um bloco B_R apresenta invariante de lugar é semelhante à descrita para um bloco B_E e, portanto, será omitida.

5.2.2 Especificação de Equivalência Compatível

Agora que já se provou que os blocos não alteram as propriedades de vivacidade e limitação do corpo da classe RPOO na qual são inseridos, pode-se definir uma especificação de equivalência compatível em que:

1. A relação $\approx_M$ define, como equivalentes, marcações resultantes de alterações *apenas* nas marcações dos lugares de entrada das transições síncronas mapeadas e nos lugares dos respectivos blocos que as substituem;

2. A relação $\approx_{BE}$ estabelece como equivalentes elementos de ligação contendo transições pertencentes a um *mesmo* bloco.

Antes de apresentar a especificação de equivalência compatível, vale uma consideração sobre o modo como um grafo de ocorrência com classe de equivalência compatível é gerado.

Construção de um Grafo de Ocorrência com Classes de Equivalência para uma Especificação de Equivalência Compatível

Segundo a Definição 5.4 (ver página 78), tem-se que para uma especificação de equivalência compatível:

$A = \{(C_1, BE, C_2) \in V \times BE_{\approx} \times V \mid \exists (M_1, be, M_2) \in C_1 \times BE \times C_2 : M_1[\tau be\rangle M_2\}$

Isto significa que (i) um arco direcionado $a \in A$, que parte de um nó $C1$, corresponde à ocorrência de um elemento de ligação be que levou à marcação M_2, pertencente ao nó $C2$ e (ii) toda uma sequência de ocorrência τ com elementos de ligação que aconteceram anteriormente a be não gerou nenhum novo nó. Em suma, dada uma marcação M_1,

pertencente a um nó $C1$, um arco direcionado, partindo de $C1$ para $C2$, só é gerado quando a aplicação da especificação de equivalência torna-se falsa, quando $M_1 \not\approx M_2$.

É importante ter esse entendimento em mente ao ler as especificações de equivalência compatíveis, que são apresentadas a seguir para o lado emissor e receptor.

Lado Emissor:

Relação de Equivalência no Conjunto de Marcações - $\approx_M$

Para duas marcações $M_1, M_2 \in [[M_0\rangle]$, tem-se que:

$$
\begin{aligned}
M_1 \approx_M M_2 \iff \bigvee_{\forall t_{sinc} \in {}^{k_i}T_E} \Bigg(& (\forall p \in ({}^{k_i}P - {}^{\bullet}t_{sinc}) : M_1(p) = M_2(p)) \\
& \wedge \Bigg(\sum_{\forall p \in {}^{\bullet}t_{sinc}} |M_1(p)| + \frac{\sum\limits_{\forall p \in {}^{\bullet}t_{sinc}} |e(p, t_{sinc})|}{\sum\limits_{\forall p' \in t_{in}^{\bullet}, t_{in} \in {}^{t_{sinc}}B_E} |e(t_{in}, p')|} \\
& \cdot \sum_{\forall p'' \in {}^{t_{sinc}}P_{B_E}} |M_1(p'')| = \sum_{\forall p \in {}^{\bullet}t_{sinc}} |M_2(p)| \\
& + \frac{\sum\limits_{\forall p \in {}^{\bullet}t_{sinc}} |e(p, t_{sinc})|}{\sum\limits_{\forall p' \in t_{in}^{\bullet}, t_{in} \in {}^{t_{sinc}}B_E} |e(t_{in}, p')|} \cdot \sum_{\forall p'' \in {}^{t_{sinc}}P_{B_E}} |M_2(p'')| \Bigg) \Bigg)
\end{aligned}
\tag{5.8}
$$

Explicação:

Duas marcações M_1 e M_2 são equivalentes quando, para *toda* transição síncrona, t_{sinc}, presente no corpo de uma classe RPOO, k_i, e que contém invocação de método síncrono, a seguinte condição é satisfeita *ao menos uma vez*:

Para todo lugar p, pertencente ao conjunto de lugares da classe não-mapeada k_i, que não é lugar de entrada da t_{sinc} a ser substituída, o multiconjunto de fichas presentes em p, na marcação M_1, é idêntico àquele presente na marcação M_2. Além disso, o somatório do número de fichas presentes em todo lugar p, que é lugar de entrada de t_{sinc}, *para marcação* M_1, *mais a razão* entre (i) o somatório do número de fichas retiradas dos lugares de entrada da transição t_{in}, do bloco B_E, quando ela ocorre, e (ii) o somatório do número de fichas depositadas nos lugares de saída dessa mesma transição, multiplicado pelo somatório do número de fichas presentes em todos os lugares de B_E, *é igual ao* somatório do número de fichas presentes em todo lugar p, que é lugar de entrada de t_{sinc}, *para marcação* M_2, *mais a razão* entre (i) o somatório do número

de fichas retiradas dos lugares de entrada da transição t_{in}, do bloco B_E, quando ela ocorre, e (ii) o somatório do número de fichas depositadas nos lugares de saída dessa mesma transição, *multiplicado* pelo somatório do número de fichas presentes em todos os lugares de B_E.

