UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE CIÊNCIAS E TECNOLOGIA

UMA ABORDAGEM PARA O DESIGN DE SISTEMAS DISTRIBUÍDOS E PROTOCOLOS DE COMUNICAÇÃO

BERNARDO GONÇALVES RISO

CAMPINA GRANDE, NOVEMBRO 1991

Uma abordagem

para o design de sistemas distribuídos

e protocolos de comunicação

R595a Riso, Bernardo Goncalves
Uma abordagem para o design de sistemas distribuidos e protocolos de comunicacao / Bernardo Goncalves Riso. - Campina Grande, 1991.
226 f.

Tese (Doutorado em Engenharia Eletrica) - Universidade Federal da Paraiba, Centro de Ciencias e Tecnologia.

1. Sistemas Distribuidos 2. Protocolos de Comunicacao 3. Tese I. Souza, Wanderley Lopes de, Dr. II. Universidade Federal da Paraiba - Campina Grande (PB) III. Título

CDU 004.75(043)

UMA ABORDAGEM PARA O DESIGN DE SISTEMAS DISTRIBUIDOS
E PROTOCOLOS DE COMUNICACAO

BERNARDO GONÇALVES RISO

TESE APROVADA EM 13.10.91

WANDERLEY LOPES DE SOUZA, Dr.Ing., UFPB
Orientador

JOSE ANTAO BELTRAO MOURA, Ph.D., UFPB
Componente da Banca

JACQUES PHILLIPE SAUVE, Ph.D., UFPB
Componente da Banca

MAURICIO FERREIRA MAGALHAES, Dr.Ing., UNICAMP
Componente da Banca

JOSE MARCOS SILVA NOGUEIRA, Dr., UFMG
Componente da Banca

ALOYSIO DE CASTRO PINTO PEDROZO, Dr., UFRJ
Componente da Banca

CAMPINA GRANDE - PB
NOVEMBRO - 1991

Para

Marli, Alexandre,

Fernanda e Vanessa

Agradecimentos

Inicialmente desejo agradecer ao professor Wanderley Lopes de Souza, meu orientador, pelas críticas e sugestões apresentadas durante a realização deste trabalho. Sem a sua orientação, este trabalho simplesmente não seria possível.

Desejo agradecer a todos os professores e funcionários do Departamento de Sistemas e Computação da UFPB-2 pelo tratamento carinhoso que me dispensaram enquanto estivemos juntos. Especialmente, agradeço aos professores José Antão Beltrão Moura (meu orientador acadêmico) e Jacques Phillipe Sauvé, pelo apoio e por me permitirem usar a sua sala durante a realização dos meus estudos em Campina Grande.

Desejo agradecer aos professores Bráulio Maia Junior e Miriam Costa, do Departamento de Matemática e Estatística da UFPB-2, pelo tempo que me concederam para a discussão sobre questões de álgebra.

Desejo agradecer à sra. Odete Emídio de Farias, coordenadora da Biblioteca Setorial da UFPB-2, e a todos os funcionários dessa biblioteca, pela valiosa ajuda que me prestaram para a localização e a aquisição de bibliografia.

Desejo agradecer a todos os meus colegas de pós-graduação da UFPB-2, pela alegria e pelo companheirismo de um excelente relacionamento.

Desejo agradecer à srta. Isolene Bernadete Hoffmann, pelo trabalho cuidadoso e paciente de digitação.

Acima de tudo desejo agradecer aos professores e funcionários do Departamento de Ciências Estatísticas e da Computação da UFSC pelo respeito, pela confiança e pela estima que vêm me dedicando desde a minha admissão pela UFSC.

BERNARDO GONÇALVES RISO

Uma abordagem para o design de sistemas distribuídos e protocolos de comunicação

Tese apresentada ao curso de Doutorado em Ciências no domínio da Engenharia Elétrica, vinculado ao Centro de Ciências e Tecnologia da Universidade Federal da Paraíba, como parte dos requisitos necessários à obtenção do grau de doutor em ciências.

Área de concentração: Processamento da informação

Wanderley Lopes de Souza

Orientador

UFPB

Campina Grande - PB

(13/11/91)

SUMÁRIO

LISTA DE FIGURAS

CAPÍTULO II

CAPÍTULO IV

CAPÍTULO V

CAPÍTULO VI

CAPÍTULO VII

CAPÍTULO VIII

CAPÍTULO IX

LISTA DE TABELAS

LISTA DE ABREVIAÇÕES

ABNT - Associação Brasileira de Normas Técnicas

ACCS - Asynchronous CCS

BF - Bissimulação Fraca

CCITT - Comité Consultatif International Télégraphique et Téléphonique

CCS - Calculus of Communicating Systems

Estelle - Extended State Transition Language

FTAM - File Transfer, Access and Management

ISO - International Organization for Standardization

LE - Lei de Expansão

LOTOS - Language of Temporal Ordering Specification

MEF - Máquina de Estados Finita

MEFE - MEF estendida

RM OSI - Basic Reference Model for Open Systems Interconnection

SDL - Specification and Description Language

STR - Sistema de Transições Rotuladas

TDF - Técnica de Descrição Formal

UDP - Unidade de Dados de Protocolo

RESUMO

Neste trabalho propõe-se uma abordagem, baseada em estilos de especificação e utilizando a álgebra "Calculus of Communicating Systems (CCS)", para o design de sistemas distribuídos e protocolos de comunicação. Inicialmente, o poder de expressão e o poder de análise da álgebra CCS são empregados em uma abordagem clássica, baseada no desenvolvimento passo-a-passo de sistemas. Nesse contexto é proposta uma metodologia, que parte da descrição abstrata do sistema que se quer desenvolver (pode ser a especificação de serviço), reduz gradativamente essa abstração através da realização de refinamentos sucessivos (com provas de correção a posteriori), até a obtenção da especificação final em termos de agentes básicos predefinidos (útil à tarefa de implementação). Tal metodologia é ilustrada com o desenvolvimento de um sistema de acesso por exclusão mútua. Em seguida estuda-se a utilização de CCS em uma abordagem transformacional, voltada para a automatização do desenvolvimento de sistemas. Nesse contexto é proposta uma metodologia, que emprega regras de transformação, para a obtenção das entidades de protocolo correspondentes a uma especificação de serviço. Tal metodologia é ilustrada com o desenvolvimento de uma parte do núcleo do serviço e do protocolo "File Transfer, Access and Management (FTAM)" da "International Organization for Standardization (ISO)". Finalmente, após um estudo sobre estilos de especificação, é proposta uma nova abordagem, com características intermediárias da abordagem clássica e da abordagem transformacional. Nesse novo contexto é proposta uma metodologia, que parte da especificação do serviço desejado e da especificação do serviço disponível (ambas realizadas no estilo orientado para restrições), e obtém a especificação das entidades de protocolo correspondentes (também no estilo orientado para restrições). Tal metodologia é ilustrada com o desenvolvimento de um sistema de comunicação que envolve o protocolo do bit alternante.

ABSTRACT

In this work we propose an approach based on specification styles and using the algebra Calculus of Communicating Systems (CCS) to be employed in the design of distributed systems and communication protocols. Initially, the expressive and analytical power of CCS is employed in a classical approach, based on step-by-step systems development. In this context a methodology is proposed which starts from an abstract system description (it may be a service specification) and gradually reduces its abstraction through successive refinements (with proofs of correctness a posteriori), until the final specification described with predefined basic agents (useful for implementation). An illustrative application for this methodology with the development of a mutual access system is given. We then study the use of CCS in the context of the transformational approach geared to the automatic development of systems. In this context a methodology is proposed which uses transformation rules for obtaining the protocol entities relative to a service specification. An illustrative application of this metholology involving part of the FTAM kernel (service and protocol) is given. Finally, following a study of specification styles, a new approach is proposed, which is intermediate between the classical and transformational approaches. In this new context, a methodology is proposed which starts from the intended service specification and from the available service specification (both carried out using the restriction oriented style) and obtains the corresponding protocol entities specification (also carried out using the restriction oriented style). An illustrative application of this methodology is given that involves a communicating system using the alternating bit protocol.

CAPÍTULO I

INTRODUÇÃO

A atividade de design (concepção e descrição) de sistemas complexos costuma apresentar consideráveis dificuldades. Um modo de enfrentar tais dificuldades consiste no emprego da técnica divida-e-conquiste que considera, isoladamente, diversos aspectos de um sistema. Cada aspecto isolado possui menos complexidade do que o sistema todo e, por isso, pode ser abordado mais confortavelmente.

Entretanto, para que a técnica divida-e-conquiste seja utilizada, é necessário que antes sejam identificados os aspectos do sistema que possuem um grau satisfatório de independência. Naturalmente, essa identificação surge como resultado de cuidadosa atividade de análise.

Examinando-se com essa intenção, por exemplo, os protocolos do "Basic Reference Model for Open Systems Interconnection (RM OSI)" da "International Organization for Standardization (ISO)" [ISO 83], percebe-se que dois aspectos gerais podem ser considerados: os aspectos estáticos, ligados à representação dos dados, e os aspectos dinâmicos, relativos ao comportamento dos processos envolvidos. Cada um desses aspectos contribui com uma parcela de complexidade para o projeto de protocolos. Assim, pode ser vantajoso considerá-los isoladamente, quando isso se torna possível.

A representação dos dados parece simplificar-se à medida que se caminha da camada de aplicação, onde as estruturas de dados são muito complexas por se situarem mais próximas da comunicação humana direta, em direção à camada física (através, basicamente, de envelopamentos sucessivos de mensagens). No meio de transmissão, a simplificação é máxima, uma vez que os dados são representados por sequências de bits.

Dá-se o contrário, aparentemente, em relação aos aspectos de controle. Estes seriam mais simples para os protocolos de alto nível (que podem supor resolvidos vários tipos de problemas, passíveis de ocorrer com as comunicações, tais como: perdas, danificações, duplicações, atrasos, dificuldades de roteamento etc.) e mais complexos nas camadas inferiores (onde esses problemas precisam ser, efetivamente, resolvidos).

Entre os dois aspectos existe independência conceitual. Além disso, a complexidade de cada um deles evolui com uma certa autonomia, à medida que são examinadas as diversas camadas de protocolos. Pode-se observar ainda que as técnicas empregadas para a representação dos aspectos de dados geralmente diferem das técnicas empregadas para a representação dos aspectos de controle.

As considerações acima fazem supor que, em grande número de casos, a abordagem de cada um desses aspectos pode ser realizada de modo isolado ao longo de vários estágios de desenvolvimento. Nesses casos, somente seria imperiosa a sua reunião em fases avançadas, por exemplo, quando ambos estivessem formalizados e detalhados em grau bastante acentuado.

Em particular, para a camada de aplicação, mostra-se em [BoGo 86], como os processos que exercem o controle da comunicação podem ser descritos isoladamente dos dados. (Veja-se também [RiLo 90].) Pode-se conjecturar que essa possibilidade de separação também exista (embora, talvez, em menor grau) para outros protocolos de alto nível.

Nos protocolos dos níveis inferiores, do RM OSI/ISO, os aspectos de controle assumem grande relevância. Para esses protocolos, o tratamento simultâneo dos aspectos de dados com os aspectos de controle, quando inevitável, poderia ser realizado com uma preocupação menor no que se refere aos dados, ao menos nas fases preliminares de desenvolvimento. Inicialmente os dados seriam tratados de modo informal. Posteriormente seriam integrados à especificação dinâmica.

Ambos os aspectos de protocolos requerem disciplina para o seu desenvolvimento. Esta é necessária para que se possa estabelecer uma base

sobre a qual os futuros usuários do sistema de comunicação, assim como os técnicos que o desenvolvem e os supervisores do projeto possam dialogar de modo proveitoso, certos de que as exigências de todos serão, ao final, atendidas.

Em especial, para a equipe técnica, a sistematização do desenvolvimento facilita a divisão de tarefas entre diferentes grupos, sem perda de coerência, permitindo que o produto de um grupo seja recebido por outro grupo para uma maior elaboração. Além disso, uma abordagem sistemática (ou metodologia) favorece a automatização de tarefas [HoLo 85].

Para o projeto dos aspectos dinâmicos dos protocolos, podem-se distinguir dois tipos de abordagens: o tipo clássico, baseado em refinamentos progressivos da descrição de um sistema e que utiliza provas de correção após cada refinamento; o tipo transformacional, voltado para a intensa exploração de ferramentas automáticas e que se baseia em regras de transformação cuja aplicação leva a resultados sempre corretos nos seus domínios de utilização.

1.1 Contribuições

Neste trabalho faz-se uma investigação das possibilidades de sistematização do uso de Técnicas de Descrição Formal (TDFs) algébricas para a especificação dos aspectos de controle de sistemas distribuídos. Essa investigação concentra-se no poder de expressão e no poder de análise de "Calculus of Communicating Systems (CCS)" [Miln 80,89], mas alguns dos resultados obtidos são extensíveis a outras TDFs da mesma natureza.

As questões fundamentais dessa investigação incluem o problema da estruturação, dos estilos e das provas de equivalência, que podem ser definidas para as especificações. Como resultado dessa investigação propõe-se uma abordagem que busca combinar as vantagens das abordagens clássica e transformacional em diversas etapas do desenvolvimento de protocolos.

Inicialmente, no contexto de uma abordagem clássica, CCS completo

(isto é, CCS com passagem de valores entre os agentes comunicantes) é empregado para a especificação de protocolos. Nessa abordagem adota-se a relação de equivalência de observação para a prova de correção dos refinamentos.

Nos limites da abordagem clássica, contribui-se com uma proposta para a utilização de agentes básicos predefinidos no desenvolvimento de sistemas. Tais agentes, que podem ser mapeados para implementações em software, hardware ou firmware, visam facilitar a tarefa de implementação. O emprego desses agentes básicos predefinidos é ilustrado através da especificação e da verificação de um protocolo de exclusão mútua.

A abordagem clássica apóia-se na criatividade do especificador, deixando-lhe uma larga margem de liberdade nas decisões de design. Esse fato resulta em especificações personalizadas, dificultando a sistematização (e, portanto, a automatização) do processo de desenvolvimento de sistemas.

Para investigar as possibilidades de automatização do processo de desenvolvimento de sistemas, fez-se um estudo da abordagem transformacional.

Nos limites da abordagem transformacional, contribui-se com um conjunto de regras de transformação que podem ser aplicadas a especificações realizadas em CCS Básico (isto é, CCS sem passagem de valores). Tais regras são apresentadas na forma de algoritmos que permitem a derivação de especificações de protocolo a partir de especificações de serviço simples. A utilização desses algoritmos é ilustrada através da especificação de uma parte do núcleo do serviço e do protocolo FTAM da ISO.

A abordagem transformacional tem uma aplicabilidade limitada aos sistemas simples. Outra dificuldade com esse tipo de abordagem refere-se à estruturação das especificações.

Para investigar as questões relacionadas à sistematização do processo de desenvolvimento de sistemas complexos, fez-se um estudo do poder expressivo de estilos de especificação.

Diferentes estilos de especificação são realizáveis em CCS Básico. Em

especial, o estilo orientado para restrições pode ser realizado, utilizando-se o operador de conjunção (&) para expressar a modalidade de sincronização denominada "multi-way synchronization" que caracteriza tal estilo.

Como uma contribuição, o conceito de produto estendido, apresentado em [Miln 86] para expressar "three-way synchronization" entre agentes (no caso do compartilhamento de portas receptoras), é ainda mais estendido. Tal extensão permite expressar "multi-way synchronization", onde vários agentes compartilham eventos (em portas receptoras e/ou oferecedoras de eventos). Uma nova forma para a Lei de Expansão, adequada ao operador &, é então estabelecida.

O operador & pode ser descrito a partir dos operadores de composição (|), restrição (\) e rerrotulação (\). Assim, ele aumenta o poder expressivo de CCS preservando-lhe a simplicidade.

No campo das aplicações dessas extensões, são apresentadas especificações não triviais de serviços e de entidades de protocolo, com o emprego do estilo orientado para restrições.

A abordagem proposta neste trabalho constitui uma outra contribuição. Ela sugere que os quatro estilos básicos para a especificação de sistemas distribuídos (estilos monolítico, orientado para restrições, orientado para recursos e orientado para estados) definidos em [ViSc 88], [ViSc 89] e [ISO 88] sejam utilizados ao longo do desenvolvimento de sistemas, de modo a extrair-lhes o máximo de suas potencialidades expressivas. Mostra-se, então, como a transformação entre estilos pode se dar, automaticamente, em alguns casos.

Na abordagem proposta, os estilos orientado para restrições e orientado para recursos são utilizados, alternadamente, até a especificação final. Esta é constituída de agentes básicos predefinidos, que podem ser descritos no estilo monolítico ou no estilo orientado para estados, visando facilitar a tarefa de implementação. O emprego da abordagem proposta é ilustrada através da especificação e da verificação de um sistema de

comunicação que utiliza o protocolo do bit alternante.

As restrições impostas ao comportamento das entidades de protocolo podem ser obtidas como o resultado de transformações realizadas sobre as restrições que atuam na especificação do serviço disponível e na especificação do serviço pretendido. Para realizar essas transformações, são definidas e utilizadas as operações de complementação de agentes e de projeção de agentes sobre conjuntos de portas. Além dessas operações, tais transformações envolvem a intercalação de restrições.

Uma vez que a estruturação das especificações é uma questão fundamental quando se trata do desenvolvimento de sistemas complexos, a abordagem preocupa-se em produzir especificações estruturadas sob três pontos de vista: (a) horizontalmente, em camadas, à semelhança do modelo OSI/ISO [ISO, 83], (b) verticalmente, com a utilização dos estilos orientado para restrições e orientado para recursos, e (c) hierarquicamente, à medida que são realizados os refinamentos.

1.2 Conteúdo dos capítulos

No capítulo 2 é situado o problema do design de sistemas distribuídos, com ênfase nas dificuldades de desenvolvimento dos protocolos de comunicação utilizados nas redes de computadores. Esse capítulo apresenta ainda o ciclo de vida de um protocolo, caracterizando cada uma de suas etapas.

No capítulo 3 são classificadas, e apresentadas sucintamente, algumas importantes técnicas de descrição formal (TDFs). A amostra é representativa do conjunto das TFDs que vêm sendo propostas para o desenvolvimento de sistemas distribuídos.

No capítulo 4 a TDF "Calculus of Communicating Systems (CCS)", utilizada pelo autor deste trabalho como base para a realização das suas contribuições, é apresentada em detalhes. Nessa apresentação o poder de expressão e o poder de análise de CCS ficam evidenciados.

No capítulo 5 são apresentadas as características da abordagem clássica, normalmente utilizada para o design de sistemas distribuídos. Seguindo tais características, é proposta uma metodologia para a especificação e a validação de protocolos. Essa metodologia é ilustrada através do desenvolvimento de um sistema de acesso por exclusão mútua. No final do capítulo a abordagem clássica é avaliada.

No capítulo 6 são apresentadas as características da abordagem transformacional, que visa imprimir um alto grau de automatização ao design de sistemas distribuídos. Seguindo tais características, é proposta uma metodologia para a especificação e a validação de protocolos. Essa metodologia é ilustrada através do desenvolvimento de uma parte do núcleo do serviço e do protocolo FTAM da ISO. No final do capítulo a abordagem transformacional é avaliada.

No capítulo 7 são realizadas extensões em CCS que lhe permitem expressar "multi-way synchronization" entre agentes. Como consequência dessas extensões, são desenvolvidas novas formas para a Lei de Expansão.

No capítulo 8 é realizada uma discussão sobre estilos de especificação. Os quatro estilos básicos (monolítico, orientado para restrições, orientado para recursos e orientado para estados) são representados em CCS. Em particular, o estilo orientado para restrições torna-se possível em CCS, graças às extensões propostas no capítulo 7.

No capítulo 9 é proposta uma nova abordagem para a especificação e a validação de protocolos. Essa abordagem foi desenvolvida a partir das características da abordagem clássica, mas apresenta indicativos de evolução para as características transformacionais. A abordagem proposta é ilustrada através da especificação e da validação do protocolo do bit alternante. No final do capítulo a abordagem proposta é avaliada.

No capítulo 10 são apresentadas as conclusões do autor. São apresentadas ainda sugestões de pesquisa e de desenvolvimento, em prosseguimento ao trabalho realizado nesta trabalho.

CAPÍTULO II

SISTEMAS DISTRIBUÍDOS E PROTOCOLOS DE COMUNICAÇÃO

Há muita polêmica em relação à definição dos sistemas distribuídos. Considerando-se um sistema computacional distribuído, essa distribuição pode se dar com relação ao acesso ao computador, com relação aos computadores e com relação ao processamento (base de dados e/ou algoritmo).

Qualquer que seja a distribuição, os sistemas distribuídos são frequentemente complexos. Esse fato decorre da heterogeneidade dos seus componentes assim como do próprio caráter repartido desses sistemas (Lope 89].

As redes de computadores são exemplos de sistemas distribuídos, onde a distribuição ocorre principalmente em relação aos computadores. Elas constituem sistemas de comunicação onde a interação das entidades comunicantes se dá através da troca de mensagens. Ao conjunto das regras, que garantem o intercâmbio ordenado dessas mensagens, denomina-se protocolo de comunicação ou simplesmente protocolo.

Para facilitar a compreensão dos problemas envolvidos na definição de protocolos e para estabelecer padrões, especialistas da área propuseram uma arquitetura com sete camadas hierarquicamente organizadas, denominada Modelo Básico de Referência para a Interconexão de Sistemas Abertos (RM OSI), norma da International Organization for Standardization (ISO) [ISO 83] e do Comité Consultatif International Télégraphique et Téléphonique (CCITT) [CCIT 84] (Fig. 2.1).

Em relação ao RM OSI dois conceitos são fundamentais: serviço e protocolo (Fig. 2.2). Utilizando os serviços oferecidos pela camada (N-1) as entidades da camada (N) cooperam entre si, de acordo com o protocolo (N), para fornecer um serviço (N) com mais recursos à camada (N + 1).

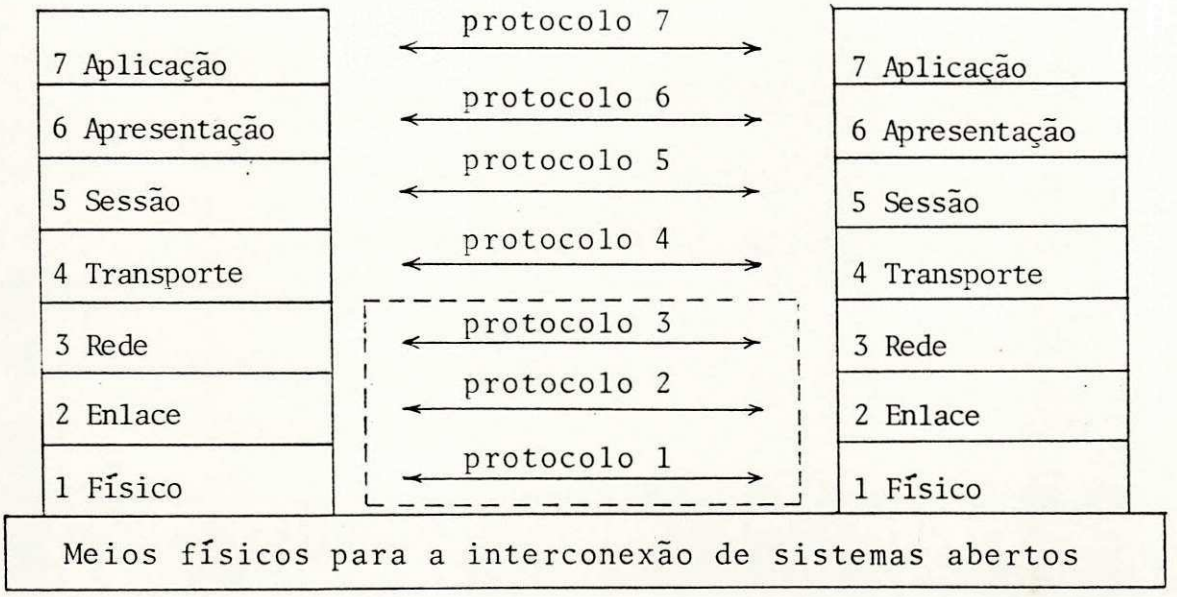


Figura 2.1 - Organização do RM OSI.

Figura 2.2 - Noções de serviço e protocolo.

O conceito de serviço oferecido pela camada (N) ignora as camadas subjacentes. A sua especificação é realizada através da descrição do comportamento observável apresentado por uma caixa preta, sujeita às trocas de primitivas de serviço com as entidades de protocolo da camada superior. Essas trocas são realizadas através dos pontos de acesso ao serviço (N) (PAS_N).

A especificação do protocolo (N) corresponde à descrição do comportamento das entidades que o executam. Essas entidades se comunicam, se sincronizam e operam de forma concorrente, utilizando os pontos de acesso ao serviço (N-1) [Boch 80].

A aplicabilidade dos conceitos de serviço e protocolo não está restrita ao contexto do RM OSI. Tampouco a aplicabilidade da estruturação em camadas. Essas noções são úteis ao desenvolvimento de sistemas distribuídos em geral.

Normalmente para o desenvolvimento de sistemas são propostos procedimentos que compreendem atividades de especificação, validação, implementação e teste.

As atividades de especificação têm como finalidade a descrição de um sistema de modo claro, conciso, sem ambiguidades e sem detalhes desnecessários. Inicialmente elas podem ser realizadas com o emprego de técnicas informais (baseadas em linguagens naturais) ou semi-formais (igualmente baseadas em linguagens naturais, mas com o acréscimo de figuras, tabelas de estados etc.). Devido às ambiguidades inerentes às linguagens naturais, especificações realizadas com tais técnicas podem gerar diferentes interpretações, comprometendo todo o ciclo de desenvolvimento de um sistema.

Como um meio de evitar a proliferação de erros de design e/ou de especificação são usadas técnicas de descrição formal (TDFs). O emprego adequado dessas técnicas leva à obtenção das especificações formais.

As atividades de validação procuram aumentar o grau de confiabilidade dos sistemas aos quais são aplicadas. Elas podem ser agrupadas segundo duas filosofias: verificação e teste.

A verificação utiliza algum tipo de raciocínio lógico para provar que um sistema possui (ou não) certas propriedades. Por exemplo, pode-se verificar a ausência de impasses em uma especificação, a equivalência entre duas especificações de um mesmo sistema (realizadas em diferentes níveis de abstração) etc.

O teste, frequentemente aplicado às implementações, busca provar a conformidade de um sistema em relação à sua especificação. Tal prova é realizada através da execução controlada desse sistema no seu ambiente e através da observação do seu comportamento. A mesma filosofia pode ser aplicada às especificações, desde que estas possam ser executadas de alguma forma.

O objetivo final do desenvolvimento de um sistema é a obtenção de uma implementação (software, hardware ou firmware), que é uma "instância" executável do sistema. Essa implementação pode ser obtida manualmente ou (semi-) automaticamente. Neste último caso é necessário que a especificação (ou parte dela) seja formalizada.

À atividade de desenvolvimento (reunindo especificação, validação, implementação e teste) de um sistema, dá-se o nome de ciclo de vida do sistema. No caso de protocolos, a reunião dessas atividades pode ser esquematizada de acordo com a Fig. 2.3.

Durante o desenvolvimento de sistemas, a utilização de TDFs permite a redução de ambiguidades e o controle dos erros. Outros benefícios, como o emprego de formalismos para a validação das especificações e do design, podem ser alcançados. Sobretudo as TDFs constituem uma base para a criação e a utilização de ferramentas automáticas, destinadas à realização de especificações, validações, implementações e testes.

O ciclo de vida de um protocolo pode ser adaptado ao desenvolvimento de outros tipos de sistemas distribuídos, sendo que as especificações do serviço e do protocolo corresponderiam, respectivamente, a uma especificação abstrata e outra refinada (com mais detalhes). As mesmas TDFs, que são

utilizadas para o desenvolvimento de protocolos, podem também ser empregadas para o desenvolvimento de outros tipos de sistemas distribuídos.

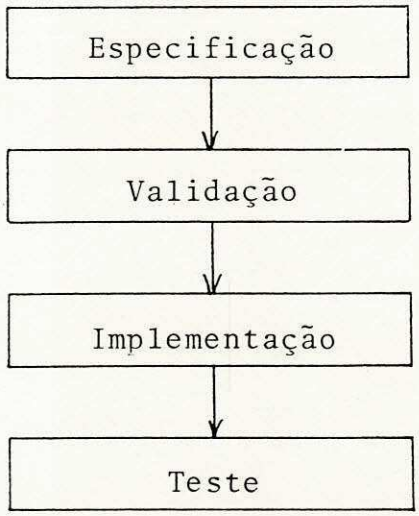


Figura 2.3 - Ciclo de vida de um protocolo.

CAPÍTULO III

TÉCNICAS DE DESCRIÇÃO FORMAL (TDFs)

Para que uma especificação formal atinja os seus objetivos é necessário que a TDF utilizada possua expressividade, poder de análise e abstração, características que nem sempre são facilmente conciliáveis.

A expressividade de uma TDF diz respeito à sua capacidade de descrever, com clareza e elegância, os aspectos relevantes dos sistemas aos quais é aplicada. No que se refere aos protocolos esses aspectos incluem comunicação, sincronização e concorrência.

A análise de um sistema é realizada quando se deseja investigar se esse sistema atende a determinados requisitos. Quando aplicada a protocolos, uma TDF deve fornecer recursos analíticos para validar o design, as especificações e as implementações.

Uma TDF possui um grau satisfatório de abstração, se além de não obrigar o especificador a incluir detalhes desnecessários (na fase de especificação), não restringe a liberdade do implementador (na fase de implementação).

Na década de 70 e no início dos anos 80 várias TDFs foram propostas para a especificação e a validação de protocolos. Para facilitar a avaliação das características apresentadas por essas diversas TDFs, elas costumam ser classificadas em três grandes categorias [Lope 89]:

(a) TDFs baseadas em modelos de transição;

(b) TDFs baseadas em linguagem de programação;

(c) TDFs híbridas.

3.1. TDFs baseadas em modelos de transição

Normalmente um protocolo pode ser considerado sob dois pontos de vista: controle e dados. O primeiro está relacionado com o estabelecimento e o encerramento de conexões, o segundo trata da transferência de dados propriamente dita.

As técnicas baseadas em modelos de transição foram motivadas pela observação de que entidades de protocolos podem ser descritas através de autômatos. Essas técnicas mostram-se eficientes para a especificação dos aspectos de controle dos protocolos, já que o espaço de estados necessário para a descrição do estabelecimento e do encerramento de conexões é pequeno. O mesmo não ocorre na fase de transferência de dados, onde a descrição de um simples parâmetro, indicando o sequenciamento das mensagens, pode provocar um fenômeno conhecido por explosão de estados.

As técnicas baseadas em modelos de transição podem ser subdivididas em:

(a) modelos baseados em autômatos puros;

(b) modelos gráficos;

(c) modelos baseados em gramáticas formais.

Na primeira categoria podem ser incluídas as Máquinas de Estados Finitas Puras (MEFs Puras) [BaSc 69], Colóquios [LeMo 73], Matriz Diálogo [Zafi 78], Diagrama de Fases [West 78] e Perturbação [ZaWe 80].

Se o espaço de estados do sistema a ser especificado é pequeno, as MEFs Puras são viáveis e permitem a análise da especificação visando detectar estados inalcançáveis, situações de impasse, laços indesejáveis etc.. Entretanto, à medida que o número de estados cresce, aumenta também a dificuldade de análise, levando à explosão de estados. Extensões a essa técnica tornam-se então necessárias a fim de aumentar o seu poder de expressão

e para, durante a análise das especificações, controlar esse fenômeno.

A teoria dos Colóquios considera a existência de interlocutores posicionados nas extremidades de um meio de comunicação. Assim, a especificação de um protocolo pode ser obtida descrevendo-se formalmente o comportamento de cada interlocutor. Em função dos tipos de mensagens que podem ser enviadas, cada interlocutor possui uma matriz de protocolo que define o seu comportamento. A introdução dessa matriz na teoria dos Colóquios representa um esforço no sentido da algebrização das especificações de protocolos. Essa tendência também pode ser observada em outras TDFs.

Na técnica Matriz Diálogo as entidades de protocolo são especificadas através de um par de grafos de ação. A partir dessas especificações, "unilogues" (caminhos que partem do estado inicial e a ele retornam) são obtidos para a construção da matriz diálogo (onde cada elemento é constituído de um par de "unilogues"). A partir da matriz diálogo é obtida uma matriz de validação, onde determinadas propriedades a respeito dos diálogos (bem comportados, não-ocorrentes e errôneos) podem ser verificadas. Essa é uma técnica importante pois auxilia o design de um protocolo, na medida em que a detecção de erros através da matriz de validação permite a correção dos diálogos e, consequentemente, das especificações.

O Diagrama de Fases é uma extensão gráfica à técnica Matriz Diálogo. Quando dois processos operam de modo assíncrono, um diálogo pode ser executado de diferentes maneiras, resultando em dificuldades de visualização. O diagrama de fases foi introduzido justamente para se obter uma compreensão melhor da dinâmica da comunicação entre as entidades.

Perturbação é uma técnica baseada na análise de alcançabilidade que permite detectar situações de impasse, recepções não-especificadas, interações não-executáveis etc.. Nesta técnica, cada estado global do sistema (composto de entidades que se comunicam através de filas) é representado por meio de uma matriz. Os elementos dessa matriz são os estados das entidades envolvidas e os conteúdos dos canais de comunicação utilizados pelas entidades. Uma árvore de

alcançabilidade pode então ser construída, com as diversas matrizes do sistema, para representar a execução do protocolo. Os erros de design e os erros de especificação do protocolo são identificados através da análise dessa árvore.

Na segunda categoria de TDFs baseadas em modelos de transições, as redes de Petri representam o modelo gráfico principal [Dant 80].

Uma rede de Petri pode ser definida como uma quádrupla <L,T,E,S>, onde L é um conjunto finito de lugares, T um conjunto de transições, E a função de entrada e S a função de saída. Nas representações gráficas das redes de Petri são usadas barras, nós e arcos direcionados. Uma barra corresponde a um evento, um nó corresponde a uma condição, um arco direcionado, conectando um nó a uma barra, corresponde a uma condição de entrada para a ocorrência de um evento e um arco direcionado, conectando uma barra a um nó, corresponde a uma condição de saída após a ocorrência de um evento.

Um estado da rede é representado por uma distribuição de fichas através dos nós da rede. Para que um evento possa ocorrer é necessário que todas as suas condições de entrada sejam satisfeitas (o que implica na presença de pelo menos uma ficha em cada nó de entrada). A ocorrência de um evento consiste na remoção de uma ficha (em cada nó de entrada) e na adição de uma ficha (em cada nó de saída).

As redes de Petri expressam bem a simultaneidade de condições e de eventos e o compartilhamento do uso de recursos (por exemplo, os meios de transmissão). Entretanto, à medida que aumenta a complexidade do sistema, aumenta a quantidade de lugares e transições da rede, acontecendo algo semelhante à explosão de estados dos autômatos.

Na terceira categoria de TDFs baseadas em modelos de transições, tem-se um conjunto de técnicas que utilizam gramáticas formais. Essas técnicas foram motivadas pela correspondência entre gramáticas formais e máquinas de estados: o símbolo inicial da grámatica formal corresponde ao estado inicial da máquina de estados, os símbolos não-terminais aos estados, os símbolos

terminais ao conjunto de interações (entradas e/ou saídas) e as regras de produção às transições.

Por outro lado, com suas regras de produção uma gramática formal gera um conjunto de sentenças e com suas regras de comunicação um protocolo define um conjunto de ações válidas. Essas evidências sugerem a utilização de gramáticas para a elaboração de modelos para a especificação de protocolos.