Relação de Equivalência no Conjunto de Ligações $\approx_{BE}$

Considerando que se denota por $t(be)$ a transição de um elemento de ligação $be \in \mathbb{BE}$, tem-se:

$$be_1 \approx_{BE} be_2 \iff \bigvee_{\forall t_{sinc} \in {}^{k_i}T_E} \left(t(be_1) \in {}^{t_{sinc}}T_{B_E} \wedge t(be_2) \in {}^{t_{sinc}}T_{B_E} \right) \tag{5.9}$$

Explicação:

Dois elementos de ligação be_1 e be_2 são equivalentes quando, para toda transição síncrona, t_{sinc}, presente no corpo de uma classe RPOO, k_i, e que contém invocação de método síncrono, a seguinte condição é satisfeita *ao menos uma vez*:

A transição, presente no elemento de ligação be_1, pertence ao bloco de sub-rede de Petri Colorida que substitui uma transição síncrona no receptor *e* a transição presente no elemento de ligação be_2 também pertence a esse mesmo bloco.

Lado Receptor:

As relações de equivalência para o lado receptor são semelhantes àquelas para o lado emissor, diferindo apenas pela substituição de cada aparição de T_E por T_R e B_E por B_R.

Por fim, vale salientar que as relações de equivalência, até então definidas para o lado emissor e receptor, não consideram o caso em que os arcos de entrada de uma transição síncrona contêm expressões condicionais. Mas, esses casos são resolvidos facilmente, bastando substituir o numerador $\sum_{\forall p' \in {}^{\bullet}t_{sinc}} |e(p, t_{sinc})|$ pela combinação do número de fichas presentes *em cada um* dos multiconjuntos da expressão condicional de *cada um* dos arcos de entrada.

5.2.3 Prova da Especificação de Equivalência Compatível

Nesta seção prova-se que as especificações de equivalência definidas na Seção 5.2.2 são compatíveis e que originam um grafo de ocorrência com mesmo tamanho do grafo de ocorrência completo que seria construído se não houvesse substituição de transição síncrona por um bloco.

Para provar que as relações de equivalência definidas na Seção 5.2.2 são compatíveis, é necessário provar que elas atendem à Definição 5.3. Ou seja,

$$M_1 \approx_M M_2 \Rightarrow [Next_\tau(M_1)] = [Next_\tau(M_2)]$$

Prova:

Dada uma marcação inicial, M_0, para um sistema RPOO, e duas marcações $M_1, M_2 \in [[M_0\rangle]$. Segundo as relações definidas nas Equações 5.8 e 5.9 (ver páginas 84 e 85), tem-se que se:

$$\begin{aligned} M_1 \approx_M M_2 \Rightarrow & M_1 = M_2 \ \vee \ M_1[\tau_1\rangle M_2[\tau_2 be\rangle M \ \vee \ M_2[\tau_1\rangle M_1[\tau_2 be\rangle M : \\ & \tau_1\tau_2 = \tau \ \wedge \ M_1 \not\approx_M M \not\approx_M M_2 \end{aligned} \tag{5.10}$$

Se $M_1 = M_2$ então tem-se diretamente que $[Next_\tau(M_1)] = [Next_\tau(M_2)]$. De forma semelhante, se $M_1[\tau_1\rangle M_2[\tau_2 be\rangle M \ \vee \ M_2[\tau_1\rangle M_1[\tau_2 be\rangle M$, ambas as marcações levarão a M e, portanto, $[Next_\tau(M_1)] = [Next_\tau(M_2)]$. Além disso, perceba que os elementos de ligação no τ entre a marcação M_1 (ou M_2) e M só podem envolver as transições t_{in}, t_2, para o lado emissor, e t_{in}, para o lado receptor. O elemento de ligação be envolve a transição t_{out}.

Então, é preciso provar que não existe um M_2, tal que $M_1 \approx_M M_2$ e M_2 *não seja* nenhuma das marcações obtidas pela ocorrência de elementos de ligação presentes em τ. A prova será por absurdo.