Em [TeLi 78] é proposto um modelo para representar as sequências de ações de um protocolo (através de uma Gramática de Ações) e as estruturas das mensagens trocadas entre as entidades comunicantes (através de uma Gramática de Mensagens). Nessa proposta pode-se observar como ambos os aspectos dos protocolos (controle e dados) são tratados pelo mesmo modelo e como, apesar disso, esses aspectos são abordados com uma certa independência.

As gramáticas formais podem ser utilizadas também como modelos para a especificação de serviços de comunicação, visando a síntese de protocolos [BoGo 86].

3.2. TDFs baseadas em linguagens de programação

As linguagens de programação de alto nível são adequadas para a especificação de protocolos graças aos seus recursos de modularidade, representação da concorrência entre processos e representação dos tipos de dados [Sten 76]. A grande vantagem desse tipo de modelo, em relação aos modelos de transição, é que não está sujeito à explosão de estados (o número de sequenciamento das mensagens de dados pode ser representado por uma simples variável).

A linguagem Pascal [JeWi 75] foi uma das primeiras linguagens de programação propostas como TDF para a especificação de protocolos [Boch 75]. Outras linguagens de programação como LISP [McAb 65], PROLOG [Colm 85] e ADA [USDD 80], foram propostas com a mesma finalidade [Lope 88].

Normalmente a realização de especificações com o uso de uma linguagem

de programação obriga à inclusão de detalhes de implementação. Esse fato reduz a abstração da especificação. Outro problema diz respeito à validação das especificações, que nesse modelo costuma basear-se na prova de asserções. Estas dependem fortemente da criatividade do analista, limitando o emprego de recursos automáticos.

Ainda dentro dessa categoria, técnicas que empregam notações mais abstratas que as linguagens de programação têm sido propostas para a especificação de protocolos. Entre elas destacam-se Lógica Temporal [Lamp 80] [ScMe 81] e Expressões Sequenciais [Schi 80].

3.3. TDFs híbridas

As TDFs baseadas em modelos de transição são mais eficientes para a especificação dos aspectos de controle dos protocolos do que para os aspectos de dados. Dá-se o inverso com relação às TDFs baseadas em linguagens de programação. Esse fato motivou a proposição de TDfs híbridas, obtidas a partir da combinação das características positivas dos dois modelos anteriores.

O uso de MEFs com o acréscimo de variáveis é proposto em [BoGe 76]. Nesse modelo a verificação de propriedades (correção parcial e correção total das especificações) é realizada por um método que reúne a análise de alcançabilidade e a prova de asserções.

Mecanismos para o estabelecimento e o encerramento de conexões em um protocolo de transporte são especificados através de uma técnica híbrida em [SuDa 78]. A técnica utilizada associa informações de contexto (relativas ao conteúdo dos pacotes que transitam no meio de transmissão) aos estados da MEF que define o protocolo. Para a verificação de propriedades tais como, ausência de impasses e progresso da comunicação com terminação eventual, é usada uma técnica baseada na análise de alcançabilidade.

Uma técnica híbrida que reúne uma linguagem baseada em PL/I [AmNa 76] e matrizes de transições de estados, é proposta em [ScRo 80]. Nesse caso as

especificações obtidas (denominadas "meta-implementações") são compiláveis, executáveis e permitem validação automática.

Uma combinação de MEFs com alguns recursos da linguagem Pascal é proposta em [Boch 80]. A verificação de propriedades é realizada através da prova de asserções. Nesse trabalho dá-se grande importância às especificações de serviço e à estruturação das especificações.

3.4. TDFs padronizadas

No início dos anos 80, preocupados com os problemas oriundos das especificações informais dos serviços e protocolos de comunicação, vários órgãos de padronização, entre eles o CCITT e a ISO, desenvolveram TDFs que se tornaram padrões internacionais.

O CCITT aprimorou a "Specification and Description Language (SDL)" [RoSa 82] [DiCh 83] [SaTi 87], uma técnica de largo uso em ambientes de telefonia, baseada em MEFs estendidas (MEFEs).

Para SDL existem atualmente três formas de representação: uma forma gráfica (a forma original), uma forma semelhante a uma linguagem de programação (desenvolvida posteriormente) e uma pictórica (a mais recente, voltada para os usuários dos sistemas).

A especificação completa de um sistema em SDL requer a definição da estrutura do sistema (em termos de MEFEs e suas interconexões), do comportamento de cada MEFE (em termos das suas interações com as outras MEFEs e com o ambiente) e das operações realizadas com os dados associados às interações.

A ISO criou o grupo de trabalho "ISO TC97/SC21/WG1 ad hoc group on FDT" visando o desenvolvimento de TDFs para a descrição dos serviços e protocolos do RM OSI. Esse grupo dividiu-se em três subgrupos. O subgrupo A definiu noções relacionadas à arquitetura dos sistemas. Essas noções foram utilizadas pelo subgrupo B, que desenvolveu a "Extended State Transition

Language (Estelle)" [ISO 88a] e pelo subgrupo C, que desenvolveu a "Language of Temporal Ordering Specification (LOTOS)" [ISO 89].

Estelle é uma técnica híbrida baseada em uma MEFE estendida pelas construções da linguagem de programação Pascal. Os projetistas de Estelle preocuparam-se em desenvolver uma técnica de fácil aprendizado que pudesse ser colocada rapidamente à disposição dos usuários. Como consequência, pode atualmente ser encontrada uma quantidade razoável de ferramentas (compiladores, simuladores, testadores etc.) que auxiliam o desenvolvimento de protocolos. A maior desvantagem é o grau de abstração da linguagem (inferior ao da linguagem LOTOS) que obriga à especificação de pequenos detalhes relativos à implementação.

Os projetistas de LOTOS preocuparam-se em desenvolver uma técnica que dispusesse de uma sólida base matemática e que utilizasse princípios que tornassem a linguagem altamente abstrata. Para a parte dinâmica, basearam-se em "Calculus of Communicating Systems (CCS)" [Miln 80, 89] e para a parte estática basearam-se em "Algebraic Specification Techniques for Correct Design of Trusty Software Systems - 1 (ACT ONE)" [EhMa 85]. Como consequência, um tempo maior para obtenção dos primeiros resultados foi necessário e mais recentemente é que começaram a surgir ferramentas para LOTOS.

Por reunir duas álgebras, LOTOS constitui-se em uma TDF complexa. Por exemplo, não existe ainda uma semântica unificada envolvendo as duas partes (estática e dinâmica) da linguagem. Esse problema leva a dificuldades na fase de validação das especificações, que costuma ser realizada em duas etapas: na primeira, são validados os aspectos de controle e, na segunda, são validados os aspectos de dados.

CAPÍTULO IV

"CALCULUS OF COMMUNICATING SYSTEMS (CCS)"

Composição síncrona é a idéia básica sobre a qual CCS está desenvolvida. Essa álgebra concebe a interação de agentes (processos) através da ocorrência de eventos atômicos (indivisíveis) dos quais os agentes comunicantes participam. Partindo dessa idéia simples, Robin Milner desenvolveu uma teoria que vem sendo estudada e explorada para a especificação e a análise de diversos tipos de sistemas.

Uma das características mais interessantes de CCS é a sua simplicidade, motivo principal da escolha dessa ágebra para a realização das especificações neste trabalho. Um pequeno conjunto de operadores (oito) é suficiente para especificar uma grande variedade de protocolos elementares, além das funções básicas de protocolos reais. Em todos esses casos, CCS contribui para uma boa compreensão dos problemas envolvidos e para a avaliação das soluções sugeridas [Holz 82] [GuPe 83] [Hung 86] [NoYe 85] [ObLo 87].

Embora CCS possua um grande poder de expressão, abstração e análise, em alguns casos essa álgebra foi submetida a variações e/ou extensões, para melhor se adaptar a situações específicas. Por exemplo, foi associada à Lógica Temporal em [PaGu 85] e à especificação de tipos abstratos de dados em [Plet 86].

Neste capítulo o fundamental da teoria clássica de CCS é apresentado em detalhes. Esse conhecimento é suficiente para o acompanhamento das aplicações apresentadas no capítulo seguinte e serve como base para os desenvolvimentos ao longo dos demais capítulos.

4.1. Conceitos arquitetônicos

Em CCS um sistema é descrito por um agente ou por um conjunto de agentes comunicantes. Cada agente pode ser representado, graficamente, por uma caixa preta dotada de portas que permitem a comunicação desse agente com o seu ambiente e com os outros agentes do sistema. Por exemplo, no sistema representado na Fig. 4.1, o agente P pode comunicar-se com o ambiente através das portas a e b e o agente Q pode fazer o mesmo através das portas $\bar{a}$ e c. P e Q podem comunicar-se entre si através das portas complementares a e $\bar{a}$.

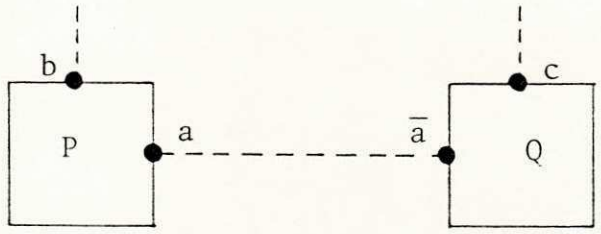


Figura 4.1 - Agentes CCS.

Na comunicação entre os agentes P e Q, o agente Q oferece um evento na porta $\bar{a}$, o qual é recebido pelo agente P em a. Esse evento é atômico, atinge simultaneamente os dois agentes e só ocorre quando ambos estão prontos.

Um evento de comunicação em CCS é uma sincronização entre dois agentes que pode se dar (ou não) com passagem de valores. Quando a teoria completa de CCS é utilizada, considera-se a passagem de valores entre agentes. Quando um subconjunto de CCS é utilizado, normalmente considera-se apenas o sincronismo puro (CCS Básico).

A teoria de CCS inclui o conceito de um observador externo ao sistema, que pode interagir com ele. O observador só é capaz de perceber uma ação atômica por vez. Se, em um determinado instante, mais de uma ação atômica é possível, uma escolha é realizada. Portanto, essa teoria não é capaz de

expressar plenamente a concorrência. CCS descreve esse fenômeno através da intercalação de eventos.

Uma especificação CCS é, na realidade, uma descrição do comportamento potencialmente observável de um sistema, em função dos eventos de comunicação a que esse sistema está sujeito. Tal especificação representa um termo da álgebra, que pode ser submetido a manipulações até ser transformado em outro termo com outra estrutura ("intension") mas com idêntico comportamento ("extension").

4.2. Sintaxe de CCS

Os termos de CCS são expressões de comportamento que utilizam rótulos ("labels") para indicar a ocorrência de eventos. Para o observador, esses eventos podem ser visíveis ou invisíveis.

$\mathcal{L} = A \cup \overline{A}$ é o conjunto de rótulos visíveis ao observador, sendo que $A = \{a, b, \ldots\}$ representa as portas que podem aceitar eventos e o conjunto de co-rótulos $\overline{A} = \{\overline{a}, \overline{b}, \ldots\}$ representa as portas que podem oferecer eventos. As comunicações entre os agentes de um sistema são indicadas pelo rótulo τ.

Por exemplo, na Fig. 4.1 os agentes P e Q comunicam-se através da porta a (de P) e $\overline{a}$ (de Q) através de um evento não observável τ.

Os rótulos podem estar vinculados a conjuntos ("tuples") de variáveis $(x_1, \ldots, x_n)$, enquanto que os co-rótulos podem estar vinculados a conjuntos de expressões de valor $(E_1, \ldots, E_n)$. Em uma comunicação, para que haja passagem de valores, é necessário que os tipos das variáveis e das expressões de valor envolvidos sejam compatíveis. Entretanto, é omitida em CCS uma definição formal dos tipos.

Se os agentes P e Q comunicam-se com passagem de valores, então a porta a de P pode estar vinculada a uma tripla de variáveis: $ax_1x_2x_3$ onde x_1, x_2 e x_3 são variáveis às quais podem ser atribuídos valores inteiros, reais e caráter (por exemplo). Nesse caso, se o agente Q oferece o evento $\overline{a}$ 5 2.0 m, a

comunicação pode ocorrer.

A cada expressão de comportamento B é associada uma espécie ("sort") $L(B) \subseteq \mathcal{L}$. Por exemplo, de acordo com a Fig. 4.1, tem-se $L(P) = \{a, b\}$ e $L(Q) = \{\bar{a}, c\}$.

A cada expressão de comportamento é associado também um conjunto de variáveis livres FV(B), que são variáveis às quais não foram ainda atribuídos valores. Considerando o agente P, se à sua porta a estão associadas as variáveis x_1, x_2 e x_3 e à sua porta b está associada a variável x_4 então, antes que P participe de algum evento, $FV(P) = \{x_1, x_2, x_3, x_4\}$.

O resultado da substituição de todas as ocorrências livres das variáveis x_i $(1 \leq i \leq n)$ em B, por expressões de valor E_i, é representado por $B\{E_1/x_1, \ldots, E_n/x_n\}$ ou, abreviadamente, $B\{E/x\}$. Se ocorreu um evento na porta a de P e os valores passados foram -5, 2.0 m, após esse evento tem-se $P(-5/x_1, 2.0/x_2, m/x_3)$.

As operações CCS, que permitem as manipulações com as expressões de comportamento, podem ser classificadas em dinâmicas e estáticas.

As operações dinâmicas descrevem os comportamentos dos agentes de uma forma indeterminística. Elas são também chamadas de operações básicas, já que qualquer agente CCS pode ser reduzido a uma expressão de comportamento constituída somente dessas operações.

NIL: expressa ausência de ação (agente inativo)

Soma (+): indica uma escolha indeterminística

Ação $(ax, \bar{a}E, \tau)$.

Em CCS utiliza-se um ponto (.) como recurso sintático para separar a representação de dois eventos sucessivos. Esse ponto não é considerado um operador.

Por exemplo, o agente P pode ter o comportamento definido por

$$P \stackrel{def}{=} axyz.NIL + bw.NIL$$

onde P pode receber um evento na porta a ou na porta b, indeterministicamente.

Nesse caso o indeterminismo (ou escolha) é resolvido pela participação do ambiente, que decide por um evento na porta a ou por um evento na porta b. Qualquer que seja a escolha, após a acorrência do evento o agente P torna-se inativo.

O agente Q, por sua vez, pode ter o comportamento definido por

$$Q \stackrel{def}{=} \overline{a}\ 5\ 2.0\ m.ct.NIL + \tau.NIL$$

onde Q pode oferecer um evento na porta $\overline{a}$ ou por decisão própria, isto é, sem participação do ambiente (o evento não observável τ ocorre), tornar-se inativo. Se $\overline{a}$ ocorre Q fica pronto para um evento em c e, quando este último ocorre, Q torna-se inativo.

As operações estáticas fixam uma estrutura de ligação entre os comportamentos dos agentes de um sistema.

- Composição (|): combina agentes, sendo que pode haver comunicação entre eles. Assim é construído o sistema composto (Fig. 4.1).

  $$C = (P|Q)$$

  onde P e Q podem se comunicar um com o outro e com o ambiente.

- Restrição (\a): esconde portas do observador. A combinação de | com \a permite restringir os eventos, que podem ocorrer em a, às comunicações τ (que passam a ser internas). O agente composto (Fig. 4.2)

  $$C' = (P|Q)\backslash a$$

  só pode comunicar-se com o ambiente através das portas b e c.

- Rerrotulação ([S]): permite a troca dos nomes dos rótulos. Essa troca é definida pela operação S. Por exemplo, S = mn/ab significa trocar a por m e b por n (Fig. 4.3).

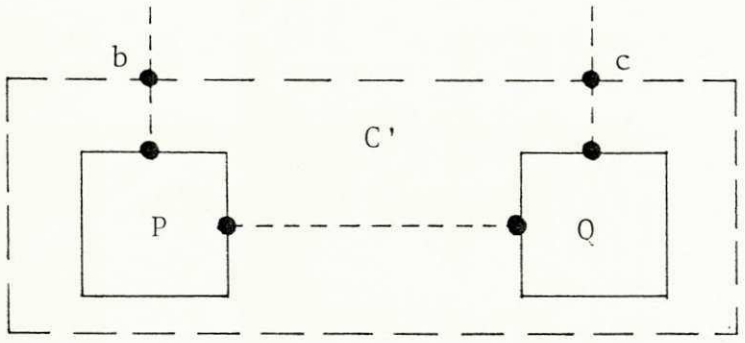


Figura 4.2 - O agente C'.

Figura 4.3 - Rerrotulação de Q.

- condição (if E then B else B'): indica uma escolha a ser realizada em função do valor assumido pela expressão booleana E. Por exemplo, o agente T (Fig. 4.4) pode ter o comportamento definido por

$$T \stackrel{def}{=} ax.(\underline{if}\ x \geq 0\ \underline{then}\ \overline{b}x.T\ \underline{else}\ \overline{c}(-x).T)$$

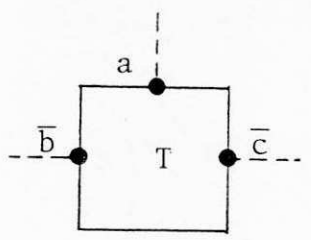


Figura 4.4 - O agente T.

T pode receber (na porta a) um número que é atribuído à variável x (de tipo adequado). Se o número for positivo ou nulo ele é oferecido na porta $\bar{b}$ e, após a ocorrência desse evento, T volta recursivamente. Se o número for negativo ele é oferecido (com o sinal trocado) na porta $\bar{c}$ e, após a ocorrência desse evento, T volta recursivamente.

- identificador ($b[\tilde{E}]$): permite a parametrização das expressões de comportamento. Por exemplo, o identificador b(y) pode ser definido por

$$b(y) \stackrel{def}{=} az.(\underline{if}\ x \geq y\ \underline{then}\ \bar{b}z.NIL\ \underline{else}\ \bar{b}y.NIL)$$

Uma instância de b(y) é o agente $B_b(3/y)$ (Fig. 4.5) onde o parâmetro y assumiu o valor 3 (por exemplo). Esse agente pode receber (na porta a) um número que é atribuído à variável z (de tipo adequado). Se o número for maior ou igual a 3, ele é devolvido ao ambiente (na porta $\bar{b}$) e o agente se torna inativo. Caso contrário , o valor 3 é devolvido (na porta $\bar{b}$) e o agente se torna inativo.

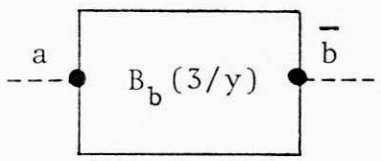


Figura 4.5 - Instância de b(y).

A precedência entre os operadores CCS é:

restrição ou rerrotulação > ação > composição > soma.

Um resumo da sintaxe de CCS é apresentado na Tabela 4.1.

TABELA 4.1 - Sintaxe das expressões de comportamento em CCS.

Operador	B"	L(B")	FV(B")
Inação	NIL	ø	ø
Soma	B + B'	L(B) ∪ L(B')	FV(B) ∪ FV(B')
Ação	$a\ x_1,\ldots,x_n.B$	L(B) ∪ {a}	$FV(B) - \{x_1,\ldots,x_n\}$
	$\bar{a}\ E_1,\ldots,E_n.B$	L(B) ∪ {$\bar{a}$}	$FV(B) \cup_i FV(E_i)$
	$\tau.B$	L(B)	FV(B)
Composição	B|B'	L(B) ∪ L(B')	FV(B) ∪ FV(B')
Restrição	B\a	L(B) - {a, $\bar{a}$}	FV(B)
Rerrotulação	B[S]	S(L(B))	FV(B)
Identificador	$b(E_1,\ldots,E_{n(b)})$	L(B)	$\cup_i FV(E_i)$
Condicional	if E then B else B'	L(B) ∪ L(B')	FV(E) ∪ FV(B) ∪ FV(B')

Fonte: [Miln 80], pág. 68.

4.3. Semântica de CCS

A semântica de CCS é definida através de um conjunto de regras de inferência, obtidas a partir de ações atômicas $\xrightarrow{\mu v}$ que, por sua vez, são definidas por indução na estrutura das expressões de comportamento. Portanto, todas as ações atômicas de uma expressão de comportamento composta podem ser inferidas a partir das ações atômicas de seus componentes.

$B \xrightarrow{\mu v} B'$: O agente B, através da ação $\xrightarrow{\mu v}$ pode transformar-se no agente B', onde $\mu \in \mathcal{L} \cup \{\tau\}$ e v é um valor de tipo apropriado a μ.

Para o caso particular $\xrightarrow{\tau}$, o valor v de tipo apropriado a τ é $\varnothing$ ("0-tuple").

- Ausência de ação

NIL não realiza ações atômicas

- Soma

$$(1)\ \frac{B_1 \xrightarrow{\mu v} B_1'}{B_1 + B_2 \xrightarrow{\mu v} B_1'}$$

A partir de $B_1 \xrightarrow{\mu v} B_1'$ infere-se que $B_1 + B_2 \xrightarrow{\mu v} B_1'$

$$(2)\ \frac{B_2 \xrightarrow{\mu v} B_2'}{B_1 + B_2 \xrightarrow{\mu v} B_2'}$$

A partir de $B_2 \xrightarrow{\mu v} B_2'$ infere-se que $B_1 + B_2 \xrightarrow{\mu v} B_2'$

Desse modo, conclui-se que as ações atômicas de uma soma são exatamente as ações atômicas das suas parcelas.

- Ação

$$(1)\ ax_1,\ldots,x_n.B \xrightarrow{a(v_1,\ldots,v_n)} B\{v_1/x_1,\ \ldots\ ,v_n/x_n\}$$

A recepção de uma "n-tuple" de valores transforma as variáveis livres $x_1,\ldots,x_n$ de B em variáveis às quais foram atribuídos esses valores.

$$(2)\ \bar{a}v_1,\ldots,v_n.B \xrightarrow{\bar{a}(v_1,\ldots,v_n)} B$$

A entrega de uma "n-tuple" de valores transforma o agente $\bar{a}v_1,\ldots,v_n.B$ no agente B.

$$(3)\ \tau.B \xrightarrow{\tau} B$$

A ocorrência do evento τ transforma o agente $\tau.B$ no agente B.

- Composição

$$(1)\ \frac{B_1 \xrightarrow{\mu v} B_1'}{B_1|B_2 \xrightarrow{\mu v} B_1'|B_2}$$

A partir de $B_1 \xrightarrow{\mu v} B_1'$ infere-se que $B_1|B_2 \xrightarrow{\mu v} B_1'|B_2$

$$(2)\ \frac{B_2 \xrightarrow{\mu v} B_2'}{B_1|B_2 \xrightarrow{\mu v} B_1|B_2'}$$

Na composição $B_1|B_2$, uma ação de B_1 ou de B_2 fornece uma ação do agente composto, na qual o outro componente não é afetado.

$$(3)\ \frac{B_1 \xrightarrow{\lambda v} B_1' \qquad B_2 \xrightarrow{\bar{\lambda} v} B_2'}{B_1 | B_2 \xrightarrow{\tau} B_1' | B_2'} \quad \lambda \in A \cup \bar{A}$$

A comunicação entre os componentes fornece uma ação τ do agente composto.

- Restrição

$$\frac{B \xrightarrow{\mu v} B'}{B\backslash a \xrightarrow{\mu v} B'\backslash a} \qquad \mu \notin \{a, \bar{a}\}$$

A condição $\mu \notin \{a, \bar{a}\}$ assegura que $B\backslash a$ não tem a ação av nem a ação $\bar{a}v$.

- Rerrotulação

$$\frac{B \xrightarrow{\mu v} B'}{B[S] \xrightarrow{(S\mu)v} B'[S]}$$

A ocorrência de uma ação na porta rerrotulada afeta o agente rerrotulado como o afetaria antes da rerrotulação.

- Identificador

Supondo que o identificador parametrizado $b(x_1, \ldots, x_{n(b)})$ é definido pela expressão de comportamento B_b, isto é,

$$b(x_1, \ldots, x_{n(b)}) \stackrel{def}{=} B_b$$

onde $FV(B_b) \subseteq \{x_1, \ldots, x_{n(b)}\}$, então a regra é

$$\frac{B_b\{v_1/x_1, \ldots, v_{n(b)}/x_{n(b)}\} \xrightarrow{\mu v} B'}{b(v_1, \ldots, v_{n(b)}) \xrightarrow{\mu v} B'}$$

Um identificador parametrizado tem exatamente as mesmas ações que a instância apropriadada do lado direito da sua definição.

- Condição

$$(1)\ \frac{B_1 \xrightarrow{\mu v} B_1'}{\underline{\text{if}}\ \text{true}\ \underline{\text{then}}\ B_1\ \text{else}\ B_2 \xrightarrow{\mu v} B_1'}$$

Quando a expressão lógica é avaliada "true", o comportamento posterior é o da primeira alternativa (B_1).

$$(2)\ \frac{B_2 \xrightarrow{\mu v} B_2'}{\underline{\text{if}}\ \text{false}\ \underline{\text{then}}\ B_1\ \underline{\text{else}}\ B_2 \xrightarrow{\mu v} B_2'}$$

Quando a expressão lógica é avaliada "false", o comportamento posterior é o da segunda alternativa (B_2).

Um resumo da semântica de CCS é apresentado na Tabela 4.2.

TABELA 4.2 - Semântica das expressões de comportamento em CCS.

Operação	Semântica
Inação	NIL não tem ações atômicas
Soma	$\dfrac{B_1 \xrightarrow{uv} B'_1}{B_1+B_2 \xrightarrow{uv} B'_1+B_2} \qquad \dfrac{B_2 \xrightarrow{uv} B'_2}{B_1+B_2 \xrightarrow{uv} B_1+B'_2}$
Ação	$ax_1,\ldots,x_n.B \xrightarrow{a(v_1,\ldots,v_n)} B\{v_1/x_1,\ldots,v_n/x_n\}$ $\bar{a}v_1,\ldots,v_n.B \xrightarrow{\bar{a}(v_1,\ldots,v_n)} B$ $\tau.B \xrightarrow{\tau} B$
Composição	$\dfrac{B_1 \xrightarrow{uv} B'_1}{B_1\|B_2 \xrightarrow{uv} B'_1\|B_2} \quad \dfrac{B_2 \xrightarrow{uv} B'_2}{B_1\|B_2 \xrightarrow{uv} B_1\|B'_2} \quad \dfrac{B_1 \xrightarrow{\lambda v} B'_1 \;\; B_2 \xrightarrow{\bar{\lambda}v} B'_2}{B_1\|B_2 \xrightarrow{\tau} B'_1\|B'_2}$
Restrição	$\dfrac{B \xrightarrow{uv} B'}{B\backslash a \xrightarrow{uv} B'\backslash a}$, $\mu \notin \{a,\bar{a}\}$
Rerrotulação	$\dfrac{B \xrightarrow{uv} B'}{B[S] \xrightarrow{(Su)v} B'[S]}$
Identificador	$\dfrac{B_b\{v_1/x_1,\ldots,v_{n(b)}/x_{n(b)}\} \xrightarrow{uv} B'}{b(v_1,\ldots,v_{n(b)}) \xrightarrow{uv} B'}$
Condicional	$\dfrac{B_1 \xrightarrow{uv} B'_1}{(\underline{\text{if}}\ \text{true}\ \underline{\text{then}}\ B_1\ \underline{\text{else}}\ B_2) \xrightarrow{uv} B'_1} \quad \dfrac{B_2 \xrightarrow{uv} B'_2}{(\underline{\text{if}}\ \text{false}\ \underline{\text{then}}\ B_1\ \underline{\text{else}}\ B_2) \xrightarrow{uv} B'_2}$

Fonte: [Miln 80], pp. 69-71.

4.4. Lei de Expansão

Para representar todas as possibilidades de ações, que um sistema composto pode executar, considera-se a intercalação das ações dos seus componentes. A expressão resultante (em termos de ações e somas) é denominada expansão do sistema. Graficamente tal expansão é representada por uma árvore

de sincronização (no CCS Básico) ou por uma árvore de comunicação (no CCS Completo).

Por exemplo, considere o sistema composto

$$K = (C|D)\backslash\{c\}$$

onde os agentes C e D são definidos (em CCS Básico) por

$$C \stackrel{def}{=} a.\bar{c}.NIL$$

$$D \stackrel{def}{=} b.c.NIL + \tau.c.NIL$$

A substituição dos identificadores C e D pelas respectivas expressões de comportamento permite uma melhor compreensão do comportamento de K:

$$K = (a.\bar{c}.NIL|b.c.NIL + \tau.c.NIL)\backslash\{c\}$$

Expandindo o comportamento desse sistema, obtém-se

$$K = a.K1 + b.K2 + \tau.K3$$

onde

$$K1 = (\bar{c}.NIL|b.c.NIL + \tau.c.NIL)\backslash\{c\}$$
$$K2 = (a.\bar{c}.NIL|c.NIL)\backslash\{c\}$$
$$K3 = (a.\bar{c}.NIL|c.NIL)\backslash\{c\}$$

Isto é, a primeira ação de K pode ocorrer na porta a (de C) ou na porta b (de D) ou, ainda, como um evento não observável τ (de D). No primeiro caso, o comportamento posterior de K é identificado por K1, no segundo, por K2 e no terceiro, por K3.

Expandindo o comportamento de K1, obtém-se

$$K1 = b.K11 + \tau.K12$$

onde

$$K11 = (\overline{c}.NIL \mid c.NIL)\backslash\{c\}$$

$$K12 = (\overline{c}.NIL \mid c.NIL)\backslash\{c\}$$

Isto é, a primeira ação de K1 pode ocorrer na porta b (de C) ou como um evento não observável τ (de D). No primeiro caso, o comportamento posterior de K1 é identificado por K11 e no segundo, por K12.

Expandindo o comportamento de K11, obtém-se

$$K11 = \tau.T$$

onde

$$T = (NIL \mid NIL)\backslash\{c\}$$

Isto é, pode haver uma sincronização dos agentes C e D (τ ocorre) através das suas portas complementares $\overline{c}$ e c, respectivamente. O comportamento posterior de K11 (identificado por T) é inativo.

Graficamente, a expansão completa de K é representada na Fig. 4.6, através de uma árvore de sincronização onde, nas folhas, T1, T2 e T3 são especificados do mesmo modo que T.

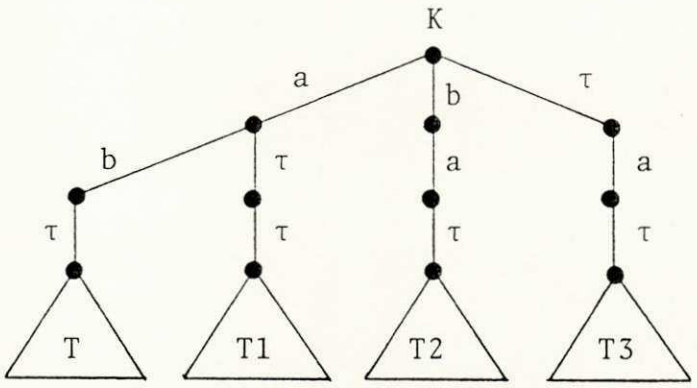


Figura 4.6 - Árvore de sincronização de K.

Uma árvore de sincronização t, correspondente ao comportamento de um sistema, pode ser expressa como

$$t = \sum_{i=1}^{n} \mu_i . t_i$$

onde cada μ_i representa uma ação e cada t_i representa uma árvore de sincronização correspondente ao comportamento do sistema posterior à ocorrência de μ_i. Na toeoria de CCS, a ação μ_i é denominada "guarda" ou "prefixo" de t_i e a função "nome", aplicada a um guarda, dá o nome de porta onde a ação da guarda ocorre. Por exemplo, nome($\bar{a}$) = a.

Seja a expressão de comportamento

$$B = (B_1 | \ldots | B_m) \backslash A$$

onde cada B_i é uma somatória de guardas g (a, $\bar{a}$ ou τ) e $A \subseteq \mathcal{L}$.

A expressão que fornece o desdobramento de B em sincronizações puras e somas é a Lei de Expansão para o CCS Básico:

$$B = \sum \{\mu . ((B_1 | \ldots | B'_i | \ldots | B_m) \backslash A) \quad 1 \leq i \leq m$$
$$\text{onde } \mu . B'_i \text{ é um termo de } B_i \text{ e nome}(\mu) \notin A\}$$
$$+ \sum \{\tau . ((B_1 | \ldots | B'_i | \ldots | B'_j | \ldots | B_m) \backslash A) \quad 1 \leq i < j \leq m,$$
$$\text{onde } \lambda . B'_i \text{ é um termo de } B_i \text{ e } \bar{\lambda} . B'_j \text{ é um termo de } B_j\}$$

A representação gráfica da expansão de uma especificação em CCS Completo (árvore de comunicação) é realizada considerando a passagem de valores. Para representar um evento de recepção (e.g., ax) considera-se todos os valores que podem ser atribuídos à variável associada a esse evento (no caso, a variável é x). Para representar um evento de oferecimento (e.g., $\bar{b}5$) basta considerar o(s) valor(es) oferecido(s) (no caso, o valor é 5). O evento τ é representado como nas árvores de sincronização. Na Fig. 4.7 é apresentada uma árvore de comunicação típica.

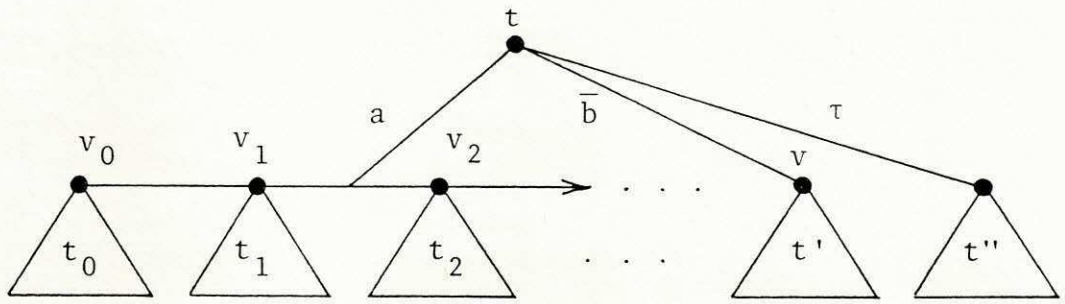


Figura 4.7 - Árvore de comunicação.

No CCS Completo, um guarda de recepção inclui a tupla de variáveis (e.g., $a\tilde{x}$) e um guarda de oferecimento inclui a tupla de expressões de valor (e.g., $\bar{a}5\ 9{,}0\ m$). Nesse caso,

$$nome(a\tilde{x}) = a$$

$$nome(\bar{a}5\ 9.0\ m) = a$$

Seja a expressão de comportamento

$$B = (B_1 | \ldots | B_m) \backslash A$$

onde cada B_i é uma somatória de guardas g ($a\tilde{x}$, $\bar{a}\tilde{E}$ ou τ) e $A \subseteq \mathcal{L}$.

A expressão que fornece o desdobramento de B em ações (com passagem de valores) e somas é a Lei de Expansão para o CCS Completo:

$B = \sum \{g.((B_1 | \ldots | B'_i | \ldots | B_m) \backslash A)$, onde $g.B'_i$ é um termo da somatória de B_i e $nome(g) \notin A\}$

$+ \sum \{\tau.((B_1 | \ldots | B'_i\{\tilde{E}/\tilde{x}\} | \ldots | B'_j | \ldots | B_m) \backslash A)$, onde $a\tilde{x}.B'_i$ é um termo da somatória de B_i e $\bar{a}\tilde{E}.B'_j$ é um termo da somatória de B_j e $i \neq j\}$.

OBS: aplicando-se sucessivamente a Lei de Expansão é possível transformar qualquer expressão de comportamento em um conjunto de ações e somas.

4.5. Relações de equivalência

Intuitivamente é possível admitir que um mesmo comportamento seja expresso de maneiras diferentes em CCS. Ou ainda que dois agentes apresentem comportamentos muito semelhantes, e.g., sejam intercambiáveis, em algum tipo de contexto, sem alterações observáveis do sistema global. Nesses casos pode-se dizer que existe algum tipo de equivalência entre os agentes.