Suponha que existe essa marcação M_2. Se ela não corresponde a uma das marcações alcançáveis pela ocorrência de elemento(s) de ligação na sequência τ, então ela é alcançável pela ocorrência de um elemento de ligação que contém uma transição que não pertence a um bloco B_E ou B_R. Como os conjuntos de transições e lugares das classes e dos blocos são disjuntos (${}^kT \cap T_B = \varnothing$ e ${}^kP \cap P_B = \varnothing$), a ocorrência de uma transição

externa ao bloco remove e/ou deposita fichas nos lugares da classe e, portanto, não satisfaz a relação de equivalência para as marcações, pois (i) os lugares da classe k não terão mais os mesmos multiconjuntos de fichas e (ii) o número de fichas nos lugares de entrada do bloco , mais a razão entre os somatórios, multiplicado pelo número de fichas nos lugares do bloco, não será mais idêntico. Portanto, para que duas marcações sejam equivalentes é preciso que sejam idênticas, ou que uma delas seja alcançável pela ocorrência de um dos elementos de ligação presentes no τ da outra. □

Por fim, discute-se por que o grafo de ocorrência gerado a partir das relações especificadas tem mesmo tamanho do grafo de ocorrência completo que seria gerado antes do mapeamento.

Como a especificação de equivalência é compatível, um novo nó somente é gerado para o grafo quando $M_1 \not\approx_M M_2$. Observando as relações das Equações 5.8 e 5.9 (ver páginas 84 e 85), percebe-se que duas marcações serão equivalentes quando os lugares da classe tiverem multiconjuntos idênticos ou houver ocorrências de transições internas aos blocos. No segundo caso, as marcações serão consideradas equivalentes e, portanto, nenhum novo nó será gerado. Então, para atender ao primeiro requisito, M_1 deve ser idêntica a M_2, pois se não for, e não satisfizer o segundo caso, originará um novo nó. Logo, o grafo de ocorrência gerado a partir da especificação de equivalência terá mesmo tamanho do grafo de ocorrência completo que seria gerado anteriormente ao mapeamento, pois ambos possuem nós para as mesmas marcações.

Capítulo 6

Conclusão

Neste trabalho, discutiu-se a introdução do suporte aos mecanismos de interação para comunicação síncrona e sincronização de métodos na ferramenta conceitual Rede de Petri Orientada a Objetos (RPOO). O acréscimo desses mecanismos deu-se pela definição de abstrações baseadas na comunicação assíncrona do formalismo original de RPOO.

Para comunicação síncrona, definiram-se duas abstrações, uma para suporte à comunicação síncrona com reconhecimento e outra, para comunicação síncrona com resposta. Ambas utilizaram o conceito de transição síncrona.

Uma transição síncrona é uma transição com invocação(ões) ou ativação de método(s) síncrono(s). Por utilizar comunicação síncrona, sua ocorrência representa um conjunto de ações que acontecem. Transições síncronas só podem estar associadas a um tipo de comunicação síncrona por vez. Para diferenciar o tipo de comunicação entre síncrona com reconhecimento e com resposta, optou-se por colocar sempre as respectivas palavras-chave ACK ou $REPLY$ como primeiro argumento de dado dos métodos invocados. Todas as transições síncronas, presentes no corpo de uma classe RPOO, foram mapeadas em blocos de sub-rede de Petri Colorida, cujas transições estão associadas apenas a métodos assíncronos.

Para a sincronização de métodos, adotou-se uma linguagem textual para facilitar a descrição dos sincronizadores e a conseqüente transformação dessa descrição numa classe RPOO, a ser inserida no sistema. Optou-se por separar um sincronizador das classes ao qual se aplica, para que os métodos dessas ainda possam ser ativados por objetos de

outras classes, que não estejam submetidas a um sincronizador. Embora não tenha sido explicitado ao longo do texto, a formalização para sincronizadores não suporta *sobreposições*, ou seja, os "conjuntos de classes" dos sincronizadores constituem partições do conjunto de classes do sistema RPOO ao qual pertencem. Esta característica não significa uma deficiência da formalização porque a superposição de sincronizadores sempre equivale a um único sincronizador com todas as relações presentes nos sincronizadores sobrepostos [Frø96].

Por fim, definiram-se relações de equivalência compatíveis que permitem usar as abstrações para comunicação síncrona, sem incorrer em aumento no tamanho do grafo de ocorrência de um dado modelo RPOO. O objetivo dessas relações foi tornar equivalentes ocorrências de transições internas aos blocos, fazendo com que marcações oriundas de tais ocorrências fossem inobserváveis. Pôde-se definir tais relações porque os blocos de sub-rede de Petri Colorida, que substituem transições síncronas, constituem um invariante de lugar e, portanto, não alteram as propriedades comportamentais dos objetos nos quais são inseridos. Como não alteram as propriedades, seus estados podem ser ignorados.