Para tornar precisa a discussão sobre as relações de equivalência entre agentes, algumas das relações definidas em CCS são:

equivalência direta ($\equiv$)

equivalência forte ($\sim$)

equivalência de observação ($\approx$)

congruência de observação ($\approx^c$)

4.5.1. Equivalência direta

Definição. Dois agentes B e C são diretamente equivalentes ($B \equiv C$) se e somente se para todo μ, v e D

$$B \xrightarrow{\mu v} D \iff C \xrightarrow{\mu v} D$$

Por exemplo, sejam os agentes B e C definidos em CCS Básico por

$$B \stackrel{def}{=} a.NIL + NIL$$
$$C \stackrel{def}{=} a.NIL$$

Os comportamentos desses agentes podem ser representados,

graficamente, por árvores de sincronização como mostra a Fig. 4.8.

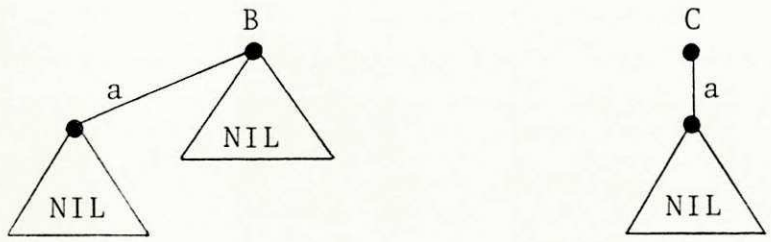


Figura 4.8 - Árvores de sincronização B e C.

O agente B oferece, como única possibilidade de ação, um evento na porta a, tornando-se depois inativo:

$$B \xrightarrow{a} NIL$$

O agente C também oferece essa possibilidade:

$$C \xrightarrow{a} NIL$$

de modo que $B \xrightarrow{a} NIL \Longrightarrow C \xrightarrow{a} NIL$. Inversamente, tem-se $C \xrightarrow{a} NIL \Longrightarrow B \xrightarrow{a} NIL$. Então $B \equiv C$.

Entretanto, o agente J definido por

$$J \stackrel{def}{=} a.NIL + \tau.NIL$$

oferece as possibilidades

$$J \xrightarrow{a} NIL$$

$$J \xrightarrow{\tau} NIL$$

A segunda possibilidade não tem correspondente em B nem em C, por isso $J \not\equiv B$ e $J \not\equiv C$.

Um resumo das propriedades da relação de equivalência direta é apresentado na Tabela 4.3.

TABELA 4.3 - Propriedades da relação de equivalência direta (≡).

Operação	Propriedades
Soma	$B+B \equiv B \qquad B1+(B2+B3) \equiv (B1+B2)+B3$ $B+NIL \equiv B \qquad B1+B2 \equiv B2+B1$
Ação	$a\tilde{x}.B \equiv a\tilde{y}.B\{\tilde{y}/\tilde{x}\}$
Restrição	$NIL\backslash B \equiv NIL \qquad (B1+B2)\backslash\beta \equiv B1\backslash\beta + B2\backslash\beta$ $(g.B)\backslash\beta \equiv \begin{cases} NIL & \text{se } \beta = nome(g) \\ g.B\backslash\beta & \text{caso contrário} \end{cases}$
Rerrotulação	$NIL[S] \equiv NIL \qquad (g.B)[S] \equiv Sg.B[S]$ $(B1+B2)[S] \equiv B1[S]+B2[S]$
Composição	Sejam B e C somas de guardas. Então $B\|C \equiv \sum \{g.(B'\|C) \;;\; g.B' \text{ uma parcela de } B\}$ $+ \sum \{g.(B\|C') \;;\; g.C' \text{ uma parcela de } C\}$ $+ \sum \{\tau.(B'\{\tilde{v}/\tilde{x}\}\|C') \;;\; a\tilde{x}.B' \text{ uma parcela de } B \text{ e } \bar{a}\tilde{v}.C' \text{ uma parcela de } C\}$ $+ \sum \{\tau.(B'\|C'\{\tilde{v}/\tilde{x}\}) \;;\; \bar{a}\tilde{v}.B' \text{ uma parcela de } B \text{ e } a\tilde{x}.C' \text{ uma parcela de } C\}$
Identificador	Seja o identificador definido por $b(\tilde{x}) \stackrel{def}{=} B_b$; então $b(\tilde{v}) \equiv B_b\{\tilde{v}/\tilde{x}\}$
Condicional	$(\underline{if}\ true\ \underline{then}\ B_1\ \underline{else}\ B2 \equiv B1$ $(\underline{if}\ false\ \underline{then}\ B_1\ \underline{else}\ B2 \equiv B2$

Fonte: [Miln 80], pp. 75-76.

A relação ≡ exige que os comportamentos posteriores às ações sejam idênticos (isto é, tenham a mesma expressão de comportamento). Uma relação de equivalência mais tolerante (abrangendo uma variedade maior de comportamentos) pode ser definida exigindo-se que os resultados das ações sejam apenas equivalentes.

4.5.2. Equivalência forte

Na relação de equivalência forte (~) a ação de um agente corresponde a uma ação igual do outro agente, mesmo para ações τ. Formalmente, a relação ~ é definida através de uma série decrescente de relações de equivalência.

Definição. Sejam B e C agentes CCS.

$B \sim_0 C$ é sempre verdade.

$B \sim_{k+1} C$ se e somente se para todo μ e ν,

$$\text{(i) se } B \xrightarrow{\mu\nu} B' \text{ então } \exists C',\ C \xrightarrow{\mu\nu} C' \text{ e } B' \sim_k C'$$

$$\text{(ii) se } C \xrightarrow{\mu\nu} C' \text{ então } \exists B',\ B \xrightarrow{\mu\nu} B' \text{ e } B' \sim_k C'$$

$B \sim C$ se e somente se para todo $k \geq 0$, $B \sim_k C$.

Por exemplo, seja o agente B definido em CCS Básico por

$$B \stackrel{def}{=} a.B + b.B$$

e o agente composto C obtido a partir da composição de B com NIL

$$C = (B|NIL)$$

Pode-se provar que $B \sim C$.

Prova. $B \sim_0 C$ é verdade. Supondo que $B \sim_k C$, pode-se provar que $B \sim_{k+1} C$, uma vez que a ocorrência de qualquer ação (a ou b) reconstitui o

agente B e tem o mesmo efeito sobre o agente C:

$$B \xrightarrow{a} B \qquad C \xrightarrow{a} C$$

$$B \xrightarrow{b} B \qquad C \xrightarrow{b} C$$

Como $B \sim_k C \Longrightarrow B \sim_{k+1} C$, então $B \sim C$.

Entretanto, $B \not\equiv C$ uma vez que o resultado da ação $\xrightarrow{a}$ sobre B não é idêntico ao resultado da ação $\xrightarrow{a}$ sobre C:

$$B \xrightarrow{a} a.B + b.B$$

$$C \xrightarrow{a} (a.B + b.B \mid NIL)$$

Um outro exemplo de equivalência forte entre agentes está relacionado com a Lei de Expansão. Na realidade a relação de equivalência entre um agente e o seu desdobramento segundo a Lei de Expansão é uma relação de equivalência forte.

Embora mais fraca do que a relação $\equiv$, a relação $\sim$ ainda pode ser considerada muito forte. Para comparar agentes que representam o mesmo comportamento externo observável, mas que tenham diferentes estruturas internas (com componentes que se intercomuniquem), é preciso definir uma relação de equivalência que ignore, em certa medida, os eventos de intercomunicação dos componentes dos sistemas.

Um resumo das propriedades da relação de equivalência forte é apresentado na Tabela 4.4.

TABELA 4.4 - Propriedades da relação de equivalência forte (~).

Operação	Propriedades
Composição	$B1 \mid NIL \sim B1 \quad B1 \mid B2 \sim B2 \mid B1$ $B1 \mid (B2 \mid B3) \sim (B1 \mid B2) \mid B3$
Restrição	$B\backslash a\backslash b \sim B\backslash b\backslash a$ $B\backslash a \sim B \quad (B:L, \ a \notin nomes(L))$ $(B1 \mid B2)\backslash a \sim B1\backslash a \mid B2\backslash a$ $(B1:L1, \ B2:L2, \ a \notin nomes(L1 \cap \overline{L2}))$
Rerrotulação	$B[I] \sim B \quad (I:L \longrightarrow L$ é a rerrotulação identidade$)$ $B[S] \sim B[S'] \ (B:L, \text{ e } S \upharpoonright L = S' \upharpoonright L)$ $B[S]\backslash b \sim B\backslash a[S] \quad (b = nome(S(a)))$ $B[S][S'] \sim B[S' \circ S]$ $(B1 \mid B2)[S] \sim B1[S] \mid B2[S]$

Fonte: [Miln 80], pp. 79-80.

4.5.3. Equivalência de observação

Informalmente, dois agentes M e N são equivalentes quanto à observação ($M \approx N$) se um observador que realiza experiências com ambos é incapaz de distingui-los. Uma vez que a relação $\approx$ permite ignorar (em certos casos) a representação dos eventos internos de um sistema, pode-se estabelecer que uma especificação abstrata ABS tem um detalhamento correto DET se $ABS \approx DET$.

A relação $\approx$ pode ser definida formalmente, através de uma sequência decrescente (cada vez menos abrangente) de relações de equivalência, de modo semelhante ao que foi feito para a relação ~.

Definição. Sejam B, B', C e C' expressões de comportamento e $B \stackrel{s}{=\!=\!\Rightarrow} B'$ a forma abreviada de

$$B \xrightarrow{\tau^{m_0}.\mu_1 v_1.\tau^{m_1}.\ldots.\mu_k v_k.\tau^{m_k}} B'$$

para k, $m_0, \ldots, m_k \geq 0$. Então:

$B \approx_0 C$ é sempre verdadeiro.

$B \approx_{k+1} C$ se e somente se para todo $s \in (\mathcal{L} \times V)^*$

(i) se $B \stackrel{s}{\Longrightarrow} B'$ então $\exists C'$ tal que $C \stackrel{s}{\Longrightarrow} C'$ e $B' \approx_k C'$

(ii) se $C \stackrel{s}{\Longrightarrow} C'$ então $\exists B'$ tal que $B \stackrel{s}{\Longrightarrow} B'$ e $B' \approx_k C'$

$B \approx C$ se e somente se para todo $k \geq 0$, $B \approx_k C$

OBS: $\mathcal{L}$xV representa o produto cartesiano do conjunto de rótulos $\mathcal{L}$ pelo conjunto de valores V; * representa uma potência inteira.

Propriedades da relação $\approx$

A prova da equivalência de observação, entre diferentes especificações de um mesmo sistema, costuma ser realizada com o auxílio das propriedades dessa relação. Por exemplo,

(1) $B \approx \tau.B$ (absorção de τ)

(2) Se $P \approx Q$ então $P|R \approx Q|R$

(3) Se $P \approx Q$ então $P\backslash L \approx Q\backslash L$

(4) Se $P \approx Q$ então $P[S] \approx Q[S]$

Entretanto, mesmo para sistemas simples, a prova da equivalência de observação pode se tornar uma tarefa difícil de ser realizada sem o auxílio de ferramentas. Uma técnica que visa, principalmente, a facilitação da realização de tais provas e que oferece uma base para a construção de ferramentas, utiliza o conceito de bissimulação [Park 81].

4.5.4. Bissimulação

A prova da equivalência de observação entre dois agentes pode ser realizada mostrando-se que existe uma relação de bissimulação fraca (BF) entre eles.

Antes da apresentação da definição formal de BF (correspondente ao CCS Básico) é introduzida a notação:

(1) P é o conjunto de todos os agentes CCS;

(2) $Act = A \cup \bar{A} \cup \{\tau\}$;

(3) Act^* representa o conjunto de sequências de eventos construídas com os elementos de Act;

(4) se $t \in Act^*$ então $\hat{t}$ representa a sequência obtida apagando-se todas as ocorrências de τ em t. Particularmente, $\hat{\tau} = \varepsilon$ (a sequência vazia);

(5) a relação de transição $\stackrel{t}{\Longrightarrow}$, onde $t \in Act^*$, é definida para sequências de eventos que podem conter τ. Se $t = a_1 \ldots a_n \in Act^*$, então $F \stackrel{t}{\Longrightarrow} F'$ se

$$F(\stackrel{\tau}{\longrightarrow})^*(\stackrel{a_1}{\longrightarrow})(\stackrel{\tau}{\longrightarrow})^* \ldots (\stackrel{\tau}{\longrightarrow})^*(\stackrel{a_n}{\longrightarrow})(\stackrel{\tau}{\longrightarrow})^* F'$$

Definição. Uma relação binária $BF \subseteq PxP$ entre os agentes P1 e P2 é uma bissimulação fraca ou, simplesmente, uma bissimulação se $\langle P1, P2\rangle \in BF$ implica que, para todo $a \in Act$,

(1) sempre que $P1 \stackrel{a}{\longrightarrow} P1'$ então $\exists$ P2',
$P2 \stackrel{\hat{a}}{\Longrightarrow} P2'$ e $\langle P1', P2'\rangle \in BF$

(2) sempre que $P2 \stackrel{a}{\longrightarrow} P2'$ então $\exists$ P1',
$P1 \stackrel{\hat{a}}{\Longrightarrow} P1'$ e $\langle P1', P2'\rangle \in BF$.

Por exemplo, pode-se provar que os agentes Q e Q', definidos por

$$Q \stackrel{def}{=} a.(b.NIL + b.NIL)$$

$$Q' \stackrel{def}{=} a.b.NIL$$

são equivalentes quanto à observação ($Q \approx Q'$) usando o conceito de

bissimulação.

Prova. A prova de Q ≈ Q' é facilitada construindo-se as árvores de sincronização de Q e Q' e, após isso, associando-se os nós (estados de Q e Q') correspondentes dessas duas árvores, como mostra a Fig. 4.9.

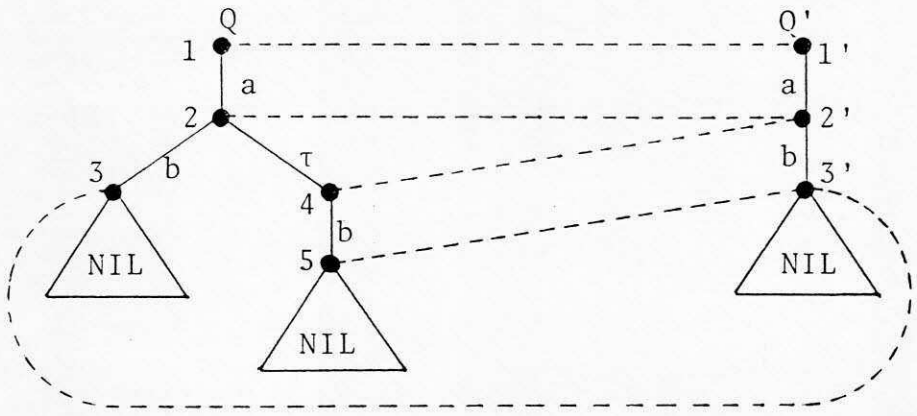


Figura 4.9 - Associação entre os estados de Q e Q'.

Com base nas árvores de sincronização de Q e Q', constrói-se a relação binária BF entre estados:

$$BF = \{\langle 1,1'\rangle, \langle 2,2'\rangle, \langle 3,3'\rangle, \langle 4,2'\rangle, \langle 5,3'\rangle\}$$

Para mostrar que BF é uma bissimulação tem-se que verificar todos os estados derivados de cada um dos oitos estados envolvidos por BF. Por exemplo, o estado 2 (de Q) tem dois estados derivados, a saber, o estado 3 e o estado 4:

$$2 \xrightarrow{b} 3 \qquad 2 \xrightarrow{\tau} 4$$

Correspondendo à primeira transição tem-se $2' \overset{b}{\Longrightarrow} 3'$, sendo que $\langle 3,3'\rangle \in BF$. Correspondendo à segunda transição, tem-se $2' \overset{\varepsilon}{\Longrightarrow} 2'$, sendo que $\langle 4,2'\rangle \in BF$.

Inversamente, correspondendo a 2' $\xrightarrow{b}$ 3' tem-se 4 $\stackrel{b}{\Longrightarrow}$ 5, sendo que $\langle 5,3' \rangle \in BF$. As outras verificações são realizadas do mesmo modo.

Um resumo das propriedades da relação de equivalência de observação é apresentado na Tabela 4.5.

TABELA 4.5 - Propriedades da relação de equivalência de observação ($\approx$).

Operação	Propriedades
Soma	$B + B \approx B$ $B + NIL \approx B$ $B1 + B2 \approx B2 + B1$ $B1 + (B2 + B3) \approx (B1 + B2) + B3$
Ação	Sejam B, B1 e B2 agentes CCS. Então, se $B1 \approx B2$, tem-se: $a.B1 \approx a.B2$ $B \approx \tau.B$
Restrição	$NIL\backslash a \approx NIL$ $B\backslash a \approx B$ se $a, \bar{a} \notin L(B)$ $(B1+B2)\backslash a \approx B1\backslash a + B2\backslash a$ $(B1 \mid B2)\backslash a \approx B1\backslash a \mid B2\backslash a$ se $a, \bar{a} \notin L(B1) \cup L(B2)$ Se $B1 \approx B2$ então $B1\backslash A \approx B2\backslash A$
Rerrotulação	Se $B1 \approx B2$ então $B1[S] \approx B2[S]$
Composição	$B1 \mid B2 \approx B2 \mid B1$ $\quad B \mid NIL \approx B$ $B1 \mid (B2 \mid B3) \approx (B1 \mid B2) \mid B3$ Se $B1 \approx B2$ então $B1 \mid B \approx B2 \mid B$

Fontes: [Miln 80] e [Shie 87].

4.5.5. Congruência de observação

Dados dois agentes Q1 e Q2, tais que $Q1 \approx Q2$, nem sempre esses agentes são intercambiáveis em todos os contextos.

Por exemplo, seja o contexto C[] onde [] é um espaço vazio que pode ser ocupado por um agente:

$$C[] \stackrel{def}{=} a.(b.NIL + [])$$

Definindo-se os agentes Q1 e Q2 por

$$Q1 \stackrel{def}{=} \tau.NIL$$
$$Q2 \stackrel{def}{=} NIL$$

podem ser construídas as árvores de sincronização de Q1 e Q2 como mostra a Fig. 4.10.

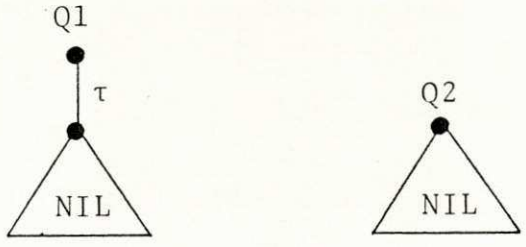


Figura 4.10 - Árvores de sincronização Q1 e Q2.

Pode-se verificar facilmente que Q1 ≈ Q2. Entretanto

$$C[Q1] \not\approx C[Q2]$$

uma vez que

$$C[Q2] \stackrel{def}{=} a.(b.NIL + NIL)$$

após a concorrência do evento a, oferece a possibilidade do evento b até que este ocorra, enquanto que, em

$$C[Q1] \stackrel{def}{=} a.(b.NIL + \tau.NIL)$$

a ocorrência de τ frustra a ocorrência de b (Fig. 4.11).

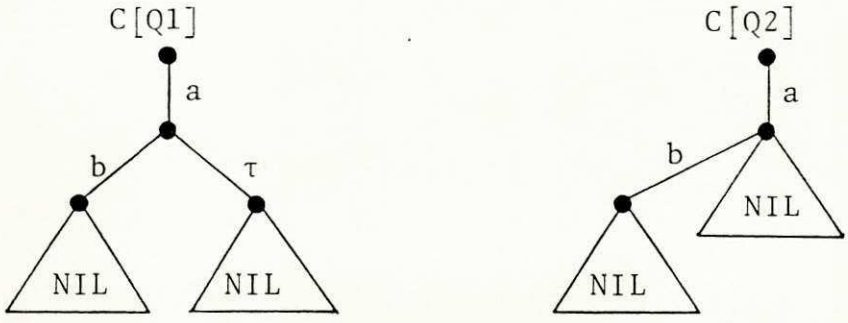


Figura 4.11 - Árvores de sincronização C[Q1] e C[Q2].

Definição. Dois agentes R1 e R2 são congruentes quanto à observação ($R1 \approx^c R2$) se e somente se, para qualquer expressão de contexto C[], $C[R1] \approx C[R2]$.

Algumas propriedades importantes da relação $\approx^c$:

(1) $g.\tau.B \approx^c g.B$

(2) $B + \tau.B \approx^c \tau.B$

(3) $g.(B + \tau.C) + g.C \approx^c g.(B + \tau.C)$

onde $g \in A \cup \overline{A}$ e $B, C \in P$ (Figs. 4.12a - 4.12c)

Por exemplo, os agentes Q e Q' definidos por

$$Q \stackrel{def}{=} a.(b.NIL + b.NIL)$$

$$Q' \stackrel{def}{=} a.b.NIL$$

são congruentes quanto à observação ($Q \approx^c Q'$).

Prova. Usando a propriedade (2) da relação $\approx^c$,

$$Q \approx^c a.(\tau.b.NIL)$$

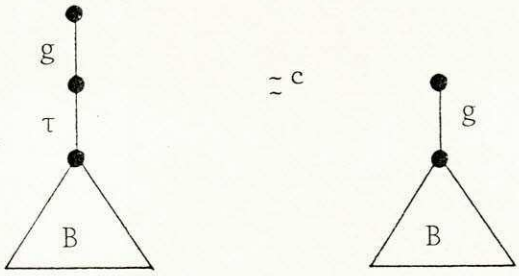
Eliminado os parênteses

$$Q \approx^c a.\tau.b.NIL$$

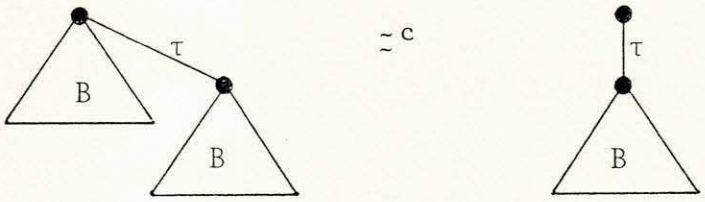
Usando a propriedade (1) da $\approx^c$,

$$Q \approx^c a.b.NIL$$

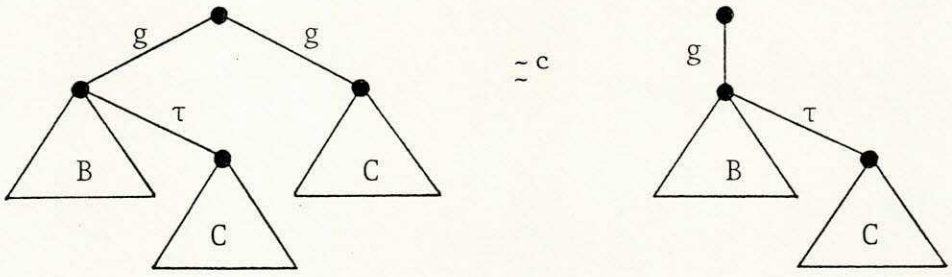
Portanto, $Q \approx^c Q'$



(a)



(b)



(c)

Figura 4.12 - Propriedades importantes da $\approx^c$.

4.6 Exemplo de um sistema distribuído

A Lei de Expansão e as relações de equivalência e congruência de observação, bem como as suas propriedades, e a bissimulação são as ferramentas utilizadas nos procedimentos de verificação da correção dos sistemas apresentados neste trabalho.

Para ilustrar a aplicação das propriedades da congruência de observação em uma situação mais complexa, na Fig. 4.13 é apresentado um sistema distribuído, constituído de um Produtor de Dados, um Consumidor de Dados e um Serviço de Comunicação Confiável. O Produtor e o Consumidor são usuários do Serviço de Comunicação.

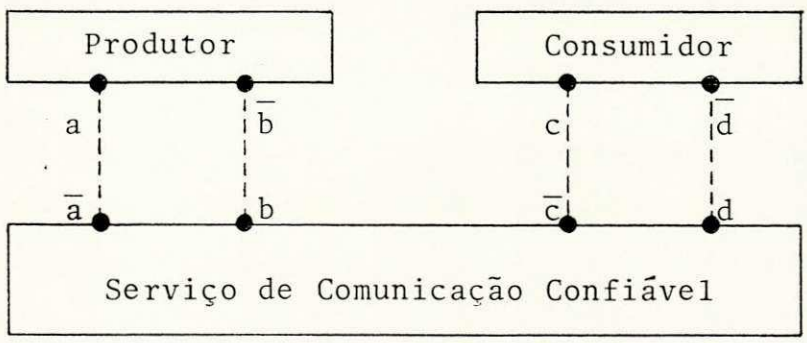


Figura 4.13 - Sistema Produtor-Consumidor.

O controle do fluxo de dados é exercido pelo Consumidor. O Produtor só envia um novo dado após ter recebido uma confirmação de que o dado anterior foi consumido (mecanismo explícito de controle de fluxo).

A descrição mais abstrata do Serviço de Comunicação (SC) pode ser realizada pela seguinte expressão de comportamento:

$$SC \stackrel{def}{=} b.\overline{c}.d.\overline{a}.SC$$

O detalhamento do Serviço de Comunicação (SC) pode ser realizado considerando três agentes: as entidades de protocolo E1 e E2 e o meio de

comunicação half-duplex confiável M (Fig. 4.14).

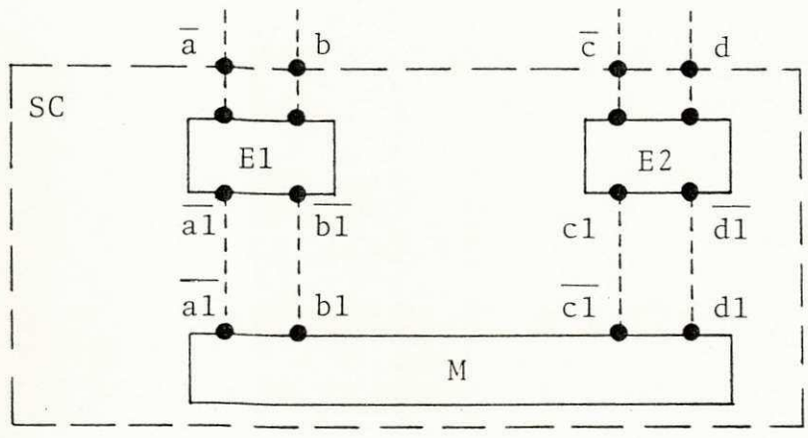


Figura 4.14 - Detalhamento de SC.

A entidade E1, relativa ao sítio do Produtor, comporta-se como

$$E1 \stackrel{def}{=} b.\overline{b1}.a1.\overline{a}.E1$$

A entidade E2, relativa ao sítio do Consumidor, comporta-se como

$$E2 \stackrel{def}{=} c1.\overline{c}.d.\overline{d1}.E2$$

O meio de comunicação M comporta-se como

$$M \stackrel{def}{=} b1.\overline{c1}.M + d1.\overline{a1}.M$$

Os agentes E1, E2 e M podem ser compostos, e as portas de intercomunicação (de nomes: a1, b1, c1 e d1) escondidas (\) do observador, obtendo-se o agente

$$SR = (E1|E2|M)\backslash\{a1,\ b1,\ c1,\ d1\}$$

Pode-se provar que SC $\approx^c$ SR.

Prova. Aplicando a Lei de Expansão a SR, obtém-se

$$SR = b.\tau.\tau.\bar{c}.d.\tau.\tau.\bar{a}.SR$$

Utilizando-se, repetidamente, a propriedade (1) da relação $\approx^c$, obtém-se o agente

$$SR' \stackrel{def}{=} b.\bar{c}.d.\bar{a}.SR'$$

tal que

$$SR \approx^c SR'$$

Como SR' e SM executam exatamente a mesma sequência infinita de eventos,

$$SR \approx^c SC$$

CAPÍTULO V

ABORDAGEM CLÁSSICA

O desenvolvimento de sistemas distribuídos pode ser realizado de acordo com uma trajetória de design (Fig. 5.1) que envolve a especificação do mesmo sistema em diversos níveis de abstração [PiLo 90].

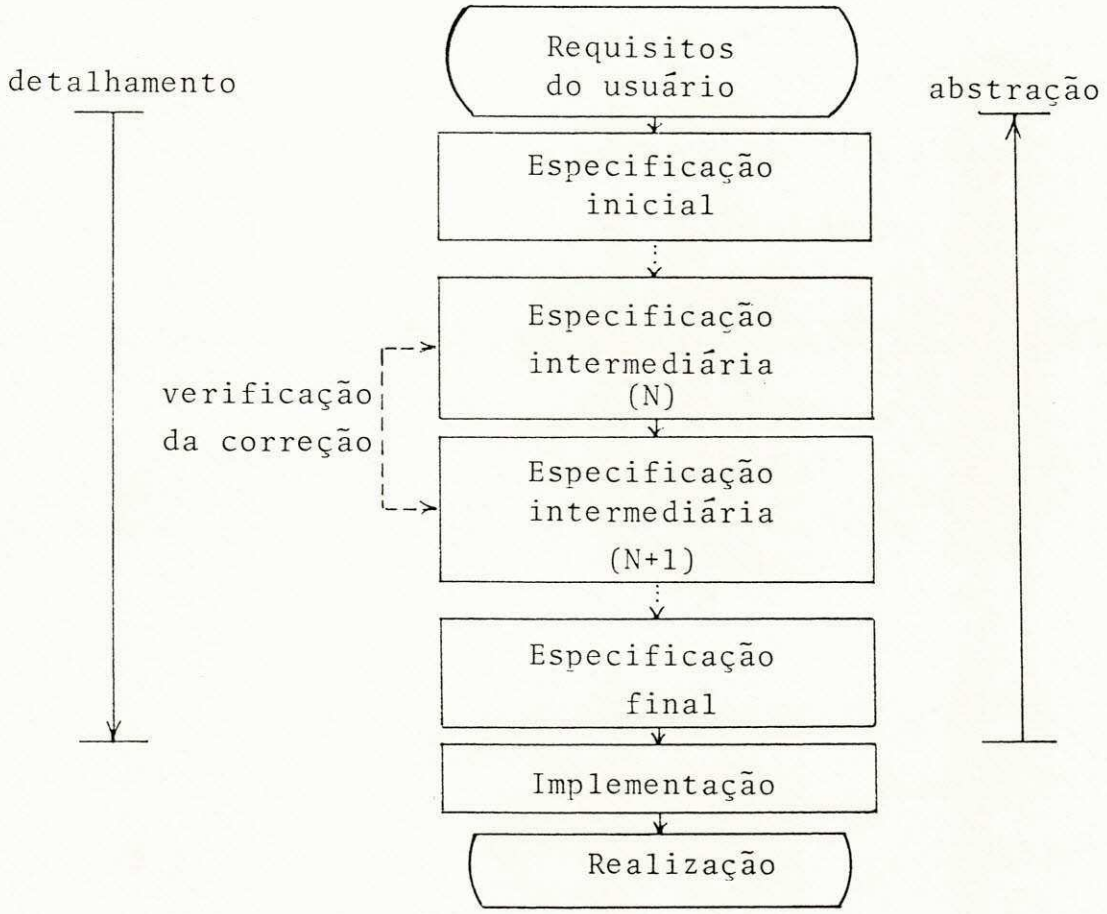


Figura 5.1 - Trajetória de design.

Os requisitos do usuário, que devem ser analisados e formalizados (possivelmente, de modo parcial), levam à especificação inicial no nível mais alto de abstração.

Partindo da especificação inicial, obtém-se uma série de especificações intermediárias, que descrevem o sistema de modo cada vez mais detalhado. Essas especificações são obtidas à medida que são incorporadas novas decisões de design, reduzindo o nível de abstração das descrições. Uma boa prática consiste em tomar uma pequena quantidade de decisões de design a cada vez.

Cada nova especificação é submetida à prova de correção. Tal prova é realizada para demonstrar que existe conformidade entre uma especificação no nível (N) e a especificação imediatamente mais abstrata, que se encontra no nível (N-1). Caso essa conformidade não se verifique, uma reavaliação das decisões tomadas no nível (N) deve ser realizada.

A sucessão de refinamentos é interrompida quando o projetista julgar suficiente o nível de detalhamento alcançado. Entretanto, essa especificação final deve ser ainda suficientemente abstrata, de modo a não limitar as escolhas do implementador. Essas escolhas podem envolver, por exemplo, decisões com relação à tecnologia a ser empregada na implementação (hardware, software ou firmware).

A partir da especificação final implementações podem ser obtidas. Em função da tecnologia escolhida para uma implementação em particular, deve-se proceder a um mapeamento dos elementos da especificação final nos elementos da implementação. Esse mapeamento pode ser realizado manualmente ou com o auxílio de ferramentas.

Finalmente a integração da implementação ao ambiente computacional para o qual esta foi destinada constitui a etapa de realização. Essa realização deve ser confrontada com relação à primeira etapa da trajetória de design, isto é, com relação aos requisitos do usuário.

A trajetória de design disciplina o trabalho do projetista na medida em que decisões de design são tomadas, justificadas e documentadas em especificações formais. Além disso, permite que os erros de design sejam detectados muito cedo e corrigidos antes que se propaguem até etapas avançadas

do design. Em última análise, a fidelidade à trajetória de design proposta contribui para a economia de tempo e esforço do projetista.

5.1. Refinamentos sucessivos e provas de correção utilizando CCS

A especificação inicial de um sistema distribuído pode ser a especificação do serviço desejado pelo usuário. CCS permite que tal especificação seja realizada, abstratamente, através da definição de um agente (uma caixa preta).

Para a realização de um refinamento o especificador deve, inicialmente, identificar as funções do agente a ser refinado. Tais funções, que podem ser executadas em paralelo ou sequencialmente, são então definidas isoladamente.

Cada função isolada (ou cada subconjunto delas) constitui uma parte do sistema refinado. Cabe ao especificador construir agentes que correspondam a cada uma dessas partes.

A especificação de um protocolo pode ser vista como um refinamento da especificação de serviço correspondente. Um tal refinamento consiste em substituir o agente provedor de serviço por uma combinação de novos agentes. Nesse caso o especificador deve identificar e isolar as funções executadas nos diversos sítios do sistema distribuído (funções locais), assim como as funções de comunicação entre os sítios (funções fim-a-fim). As primeiras resultam nas especificações das entidades de protocolo. As outras resultam na especificação do serviço subjacente utilizado por essas entidades.

Cada entidade de protocolo pode ser especificada como um agente CCS e o mesmo pode ser feito em relação ao serviço subjacente. Uma vez escondidos do observador os eventos de comunicação entre as entidades de protocolo e o provedor do serviço subjacente, a ação conjunta desses agentes deve ser equivalente quanto à observação ($\approx$) ao serviço desejado.

Os refinamentos seguintes são realizados para reduzir,

gradativamente, o nível de abstração das entidades de protocolo. Cada refinamento consiste na substituição de um agente CCS (uma caixa preta) por uma combinação de outros agentes CCS (outras caixas pretas), que devem apresentar o mesmo comportamento externo do agente substituído.

Tal sistemática prossegue até a obtenção da especificação final. Esta pode ser realizada (no todo ou em parte) através de agentes CCS previamente definidos.

5.2. Utilização de agentes básicos predefinidos

Em CCS pode-se definir um conjunto de agentes simples com um comportamento bem genérico. Esses agentes podem ser utilizados nas especificações de sistemas reais, desde que os nomes genéricos de suas portas de comunicação sejam substituídos (através de operações de rerrotulação [S]), pelos nomes reais.

Para permitir a ligação de n-uplas de variáveis ($\tilde{x}$) às portas receptoras (rótulos) e para permitir a ligação de n-uplas de expressões de valor ($\tilde{E}$) às portas oferecedoras (co-rótulos), são feitas extensões à notação usada em [Miln 80].

(a) função n-ária:

Função f $\stackrel{def}{=}$ $i\tilde{x}.\bar{o}(f(\tilde{x})).$(Função f)

Recebe valores para $\tilde{x}$ em i e calcula $f(\tilde{x})$ oferecendo-o em $\bar{o}$ (Fig. 5.2).

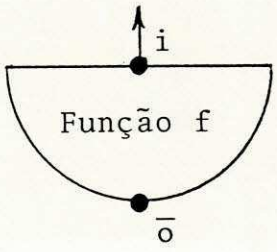


Figura 5.2 - O agente Função f.

(b) predicado n-ário:

Verificador p $\stackrel{def}{=}$ $i\tilde{x}$. if $p(\tilde{x})$ then $\bar{o}_1\tilde{x}$. (Verificador p)

else $\bar{o}_2\tilde{x}$. (Verificador p)

O valor recebido na entrada é transmitido em uma das saídas ($\bar{o}_1$ ou $\bar{o}_2$) de acordo com a avaliação do predicado booleano $p(\tilde{x})$ (Fig. 5.3).

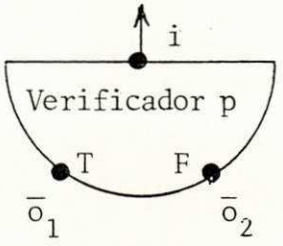


Figura 5.3 - O agente Verificador p.

(c) comporta:

Comporta $\stackrel{def}{=}$ $i\tilde{x}.\bar{\tilde{o}}\tilde{x}$. c. Comporta

Recebe uma n-upla de valores em i, transmite-a em $\bar{o}$ e espera pela sincronização em c para ser reativado (Fig. 5.4).

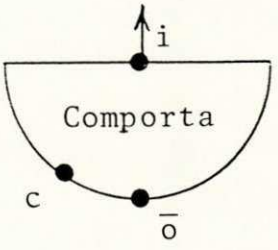


Figura 5.4 - O agente Comporta.

(d) Fonte:

Constante $\tilde{v} \stackrel{def}{=} i.\overline{o}\tilde{v}.(\text{Constante } \tilde{v})$

Gera uma n-upla de valores constantes $\tilde{v}$ após uma sincronização em i (Fig. 5.5).

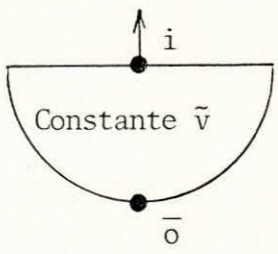


Figura 5.5 - O agente Constante $\tilde{v}$.

(e) ciclo:

Ciclo $\stackrel{def}{=} a\tilde{x}.b\tilde{x}.\overline{c}\tilde{v}.\text{Ciclo}$

Comporta-se ciclicamente, participando de uma sequência fixa de eventos (Fig. 5.6).

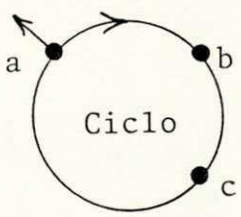


Figura 5.6 - O agente Ciclo.

(f) sorvedouro

Sorvedouro $\stackrel{def}{=}$ $i\tilde{x}$. Sorvedouro

Recebe uma n-upla de valores em i e consome-a (Fig. 5.7).

Figura 5.7 - O agente Sorvedouro.

(g) chave:

Chave $\stackrel{def}{=}$ $i\tilde{x}.(c_1.\bar{o}_1\tilde{x}.\text{Chave} + c_2.\bar{o}_2\tilde{x}.\text{Chave})$

Recebe uma n-upla de valores em i e espera por uma sincronização que pode ocorrer, indeterministicamente, em c_1 ou c_2. Se a primeira alternativa ocorre, oferece a n-upla de valores em $\bar{o}_1$. Se a segunda alternativa ocorre, ofereca a n-upla de valores em $\bar{o}_2$. Em qualquer caso, após a entrega da n-upla de valores, volta recursivamente (Fig. 5.8).

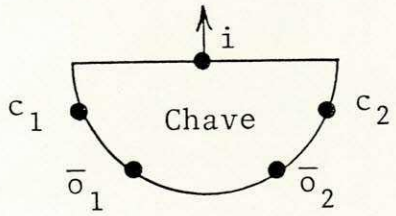


Figura 5.8 - O agente Chave.

5.3. Aplicação: um sistema de acesso por exclusão mútua

A trajetória de design, apresentada na seção 5.1, é adotada como guia para o desenvolvimento de um sistema de acesso por exclusão mútua, que coordena o acesso de n usuários a um recurso compartilhado (e.g., uma impressora, um disco, etc.).

O sistema a ser desenvolvido é definido em [BoVe 87]. Uma versão em CCS é apresentada em [LoRi 88]. Diferentemente de [BoVe 87], em [LoRi 88] e neste capítulo não são consideradas possibilidades de falhas do sistema.

Os requisitos do usuário estabelecem que os n usuários do sistema de acesso por exclusão mútua estão situados em diferentes sítios (sistema distribuído). A utilização do recurso pelos n usuários se dá de modo "justo" (Fig. 5.9).

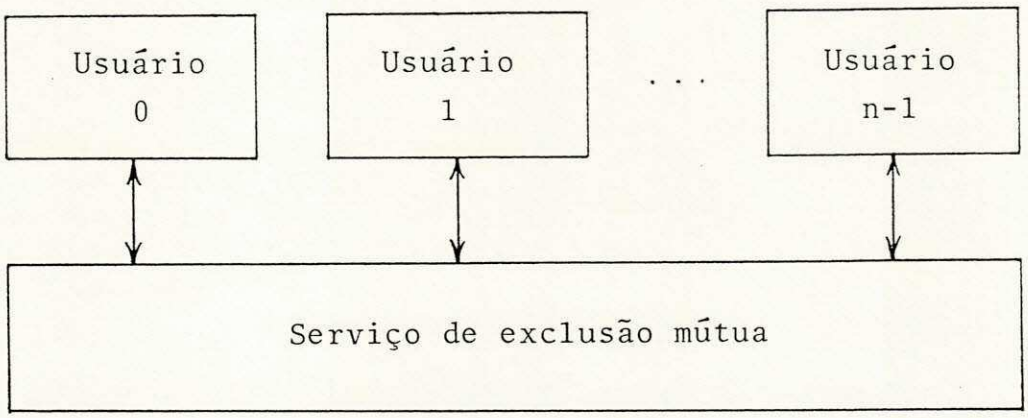


Figura 5.9 - Sistema de acesso por exclusão mútua.

Os usuários desejam um serviço de exclusão mútua. Considerando que o serviço disponível é um anel virtual, que permite simplesmente uma troca confiável de primitivas, é necessário que cada usuário seja conectado ao anel através de um controlador de exclusão mútua (Controlador_EM), cuja função é verificar se o seu usuário pode (ou não) ter acesso ao recurso compartilhado (Fig. 5.10).

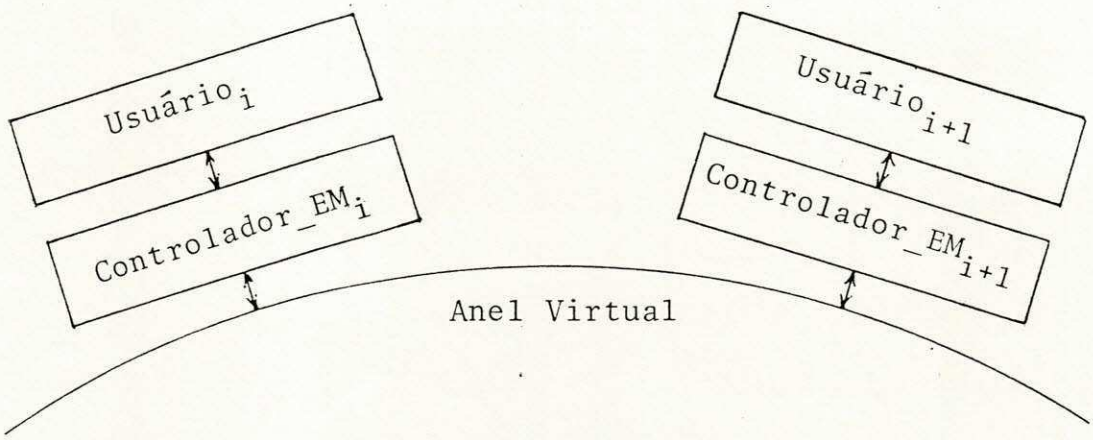


Figura 5.10 - Arquitetura do sistema.

Os controladores utilizam o serviço de anel virtual para oferecerem aos usuários um serviço de comunicação com o mesmo grau de transparência do serviço desejado. Portanto esses controladores podem ser vistos como entidades de protocolo, que podem ser obtidas a partir do refinamento do serviço de exclusão mútua (Fig. 5.11).

Neste capítulo a atenção estará concentrada no comportamento do Controlador_EM (referido simplesmente, como controlador), pois este representa o protocolo de exclusão mútua. Cada controlador decide se o seu usuário pode (ou não) ter acesso ao recurso compartilhado (o privilégio) de acordo com o estado (variável e_e) do seu vizinho à esquerda e de acordo com o seu próprio

estado (variável m_e). Após usar (ou não) o privilégio há uma troca de estado em m_e, para dar oportunidade aos demais usuários de terem acesso ao privilégio.

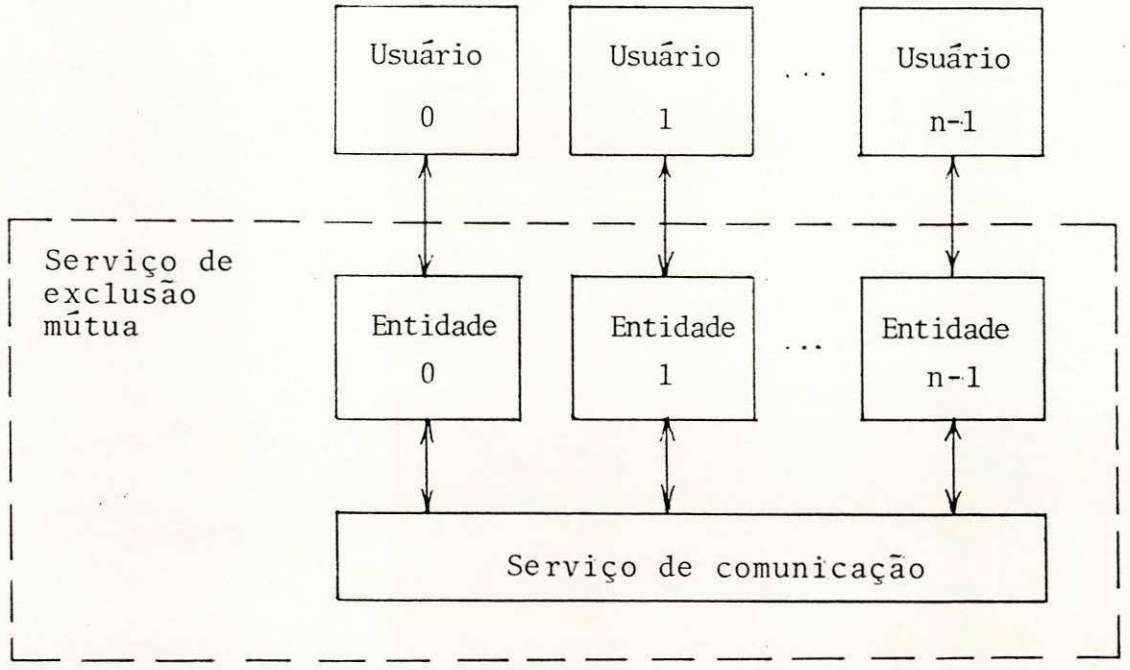


Figura 5.11 - Refinamento do serviço de exclusão mútua.

Para solicitar o estado do seu vizinho à esquerda, o controlador usa uma primitiva S_REQ, que resulta em uma primitiva S_IND para esse vizinho. Este, por sua vez, responde com uma primitiva S_RESP (acompanhada do estado solicitado) que resulta em uma primitiva S_CONF para o controlador solicitante.

O controlador pode ser especificado, no nível mais alto de abstração, como o agente Controlador_EM mostrado na Fig. 5.12. Esse agente possui uma porta oferecedora de eventos $\bar{a}$ e três portas receptoras b, c e d.

A porta $\bar{a}$ é usada para o envio das primitivas S_REQ (para o vizinho da esquerda) e S_RESP (para o vizinho da direita). A porta b é usada para a recepção das primitivas S_CONF (proveniente do vizinho da esquerda) e S_IND

(proveniente do vizinho da direita). A porta c é usada, exclusivamente, no momento da ativação do controlador e não volta a ser usada durante o funcionamento normal deste. Finalmente, a porta d é usada para as indicações de posse e liberação do privilégio.

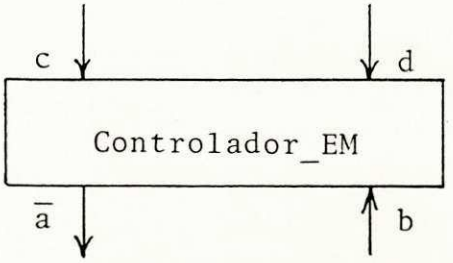


Figura 5.12 - O agente controlador de exclusão mútua.

O comportamento do Controlador_EM é dado pelas expressões (eq. 5.1 - eq. 5.4).

$$\text{Controlador\_EM} \stackrel{\text{def}}{=} c.\text{Control\_EM(E\_I)} \qquad \text{(eq. 5.1)}$$

onde a (eq. 5.1) descreve a ativação do controlador, como consequência da sincronização que ocorre entre ele e o seu usuário, nas portas c e $\bar{c}$, respectivamente. Ao ser ativado, o Controlador_EM passa a comportar-se como Control_EM(m_e) onde m_e assume o valor E_I (estado inicial).

$$\text{Control\_EM(m\_e)} \stackrel{\text{def}}{=} \bar{a}\text{S\_REQ.Control\_esp} \qquad \text{(eq. 5.2)}$$

onde a primitiva S_REQ (solicitação do estado do vizinho à esquerda) é oferecida na porta $\bar{a}$. Após a ocorrência dessa primitiva, o controlador passa a

comportar-se como Control_esp.

$$\text{Control\_esp} \stackrel{def}{=} \text{bprim e\_e.}\underline{\text{if}}\ \text{conf(prim)}\ \underline{\text{then}}\ \text{Control\_priv}\ \underline{\text{else}}\ \overline{a}\text{S\_RESP m\_e.}\ \text{Control\_esp} \qquad \text{(eq. 5.3)}$$

A expressão de comportamento (eq. 5.3) indica que o controlador pode receber (na porta b) duas expressões de valor compatíveis com os tipos de prim e de e_e (primitiva e estado, respectivamente). É oportuno mencionar que 0 é aceitável para e_e.

A variável prim pode ter os seguintes valores:

(a) S_CONF, que é a confirmação do vizinho à esquerda à primitiva S_REQ. A S_CONF deve ser acompanhada do estado E_E (um valor a ser assumido pela variável e_e);

(b) S_IND (acompanhada de 0) é a indicação de uma primitiva S_REQ enviada pelo vizinho da direita.

O predicado booleano conf(prim) avalia prim. Se ocorre uma S_IND então o controlador oferece a primitiva S_RESP, com o valor corrente de m_e (na porta $\overline{a}$), e volta recursivamente como Control_esp. Caso contrário ele comporta-se como

$$\text{Control-priv} \stackrel{def}{=} \underline{\text{if}}\ \text{priv(m\_e,e\_e)}\ \underline{\text{then}}\ \text{d.d.Control\_EM(N\_E)} + \tau\text{.Control\_EM(N\_E)}\ \underline{\text{else}}\ \text{Control\_EM(m\_e)} \qquad \text{(eq. 5.4)}$$

onde priv(m_e,e_e) é um predicado booleano que verifica se o controlador pode

volta a comportar-se como Control_EM(m_e) (eq. 5.2), mantendo o valor corrente de m_e. Caso contrário, dá-se uma escolha indeterminística entre duas alternativas:

(a) o usuário solicita o recurso através de uma sincronização ($\bar{d}$ para o usuário e d para o controlador), liberando-o com uma sincronização semelhante;

(b) o usuário não solicita o recurso e o controlador evolui sem a ocorrência de eventos externos.

Em ambos os casos o controlador volta a comportar-se de acordo com a (eq. 5.2), mas com o novo valor N_E (novo estado) para o seu estado.

5.3.1. Refinamento do agente Controlador_EM

Dois tipos de funções podem ser identificados no controlador: funções relacionadas ao armazenamento de informações (memória), para guardar o estado corrente do controlador, e funções relacionadas ao controle da comunicação, para realizar as sincronizações com o usuário (na interface superior) e a troca de primitivas com o provedor do serviço de anel virtual (na interface inferior).

As funções da memória podem ser realizadas pelo agente Mem. As funções do controle podem ser realizadas pela ação conjunta de vários agentes: Constante E_I para ativar o controlador e gerar o seu estado inicial, Comporta 1 para gravar em Mem o estado inicial e para ativar o agente Cont, este para oferecer a primitiva S_REQ e reconhecer a primitiva (S_CONF ou S_IND) recebida do anel virtual, Ct_conf para realizar o comportamento posterior à recepção de uma primitiva S_CONF e Ct_ind para realizar o comportamento posterior à recepção de uma primitiva S_IND (Fig. 5.13).

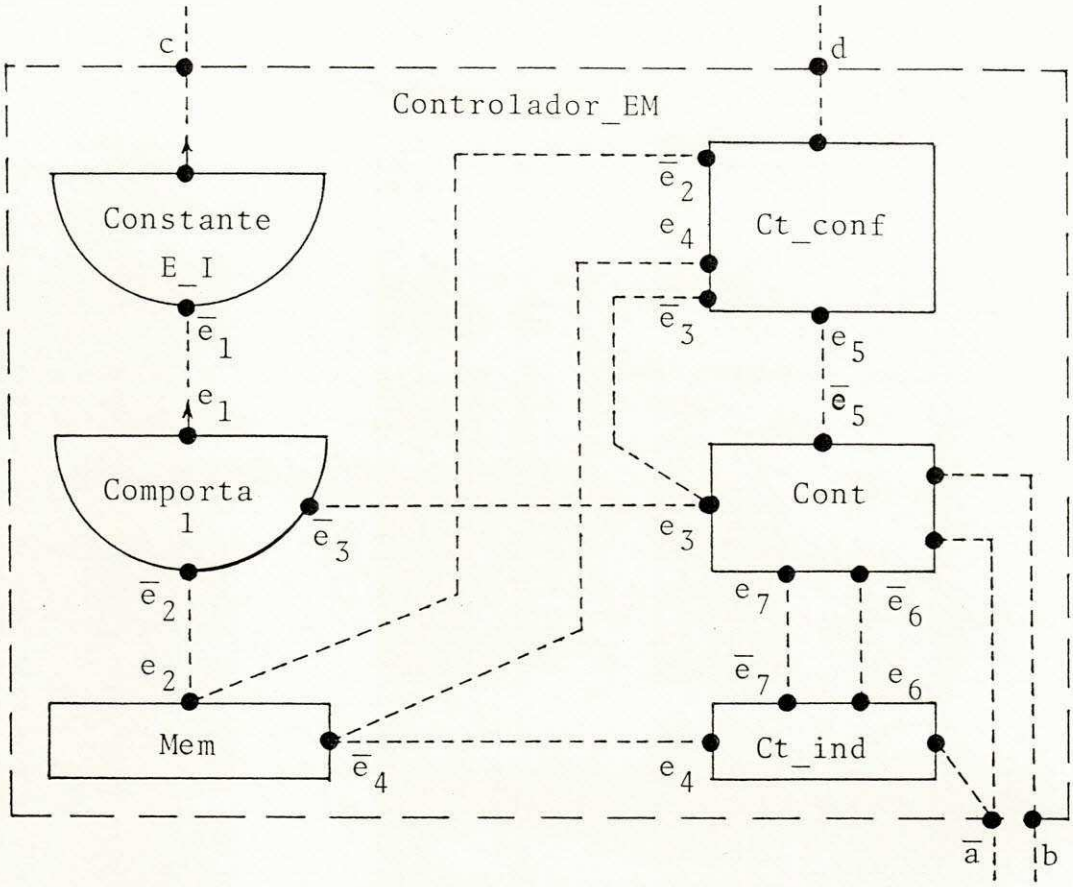


Figura 5.13 - Refinamento do agente Controlador_EM.

As expressões de comportamento desses agentes são:

$$\text{Constante E\_I} \stackrel{def}{=} c.\bar{e}_1 \text{E\_I.NIL} \qquad \text{(eq. 5.5)}$$

onde Constante E_I pode sincronizar-se com o usuário em c, oferecer o estado inicial do controlador em $\bar{e}_1$ e, após esse evento, tornar-se inativo.

$$\text{Comporta 1} = \text{Comporta}[i\bar{o}\ \bar{c}/e_1\bar{e}_2\bar{e}_3] \qquad \text{(eq. 5.6)}$$

onde Comporta 1 pode receber (na porta e_1) o estado inicial do controlador, oferecê-lo (na porta $\bar{e}_2$) e ativar o agente Cont com um evento (na porta $\bar{e}_3$). Comporta 1 não participa de mais eventos no funcionamento normal do controlador.

$$Mem \stackrel{def}{=} e_2 m\_e.Cel(m\_e) \qquad \text{(eq. 5.7)}$$

onde Mem pode receber (na porta e_2) e guardar (em m_e) o estado inicial do controlador. Após esse evento, comporta-se como Cel(m_e).

$$Cel(v) \stackrel{def}{=} e_2 m\_e.Cel(m\_e) + \bar{e}_4 v.Cel(v) \qquad \text{(eq. 5.8)}$$

onde Cel(v) apresenta um comportamento indeterminístico: na primeira alternativa pode receber (na porta e_2) um valor para a variável m_e e voltar recursivamente, na segunda alternativa pode entregar (na porta $\bar{e}_4$) o estado corrente do controlador e voltar recursivamente.

$$Cont \stackrel{def}{=} e_3.\bar{a}S\_REQ.Cont\_esp + e_7.Cont\_esp \qquad \text{(eq. 5.9)}$$

onde Cont apresenta um comportamento indeterminístico: na primeira alternativa ele pode ser ativado (inicialmente por Comporta 1 e, durante o funcionamento normal do controlador, por Ct_conf) com um evento na porta e_3, enviar a primitiva S_REQ e passar a comportar-se como Cont_esp. Na segunda alternativa Cont pode ser ativado pelo agente Ct_ind, com um evento na porta e_7, e passar a comportar-se como Cont_esp.

$$Cont\_esp \stackrel{def}{=} bprim\ e\_e.\underline{if}\ conf(prim)\ \underline{then}\ \bar{e}_5 e\_e.Cont\ \underline{else}\ \bar{e}_6.Cont \qquad \text{(eq. 5.10)}$$

onde Cont_esp pode receber e fazer o reconhecimento da primitiva recebida do anel virtual (na porta b). Se é uma S_CONF o estado E_E é oferecido (na porta $\bar{e}_5$) a Ct_conf. Caso contrário (S_IND) é estabelecida uma sincronização (na porta $\bar{e}_6$) com o agente Ct-ind. Em ambos os casos o agente Cont_esp passa a comportar-se como o agente Cont, que irá sincronizar-se com Ct_conf ou com Ct_ind.

$$Ct\_conf \stackrel{def}{=} e_5 e\_e . e_4 m\_e .$$
$$\underline{if}\ priv(m\_e, e\_e)\ \underline{then}\ d.d.\bar{e}_2 N\_E.\bar{e}_3 . Ct\_conf$$
$$+$$
$$\tau.\bar{e}_2 N\_E.\bar{e}_3 . Ct\_conf$$
$$\underline{else}\ \bar{e}_3 . Ct\_conf \qquad \text{(eq. 5.11)}$$

onde Ct_conf combina E_E e M_E (este, obtido na Mem) e decide se pode (ou não) ceder o privilégio ao seu usuário. Se isso for possível, Ct_conf oferecerá um N_E a Mem mesmo que o privilégio não seja solicitado. A última tarefa de um ciclo de Ct_conf é sincronizar-se com Cont.

$$Ct\_ind \stackrel{def}{=} e_6 . e_4 m\_e . \bar{a} S\_RESP\ m\_e . \bar{e}_7 . Ct\_ind \qquad \text{(eq. 5.12)}$$

onde Ct_ind envia a primitiva S_RESP, acompanhada por M_E (obtido na Mem), para o anel virtual. A última tarefa de um ciclo Ct-ind é sincronizar-se com Cont.

A composição dos agentes especificados acima (eq. 5.5 - eq. 5.12) define um novo sistema (Cont_EM), cuja especificação é dada por

$$Cont\_EM = (Constante\ E\_I \mid Comporta\ 1 \mid Mem \mid Cont \mid Ct\_conf \mid Ct\_ind)$$

Para $L = \{e_1, \bar{e}_1, \ldots, e_7, \bar{e}_7\}$ e para

$$Controlador\_EM' = Cont\_EM \backslash L$$

na seção seguinte prova-se que

$$Controlador\_EM \approx Controlador\_EM'$$

5.3.2. Equivalência de observação entre Controlador_EM e Controlador_EM'

Usando a Lei de Expansão e as propriedades das relações de equivalência forte (~), equivalência de observação (≈) e congruência de observação ($\approx^c$), pode-se provar que Controlador_EM ≈ Controlador_EM'.

Prova. Substituindo na definição do Controlador_EM' os identificadores Constante E_I, Comporta 1, Mem, Cont, Ct_conf e Ct_ind pelas respectivas expressões de comportamento, obtém-se

$$
\begin{aligned}
\text{Controlador\_EM'} \sim\ & (c.\overline{e}_1 E\_I.NIL \mid e_1 x.\overline{e}_2 x.\overline{e}_3.\text{Comporta 1} \mid \\
& e_2 m\_e.Cel(m\_e) \mid \\
& e_3.\overline{a}S\_REQ.\text{Cont-esp} + e_7.Cont\_esp \mid \\
& e_5 e\_e.e_4 m\_e. \\
& \underline{\text{if}}\ priv(m\_e, e\_e)\ \underline{\text{then}}\ d.d.\overline{e}_2 N\_E.\overline{e}_3.Ct\_conf \\
& \qquad\qquad + \\
& \qquad\qquad \tau.\overline{e}_2 N\_E.\overline{e}_3.Ct\_conf \\
& \qquad \underline{\text{else}}\ \overline{e}_3.Ct\_conf \mid \\
& e_6.e_4 m\_e.\overline{a}S\_RESP\ m\_e.\overline{e}_7.Ct\_ind)\backslash L
\end{aligned}
$$

De acordo com a Lei de Expansão, na inicialização,

$$
\begin{aligned}
\text{Controlador\_EM'} \sim\ & c.(\overline{e}_1 E\_I.NIL \mid e_1 x.\overline{e}_2 x.\overline{e}_3.\text{Comporta 1} \mid \\
& e_2 m\_e.Cel(m\_e) \mid \\
& e_3.\overline{a}S\_REQ.Cont\_esp + e_7.Cont\_esp \mid \\
& e_5 e\_e.e_4 m\_e. \\
& \underline{\text{if}}\ priv(m\_e, e\_e)\ \underline{\text{then}}\ d.d.\overline{e}_2 N\_E.\overline{e}_3.Ct\_conf \\
& \qquad\qquad + \\
& \qquad\qquad \tau.\overline{e}_2 N\_E.\overline{e}_3.Ct\_conf \\
& \qquad \underline{\text{else}}\ \overline{e}_3.Ct\_conf \mid \\
& e_6.e_4 m\_e.\overline{a}S\text{-}RESP\ m\_e.\overline{e}_7.Ct\_ind)\backslash L
\end{aligned}
$$

onde o evento na porta c indica a ativação do controlador realizada pelo seu

usuário.

De acordo com a Lei de Expansão,

$$\begin{aligned}\text{Controlador\_EM'} \sim c.\tau_1.(&NIL \mid \bar{e}_2 E\_I.\bar{e}_3.\text{Comporta 1} \mid e_2 m\_e.\text{Cel}(m\_e) \mid \\ &e_3.\bar{a}S\_REQ.\text{Cont\_esp} + e_7.\text{Cont\_esp} \mid \\ &e_5 e\_e.e_4 m\_e. \\ &\underline{\text{if}}\ \text{priv}(m\_e, e\_e)\ \underline{\text{then}}\ d.d.\bar{e}_2 N\_E.\bar{e}_3.\text{Ct\_conf} \\ &\qquad + \\ &\qquad \tau.\bar{e}_2 N\_E.\bar{e}_3.\text{Ct\_conf} \\ &\qquad \underline{\text{else}}\ \bar{e}_3.\text{Ct\_conf} \mid \\ &e_6.e_4 m\_e.\bar{a}S\text{-}RESP\ m\_e.\bar{e}_7.\text{Ct\_ind})\backslash L\end{aligned}$$

onde τ_1 indica a ocorrência de um evento interno na porta e_1 (o estado inicial E_I é gerado pelo agente Constante E_I e entregue ao agente Comporta 1). Daqui em diante τ_i representa um evento interno envolvendo as portas com nome e_i.

De acordo com a Lei de Expansão,

$$\text{Controlador\_EM'} \sim c.\tau_1.\tau_2.\tau_3.\text{Control\_EM'}(E\_I)$$

Com o evento τ_2 o estado inicial é guardado em Mem e com o evento τ_3 o agente Cont é ativado.

Usando a propriedade 1 da relação $\approx^c$ (seção 4.5.5), que permite a absorção de τ's,

$$\text{Controlador\_EM'} \approx^c c.\text{Control\_EM'}(E\_I)$$

que, comparando com a (eq. 5.1),

$$\text{Controlador\_EM} \approx \text{Controlador\_EM'}$$

se e somente se

$$Control\_EM(m\_e) \approx Control\_EM'(m\_e)$$

onde

$$Control\_EM'(m\_e) = (Cel(m\_e) \mid Ct\_conf \mid Ct\_ind \mid \overline{a}S\_REQ.Cont\_esp)\backslash L$$

Para evitar a repetição das expressões de comportamento dos componentes dos agentes compostos, a partir daqui serão apresentados, de preferência, os identificadores correspondentes.

A comparação entre o comportamento do Controlador_EM e o comportamento do Controlador_EM' também pode ser feita através de suas respectivas árvores de comunicação (Fig. 5.14). Pode-se verificar como a supressão de τ_i, i = 1, 2 e 3 (na árvore de comunicação correspondente ao Controlador_EM') resulta em árvores com os mesmos eventos para os dois agentes.

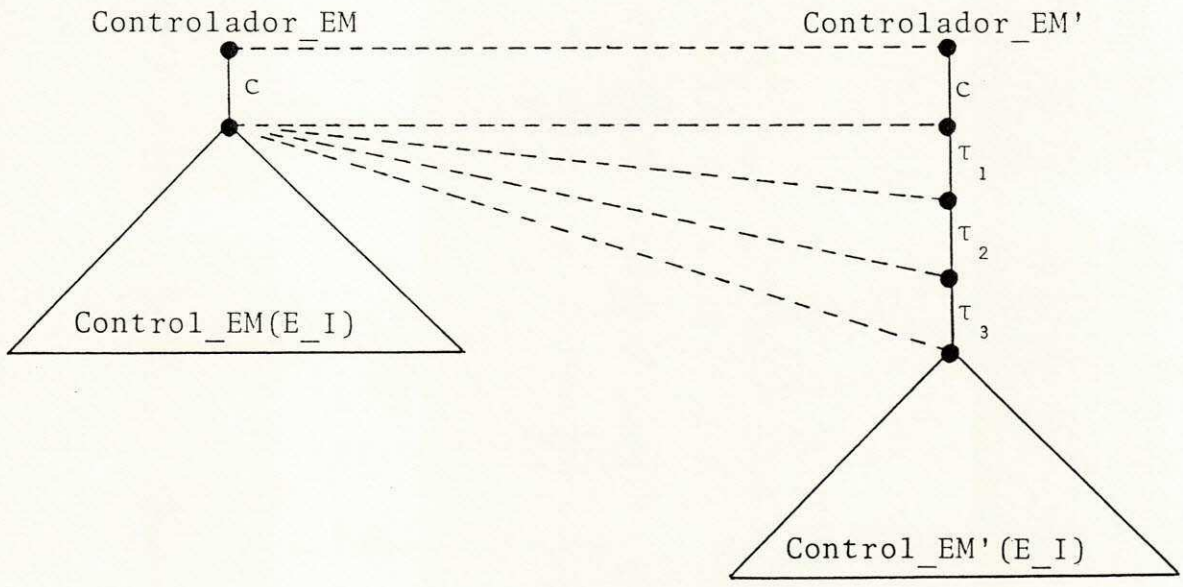


Figura 5.14 - Correspondência entre os estados de Controlador_EM e Controlador_EM'.

De acordo com a Lei de Expansão,

$$Control\_EM'(m\_e) \sim \bar{a}S\_REQ.Control\_esp'$$

que, comparando com a (eq. 5.2),

$$Control\_EM(m\_e) \approx Control\_EM'(m\_e)$$

se e somente se

$$Control\_esp \approx Control\_esp'$$

onde

$$Control\_esp' = (Cel(m\_e)|Ct\_conf|Ct\_ind|Cont\_esp)\backslash L$$

A comparação entre o comportamento de Control_EM(m_e) e o comportamento de Control_EM'(m_e) também pode ser feita através de suas respectivas árvores de comunicação (Fig. 5.15). Nesse caso, não há eventos internos a suprimir e ambas apresentam exatamente o mesmo evento.

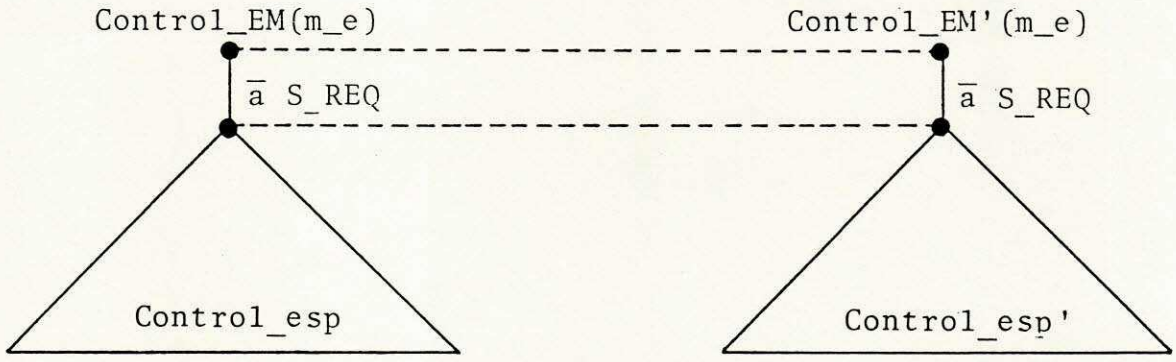


Figura 5.15 - Correspondência entre os estados de Control_EM(m_e) e Control_EM'(m_e).

Após o envio da primitiva S_REQ, o Controlador_EM' passa a esperar a ocorrência de uma primitiva. Prosseguindo com o próximo evento possível,

$$\begin{aligned} \text{Control\_esp'} \sim \text{bprim } e\_e.(\text{Cel}(m\_e) \mid \text{Ct\_conf} \mid \text{Ct\_ind} \mid \\ \underline{\text{if}}\ \text{conf(prim)}\ \underline{\text{then}}\ \bar{e}_5 e\_e.\text{Cont} \\ \underline{\text{else}}\ \bar{e}_6.\text{Cont}) \backslash L \end{aligned}$$

Uma primitiva ocorre na porta b e passa a ser examinada pelo agente Cont.