6.1 Trabalhos Futuros

Mediante as conclusões expostas, sugerem-se os seguintes trabalhos futuros para melhorar o suporte a mecanismos de interação em RPOO:

- Estudar como a sobreposição de sincronizadores pode ser realizada em RPOO, tornando possível uma especificação modular dos sincronizadores;
- Implementar uma ferramenta para edição e análise de sistemas RPOO. Em especial, investigar e implementar a construção de grafos de ocorrência, baseados em especificações de equivalência compatíveis, para essa ferramenta.

Bibliografia

[AFK+93] G. Agha, S. Frølund, W. Kim, R. Panwar, A. Patterson, and D. Sturman. Abstraction and modularity mechanisms for concurrent computing. In G. Agha, P. Wegner, and A. Yonezawa, editors, *Research Directions in Concurrent Object-Oriented Programming*. MIT Press, Cambridge, MA, 1993.

[Agh86] G. Agha. *Actors: a Model of Concurrent Computation in Distributed Systems*. MIT Press, Cambridge, MA, 1986.

[AP98] V. S. Alagar and K. Periyasamy. *Specification of Software Systems*. Springer-Verlag, Berlin, 1998.

[BM93] Ö. Babaoğlu and K. Marzullo. Consistent global states of distributed systems: Fundamental concepts and mechanisms. In Sape Mullender, editor, *Distributed Systems*, pages 55–96. ACM Press, 1993.

[Boo94] G. Booch. *Object-Oriented Analysis and Design*. Object-Oriented Software Engineering. Benjamin/Cummings Publishing Company, Santa Clara, California, USA, 2^{nd} edition, 1994.

[CGdFP98] S.A.D. Costa, D.D.S. Guerrero, J.C.A. de Figueiredo, and A. Perkusich. Aspectos de herança em uma ferramenta de modelagem de sistemas baseada em redes de Petri. In *Anais do SBES'98, Simpósio Brasileiro de Engenharia de Software*, pages 297–312, Maringá, PR, October 1998.

[CW96] E. M. Clarke and J. M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys*, September 1996. Special Issue: ACM Strategic Directions in Computing Research.

[dMGdFP98] A.K.A. de Medeiros, D.D.S. Guerrero, J.C.A. de Figueiredo, and A. Perkusich. An object-oriented Petri-net modeling tool and abstraction mechanisms for cooperative systems. In *Proceedings of IEEE International Conference on Systems Man and Cybernetics*, pages 172–177, San Diego, USA, October 1998.

[Frø96] S. Frølund. *Coordinating Distributed Objects: An Actor-Based Approach to Synchronization*. MIT Press, Cambridge, Massachusetts, 1996.

[Gen87] H.J. Genrich. Predicate/Transition nets. In W. Brauer, W. Reisig, and G. Rozemberg, editors, *Petri Nets: Central Models and Their Properties*, volume 254 of *Lecture Notes in Computer Science*, pages 207–247. Springer-Verlag, 1987.

[Gue99] D. D. S. Guerrero. Especificação e análise de sistemas concorrentes utilizando redes de Petri e orientação a objetos. Relatório técnico, Coordenação de Pós-graduação em Engenharia Elétrica - COPELE/UFPB, Campina Grande, PB, February 1999.

[Jen92] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis, Methods and Practical Use*, volume 1 of *EACTS – Monographs on Theoretical Computer Science*. Springer-Verlag, 1992.

[Jen95] K. Jensen. *Coloured Petri Nets: Basic Concepts, Analysis, Methods and Practical Use*, volume 2 of *EACTS – Monographs on Theoretical Computer Science*. Springer-Verlag, 1995.

[MBC+95] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceshinis. *Modeling with Generalized Stochastic Petri Nets*. John Wiley and Sons, 1995.

[Mul93] S. Mullender, editor. *Distributed Systems*. ACM Press, 1993.

[Mur89] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen. *Object-Oriented Modelling and Design.* Prentice-Hall, Englewood Cliffs, New Jersey, 1991.

[Thi86] P. S. Thiagarajan. Elementary net systems. In W. Brauer, editor, *Petri nets: central models and their properties, Advances in Petri nets, Proceedings of an advanced course, Bad Honnef, 8.-19. Sept. 1986, Vol. 1*, number 254 in Lecture Notes in Computer Science, pages 26–59, Berlin-Heidelberg-New York, 1986. Springer.