Se conf(prim) = falso, a primitiva é uma S_IND. Nesse caso

$$\text{Control\_esp'} \sim \text{bprim } e\_e.\tau_6.\tau_4.\bar{a}\ \text{S\_RESP}m\_e.\tau_7.\text{Control\_esp'}$$

Com o evento τ_6 o agente Cont ativa o agente Ct_ind. Com o evento τ_4 o agente Ct_ind lê o estado corrente do Controlador_EM' em Mem. Após isso, uma primitiva S_RESP é enviada e τ_7 ocorre quando o agente Ct_ind devolve o controle ao agente Cont.

Caso contrário, isto é, conf(prim) = verdadeiro, a primitiva é uma S_CONF. Nesse outro caso

$$\text{Control\_esp'} \sim \text{bprim } e\_e.\tau_5.\tau_4.\text{Control\_priv'}$$

Com o evento τ_5 o agente Cont ativa o agente Ct_conf. Com o evento τ_4 o agente Ct_conf lê o estado corrente do Controlador_EM' em Mem. Após isso, o Controlador_EM' passa a comportar-se como Control_priv'.

Absorvendo os τ's em ambas as expressões de Control_esp', e comparando com a (eq. 5.3),

$$\text{Control\_esp} \approx \text{Control\_esp'}$$

se e somente se

$$Control\_priv \approx Control\_priv'$$

onde

$$Control\_priv' = (Cel(m\_e) \mid Ct\_ind \mid Cont \mid$$
$$\underline{if}\ priv(m\_e, e\_e)\ \underline{then}\ d.d.\bar{e}_2 N\_E.\bar{e}_3.Ct\_conf$$
$$+$$
$$\tau.\bar{e}_2 N\_E.\bar{e}_3.Ct\_conf$$
$$\underline{else}\ \bar{e}_3.Ct\_conf) \backslash L$$

A comparação entre o comportamento de Control_esp e o comportamento de Control_esp' também pode ser feita através de suas respectivas árvores de comunicação (Fig. 5.16). Pode-se verificar como a supressão de τ_i, i = 4, 5, 6 e 7 (na árvore de comunicação correspondente a Control_esp') resulta em árvores com os mesmos eventos para os dois agentes.

Comportando-se como Control_priv', o Controlador_EM' compara o seu próprio estado com o estado do vizinho à esquerda (o agente Ct_conf faz tal comparação utilizando o predicado priv e os valores das variáveis m_e e e_e).

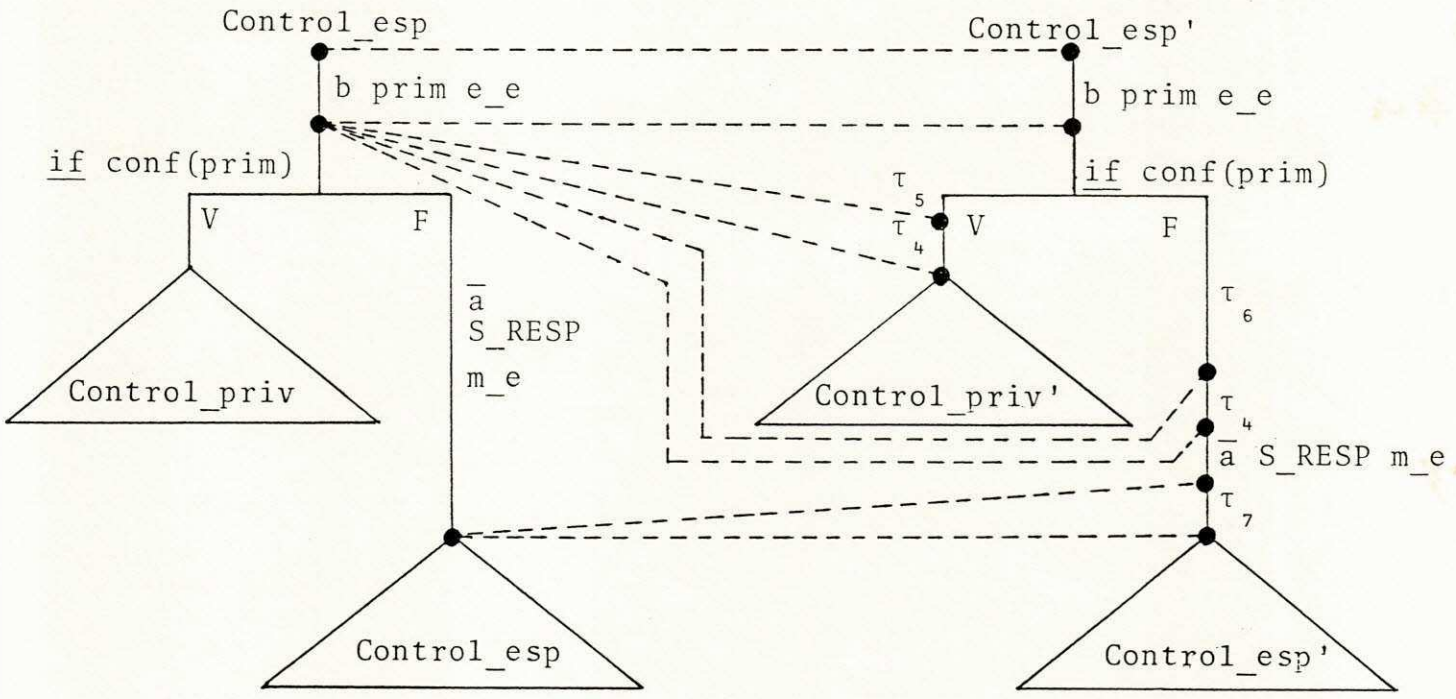


Figura 5.16 - Correspondência entre os estados de Control_esp e Control_esp'.

Se priv(m_e,e_e) = falso o usuário não pode ter acesso ao recurso compartilhado. Nesse caso,

$$Control\_priv' \sim \tau_3.Control\_EM'(m\_e)$$

Com o evento τ_3 o agente Ct_conf devolve o controle ao agente Cont e o Controlador_EM' volta a comportar-se como Control_EM'(m_e).

Caso contrário, isto é, priv(m_e,e_e) = verdadeiro o usuário pode ter acesso ao recurso compartilhado. Nesse outro caso dá-se uma escolha indeterminística entre duas alternativas:

(a) supondo que o usuário não solicita o recurso compartilhado em tempo hábil, ocorre a segunda alternativa da escolha indeterminística.

$$Control\_priv' \sim \tau.\tau_2.\tau_3.Control\_EM'(N\_E)$$

Com o evento τ o agente Cont_conf evolui para uma situação em que o usuário não pode mais solicitar o recurso. Com o evento τ_2 o agente Ct_conf envia o novo estado do Controlador_EM' para Mem. Com o evento τ_3 o agente Ct_conf devolve o controle para o agente Cont. A partir daí o Controlador_EM' volta a comportar-se como Control_EM' mas com um novo estado (N_E).

(b) supondo que o usuário obtém o recurso compartilhado, ocorre a primeira alternativa da escolha indeterminística.

$$Control\_priv' \sim d.d.\tau_2.\tau_3.Control\_EM'(N\_E)$$

Com o primeiro evento na porta d, o usuário obtém o recurso compartilhado e com o segundo evento na mesma porta, libera-o. Após a

atualização do estado do Controlador_EM' e a devolução do controle para o agente Cont, o Controlador_EM' volta a comportar-se como Control_EM' mas com um novo estado (N_E).

Absorvendo as ocorrências de τ_2 e τ_3 nas três expressões de Control_priv' apresentados acima e comparando com a (eq. 5.4),

$$\text{Control\_priv} \approx \text{Control\_priv'}$$

se e somente se

$$\text{Control\_EM(m\_e)} \approx \text{Control\_EM'(m\_e)}$$

A comparação entre o comportamento de Control_priv e Control_priv' também pode ser feita através de suas respectivas árvores de comunicação (Fig. 5.17). Pode-se verificar como a supressão de τ_i, i = 2 e 3 (na árvore de comunicação correspondente a Control_priv') resulta em árvores de comunicação com os mesmos eventos para os dois agentes.

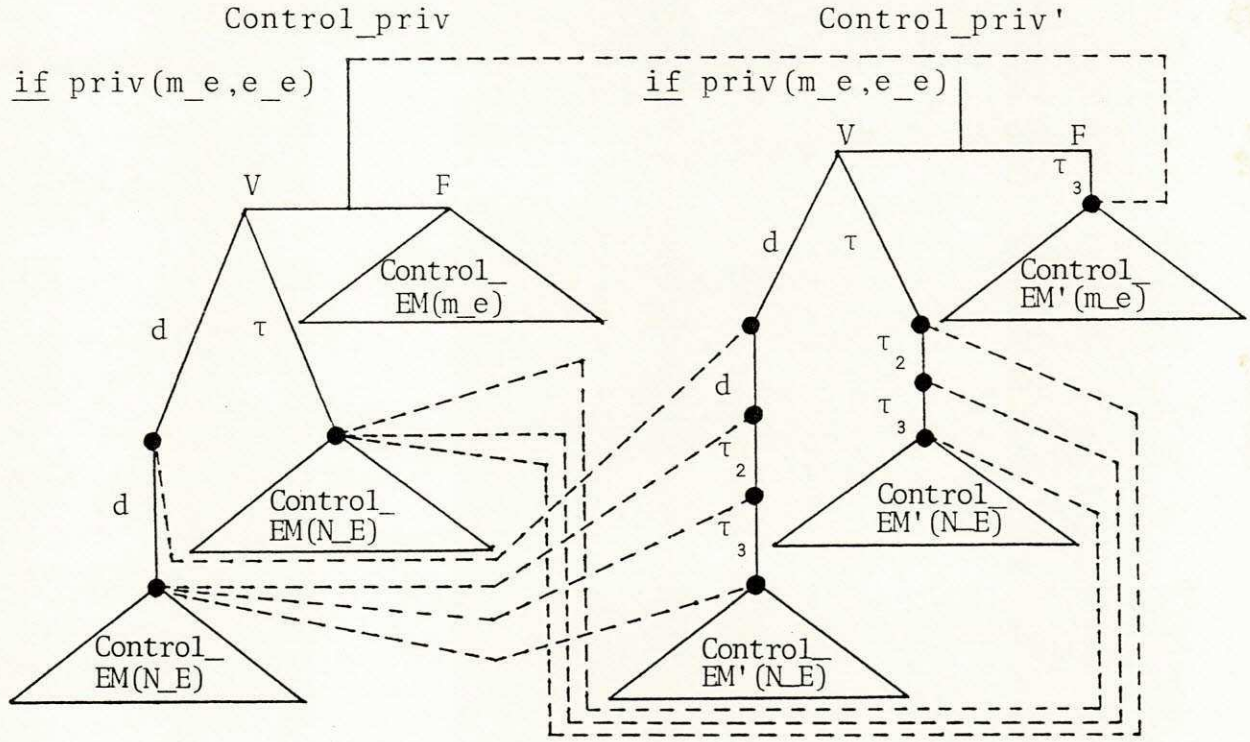


Figura 5.17 - Correspondência entre os estados de Control_priv e Control_priv'.

O Controlador_EM' completa um ciclo de atividades ao retomar o seu comportamento como Control_EM'. Durante esse ciclo, ele comporta-se de modo equivalente quanto à observação relativamente ao Controlador_EM. Por indução, resulta que ambos permanecem equivalentes quanto à observação durante os ciclos seguintes, isto é,

$$\text{Controlador\_EM} \approx \text{Controlador\_EM'}$$

como se desejava provar.

5.3.3. Refinamento passo-a-passo do agente Ct_conf

Uma vez que o agente Ct_conf apresenta o comportamento mais interessante do refinamento do Controlador_EM, ele é o escolhido para a continuação do processo de refinamentos sucessivos.

Na Fig. 5.18 todos os agentes são predefinidos, com exceção do agente Privilégio. Este agente está diretamente relacionado com o uso do privilégio.

$$\begin{aligned}\text{Privilégio} \stackrel{\text{def}}{=} m_8 m\_e.(&d.d.\bar{e}_2 N\_E.\bar{e}_3.\bar{m}_2.\text{Privilégio} \\ &+ \\ &\tau.\bar{e}_2 N\_E.\bar{e}_3.\bar{m}_2.\text{Privilégio})\end{aligned}$$

Onde o agente Privilégio recebe (na porta m_8) o novo estado do Controlador_EM que é atribuído à variável m_e. Após esse evento, dá-se uma escolha indeterminística.

Na primeira alternativa o usuário ocupa o recurso compartilhado (d ocorre) e depois libera-o (novo d ocorre). O agente Privilégio envia N_E para Mem, devolve o controle para o agente Cont e desbloqueia a Comporta 2. No final, volta recursivamente.

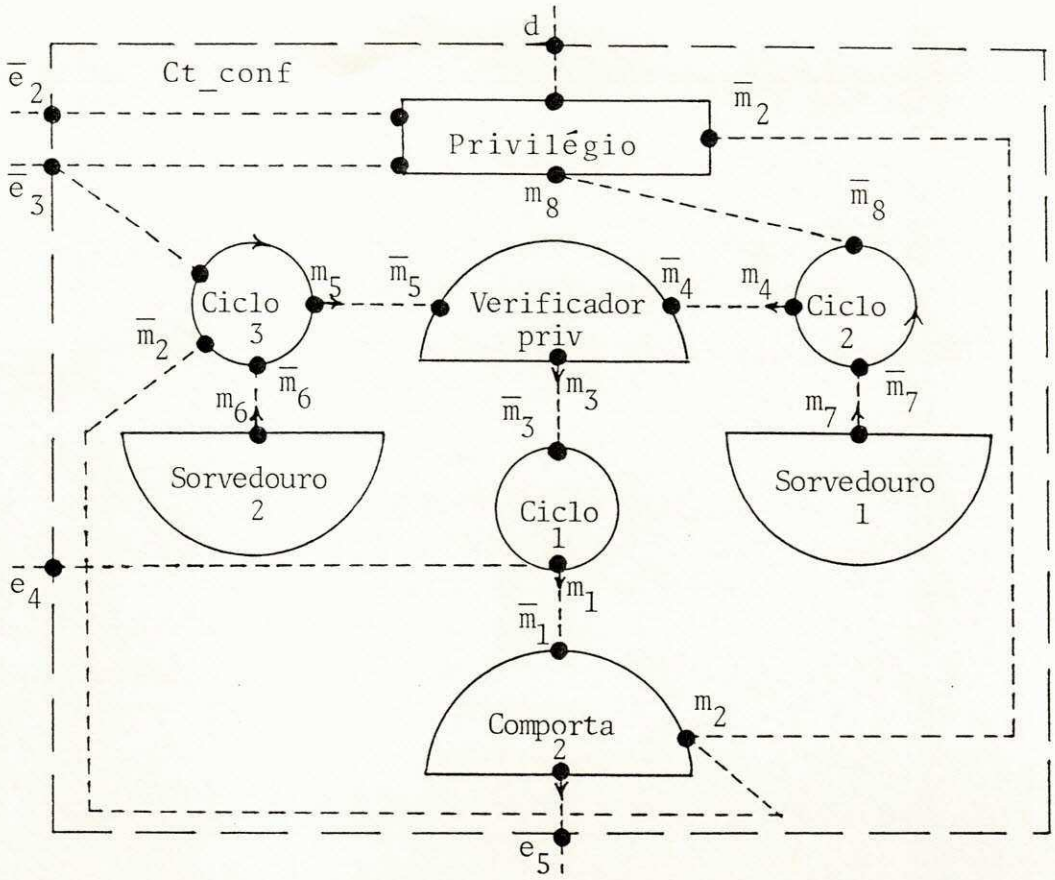


Figura 5.18 - Refinamento do agente Ct_conf.

Na segunda alternativa o usuário não solicita o recurso em tempo hábil. O agente Privilégio evolui (τ ocorre), envia N_E para Mem, devolve o controle para o agente Cont e desbloqueia a Comporta 2. No final, também volta recursivamente.

O agente Privilégio pode ser refinado, como mostra a Fig. 5.19, onde todos os agentes são básicos com exceção de Temporizador. Este agente faz com que o usuário tenha um prazo para a solicitação do recurso compartilhado.

$$\text{Temporizador} \stackrel{\text{def}}{=} n_4 . T_i(0)$$

onde o Temporizador, após ser ativado por um evento na porta n_4, comporta-se como $T_i(0)$.

$$T_i(t) \stackrel{\text{def}}{=} \underline{\text{if}}\ p(t)\ \underline{\text{then}}\ (\text{Temporizador} + T_i(t+1))$$
$$\underline{\text{else}}\ \bar{n}_6 . \text{Temporizador}$$

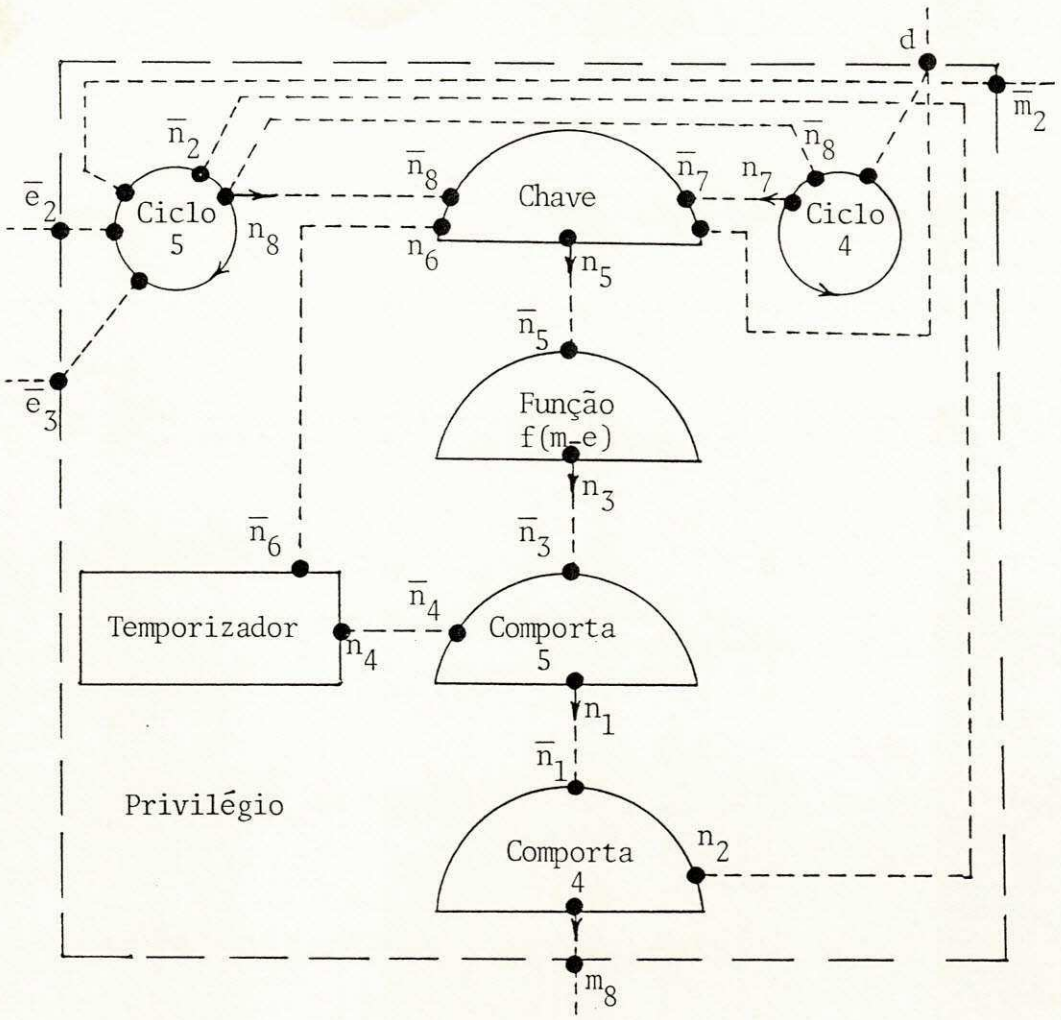


Figura 5.19 - Refinamento do agente Privilégio.

O predicado p(t) é avaliado por um processo dependente da implementação. Se t é menor do que um valor máximo (p(t) = verdadeiro) então haverá um incremento no valor de t ou o Temporizador é reinicializado (essa escolha indeterminística permite que a incrementação no valor de t seja interrompida pelo usuário, ao solicitar o recurso compartilhado, antes do estouro da temporização). Caso contrário (p(t) = falso) uma sincronização de temporização ocorrerá na porta n_6.

O processo de refinamentos sucessivos pode continuar com os agentes Mem, Cont, Ct_ind e Temporizador até que o Controlador_EM esteja completamente especificado em termos de agentes básicos predefinidos. Nesse caso a implementação final consistirá da implementação e da interconexão desses agentes.

5.4. Avaliação da abordagem clássica

Esse tipo de abordagem permite ao especificador explorar amplamente a sua criatividade no processo de obtenção das especificações. Dependendo da sua experiência ele pode obter especificações simples, elegantes e concisas, que terão também a sua marca pessoal.

Especificações personalizadas podem se tornar inconvenientes para fins de sistematização, estruturação, trabalho em equipe, padronização e automatização.

No caso da trajetória de design, a sistematização é uma disciplina de trabalho que visa agilizar a produção de especificações, minimizando tempo e custos. Como sistematização e liberdade são normalmente duas características conflitantes, um compromisso deve ser estabelecido entre ambas. O estabelecimento desse compromisso envolve a definição de um conjunto de princípios (ortogonalidade, generalidade e flexibilidade) que deve nortear o especificador durante a trajetória de design.

A abordagem clássica não dispõe de regras para impor estruturação às especificações. Ela se baseia no processo de refinamentos sucessivos (levando à hierarquização das diversas especificações de um mesmo sistema) mas nada diz sobre a organização das partes que compõem uma especificação em um determinado nível de abstração.

A ausência de estruturação nas especificações dificulta o trabalho em equipe. Se as partes de um sistema não podem ser tratadas independentemente umas das outras, cada especificador é obrigado a ter uma visão global do sistema. Assim o trabalho em paralelo (diferentes partes do sistema sendo desenvolvidas simultaneamente) fica prejudicado.

A ausência de sistematização e/ou estruturação torna-se ainda mais prejudicial quando as especificações a serem desenvolvidas são complexas, de grande porte e destinadas à padronização. Esse é o caso dos serviços e protocolos relativos ao Modelo de Referência OSI/ISO, os quais são normalmente

desenvolvidos por equipes multinacionais de projetistas.

Outro problema é a dificuldade de automatização. A margem de variações nas soluções permitidas pela abordagem clássica é larga demais para ser tratada de modo prático através de recursos computacionais. Como atualmente crescem as necessidades de produção industrial de software de comunicação, metodologias transformacionais que possibilitem a produção (semi-) automática desse tipo de software têm sido investigadas.

CAPÍTULO VI

ABORDAGEM TRANSFORMACIONAL

O objetivo principal deste tipo de abordagem é o de fornecer ao projetista um conjunto de recursos automáticos para a obtenção do design de sistemas. Tais recursos baseiam-se em regras de transformação, que uma vez aplicadas à especificação de um sistema (com um certo grau de detalhamento), permitem a obtenção de outra versão do mesmo sistema (com um grau de detalhamento ainda maior).

As regras de transformação são definidas de modo a produzirem resultados sempre corretos, quando criteriosamente aplicadas. Desse modo dispensam a prova de correção das especificações a posteriori.

Em [MeBo 83] apresenta-se uma metodologia transformacional para a definição de um dos componentes de um sistema, a partir da especificação do serviço oferecido pelo sistema e da especificação de cada um dos outros componentes desse sistema. De acordo com tal metodologia, obtém-se o conjunto de ações do componente desconhecido, subtraindo as ações realizadas pelos componentes conhecidos do conjunto das ações realizadas pelo provedor de serviço.

O comportamento do componente desconhecido é obtido através de regras de transformação, que envolvem "projeções" das sequências de ações dos componentes conhecidos sobre o conjunto de ações que o componente desconhecido pode realizar. Tais projeções apagam nessas sequências todas as ações que não podem ser realizadas pelo componente desconhecido.

Em um dos exemplos apresentados pelos autores, realiza-se o refinamento de um sistema de comunicação que utiliza o protocolo do bit alternante. Nesse exemplo todas as especificações são realizadas com o emprego

de máquinas de estados finitas (MEFs). A MEF correspondente à entidade receptora é obtida a partir da MEF definida para o seviço de comunicação e da MEF definida para a entidade emissora.

A metodologia adotada em [BoGo 86] está voltada para a obtenção da especificação do protocolo correspondente a uma dada especificação de serviço, mas não necessita do conhecimento prévio de nenhum dos componentes do sistema.

Essa metodologia exige, além da especificação do serviço oferecido pelo sistema, a inclusão de informações sobre os locais onde as primitivas de serviço devem ocorrer. Cada entidade de protocolo é obtida a partir de "projeções" da especificação do serviço sobre o local onde a entidade pode realizar as suas ações.

A sintaxe das expressões de comportamento é definida através das regras de produção de uma gramática livre-de-contexto, envolvendo apenas operadores para expressar a sequência de ações, o paralelismo de execução e a escolha entre alternativas de comportamento. O algoritmo apresentado permite a dedução das expressões de comportamento das entidades de protocolo, que se comunicam através de um serviço subjacente confiável.

Uma extensão dessa metodologia é apresentada em [KhBo 89]. Nessa extensão incluem-se outros operadores e admite-se um serviço subjacente não confiável. Outra característica dessa extensão é a obtenção de resultados mais otimizados.

Nova extensão, ainda mais completa, é apresentada em [KaHi 90]. Nesse caso a linguagem de especificação é LOTOS Básico, isto é, LOTOS sem passagem de valores.

Uma metodologia transformacional para a realização de especificações em CCS é apresentada em [Parr 90]. Tal metodologia permite que o componente desconhecido de um sistema distribuído seja obtido a partir da especificação do serviço oferecido pelo sistema e da especificação dos componentes conhecidos. Para facilitar a aplicação dessa metodologia, o autor utilizou uma ferramenta semi-automática.

A especificação de duas entidades de protocolo correspondentes a uma especificação de serviço é obtida algoritmicamente em [Lang 90]. Os algoritmos apresentados nesse trabalho supõem LOTOS Básico como linguagem de especificação. Duas situações são consideradas: na primeira supõe-se que as entidades de protocolo comunicam-se diretamente, enquanto que na segunda supõe-se que as entidades de protocolo comunicam-se através de um meio de comunicação confiável. Para ambas as situações são desenvolvidos algoritmos para casos cada vez mais complexos. Inicialmente, impõe-se que a especificação do serviço não tenha eventos internos nem escolhas entre eventos observáveis em sítios diferentes. O algoritmo desenvolvido para atender a essas imposições é depois estendido em duas etapas, sendo que em cada uma dessas etapas é retirada uma imposição.

6.1. Uma abordagem transformacional em CCS

Nesta seção definem-se regras de transformação que permitem obter uma especificação refinada de um sistema a partir de uma especificação abstrata do mesmo sistema. Para dar simplicidade às regras de transformação, utiliza-se um subconjunto de CCS Básico. Tal subconjunto compreende apenas o uso dos operadores NIL, ação $(a, \bar{a}, \tau)$, soma (+), composição (|) e restrição (\).

As regras de transformação aplicam-se a um modelo arquitetônico constituído de três agentes. São eles o provedor de serviço Pro_Serv, a entidade de protocolo E1, localizada no sítio 1 e a entidade de protocolo E2, localizada no sítio 2 (Fig. 6.1).

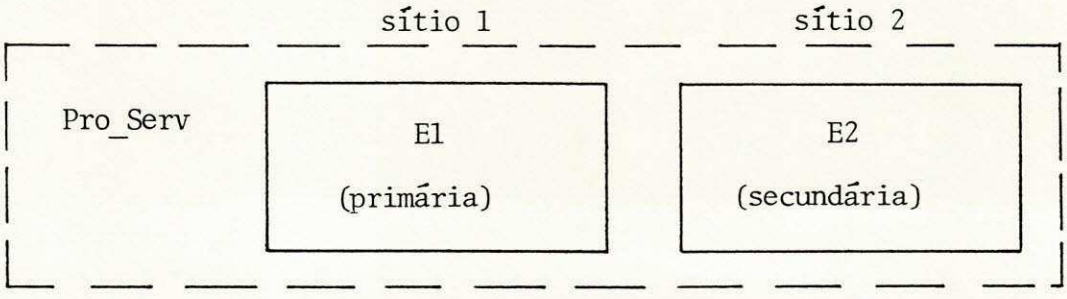


Figura 6.1 - Modelo arquitetônico.

O agente Pro_Serv representa a versão mais abstrata do sistema. Considera-se que o seu comportamento é conhecido.

Os agentes E1 e E2 representam, compostos, a versão refinada do sistema. A especificação do comportamento de cada um desses agentes deve ser alcançada com a aplicação das regras de transformação.

Considera-se que o primeiro evento realizado pelo sistema ocorre no sítio 1. Como esse evento envolve a entidade de protocolo E1, esta é designada entidade primária. A entidade de protocolo E2 é designada entidade secundária.

As entidades de protocolo comunicam-se diretamente para apresentar o mesmo comportamento observável do provedor de serviço. A comunicação direta entre as entidades subentende a utilização de um serviço subjacente. Tal serviço, por ser confiável, não necessita representação no modelo adotado.

Para a definição das regras de transformação analisa-se um conjunto de aspectos isolados que podem estar presentes na especificação de serviço. Cada aspecto permite a criação de uma regra que mapeia a estrutura da especificação de serviço na estrutura do protocolo correspondente. As regras são validadas e apresentadas de forma algorítmica.

A análise da especificação de serviço considera quatro aspectos: as ações isoladas, o sequenciamento finito de ações, o comportamento recursivo do sistema e a escolha indeterminística entre ações.

6.1.1. Ações isoladas

A cada ação observável (por exemplo a, $\bar{a}$) ou não observável (τ) da especificação de serviço corresponde uma ação igual definida na especificação de uma das entidades de protocolo.

Se a ação definida na especificação de serviço ocorre no sítio 1, então essa mesma ação deve participar do comportamento da entidade de protocolo do sítio 1, isto é, de E1. Caso contrário, deve participar do comportamento da entidade de protocolo do sítio 2, isto é, de E2.

Por exemplo (Fig. 6.2), no caso em que

$$Pro\_Serv \stackrel{def}{=} a1.a2.NIL$$

onde a ação a1 ocorre no sítio 1 e a ação a2 ocorre no sítio 2, os comportamentos das entidades de protocolo E1 e E2 são tais que

$a1 \in L(E1)$ (a1 é um dos rótulos de E1)

$a2 \in L(E2)$ (a2 é um dos rótulos de E2)

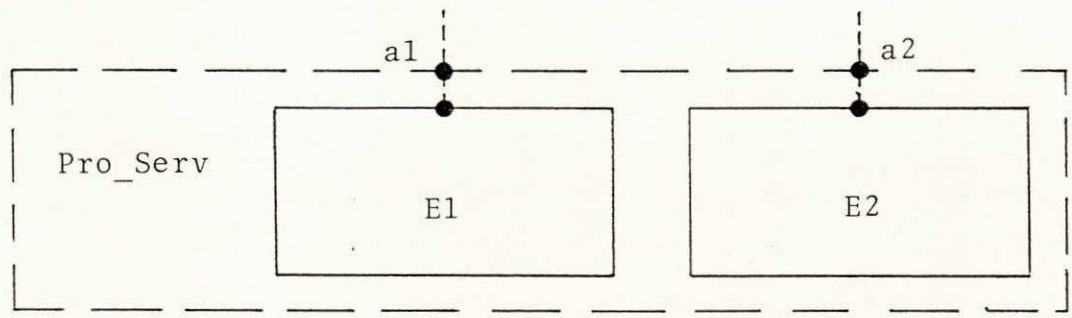


Figura 6.2 - Caso em que a1 ocorre no sítio 1
e a2 ocorre no sítio 2.

Algoritmo 1

Dado

$$Pro\_Serv \stackrel{def}{=} a(1). \ldots .a(n).Pro\_Serv'$$

$L(E1) \Leftarrow \emptyset$; $L(E2) \Leftarrow \emptyset$;

$i \Leftarrow 1$

Enquanto $i \leq n$ faça

```
        Se a(i) ocorre no sítio 1 então
           Se a(i) é um evento observável então
              L(E1)  <==  L(E1) ∪ {a(i)}
           senão inclua τ no comportamento de E1
        caso contrário
           Se a(i) é um evento observável então
              L(E2)  <==  L(E2) ∪ {a(i)}
           senão inclua τ no comportamento de E2
        fim se
i <== i + 1
fim enquanto.
```

O algoritmo 1 pode ser aplicado às sequências de ações do provedor de serviço Pro_Serv. Ele divide o conjunto de ações de Pro_Serv em duas partes, conforme o sítio onde as ações ocorrem. As ações observáveis no sítio 1 passam a pertencer ao conjunto de rótulos da entidade de protocolo E1 e as ações observáveis no sítio 2 passam a pertencer ao conjunto de rótulos da entidade de protocolo E2.

As ações não observáveis τ não podem ser incluídas em L(E1) nem em L(E2) mas, uma vez presentes no comportamento de Pro_Serv, devem constar do comportamento E1 ou do comportamento de E2, conforme o sítio onde se considera que essas ações ocorrem.

6.1.2. Sequenciamento finito de ações

Se a especificação de serviço apresenta uma sequência finita de ações (observáveis ou não), há dois casos importantes a considerar:

1º caso: as ações realizam-se em um mesmo sítio. Nesse caso, o comportamento da entidade de protocolo localizada em tal sítio deve apresentar a mesma sequência de ações. A outra entidade de protocolo permance indiferente a essas ações.

Por exemplo (Fig. 6.3), no caso em que

$$Pro\_Serv \stackrel{def}{=} a1.b1.c1.NIL$$

onde as ações a1, b1 e c1 ocorrem no sítio 1, os comportamentos das entidades de protocolo E1 e E2 podem ser definidos por

$$E1 \stackrel{def}{=} a1.b1.c1.NIL$$

$$E1 \stackrel{def}{=} NIL$$

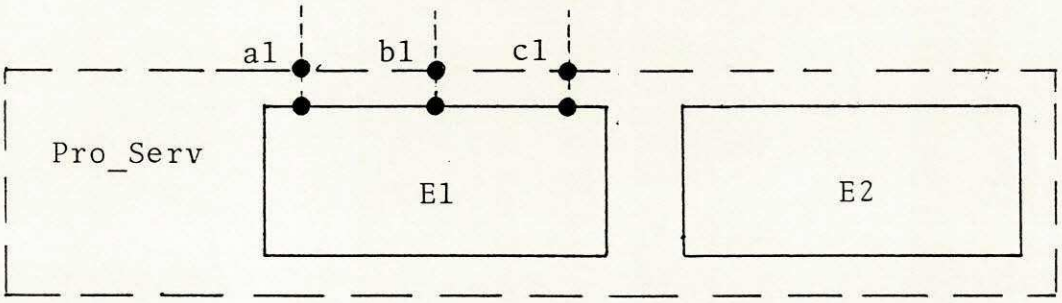


Figura 6.3 - Caso em que a1, b1 e c1 ocorrem no sítio 1.

Algoritmo 2

Dado

$$Pro\_Serv \stackrel{def}{=} a(1). \ldots .a(n).Pro\_Serv'$$

as entidades de protocolo E1 e E2 têm comportamentos definidos por:

$$E1 \stackrel{def}{=} A(1). \ldots .A(n).E1'$$

$$E2 \stackrel{def}{=} B(1). \ldots .B(n).E2'$$

```
Se as ações a(1). ... .a(n) ocorrem no sítio 1 então
        i <== 1
        Enquanto  i ≤ n faça
                A(i) = a(i)
                B(i) = ε        (sequência vazia de eventos)
                i  <== i + 1
        fim enquanto
fim se
Se as ações a(1). ... .a(n) ocorrem no sítio 2 então
        i <== 1
        Enquanto  i ≤ n faça
                A(i) = ε        (sequência vazia de eventos)
                B(i) = a(i)
                i  <== i + 1
        fim enquanto
fim se.
```

Para provar a validade da regra de transformação expressa através do algoritmo 2 pode-se considerar a Fig. 6.4. Nessa figura, a árvore E1|E2 tem os mesmos eventos que a árvore Pro_Serv e com a mesma ordenação.

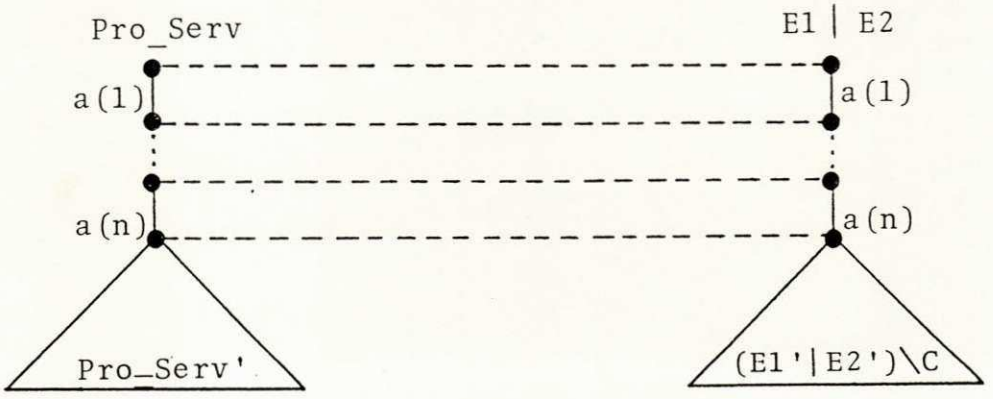


Figura 6.4 - Bissimulação entre Pro_Serv e E1|E2.

Considerando que

$$Pro\_Serv' \approx (E1' | E2') \backslash C$$

onde C é um conjunto de ações restritas, pode-se traçar as linhas interrompidas da Fig. 6.4 que representam a bissimulação entre Pro_Serv e E1|E2. Tal bissimulação prova a equivalência requerida.

2º caso: as ações realizam-se em sítios diferentes. Nesse caso, a cada mudança de sítio corresponde uma sincronização das entidades de protocolo.

Por exemplo (Fig. 6.5), no caso em que

$$Pro\_Serv \stackrel{def}{=} a1.b1.a2.b2.NIL$$

onde as ações a1 e b1 ocorrem no sítio 1 e as ações a2 e b2 ocorrem no sítio 2, os comportamentos das entidades de protocolo E1 e E2 podem ser definidos por

$$E1 \stackrel{def}{=} a1.b1.\overline{c}.NIL$$
$$E2 \stackrel{def}{=} c.a2.b2.NIL$$

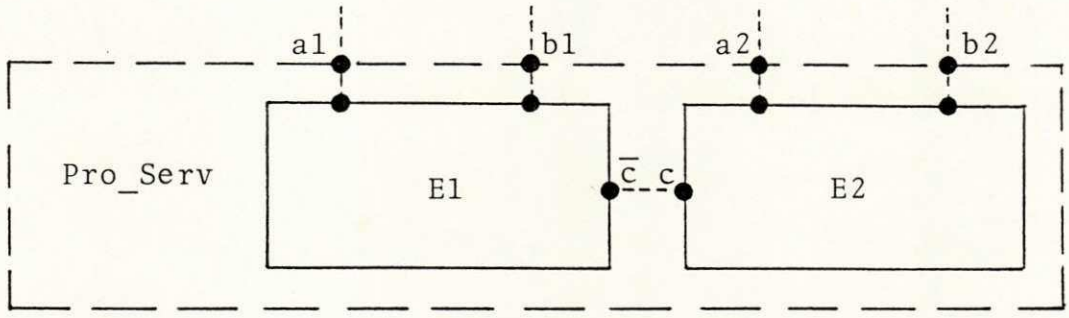


Figura 6.5 - Caso em que a1 e b1 ocorrem no sítio 1 e a2 e b2 ocorrem no sítio 2.

Nesse exemplo, a sincronização das entidades de protocolo E1 e E2 se dá através das portas complementares c e $\bar{c}$. Essas portas devem ser ocultadas do observador para fins de verificação de equivalência de observação.

$$(E1 \mid E2) \backslash \{c\} \approx Pro\_Serv$$

Nos casos simples em que o provedor de serviço executa uma sequência de ações em um dos sítios e, posteriormente, executa uma sequência de ações no outro sítio, tornando-se depois inativo, as entidades de protocolo realizam apenas uma sincronização.

Algoritmo 3

Dado

$$Pro\_Serv \stackrel{def}{=} a(1). \ldots .a(n).Pro\_Serv'$$

as entidades de protocolo E1 e E2 têm comportamentos definidos por:

Se a(1) ocorre no sítio 1 então

$E1 \stackrel{def}{=} A(1). \ldots .A(n).\bar{c}.E1'$

$E2 \stackrel{def}{=} c.B(1). \ldots .B(n).E2'$

caso contrário

$E1 \stackrel{def}{=} c.A(1). \ldots .A(n).E1'$

$E2 \stackrel{def}{=} B(1). \ldots .B(n).E2'$

fim se

i <== 1

Enquanto i ≤ n faça

Se a(i) ocorre no sítio 1 então

A(i) = a(i)

B(i) = ε (sequência vazia de eventos)

caso contrário

A(i) = ε

B(i) = a(i)

fim se

i <== i + 1

fim enquanto.

Para provar a validade da regra de transformação expressa através do algoritmo 3 pode-se considerar a Fig. 6.6. Nessa figura, a árvore (E1|E2)\{c} tem os mesmos eventos que a árvore Pro-Serv e com a mesma ordenação, mas (E1|E2)\{c} intercala uma vez o evento não observável τ, correspondente à sincronização da entidade de protocolo E1 com a entidade de protocolo E2, na porta de comunicação de nome c.

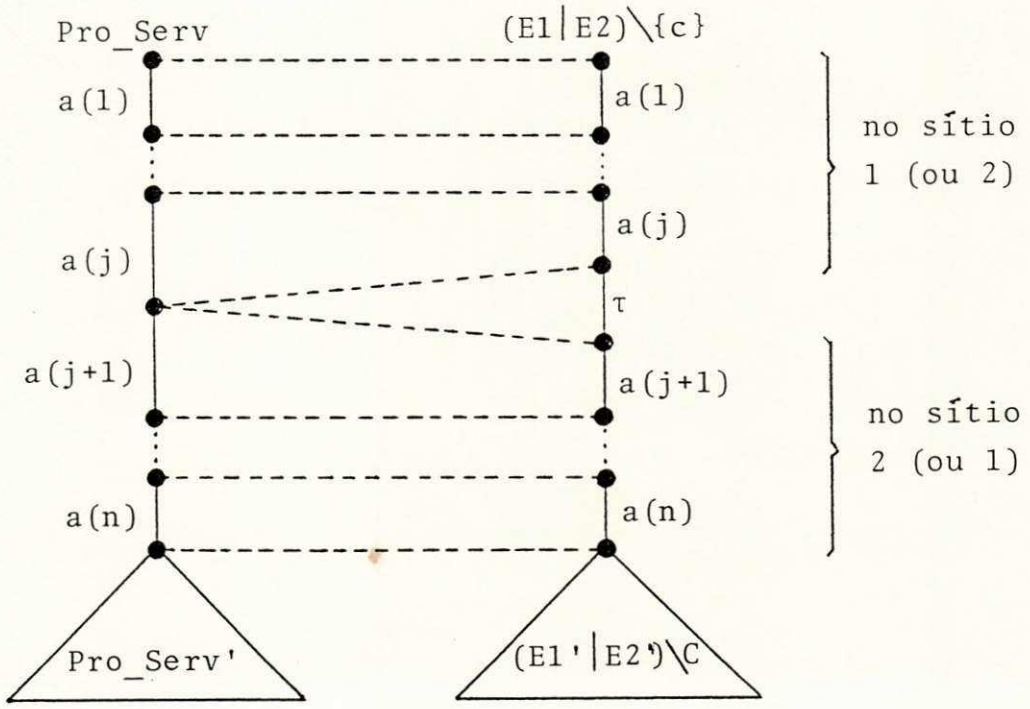


Figura 6.6 - Bissimulação entre Pro_Serv e (E1|E2)\{c}.

Considerando que

$$Pro\_Serv' \approx (E1' \mid E2')\backslash C$$

onde C é um conjunto de ações restritas, pode-se traçar as linhas interrompidas da Fig. 6.6 que representam a bissimulação entre Pro_Serv e (E1'|E2')\C. Tal bissimulação prova a equivalência requerida.

A situação é mais complexa quando o provedor de serviço alterna sequências de ações do sítio 1 com sequências de ações do sítio 2. Por exemplo (Fig. 6.7), no caso em que

$$Pro\_Serv \stackrel{def}{=} a1.a2.a1.b1.a2.b2.NIL$$

onde as sequências de ações são tais que

a1 ocorre no sítio 1,

a2 ocorre no sítio 2,

a1.b1 ocorre no sítio 1 e

a2.b2 ocorre no sítio 2,

as entidades de protocolo E1 e E2 podem ser definidas por

$$E1 \stackrel{def}{=} a1.\overline{c}.d.a1.b1.\overline{c}.NIL$$

$$E2 \stackrel{def}{=} c.a2.\overline{d}.c.a2.b2.NIL$$

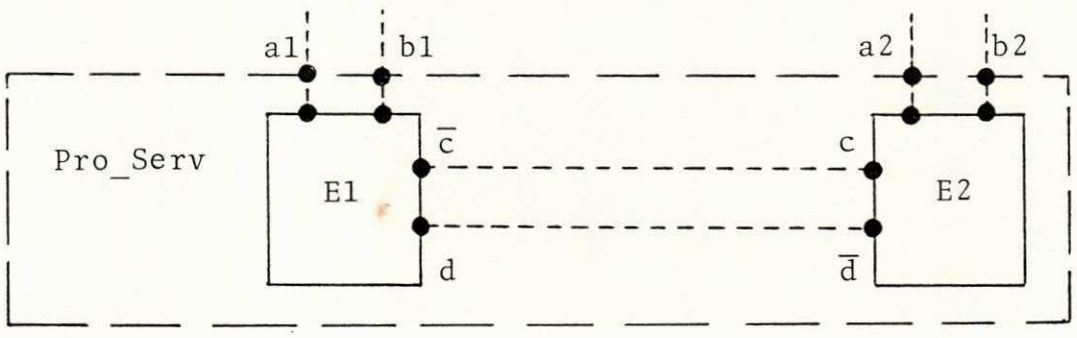


Figura 6.7 - Sincronizações de E1 com E2 para o caso de alternância de eventos nos sítios 1 e 2.

Nesse exemplo, uma entidade transfere o comportamento ativo para a outra entidade através de uma sincronização. Após isso, espera nova sincronização para retomar o comportamento ativo.

Algoritmo 4

Dado

$$Pro\_Serv \stackrel{def}{=} a(1). \ldots .(a(n).Pro\_Serv'$$

as entidades de protocolo E1 e E2 têm comportamentos definidos por

$$E1 \stackrel{def}{=} A(1). \ldots .A(n).E1'$$
$$E2 \stackrel{def}{=} B(1). \ldots .B(n).E2'$$

i <== 1

A(i) = a(1) (considerando E1 como entidade primária)

B(i) = ε (onde ε é a sequência vazia de eventos)

i <== i + 1

Enquanto i ≤ n faça

  Se a(i-1) e a(i) ocorrem no mesmo sítio então

    Se a(i-1) e a(i) ocorrem no sítio 1 então

      A(i) = a(i); B(i) = ε

    senão (a(i-1) e a(i) ocorrem no sítio 2)

      A(i) = ε; B(i) = a(i)

    fim se

  senão (a(i-1) e a(i) ocorrem em sítios diferentes)

    Se a(i-1) ocorre no sítio 1 e a(i) ocorre no sítio 2 então

      A(i) = $\overline{c}$; B(i) = c.a(i)

    senão (a(i-1) ocorre no sítio 2 e a(i) ocorre no sítio 1)

      A(i) = d.a(i); B(i) = $\overline{d}$

    fim se

```
        fim se
        i <== i + 1
fim enquanto.
```

No caso em que a especificação de serviço é definida por uma sequência finita de eventos, o algoritmo 4 pode ser utilizado para a derivação automática da especificação do protocolo correspondente. O algoritmo 4 copia os eventos da sequência, colocando-os com o mesmo ordenamento no comportamento de E1 ou no comportamento de E2, conforme o sítio em que ocorrem.

Além disso, o algoritmo 4 introduz eventos novos. Estes eventos correspondem às sincronizações das entidades de protocolo E1 e E2 e indicam a transferência do comportamento ativo, que vinha sendo exercido por uma entidade, para a outra entidade que, então, assume a continuação das ações.

Para provar a validade da regra de transformação expressa através do algoritmo 4 pode-se considerar a Fig. 6.8. Nessa figura, a árvore (E1|E2)\{c,d} tem os mesmos eventos que a árvore Pro_Serv, e com a mesma ordenação. Entretanto, (E1|E2)\{c,d} intercala os eventos não observáveis τ correspondentes às sincronizações das entidades de protocolo E1 e E2 nas portas de comunicação de nomes c e d.

Considerando que

$$Pro\_Serv' \approx (E1'|E2')\backslash C$$

onde C é um conjunto de ações restritas, pode-se traçar as linhas interrompidas da Figura 6.8 que representam a bissimulação entre Pro_Serv e (E1|E2)\{c,d}. Tal bissimulação prova a equivalência requerida.

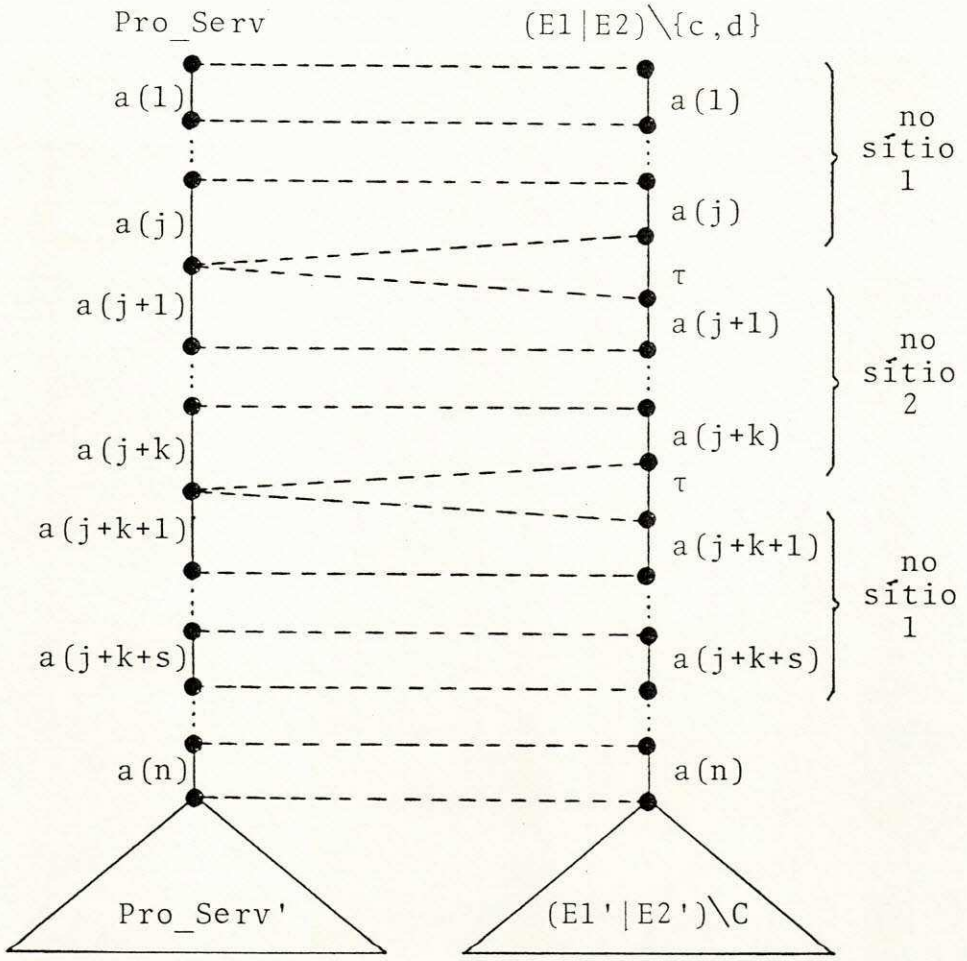


Figura 6.8 - Bissimulação entre Pro_Serv e (E1|E2)\{c,d} no caso de comportamento finito.

6.1.3. Comportamento recursivo

Se o agente provedor de serviço (identificado por Pro_Serv) apresenta um comportamento infinito, e esse comportamento inclui eventos nos dois sítios do sistema, então os agentes derivados (as entidades de protocolo identificadas por E1 e E2) também devem apresentar comportamentos infinitos.

Assim, o identificador E1 deve ser acrescentado ao final da expressão de comportamento da entidade E1 e o identificador E2 deve ser acrescentado ao final da expressão de comportamento da entidade E2. Entretanto, essas medidas não são suficientes para que a derivação esteja correta.

Por exemplo, pode-se considerar

$$Pro\_Serv \stackrel{def}{=} a1.a2.Pro\_Serv$$

onde a ação a1 ocorre no sítio 1 e a ação a2 ocorre no sítio 2. Aparentemente, os comportamentos das entidades de protocolo E1 e E2 seriam

$$E1 \stackrel{def}{=} a1.\overline{c}.E1$$
$$E2 \stackrel{def}{=} c.a2.E2$$

Se fosse assim, a entidade E1 poderia realizar a ação a1 logo após a sua sincronização com a entidade E2. Desse modo, o refinamento estaria errado.

Para garantir que a sequência de eventos observáveis apresentada por Pro_Serv seja também apresentada pela composição de E1 e E2, introduz-se um evento de sincronização na porta d (Fig. 6.9).

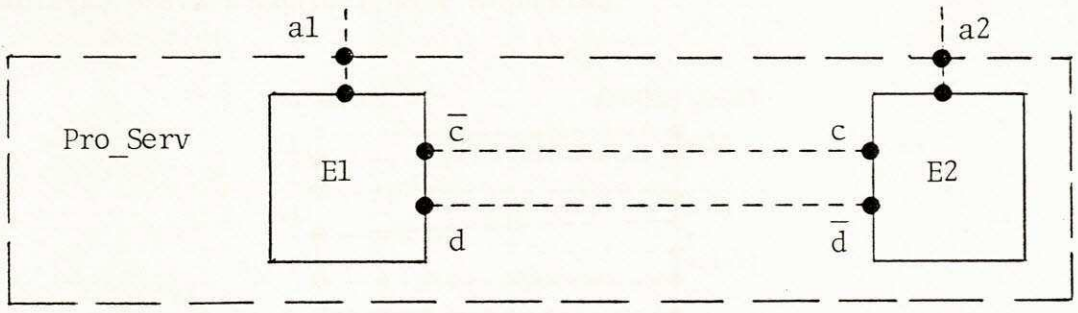


Figura 6.9 - Sincronizações de E1 com E2 para o caso de comportamento infinito.

Desse modo,

$$E1 \stackrel{def}{=} a1.\overline{c}.d.E1$$
$$E2 \stackrel{def}{=} c.a2.\overline{d}.E2$$

Algoritmo 5

Dado

$$Pro\_Serv \stackrel{def}{=} a(1). \ldots .a(n).Pro\_Serv$$

as entidades de protocolo E1 e E2 têm comportamentos definidos por

$$E1 \stackrel{def}{=} A(1). \ldots .A(n).d.E1$$
$$E2 \stackrel{def}{=} B(1). \ldots .B(n).\bar{d}.E2$$

no mais, repete-se o algoritmo 4.

Para provar a validade da regra de transformação expressa através do algoritmo 4 pode-se considerar a Fig. 6.10. Nessa figura, a árvore $(E1|E2)\backslash\{c,d\}$ tem os mesmos eventos que a árvore Pro_Serv, e com a mesma ordenação. Entretanto, $(E1|E2)\backslash\{c,d\}$ intercala os eventos τ (se a(n) ocorre no sítio 1, o último τ é desnecessário) correspondentes às sincronizações das entidades de protocolo E1 e E2, nas portas de nomes c e d. As linhas interrompidas representam a bissimulação entre Pro_Serv e $(E1|E2)\backslash\{c,d\}$. Tal bissimulação prova a equivalência requerida.

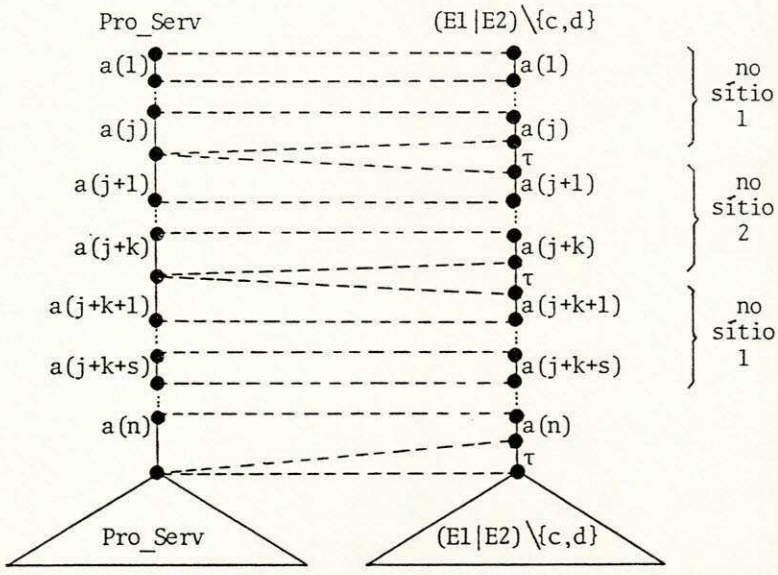


Figura 6.10 - Bissimulação entre Pro_Serv e $(E1|E2)\backslash\{c,d\}$ no caso de comportamento infinito.

6.1.4. Escolha indeterminística entre ações

O agente Pro_Serv pode apresentar um comportamento com escolhas indeterminísticas entre ações. Para fins de simplificação, no presente trabalho supõe-se que essas escolhas se dão entre ações de um mesmo sítio.

Por exemplo, pode-se considerar

$$\text{Pro\_Serv} \stackrel{\text{def}}{=} a1.a2.NIL + b1.b2.NIL$$

onde as ações a1 e b1 ocorrem no sítio 1 e as ações a2 e b2 ocorrem no sítio 2. Nesse exemplo, a escolha se dá entre eventos do sítio 1 (isto é, entre o evento a1 e o evento b1).

O comportamento das entidades de protocolo E1 e E2 pode ser dado por

$$E1 \stackrel{\text{def}}{=} a1.\overline{c1}.NIL + b1.\overline{c2}.NIL$$

$$E2 \stackrel{\text{def}}{=} c1.a2.NIL + c2.b2.NIL$$

Para fins de derivação das entidades de protocolo, quando uma escolha se apresenta em um sítio do sistema, ela deve ser reproduzida pela entidade localizada nesse sítio e ignorada pela outra entidade. No exemplo, como a escolha apresenta-se entre eventos do sítio 1, ela reproduz-se no comportamento da entidade E1.

A questão da sincronização das entidades de protocolo, posteriormente à realização de uma escolha com a participação de uma delas, diz respeito à

transferência de comportamento ativo de uma para a outra entidade. Só a discriminação entre as sincronizações possíveis tem a ver com a escolha realizada.

No exemplo, se a escolha entre a1 e b1 recai sobre a1 a sincronização de E1 com E2 se dá em c1 (neste evento, o índice 1 lembra a primeira escolha). Se a escolha recai sobre b1 a sincronização de E1 com E2 se dá em c2 (agora, o índice 2 lembra a segunda escolha).

Em geral, pode-se considerar que uma escolha presente no comportamento do provedor de serviço resulta em uma escolha no comportamento de cada entidade de protocolo. Pode-se considerar ainda que tais escolhas têm igual número de alternativas, a menos que haja preocupação com relação à otimização das derivações. Neste trabalho considera-se que a otimização pode ser realizada em uma etapa posterior.

Algoritmo 6

Dado

$$Pro\_Serv \stackrel{def}{=} \sum_{i=1}^{n} a(i).Pro\_Serv(i)$$

se a escolha se dá entre eventos do sítio 1 então as entidades de protocolo E1 e E2 têm comportamentos definidos por

$$E1 \stackrel{def}{=} \sum_{i=1}^{n} a(i).E1(i)$$

$$E2 \stackrel{def}{=} \sum_{i=1}^{n} \varepsilon.E2(i)$$

caso contrário

$$E1 \stackrel{def}{=} \sum_{i=1}^{n} \varepsilon.E1(i)$$

$$E2 \stackrel{def}{=} \sum_{i=1}^{n} a(i).E1(i)$$

fim se.

Cada um dos algoritmos apresentados até aqui contempla um aspecto isolado, que pode estar presente na especificação de serviço e que necessita de mapeamento para a especificação do protocolo correspondente. O último aspecto é o da escolha indeterminística entre ações.

O algoritmo 6 realiza tal mapeamento através da reprodução da escolha no comportamento de uma das entidades de protocolo, conforme o sítio onde a escolha se dá.

Entretanto, os comportamentos dos agentes Pro_Serv, E1 e E2 posteriores à escolha não são considerados nesse algoritmo. Isso se dá porque esses comportamentos podem envolver vários outros aspectos como o de sequenciamento de ações (finito ou infinito), o de mudanças de sítio e mesmo o de novas escolhas indeterminísticas entre ações. Tais aspectos exigem a aplicação de algoritmos anteriores e, no último caso, a reaplicação do algoritmo 6.

Para provar a validade da regra de transformação expressa através do algoritmo 6 pode-se considerar a Fig. 6.11. Nessa figura, a árvore (E1|E2)\C (onde C representa um conjunto de ações restritas relativas aos comportamentos de E1(i) e E2(i), i = 1,...,n) tem a mesma escolha de eventos apresentada pela árvore Pro_Serv, no primeiro nível de ramificações de ambas.

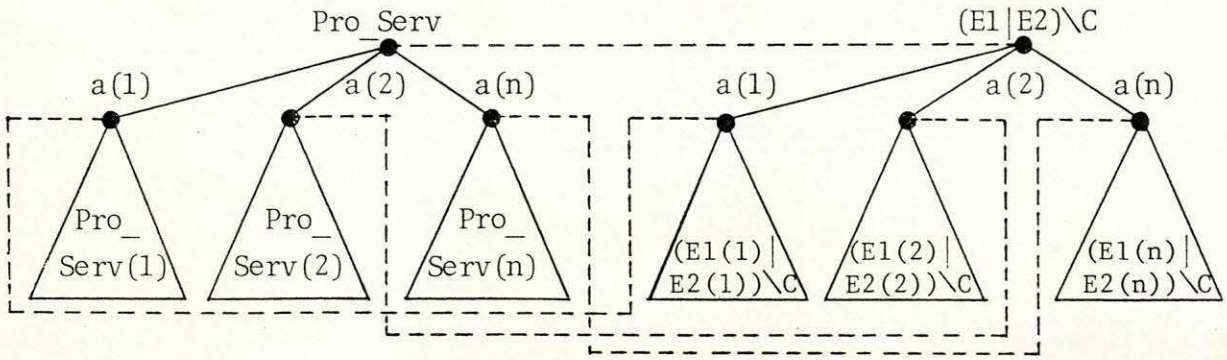


Figura 6.11 - Bissimulação entre Pro_Serv e (E1|E2)\C.

Considerando que os comportamentos posteriores às escolhas correspondentes são equivalentes quanto à observação nas duas árvores, isto é,

$$(E1(i)|E2(i))\backslash C \approx Pro\_Serv(i),\ i=1,2,\ldots,n$$

pode-se estabelecer a bissimulação indicada na Fig. 6.11 com o uso de linhas interrompidas. Tal bissimulação prova a equivalência requerida.

6.2. Aplicação: o núcleo FTAM do RM OSI/ISO

Para ilustrar a aplicabilidade dos algoritmos propostos, nesta seção deriva-se o comportamento de entidades de protocolo correspondentes a uma parte do núcleo do serviço e do protocolo "File Transfer, Access and Management (FTAM)" [ISO 86]. Para essa ilustração foi escolhido o estabelecimento de uma associação FTAM sem considerar a possibilidade de interrupções abruptas.

Na Fig. 6.12 o provedor dos serviços do núcleo FTAM, o agente Serv, é

representado por uma caixa preta. O seu comportamento observável (o serviço que oferece) é descrito através dos eventos (primitivas de serviço) que podem ocorrer nas portas de comunicação com os seus usuários.

Inicialmente, o provedor dos serviços do núcleo FTAM comporta-se como

$$Serv \stackrel{def}{=} F\_INIRQ.Serv\_1 \qquad \text{(eq. 6.2)}$$

isto é, o provedor pode receber um pedido de estabelecimento de uma associação FTAM (F_INIRQ ocorre). Após esse evento, o provedor comporta-se como

$$Serv\_1 \stackrel{def}{=} \tau.Serv\_2 + \tau.Serv\_3 \qquad \text{(eq. 6.1)}$$

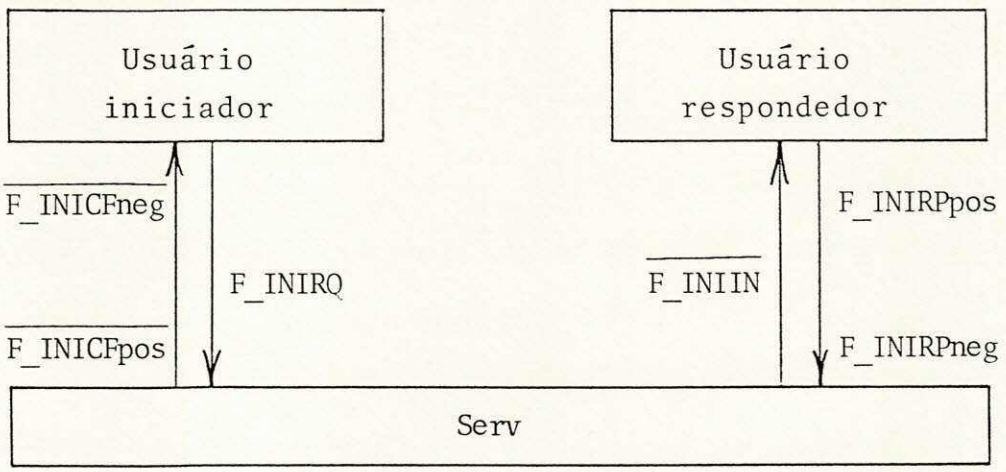


Figura 6.12 - Provedor de serviços FTAM e seus usuários.

isto é, o provedor pode executar um entre dois eventos não observáveis (uma

escolha indeterminística). Conforme o caso, passa a comportar-se como Serv_2 ou Serv_3.

$$Serv\_2 \stackrel{def}{=} \overline{F\_INICFneg}.Serv \qquad \text{(eq. 6.3)}$$

isto é, a primeira alternativa da escolha indeterminística indica uma recusa local ao pedido do usuário iniciador para o estabelecimento de uma associação FTAM. Nesse caso a recusa é oferecida a esse usário ($\overline{F\_INICFneg}$ ocorre). Após tal evento o provedor volta a comportar-se como Serv, onde poderá receber outra solicitação.

$$Serv\_3 \stackrel{def}{=} \tau.Serv\_2 + \tau.Serv\_4 \qquad \text{(eq. 6.4)}$$

isto é, o provedor realiza uma nova escolha indeterminística. Conforme o caso, volta a comportar-se como Serv_2 (indicando a recusa remota, no sítio do usuário respondedor) ou passa a comportar-se como Serv_4 (indicando a aceitação remota).

$$Serv\_4 \stackrel{def}{=} \overline{F\_INIIN}.Serv\_5 \qquad \text{(eq. 6.5)}$$

isto é, o provedor oferece uma indicação do pedido de associação FTAM ao usuário respondedor. Após a ocorrência de $\overline{F\_INIIN}$ o provedor passa a comportar-se como Serv_5 para receber a resposta desse usuário.

$$Serv\_5 \stackrel{def}{=} F\_INIRPneg.Serv\_2 + F\_INIRPpos.Serv\_6 \qquad \text{(eq. 6.6)}$$

isto é, o provedor apresenta uma escolha indeterminística. Na primeira alternativa recebe a recusa do usuário respondedor (F_INIRPneg ocorre) e volta a comportar-se como Serv_2. Na segunda alternativa recebe a aceitação desse usuário (F_INIRPpos ocorre) e passa a comportar-se como Serv_6.

$$Serv\_6 \stackrel{def}{=} \overline{F\_INICFpos}.Assoc \qquad \text{(eq. 6.7)}$$

isto é, o provedor oferece ao usuário iniciador a confirmação de que o seu pedido foi atendido ($\overline{F\_INICFpos}$ ocorre) e passa a comportar-se como Assoc. O comportamento de Assoc não é apresentado neste trabalho. Uma especificação mais completa do núcleo FTAM em CCS encontra-se em [RiLo 90].

6.2.1. Derivação das entidades de protocolo E1 e E2

Através da aplicação do algoritmo 1 da seção 6.1, os eventos observáveis representados na especificação do provedor de serviços do núcleo FTAM, isto é, as seis primitivas de serviço consideradas, dividem-se em duas partes. A primeira delas é

$$L(E1) = \left\{F\_INIRQ,\ \overline{F\_INICFneg},\ \ \overline{F\_INICFpos}\right\}$$

correspondente ao conjunto de rótulos da entidade de protocolo primária E1, localizada no sítio 1, e a segunda é

$$L(E2) = \left\{\overline{F\_INIIN},\ F\_INIRPneg,\ \ F\_INIRPpos\right\}$$

correspondente ao conjunto de rótulos da entidade de protocolo secundária E2, localizada no sítio 2.

Para a definição do comportamento da entidade E1 devem ser considerados não apenas os eventos pertencentes ao conjunto L(E1),

observáveis, mas também os eventos não observáveis, descritos em Serv, que ocorrem no sítio 1.

De modo semelhante, para a definição do comportamento da entidade E2 devem ser considerados não apenas os eventos pertencentes ao conjunto L(E2), observáveis, mas também os eventos não observáveis, descritos em Serv, que ocorrem no sítio 2.

As entidades E1 e E2 comunicam-se diretamente (comunicação virtual) através da troca de unidades de dados de protocolo (UDPs) (Fig. 6.13).

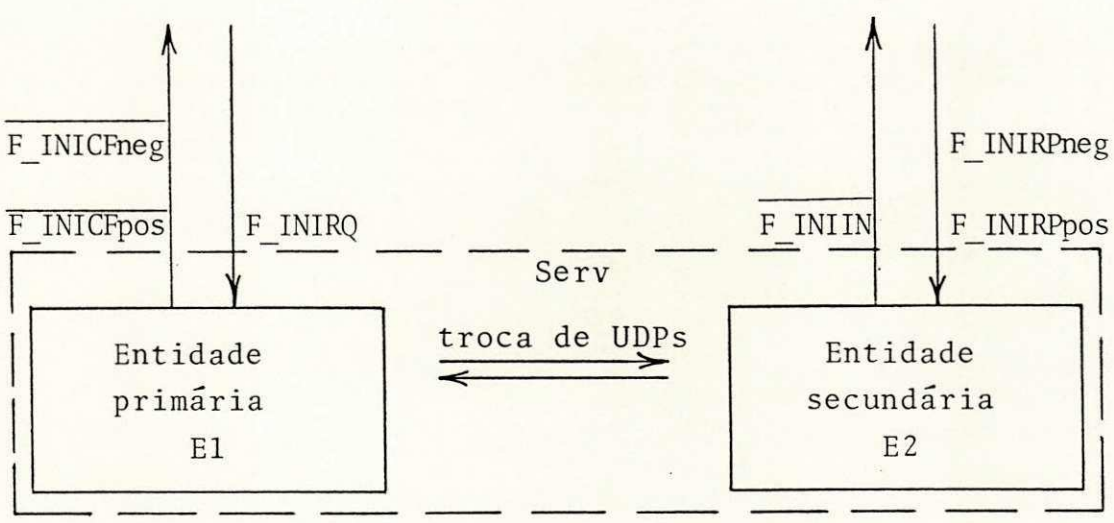


Figura 6.13 - Refinamento do agente Serv.

Para a definição das entidades E1 e E2, as equações que descrevem o comportamento de Serv são tratadas pelos algoritmos da seção 6.1.

A (eq. 6.1) descreve o primeiro evento do comportamento do provedor de serviços Serv. Esse evento é F_INIRQ e ocorre no sítio 1.

Aplicando o algoritmo 2 (com n=1) a esse comportamento obtém-se

$$E1 \stackrel{def}{=} F\_INIRQ.E1\_1$$

$$E2 \stackrel{def}{=} \varepsilon.E2\_1$$

onde o agente E1 realiza o mesmo evento e passa a comportar-se como E1_1. O agente E2 ignora o evento (executa uma sequência vazia de eventos) e passa a comportar-se como E2_1.

A (eq. 6.2) descreve uma escolha indeterminística entre dois eventos não observáveis. Essa escolha ocorre no sítio 1.

Aplicando o algoritmo 6 (com n=2) a esse comportamento obtém-se

$$E1\_1 \stackrel{def}{=} \tau.E1\_2 + \tau.E1\_3$$

$$E2\_1 \stackrel{def}{=} \varepsilon.E2\_2 + \varepsilon.E2\_3$$

de modo que, para o agente E1_1, se a primeira alternativa ocorre ele passa a comportar-se como E1_2, e se a segunda alternativa ocorre ele passa a comportar-se como E1_3. Para o agente E2_1 dá-se uma escolha entre sequências vazias de eventos: se a primeira alternativa ocorre ele passa a comportar-se como E2_2, e se a segunda alternativa ocorre ele passa a comportar-se como E2_3.

A (eq. 6.3) descreve o evento $\overline{F\_INICFneg}$ no sítio 1. Aplicando o algoritmo 2 (com n=1) obtém-se

$$E1\_2 \stackrel{def}{=} \overline{F\_INICFneg}.E1$$
$$E2\_2 \stackrel{def}{=} \varepsilon.E2$$

onde o agente E1_2 realiza o mesmo evento e volta a comportar-se como E1. O agente E2_2 executa uma sequência vazia de eventos e volta a comportar-se como E2.

A (eq. 6.4) descreve uma escolha indeterminística entre eventos não observáveis no sítio 2. Aplicando o algoritmo 3, introduz-se uma sincronização das entidades de protocolo nas portas complementares de nome c1, devida à mudança de sítio:

$$E1\_3 \stackrel{def}{=} \overline{c1}.E1\_4$$
$$E2\_3 \stackrel{def}{=} c1.E2\_4$$

Aplicando o algoritmo 6 (com n=2), reproduz-se a escolha indeterminística no comportamento de E2_4. O agente E1_4 ignora essa escolha:

$$E1\_4 \stackrel{def}{=} \varepsilon.E1\_5 + \varepsilon.E1\_6$$

$$E2\_4 \stackrel{def}{=} \tau.E2\_5 + \tau.E2\_6$$

Se ocorre a primeira alternativa da escolha apresentada na (eq. 6.4), a atividade do provedor de serviço volta ao sítio 1. Aplicando o algoritmo 3 a esse comportamento introduz-se uma sincronização das entidades de protocolo nas portas complementares de nome c2. Assim comunica-se a recusa do pedido de associação FTAM:

$$E1\_5 \stackrel{def}{=} c2.E1\_2$$
$$E2\_5 \stackrel{def}{=} \overline{c2}.E2\_2$$

Se ocorre a segunda alternativa da escolha apresentada na (eq. 6.4), a atividade do provedor de serviço permanece no sítio 2. Nesse caso, a

entidade de protocolo primária passa a comportar-se como E1_6 e a entidade de protocolo secundária passa a comportar-se como E2_6.

A (eq. 6.5) descreve a ocorrência do evento $\overline{\text{F\_INIIN}}$ no sítio 2. Aplicando o algoritmo 2 (com n=2) obtém-se

$$E1\_6 \stackrel{def}{=} \varepsilon.E1\_7$$
$$E2\_6 \stackrel{def}{=} \overline{\text{F\_INIIN}}.E2\_7$$

onde o agente E1_6 executa uma sequência vazia de eventos e passa a comportar-se como E1_7. O agente E2_6 executa o evento $\overline{\text{F\_INIIN}}$ e passa a comportar-se como E2_7.

A (eq. 6.6) descreve uma escolha indeterminística entre os eventos F_INIRPneg e F_INIRPpos no sítio 2. Aplicando o algoritmo 6 (com n=2) obtém-se

$$E1\_7 \stackrel{def}{=} \varepsilon.E1\_8 + \varepsilon.E1\_9$$

$$E2\_7 \stackrel{def}{=} F\_INIRPneg.E2\_8 + F\_INIRPpos.E2\_9$$

onde o agente E1_7 apresenta uma escolha indeterminística entre duas sequências vazias de eventos. Se ocorre a primeira alternativa passa a comportar-se como E1_8, e se ocorre a segunda alternativa passa a comportar-se como E1_9.

O agente E2_7 apresenta uma escolha indeterminística com duas alternativas. Na primeira alternativa executa o evento observável F_INIRPneg e passa a comportar-se como E2_8, e na segunda alternativa executa o evento observável F_INIRPpos e passa a comportar-se como E2_9.

Se ocorre a primeira alternativa da escolha apresentada na (eq. 6.6), a atividade do provedor de serviço volta ao sítio 1. Aplicando o algoritmo 3 a esse comportamento, introduz-se uma sincronização das entidades de protocolo nas portas complementares de nome c2. Assim comunica-se a recusa do pedido de associação FTAM:

$$E1\_8 \stackrel{def}{=} c2.E1\_2$$
$$E2\_8 \stackrel{def}{=} \overline{c2}.E2\_2$$

Se ocorre a segunda alternativa da escolha apresentada na (eq. 6.6), a atividade do provedor de serviço também volta ao sítio 1. Aplicando o algoritmo 3 a esse comportamento, introduz-se uma sincronização das entidades de protocolo nas portas complementares de nome c3. Assim comunica-se a aceitação do pedido de associação FTAM:

$$E1\_9 \stackrel{def}{=} c3.E1\_10$$
$$E2\_9 \stackrel{def}{=} \overline{c3}.E2\_10$$

Finalmente, a (eq. 6.7) descreve o evento observável $\overline{F\_INICFpos}$ no sítio 1. Aplicando o algoritmo 2 obtém-se

$$E1\_10 \stackrel{def}{=} \overline{F\_INICFpos}.E1\_Assoc$$
$$E2\_10 \stackrel{def}{=} \varepsilon.E2\_Assoc$$

onde o agente E1_10 executa o evento observável $\overline{F\_INICFpos}$ e passa a comportar-se como E1_Assoc que identifica a atividade da entidade de protocolo E1 no regime de associação FTAM.

O agente E2_10 executa a sequência vazia de eventos e passa a comportar-se como E2_Assoc que identifica a atividade da entidade de protocolo E2 no regime de associação FTAM.

6.2.2. Simplificação das especificações derivadas

Embora os comportamentos derivados de Serv para as entidades de protoclo E1 e E2 estejam corretos, eles podem ser simplificados através da eliminação das sequências vazias de eventos ε. Com essa simplificação obtém-se para a entidade E1:

$$E1 \stackrel{def}{=} F\_INIRQ.E1\_1$$

$$E1\_1 \stackrel{def}{=} \tau.E1\_2 + \tau.E1\_3$$

$$E1\_2 \stackrel{def}{=} \overline{F\_INICFneg}.E1$$

$$E1\_3 \stackrel{def}{=} \overline{c1}.E1\_4$$

$$E1\_4 \stackrel{def}{=} E1\_5 + E1\_6$$

$$E1\_5 \stackrel{def}{=} c2.E1\_2$$

$$E1\_6 \stackrel{def}{=} E1\_7$$

$$E1\_7 \stackrel{def}{=} E1\_8 + E1\_9$$

$$E1\_8 \stackrel{def}{=} c2.E1\_2$$

$$E1\_9 \stackrel{def}{=} c3.E1\_10$$

$$E1\_10 \stackrel{def}{=} \overline{F\_INICFpos}.E1\_Assoc$$

E para a entidde E2:

$$E2 \stackrel{def}{=} E2\_1$$

$$E2\_1 \stackrel{def}{=} E2\_2 + E2\_3$$

$$E2\_2 \stackrel{def}{=} E2$$

$$E2\_3 \stackrel{def}{=} c1.E2\_4$$

$$E2\_4 \stackrel{def}{=} \tau.E2\_5 + \tau.E2\_6$$

$$E2\_5 \stackrel{def}{=} \overline{c2}.E2\_2$$

$$E2\_6 \stackrel{def}{=} \overline{F\_INIIN}.E2\_7$$

$$E2\_7 \stackrel{def}{=} F\_INIRPneg.E2\_8 + F\_INIRPpos.E2\_9$$

$$E2\_8 \stackrel{def}{=} \overline{c2}.E2\_2$$

$$E2\_9 \stackrel{def}{=} \overline{c3}.E2\_10$$

$$E2\_10 \stackrel{def}{=} E2\_Assoc$$

A simplificação dos comportamentos de E1 e de E2 pode prosseguir através da redução do uso de identificadores. Com essa redução obtém-se para a entidade E1:

$$E1 \stackrel{def}{=} F\_INIRQ.E1\_1$$

$$E1\_1 \stackrel{def}{=} \tau.E1\_2 + \tau.E1\_3$$

$$E1\_2 \stackrel{def}{=} \overline{F\_INICFneg}.E1$$

$$E1\_3 \stackrel{def}{=} \overline{c1}.E1\_4$$

$$E1\_4 \stackrel{def}{=} c2.E1\_2 + c3.E1\_5$$

$$E1\_5 \stackrel{def}{=} \overline{F\_INICFpos}.E1\_Assoc$$

E para a entidade E2:

$$E2 \stackrel{def}{=} c1.E2\_1$$

$$E2\_1 \stackrel{def}{=} \tau.E2\_2 + \tau.E2\_3$$

$$E2\_2 \stackrel{def}{=} \overline{c2}.E2$$

$$E2\_3 \stackrel{def}{=} \overline{F\_INIIN}.E2\_4$$

$$E2\_4 \stackrel{def}{=} F\_INIRPneg.E2\_2 + F\_INIRPpos\text{-}E2\_5$$

$$E2\_5 \stackrel{def}{=} \overline{c3}.E2\_Assoc$$

As sincronizações das entidades de protoclo E1 e E2 correspondem às trocas de UDPs. A sincronização na porta c1 corresponde à UDP IRNIRQ, a sincronização na porta c2 corresponde à UDP INIRPneg e a sincronização na porta c3 corresponde à UDP INIRPpos.

Denotando por R a operação de rerrotulação que altera o nome das portas c1, c2 e c3 para associá-las à troca de UDPs,

$$R = [INIRQ,\ INIRPneg,\ INIRPpos\backslash c1,\ c2,\ c3]$$

A rerrotulação R das entidades de protocolo E1 e E2 fornece o refinamento do provedor de serviços Serv apresentado na Fig. 6.14.

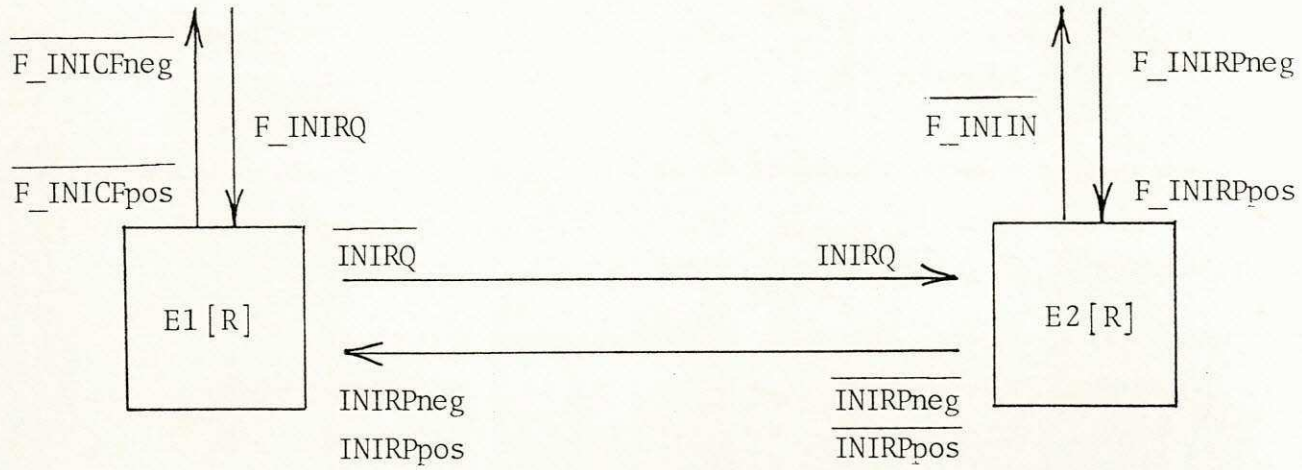


Figura 6.14 - Entidades de protocolo E1[R] e E2[R].

O comportamento de E1[R] é dado por

$$E1[R] \stackrel{def}{=} F\_INIRQ.E1\_1[R]$$

$$E1\_1[R] \stackrel{def}{=} \tau.E1\_2[R] + \tau.E1\_3[R]$$

$$E1\_2[R] \stackrel{def}{=} \overline{F\_INICFneg}.E1[R]$$

$$E1\_3[R] \stackrel{def}{=} \overline{INIRQ}.E1\_4[R]$$

$$E1\_4[R] \stackrel{def}{=} INIRPneg.E1\_2[R] + INIRPpos.E1\_5[R]$$

$$E1\_5[R] \stackrel{def}{=} \overline{F\_INICFpos}.E1\_Assoc[R]$$

O comportamento de E2[R] é dado por

$$E2[R] \stackrel{def}{=} INIRQ.E2\_1[R]$$

$$E2\_1[R] \stackrel{def}{=} \tau.E2\_2[R] + \tau.E2\_3[R]$$

$$E2\_2[R] \stackrel{def}{=} \overline{INIRPneg}.E2[R]$$

$$E2\_3[R] \stackrel{def}{=} \overline{F\_INIIN}.E2\_4[R]$$

$$E2\_4[R] \stackrel{def}{=} F\_INIRPneg.E2\_2[R] + F\_INIRPpos.E2\_5[R]$$

$$E2\_5[R] \stackrel{def}{=} \overline{INIRPpos}.E2\_Assoc[R]$$

6.3 - Avaliação da abordagem transformacional

A abordagem transformacional pode ser empregada para a derivação de especificações de protocolo a partir de especificações de serviço. Tais derivações devem ser realizadas com o mínimo de participação dos projetistas.

Uma das características mais importantes desse tipo de abordagem é a utilização de regras de transformação. Elas levam a resultados sempre corretos quando são aplicadas no domínio para o qual foram definidas. Desse modo pode-se evitar a realização de provas de correção a posteriori.

O grande problema com esse tipo de abordagem, é que os algoritmos obtidos normalmente só podem ser aplicados a casos simples. Entretanto, uma intensa pesquisa nessa área vem sendo desenvolvida, visando a obtenção de algoritmos mais poderosos que possam ser aplicados a situações mais abrangentes.

Neste capítulo buscou-se dar uma contribuição no âmbito da abordagem transformacional. Para alcançar esse objetivo foram propostos algoritmos que permitem derivar especificações de protocolo a partir de especificações de serviço.

Os algoritmos apresentados têm aplicabilidade bastante restrita, exigindo que a especificação de serviço envolva apenas escolhas e sequenciamentos. Além disso, as escolhas não se podem dar entre eventos em sítios diferentes. Esses sítios são apenas dois e duas as entidades de protocolo que podem ser derivadas.

Não obstante a limitação dos algoritmos propóstos, procurou-se ilustrá-los com uma aplicação a um subconjunto de um protocolo real. Nesse exemplo derivou-se a especificação de uma parte do núcleo do protocolo FTAM da ISO.

CAPÍTULO VII

EXTENSÕES A CCS

Uma vez que CCS é uma linguagem recente e está em evolução, novas construções e novos operadores podem ser adicionados a CCS, visando aumentar a sua expressividade e consequentemente permitindo a descrição de novos aspectos.

No CCS clássico um agente pode sincronizar-se com o ambiente ou com outro agente. Tais sincronizações envolvem eventos atômicos (indivisíveis) que atingem, simultaneamente, o agente e o seu ambiente (no primeiro caso) ou os dois agentes comunicantes (no segundo caso).

Os agentes M1 e M2 (Fig. 7.1) dispõem, cada um, de uma porta receptora de eventos com o rótulo a.

Figura 7.1 - Agentes M1 e M2.

Apesar de M1 e M2 possuírem portas com o mesmo rótulo, ou ocorre um evento na porta a de M1 ou ocorre um evento na porta a de M2. O CCS clássico não admite que M1 e M2 sincronizem-se simultaneamente com o ambiente.

Realizando-se a composição de M1 e M2 com um outro agente M dotado da porta $\bar{a}$ (Fig. 7.2), ainda assim as sincronizações ocorrerão entre M1 e M ou entre M2 e M.

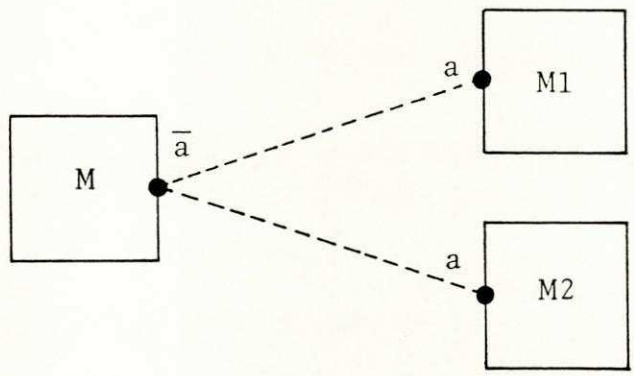


Figura 7.2 - Agente (M|M1|M2).

Embora suficiente em grande número de situações, esse tipo de sincronização apresenta limitações quando o especificador deseja descrever um sistema que, para executar uma ação, precisa que várias condições (expressas através do comportamento de agentes) sejam atendidas simultaneamente.

Portanto é útil considerar a possibilidade do ambiente sincronizar-se com vários agentes simultaneamente. É útil também considerar a possibilidade de um agente sincronizar-se com vários outros, através de um único evento que atinja a todos simultaneamente. Tais modos de sincronização são reunidos sob a denominação "multi-way synchronization".

7.1. CCS com "three-way synchronization"

Em [Miln 86] é proposto um operador de conjunção ($\&_{\{a\}}$) em que o ambiente sincroniza-se, ao mesmo tempo, com dois agentes que possuem portas receptoras com o mesmo rótulo a. Esse tipo de sincronização é denominado "three-way synchronization".

A construção apresentada na Fig. 7.3 permite que o ambiente (ou um agente) ofereça um evento na sua porta $\bar{a}$ e os agentes $M1[a_1/a]$ e $M2[a_2/a]$ participem (como receptores) desse evento. Para isso tornar-se possível, a

porta a de M1 foi rerrotulada como a_1 e a porta a de M2 foi rerrotulada como a_2 e ambas foram restritas. Além disso foi introduzido o agente

$$S \stackrel{def}{=} (\bar{a}_1 \bar{a}_2 a)^w$$

dotado da porta receptora a e das portas oferecedoras $\bar{a}_1$ e $\bar{a}_2$.

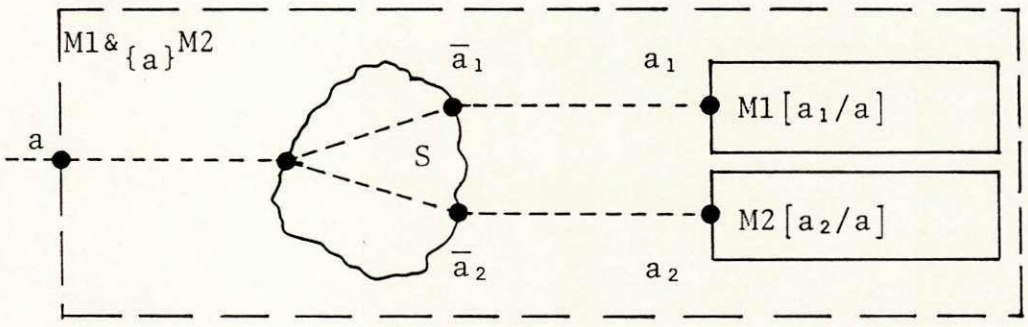


Figura 7.3 - Agente conjunto $M1\&_{\{a\}}M2$.

Do modo como foi definido, S permite que os eventos nas portas $\bar{a}_1$, $\bar{a}_2$ e a ocorram simultaneamente (e repetidamente, dependendo do valor do expoente w) quando $M1[a_1/a]$ e $M2[a_2/a]$ estão prontos. Na definição do agente S deve-se observar que as portas $\bar{a}_1$, $\bar{a}_2$ e a não correspondem a eventos distintos. Trata-se, na realidade, de um único evento atômico envolvendo essas três portas.

A construção apresentada na Fig. 7.3 pode ser descrita em CCS como

$$M1\&_{\{a\}}M2 = (M1[a_1/a] \mid M2[a_2/a] \mid (\bar{a}_1 \bar{a}_2 a)^w) \setminus \{a_1, a_2\}$$

onde os rótulos a_1 e a_2 são escolhidos de modo que não ocorram previamente em M1 nem em M2.

O produto $a_1 a_2 \bar{a}_1 \bar{a}_2 a = a$ é uma extensão à sincronização normalmente definida para duas portas complementares com o mesmo nome e que resulta no evento não observável τ (e.g . $c\bar{c} = \tau$). Esse produto estendido, juntamente com a restrição das portas a_1 e a_2, assegura que o agente conjunto $M1\&_{\{a\}}M2$ realiza a ação $a_1 a_2 \bar{a}_1 \bar{a}_2 a = a$ se e somente se ambos, M1 e M2 realizam o evento a simultaneamente ao sincronizarem-se com o ambiente.

No caso em que o agente conjunto $M1\&_{\{a\}}M2$ sincroniza-se com outro agente M dotado da porta $\bar{a}$ (Fig. 7.4), ocorre a ação $a_1 a_2 \bar{a}_1 \bar{a}_2 a \bar{a} = \tau$.

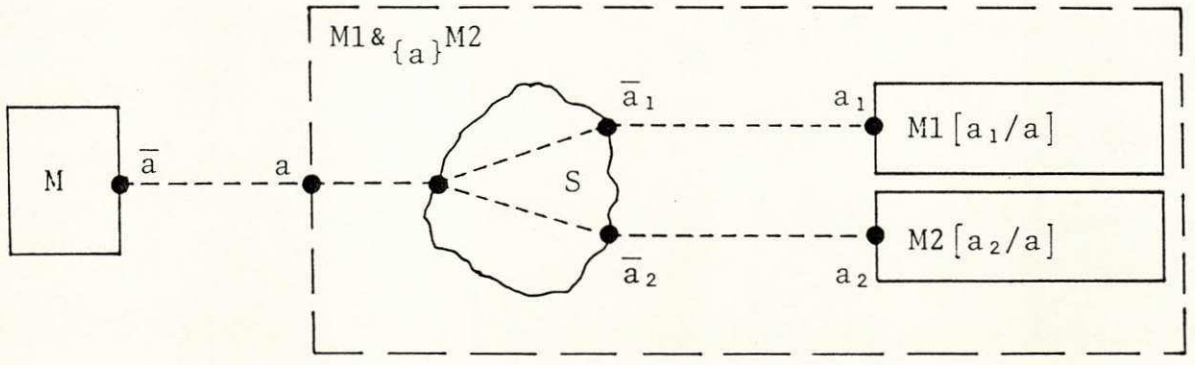


Figura 7.4 - $M1\&_{\{a\}}M2$ pode sincronizar-se com M.

As construções nas Figs. 7.3 e 7.4 permitem que os agentes M1 e M2 compartilhem eventos na porta receptora a (que pode ser vista como uma porta comum aos dois agentes). Esta construção pode ser estendida para o caso em que o comportamento se dá em uma porta oferecedora de eventos.

Os agentes N1 e N2 (Fig. 7.5) dispõem, cada um, de uma porta oferecedora de eventos com o co-rótulo $\bar{b}$.

A construção apresentada na Fig. 7.6 permite que o ambiente (ou um agente) receba um evento na sua porta b e os agentes N1 e N2 participem (como oferecedores) desse evento.

Figura 7.5 - Os agentes N1 e N2.

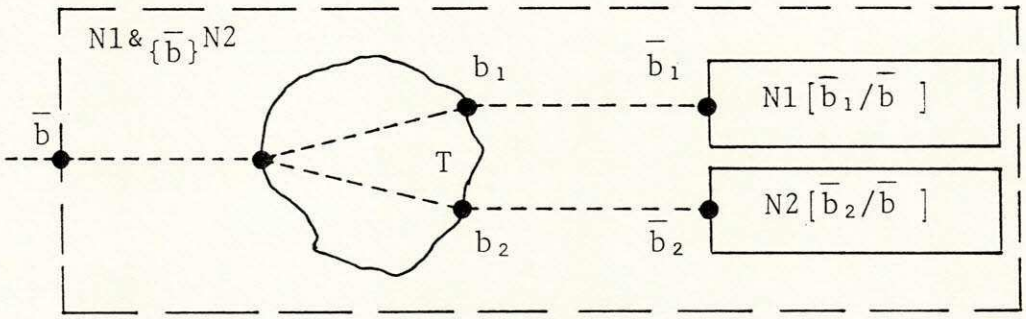


Figura 7.6 - O agente conjunto $N1\&_{\{b\}}N2$.

A construção apresentada na Fig. 7.6 pode ser descrita em CCS como

$$N1\&_{\{b\}}N2 = (N1[\bar{b}_1/\bar{b}] \mid N2[\bar{b}_2/\bar{b}] \mid (b_1 b_2 \bar{b})^{w}) \setminus \{b_1, b_2\}$$

onde os rótulos b_1 e b_2 são escolhidos de modo que não ocorram previamente em N1 nem em N2.

O produto estendido $b_1 b_2 \bar{b}_1 \bar{b}_2 \bar{b} = \bar{b}$, juntamente com a restrição das portas b_1 e b_2 assegura que o agente conjunto $N1\&_{\{b\}}N2$ realiza a ação $b_1 b_2 \bar{b}_1 \bar{b}_2 \bar{b} = \bar{b}$, se e somente se ambos, N1 e N2 realizam $\bar{b}$ simultaneamente ao sincronizarem-se com o ambiente.

No caso em que o agente conjunto $N1\&_{\{b\}}N2$ sincroniza-se com outro agente N dotado da porta receptora b (Fig. 7.7), ocorre a ação $b_1 b_2 \bar{b}_1 \bar{b}_2 \bar{b} b = \tau$.

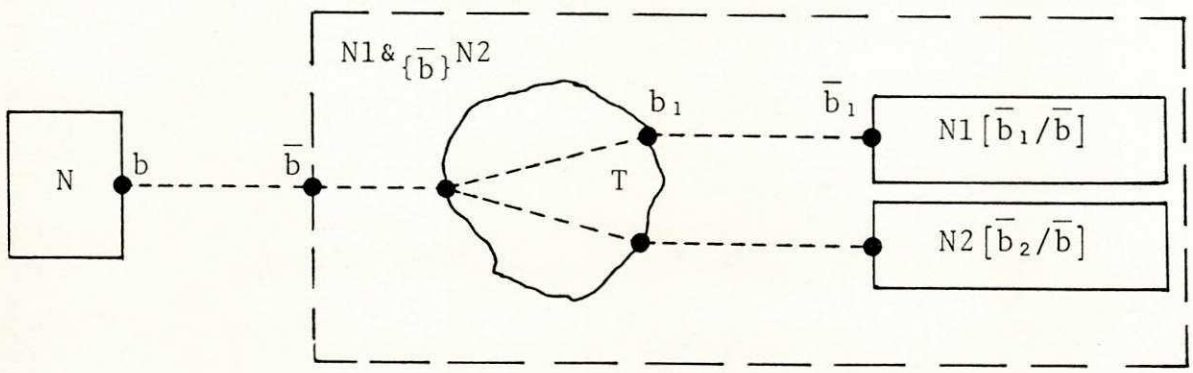


Figura 7.7 - $N1\&_{\{b\}}N2$ pode sincronizar-se com N.

O operador binário $\&_A$ força agentes a compartilharem ações nas portas pertencentes ao conjunto A, mas deixa esses agentes realizarem outras ações livremente. As regras de inferência que definem a semântica do operador $\&_A$ são as seguintes:

$$1)\ \frac{P \xrightarrow{a} P' \quad Q \xrightarrow{a} Q'}{P\&_A Q \xrightarrow{a} P'\&_A Q'} \qquad (a \in A)$$

isto é, se P pode realizar a ação a e, a partir desse evento, passar a comportar-se como P' e Q pode realizar a mesma ação e, após esse evento, passar a comportar-se como Q', então o agente conjunto $p\&_A Q$ (onde $a \in A$) pode realizar a ação a e passar a comportar-se como o agente conjunto $P'\&_A Q'$ (P e Q são atingidos).

$$2)\ \frac{P \xrightarrow{b} P'}{P\&_A Q \xrightarrow{b} P'\&_A Q} \ (b \notin A)$$

isto é, se P pode realizar a ação b e, a partir desse evento, passar a

comportar-se como P', então o agente conjunto $P\&_A Q$ (onde $b \notin A$) pode realizar a ação b e passar a comportar-se como o agente conjunto $P'\&_A Q$ (só o agente P é atingido).

$$3)\ \frac{Q \xrightarrow{b} Q'}{P\&_A Q \xrightarrow{b} P\ \&_A\ Q'} \qquad (b \notin A)$$

isto é, se Q pode realizar a ação b e, a partir desse evento, passar a comportar-se como Q', então o agente conjunto $P\&_A Q$ (onde $b \notin A$) pode realizar a ação b e passar a comportar-se como o agente conjunto $P\&_A Q'$ (só o agente Q é atingido).

Os casos em que dois agentes compartilham um evento (sincronizando-se com o ambiente ou com um terceiro agente) podem ser generalizados para envolverem vários agentes.

7.2. CCS com "multi-way synchronization"

O operador $\&_A$ pode ser utilizado para a realização de construções que permitem a vários agentes compartilharem eventos em uma mesma porta simultaneamente. Como $\&_A$ é um operador binário, ele precisa ser usado repetidamente nessas construções.

Os agentes M1, M2 e M3 (Fig. 7.8) dispõem, cada um, de uma porta receptora de eventos com o rótulo a.

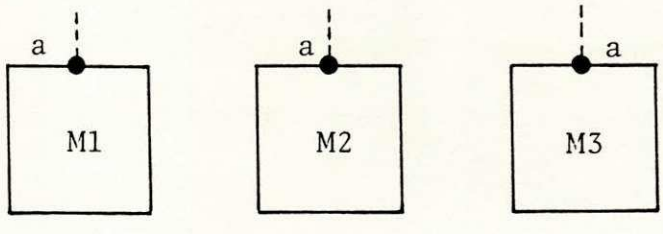


Figura 7.8 - Agentes M1, M2 e M3.

Usando o operador $\&_{\{a\}}$ pode-se realizar uma construção que permite aos agentes M1, M2 e M3 compartilharem um (ou mais) eventos na porta a (Fig. 7.9). Tal construção pode ser descrita em CCS como

$$(M1\&_{\{a\}}M2)\&_{\{a\}}M3 = (((M1[a_1/a]\,|\,M2[a_2/a]\,|\,S1)\backslash\{a_1,\ a_2\})$$
$$[s_1/a]\,|\,M3[a_3/a]\,|\,S2)\backslash\{s_1,\ a_3\})$$

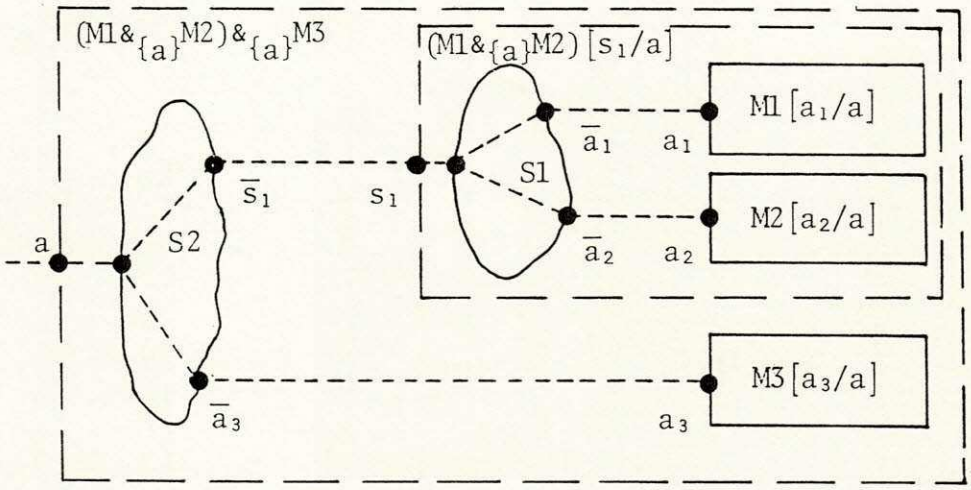


Figura 7.9 - Agente $(M1\&_{\{a\}}M2)\&_{\{a\}}M3$.

A construção apresentada na Fig. 7.9 é realizada em duas etapas. Na primeira etapa obtém-se $M1\&_{\{a\}}M2$ através da rerrotulação dos agentes M1 e M2. Nessa etapa é introduzido o agente

$$S1 \stackrel{def}{=} (\bar{a}_1\bar{a}_2a)^w$$

Na segunda etapa obtém-se $(M1\&_{\{a\}}M2)\&_{\{a\}}M3$ através da rerrotulação dos agentes $(M1\&_{\{a\}}M2)$ e M3.

Nessa etapa é introduzido o agente

$$S2 \stackrel{def}{=} (\bar{s}_1 \bar{a}_3 a)^w$$

A construção correspondente a três agentes (N1, N2 e N3) que compartilham eventos em uma porta oferecedora ($\bar{b}$) é apresentada na Fig. 7.10. Tal construção pode ser descrita em CCS como

$$(N1 \&_{\{b\}} N2) \&_{\{b\}} N3 = (((N1[\bar{b}_1/\bar{b}] \mid N2[\bar{b}_2/\bar{b}] \mid T1) \setminus \{b_1, b_2\}) \\ [\bar{t}_1/\bar{b}] \mid N3[\bar{b}_3/\bar{b}] \mid T2) \setminus \{t_1, b_3\})$$

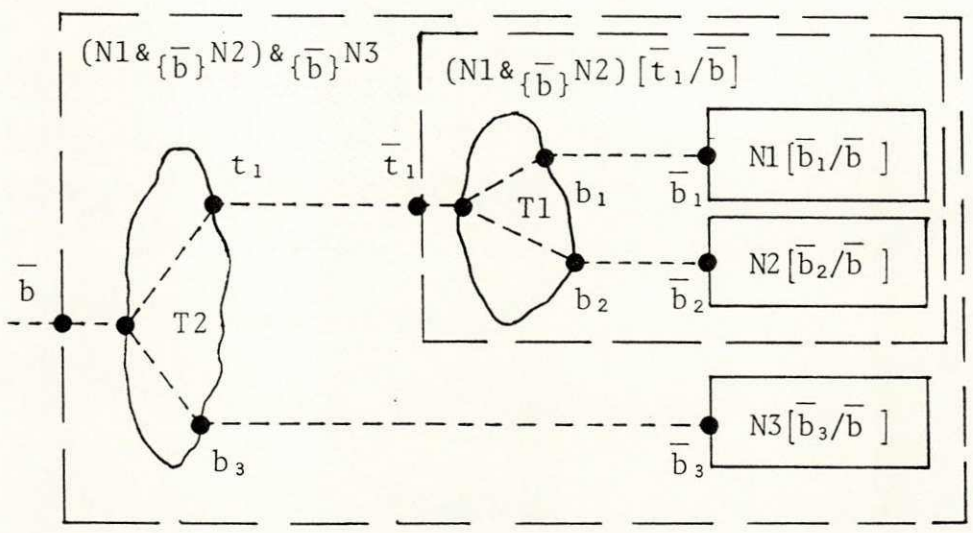


Figura 7.10 - Agente $(N1 \&_{\{b\}} N2) \&_{\{b\}} N3$.

A construção apresentada na Fig. 7.10 é igualmente realizada em duas etapas. Na primeira etapa obtém-se $N1 \&_{\{b\}} N2$ através da rerrotulação dos agentes N1 e N2. Nessa etapa é introduzido o agente

$$T1 \stackrel{def}{=} (b_1 b_3 \bar{b})^w$$

Na segunda etapa obtém-se $(N1 \&_{\{b\}} N2) \&_{\{b\}} N3$ através da rerrotulação dos agentes $(N1 \&_{\{b\}} N2)$ e N3. Nessa etapa é introduzido o agente

$$T2 \stackrel{def}{=} (t_1 b_3 \bar{b})^w$$

Para considerar o caso em que n agentes (M1,...,Mn) compartilham eventos na porta a, tem-se a construção

$$(M1\&_{\{a\}}M2)...\&_{\{a\}}Mn = (((M1[a_1/a]|M2[a_2/a]|S_1)\backslash\{a_1,\ a_2\})$$
$$[s_1/a]|...|Mn[a_n/a]|S_{n-1})\backslash\{s_{n-2},\ a_n\}$$

onde

$$S_1 \stackrel{def}{=} (\bar{a}_1\bar{a}_2 a)^w$$
$$S_i \stackrel{def}{=} (\bar{s}_{i-1}\bar{a}_{i+1} a)^w \qquad (2 \leq i \leq n-2)$$

Para considerar o caso em que n agentes (N1,...,Nn) compartilham eventos na porta $\bar{b}$, tem-se a construção

$$(N1\&_{\{\bar{b}\}}N2)...\&_{\{\bar{b}\}}Nn = (((N1[\bar{b}_1/\bar{b}]|N2[\bar{b}_2/\bar{b}]|T_1)\backslash\{b_1,\ b_2\})$$
$$[\bar{t}_1/\bar{b}]|...|Nn[\bar{b}_n/\bar{b}]|T_{n-1})\backslash\{t_{n-2},\ b_n\}$$

onde

$$T_1 \stackrel{def}{=} (b_1 b_2 \bar{b})^w$$
$$T_i \stackrel{def}{=} (t_{i-1} b_{i+1} \bar{b})^w \qquad (2 \leq i \leq n-2)$$

Uma situação diferente surge quando o conjunto de portas compartilhadas possui mais de um elemento (essas portas podem ser receptoras ou oferecedoras de eventos). Para ilustrar um casos desses, os agentes L1, L2 e F (Fig. 7.11) podem ser considerados.

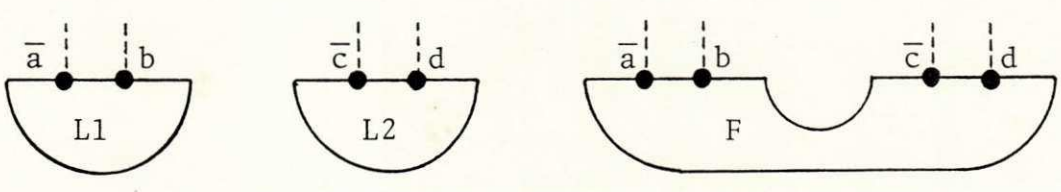


Figura 7.11 - Agentes L1, L2 e F.

Na Fig. 7.11 os agentes L1 e L2 não têm portas onde possam compartilhar eventos um com o outro, mas L1 e F podem compartilhá-los nas portas $\bar{a}$ e b enquanto que L2 e F podem compartilhá-los nas portas c e $\bar{d}$. Uma construção que permite aos agentes L1, L2 e F esse compartilhamento de eventos (Fig. 7.12) pode ser descrita em CCS como

$$(L1|L2)\&_U F = (L1[[a_1, b_1/a, b]|L2[c_1, d_1/c, d]| F[a_2, b_2, c_2, d_2/a, b, c, d]|H)\setminus \{a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2\}$$

onde $U = \{\bar{a}, b, \bar{c}, d\}$ e

$$H \stackrel{def}{=} a_1 a_2 \bar{a}.H + \bar{b}_1 \bar{b}_2 b.H + c_1 c_2 \bar{c}.H + \bar{d}_1 \bar{d}_2 d.H$$

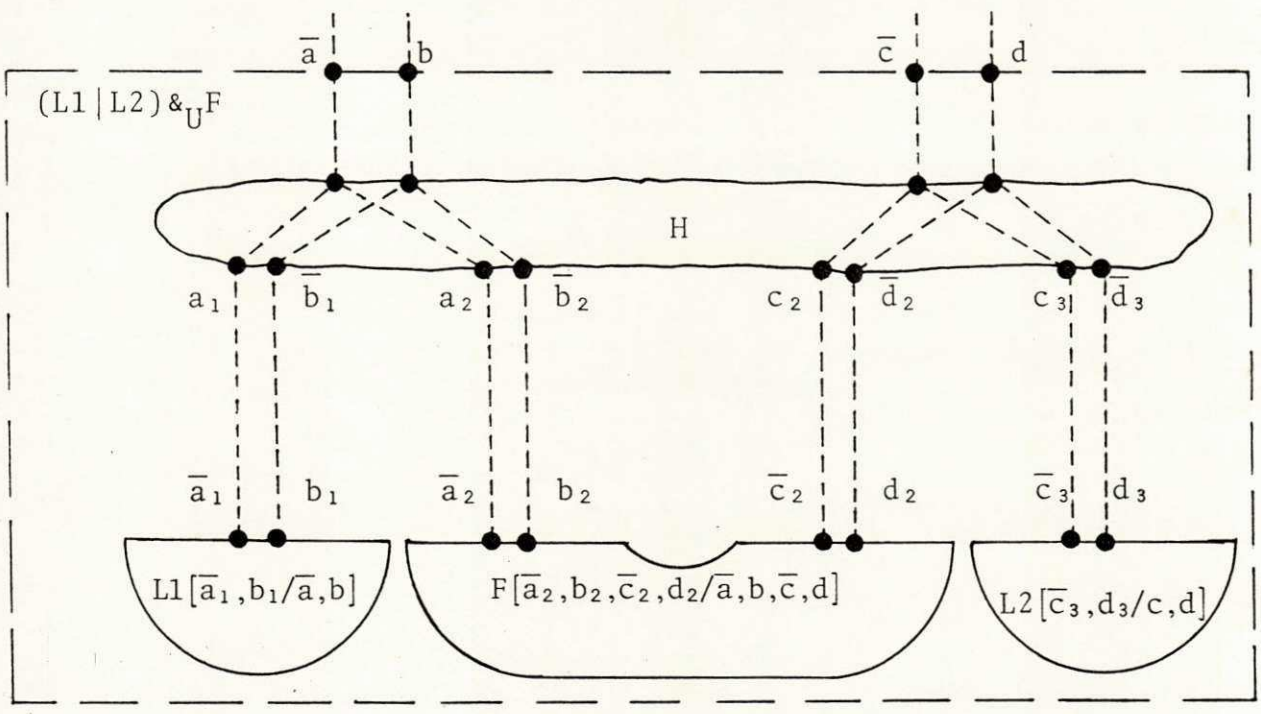


Figura 7.12 - Agente $(L1|L2)\&_U F$.

O agente H oferece permanentemente a escolha indeterminística entre

quatro produtos estendidos. Tais produtos permitem o compartilhamento de eventos nas portas $\bar{a}$, b, $\bar{c}$ e d.

A conjunção $(L1|L2)\&_U F$ pode ser utilizada como um modelo para a realização de especificações de serviço. Definindo-se

$$Serv = (L1|L2)\&_U F$$

os agentes L1 e L2 podem ser interpretados como condições (ou restrições) locais, impostas ao provedor de serviço Serv em dois sítios (sítio 1 e sítio 2, respectivamente), enquanto F pode ser interpretado como a restrição fim-a-fim (relativa à comunicação entre os dois sítios) (Fig. 7.13).

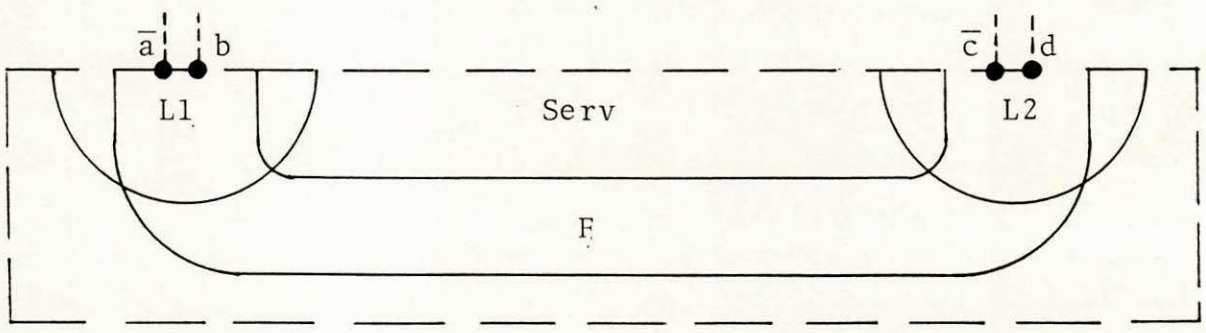


Figura 7.13 - Agente Serv.

7.3. Conjunção e a Lei da Expansão

Para o caso de agentes construídos com o emprego do operador de composição (|), somente dois tipos de ações podem ocorrer:

(1) uma ação $a \in A \cup \bar{A} \cup \{\tau\}$ envolvendo apenas um dos componentes;

(2) uma ação τ resultante da sincronização entre dois componentes com portas complementares.

No caso de agentes construídos com o emprego do operador de conjunção ($\&_C$), faz-se necessária uma nova forma para a Lei de Expansão, diferente daquelas apresentadas na seção 4.4 (para o CCS Básico e para o CCS Completo). Essa nova forma permite considerar o compartilhamento de ações entre agentes ("multi-way synchronization").

Considerando um sistema Q especificado através da conjunção de dois agentes Q1 e Q2, que compartilham eventos nas portas pertencentes ao conjunto C

$$Q = (Q1 \ \&_C \ Q2)$$

a expressão que fornece o desdobramento de Q em sincronizações puras (sem passagem de valor) e escolhas indeterminísticas (somas) é a Lei da Expansão para o CCS Básico com "multi-way synchronization".

$$\begin{aligned} Q \sim & \sum \{a.(Q1' \&_C Q2) \quad \text{onde } Q1 \xrightarrow{a} Q1' \text{ e } a \notin C\} \\ & + \sum \{b.(Q1 \&_C Q2') \quad \text{onde } Q2 \xrightarrow{b} Q2' \text{ e } b \notin C\} \\ & + \sum \{\tau.(Q1' \&_C Q2') \text{ onde } Q1 \xrightarrow{c} Q1', \ Q2 \xrightarrow{\overline{c}} Q2' \text{ e } c \notin C\} \\ & + \sum \{d.(Q1' \&_C Q2') \text{ onde } Q1 \xrightarrow{d} Q1', \ Q2 \xrightarrow{d} Q2' \text{ e } d \in C\} \end{aligned}$$

Com o primeiro somatório exprime-se a possibilidade de ocorrências de eventos que atingem apenas o agente Q1. Esses eventos podem ser sincronizações com o ambiente (não compartilhadas com Q2) ou eventos não observáveis τ.

Com o segundo somatório exprime-se a possibilidade de ocorrências de eventos que atingem apenas o agente Q2. Esses eventos são do mesmo tipo que aqueles correspondentes ao primeiro somatório.

Com o terceiro somatório exprime-se a possibilidade de sincronização entre Q1 e Q2 através de portas complementares, resultando no evento não observável τ.

Com o último somatório exprime-se a possibilidade do compartilhamento

de eventos ("multi-way synchronization") em portas receptoras ou oferecedoras, que pertencem ao conjunto C.

O agente Serv (apresentado na seção 7.2) pode oferecer um serviço de comunicação bidirecional confiável entre dois usuários. O usuário localizado no sítio 1 (Questionador) envia uma pergunta para o usuário localizado no sítio 2 (Respondedor). Este, por sua vez, envia a resposta para o Questionador (Fig. 7.14).

Para o agente Serv, as restrições locais L1 e L2 e a restrição fim-a-fim F (Fig. 7.13) podem ser definidas como segue.

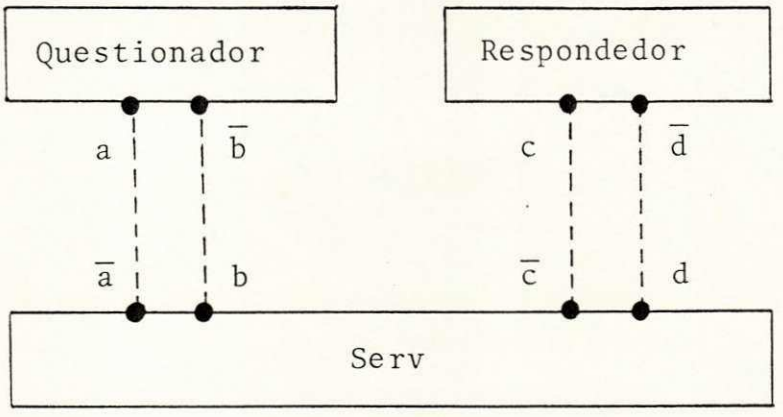


Figura 7.14 - Sistema Pergunta_e_Resposta.

$$L1 \stackrel{def}{=} b.\overline{a}.NIL$$

indicando que Serv deve receber a pergunta do Questionador (na porta b) e depois entregar a ele a resposta (na porta $\overline{a}$). L1 tem comportamento finito.

$$L2 \stackrel{def}{=} \overline{c}.d.NIL$$

indicando que Serv deve entregar a pergunta ao Respondedor (na porta $\overline{c}$) e

depois receber dele a resposta (na porta d). L2 tem comportamento finito.

$$F \stackrel{def}{=} b.\bar{c}.F + d.\bar{a}.F$$

indicando que Serv não pode receber uma nova mensagem (pergunta ou resposta) antes de completar a transmissão da mensagem anterior. F não impõe comportamento finito a Serv.

De acordo com a Lei de Expansão para o CCS Básico com "multi-way synchronization" a conjunção

$$Serv = (L1 \mid L2)\&_U F$$

onde $U = \{\bar{a}, b, \bar{c}, d\}$, apresenta a seguinte expansão:

$$Serv \sim b.Serv1$$

onde

$$Serv1 = (\bar{a}.NIL \mid \bar{c}.d.NIL)\&_U \bar{c}.F$$

indicando que Serv recebeu a pergunda do Questionador.

$$Serv1 \sim \bar{c}.Serv2$$

onde

$$Serv2 = (\bar{a}.NIL \mid d.NIL)\&_U (b.\bar{c}.F + d.\bar{a}.F)$$

indicando que Serv entregou a pergunta ao Respondedor.

$$Serv2 \sim d.Serv3$$

onde

$$Serv3 = (\bar{a}.NIL \mid NIL)\&_U (\bar{a}.F)$$

indicando que Serv recebeu a resposta do Respondedor.

$$Serv3 = \overline{a}.Serv4$$

onde

$$Serv4 \sim (NIL|NIL)\&_U F$$

indicando que Serv entregou a resposta ao Questionador, tornando-se inativo.

O comportamento de Serv pode ser apresentado graficamente na forma de uma árvore de sincronização (Fig. 7.15).

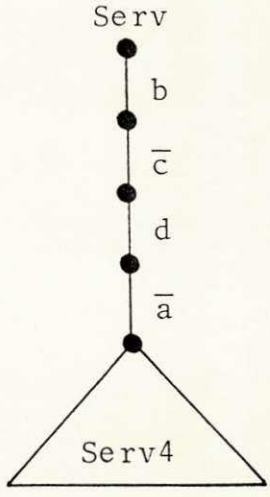


Figura 7.15 - Árvore de sincronização Serv.

CAPÍTULO VIII

ESTILOS DE ESPECIFICAÇÃO EM CCS COM "MULTI-WAY SYNCHRONIZATION"

As diferentes especificações de um mesmo sistema, obtidas ao longo da trajetória de design, podem ser avaliadas de acordo com três princípios definidos em [ViSc 88]; ortogonalidade, generalidade e flexibilidade.

A ortogonalidade é o princípio segundo o qual aspectos independentes de um sistema devem ser especificados independentemente. No caso da especificação do serviço oferecido por um sistema distribuído, o especificador deve procurar descrever as restrições impostas sobre o serviço em cada sítio (restrições locais) isoladamente das restrições impostas à comunicação entre os sítios (restrições fim-a-fim).

Para atender ao princípio de generalidade, o especificador deve procurar definir construções gerais que possam ser reutilizadas. Expressões de comportamento parametrizadas variam as suas características quando os parâmetros assumem valores particulares. Portanto, essas expressões podem ser reutilizadas em diferentes situações. Agentes básicos predefinidos podem ser rerrotulados, para serem reutilizados em diferentes partes de um sistema, como foi o caso da especificação de um controlador de exclusão mútua apresentado no Capítulo V.

A flexibilidade de uma especificação diz respeito à facilidade com que ela pode ser submetida a adaptações e extensões, depois de pronta, para atender a novas necessidades dos usuários do sistema. A especificação parcial de um sistema deve ser estruturada de tal modo que possa ser completada sem dificuldade.

Os princípios de ortogonalidade, generalidade e flexibilidade podem

ser empregados na análise de diferentes estilos de especificação. Tais estilos referem-se ao modo como cada especificador realiza a descrição de sistemas a partir dos recursos descritivos da linguagem de especificação adotada.

Para simplificar a avaliação dos estilos de especificação segundo os princípios acima, quatro estilos básicos foram definidos em [ViSc 88]: estilo monolítico, estilo orientado para restrições, estilo orientado para recursos e estilo orientado para estados. Cada um desses estilos dá ênfase a um conjunto de características normalmente presentes em especificações reais.

Os estilos monolítico e orientado para restrições são do tipo "extensional", isto é, descrevem o que o sistema faz sem apresentar a estrutura interna desse sistema. Portanto, com tais estilos só os aspectos observáveis dos sistemas podem ser descritos.

Os estilos orientado para recursos e orientado para estados são do tipo "intensional", isto é, descrevem como o sistema realiza as suas atividades, exibindo aspectos internos dos sistemas. No caso do estilo orientado para recursos a própria estrutura interna do sistema é revelada.

A fim de ilustrar os estilos básicos de especificação, na Fig. 8.1 é apresentado um sistema distribuído e constituído de um Produtor de Dados, um Consumidor de Dados e um Serviço de Comunicação confiável. O Produtor e o Consumidor são usuários do Serviço de Comunicação.

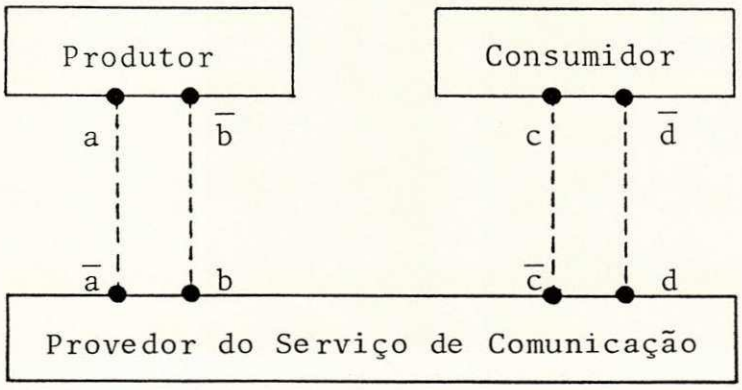


Figura 8.1 - Sistema Produtor-Consumidor.

O controle do fluxo de dados é exercido pelo Consumidor. O Produtor só envia um novo dado após ter recebido uma confirmação de que o dado anterior foi consumido (mecanismo explícito de controle de fluxo).

8.1. Estilo monolítico

No estilo monolítico busca-se uma especificação que descreva apenas o sequenciamento e a escolha das ações de um sistema. Portanto, os operadores dinâmicos do CCS clássico são suficientes para expressar o comportamento desse sistema.

A especificação inicial do Serviço de Comunicação (Fig. 8.1), no estilo monolítico, é dada pela seguinte expressão de comportamento:

$$SM \stackrel{def}{=} b.\bar{c}.d.\bar{a}.SM$$

Em uma especificação monolítica não há estruturação, o que obriga o projetista a considerar sempre o comportamento global do sistema que está sendo desenvolvido.

Para sistemas simples, esse estilo pode levar a especificações concisas onde a ausência de estruturação não chega a constituir um problema. Entretanto, no desenvolvimento de um sistema complexo, que normalmente envolve uma equipe de projetistas, a estruturação da especificação inicial do sistema torna-se fundamental.

Os princípios de ortogonalidade e flexibilidade não são atendidos pelo estilo monolítico devido à ausência de estruturação desse estilo. Pelo mesmo motivo, o princípio de generalidade é atendido apenas parcialmente.

Como o estilo monolítico impede que os aspectos independentes de um sistema sejam tratados de modo isolado, a reutilização das especificações realizadas nesse estilo só pode se dar a nível global, isto é, considerando o sistema todo e não a nível de cada aspecto particular do sistema. Por exemplo,

no caso da especificação do serviço oferecido por um sistema distribuído, os aspectos locais não recebem tratamento independente dos aspectos fim-a-fim do mesmo sistema. Esse fato obriga à reutilização da especificação de serviço como um todo (se possível), impedindo que os aspectos locais sejam isolados e as especificações correspondentes reutilizadas.

8.2. Estilo orientado para restrições

No estilo orientado para restrições busca-se uma especificação estruturada para o sistema. Essa especificação é obtida identificando-se aspectos desse sistema que podem ser considerados isoladamente. A especificação do sistema é obtida através de uma combinação desses aspectos.

No Serviço de Comunicação SC da Fig. 8.2 dois aspectos (restrições) locais e uma restrição fim-a-fim estão representadas.

A restrição local L1, relativa ao sítio do Produtor, indica que não pode haver a entrega de uma confirmação (na porta $\bar{a}$) sem antes ter ocorrido o recebimento de um dado (na porta b):

$$L1 \stackrel{def}{=} b.\bar{a}.L1$$

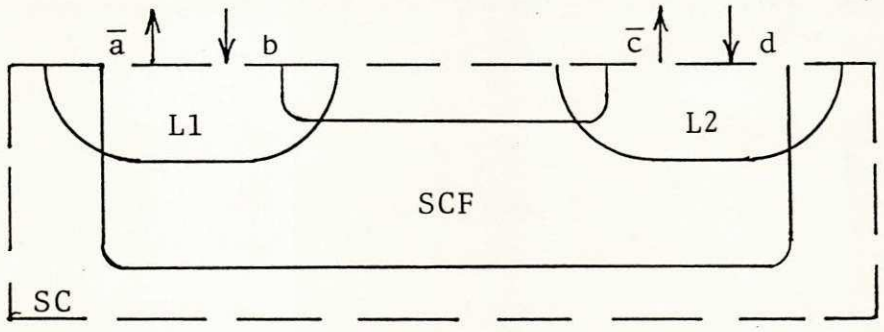


Figura 8.2 - Restrições de SC.

A restrição local L2, relativa ao sítio do Consumidor, indica que não pode haver o recebimento de uma confirmação (na porta b) sem antes ter ocorrido a entrega de um dado (na porta $\overline{c}$):

$$L2 \stackrel{def}{=} \overline{c}.d.L2$$

A restrição fim-a-fim SCF pode ser estruturada considerando-se os aspectos do fluxo de dados (F1) isoladamente dos aspectos do fluxo de confirmações (F2) (Fig. 8.3):

$$SCF = (F1 | F2)$$

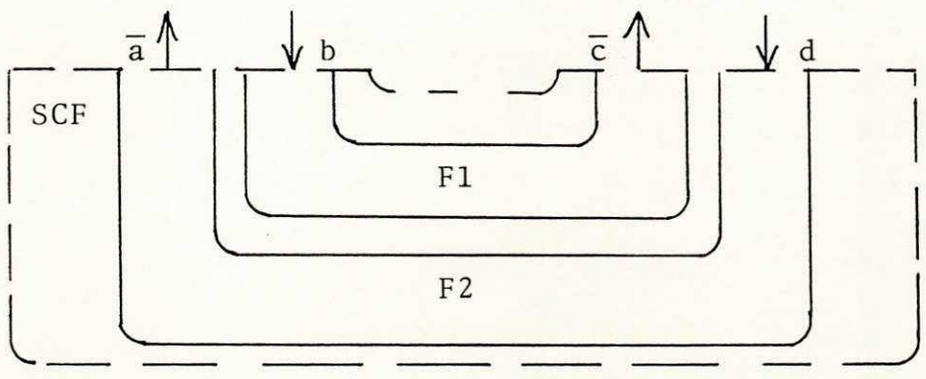


Figura 8.3 - Estrutura de SCF.

A restrição F1 indica que um dado não pode ser entregue ao Consumidor (na porta $\overline{c}$) sem antes ter sido recebido do Produtor (na porta b):

$$F1 \stackrel{def}{=} b.\overline{c}.F1$$

A restrição F2 indica que uma confirmação não pode ser entregue ao Produtor (na porta $\overline{a}$) sem antes ter sido recebida do Consumidor (na porta d):

$$F2 \stackrel{def}{=} d.\overline{a}.F2$$

As restrições L1 e L2 não compartilham eventos entre si. Tampouco as

restrições F1 e F2. Desse modo, a especificação inicial de SC pode ser expressa, no estilo orientado para restrições, pela conjunção

$$SC = (L1 \mid L2) \&_C (F1 \mid F2)$$

onde $C = \{\overline{a}, b, \overline{c}, d\}$ é o conjunto de portas compartilhadas por $(L1 \mid L2)$ e $(F1 \mid F2)$.

O serviço de comunicação SC é equivalente quanto à observação ao serviço de comunicação SM ($SC \approx SM$), uma vez que há entre ambos uma relação de bissimulação (Fig. 8.4.).

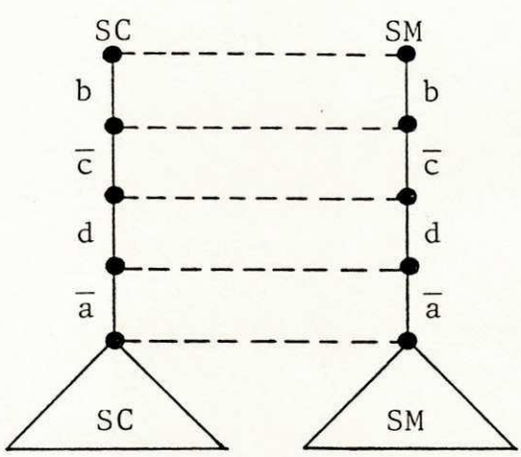


Figura 8.4 - Bissimulação entre SC e SM.

O estilo orientado para restrições pode levar a especificações mais extensas que o estilo monolítico, mas a sua estruturação permite um melhor entendimento das peculiaridades do sistema. Tal entendimento se dá através da identificação e do isolamento das restrições que são impostas ao sequenciamento de eventos.

No caso de um sistema complexo as expressões de comportamento, que representam as restrições, podem ser desenvolvidas independentemente pela

equipe de especificadores. Essas restrições podem ainda ser reutilizadas para a especificação dos recursos que constituem a estrutura interna desse sistema.

Graças à estruturação imposta às especificações, o estilo orientado para restrições atende aos três princípios de design citados neste capítulo. No caso do princípio de ortogonalidade, os aspectos independentes são especificados através de restrições independentes cuja conjunção ($\&_A$) define o comportamento do sistema. O princípio de generalidade é atendido, não apenas a nível global (sistema completo) como também a nível dos aspectos independentes do sistema (restrições). Esse estilo permite também a produção de especificações flexíveis na medida em que extensões ao sistema especificado podem ser descritas através de novas restrições que, por sua vez, podem ser facilmente incorporadas à especificação anterior.

8.3. Estilo orientado para recursos

No estilo orientado para recursos busca-se descrever, estruturadamente, um sistema a partir dos seus componentes internos (recursos). Uma vez que a intercomunicação desses recursos é escondida do observador, o emprego do operador de restrição (\) caracteriza esse estilo.

No Serviço de Comunicação SR (Fig. 8.5) três recursos estão representados: E1, E2 e M. Os recursos E1 e E2 correspondem às entidades de protocolo do sítio do Produtor e do sítio do Consumidor, respectivamente. E1 e E2 comunicam-se utilizando o meio de comunicação half-duplex confiável M.

A entidade E1 comporta-se como

$$E1 \stackrel{def}{=} b.\overline{b1}.a1.\overline{a}.E1$$

A entidade E2 comporta-se como

$$E2 \stackrel{def}{=} c1.\overline{c}.d.\overline{d1}.E2$$

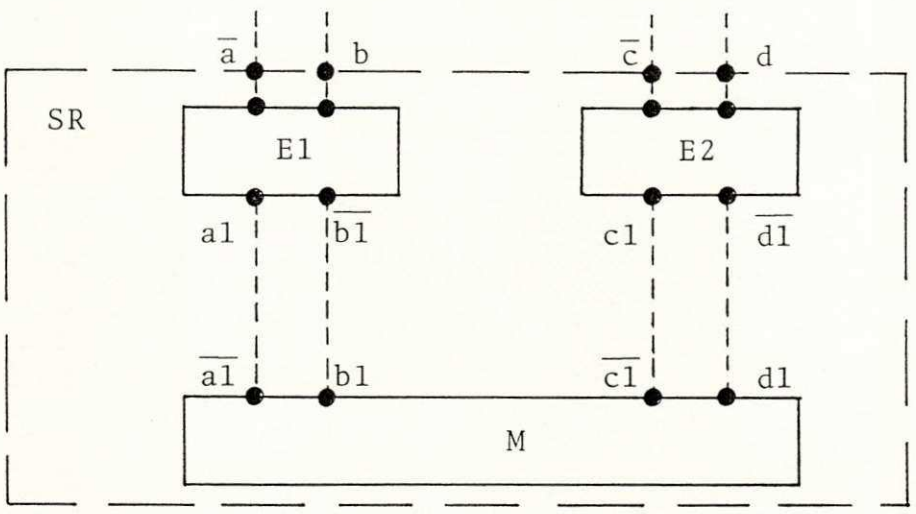


Figura 8.5 - Recursos de SR.

O meio de comunicação M comporta-se como

$$M \stackrel{def}{=} b1.\overline{c1}.M + d1.\overline{a1}.M$$

O serviço de comunicação SR é equivalente quanto a observação ao serviço de comunicação SM (SR ≈ SM), uma vez que há entre ambos uma relação de bissimulação (Fig. 8.6).

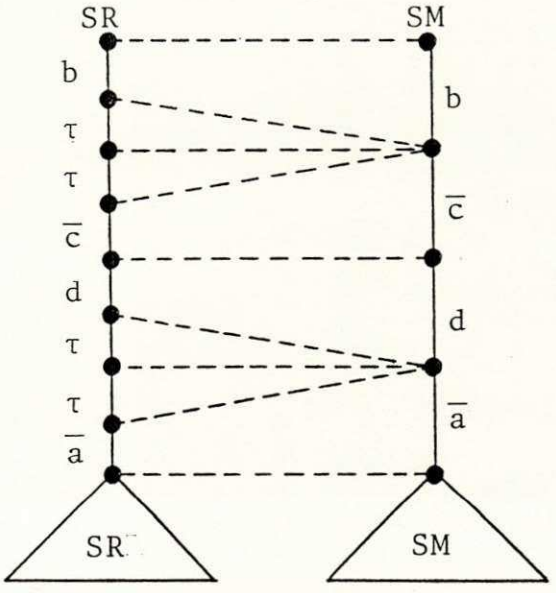


Figura 8.6 - Bissimulação entre SR e SM.

A descrição de um sistema no estilo orientado para recursos pode ser obtida pelo refinamento de uma especificação mais abstrata desse sistema. Pode-se desenvolver uma abordagem, baseada em refinamentos sucessivos, na qual os dois estilos estruturados (orientado para restrições e orientado para recursos) sejam utilizados, alternadamente, até a obtenção da especificação final.

Assim como o estilo orientado para restrições, o estilo orientado para recursos atende a todos os princípios de design. No caso do princípio de ortogonalidade, os aspectos independentes são tratados através de recursos independentes. Esses recursos permitem que o princípio de generalidade seja atendido não apenas a nível global (sistema completo) como também a nível dos seus componentes (recursos do sistema). A flexibilidade oferecida por esse estilo permite que acréscimos e modificações afetem exclusivamente alguns recursos, deixando intactos os outros recursos que constituem o restante do sistema.

8.4. Estilo orientado para estados

No estilo orientado para estados busca-se descrever um sistema a partir dos estados que esse sistema assume e das ações que pode executar em cada estado. Nas expressões que definem o sistema, cada estado é associado a um identificador de comportamento.

O Serviço de Comunicação SE, capaz de transmitir dados e confirmações entre um Produtor e um Consumidor, pode ser expresso no estilo orientado para estados como

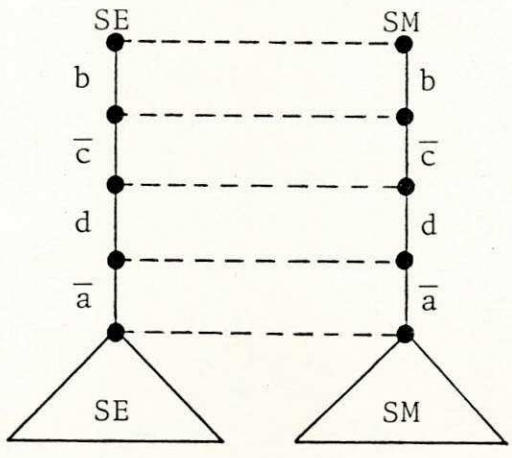
$$SE \stackrel{def}{=} b.S1$$

$$S1 \stackrel{def}{=} \overline{c}.S2$$

$$S2 \stackrel{def}{=} d.S3$$

$$S3 \stackrel{def}{=} \overline{a}.SE$$

O serviço de comunicação SE é equivalente quanto à observação ao serviço de comunicação SM (SE ≈ SM), uma vez que há entre ambos uma relação de bissimulação (Fig. 8.7).



Figira 8.7 - Bissimulação entre SE e SM.

No estilo orientado para estados, cada estado representa uma situação interna do sistema. Esse estilo, assim como o monolítico, não permite a estruturação das especificações. Entretanto, esses estilos podem ser utilizados para a especificação de sistemas simples e também para a especificação de agentes básicos, que não são mais refinados.

De modo semelhante ao estilo monolítico, os princípios de ortogonalidade e flexibilidade não são atendidos pelo estilo orientado para estados. O princípio da generalidade só é atendido a nível global uma vez que a ausência de estruturação não permite considerar partes isoladas de um mesmo sistema.

Dos quatro estilos básicos, somente o estilo orientado para restrições e o estilo orientado para recursos atendem aos três princípios de design considerados neste capítulo. Como consequência esses dois estilos devem gozar da preferência dos projetistas.

A especificação do serviço oferecido por um sistema distribuído pode ser realizada no estilo orientado para restrições. A especificação do protocolo correspondente (um refinamento da especificação de serviço) pode ser realizada no estilo orientado para recursos, onde cada recurso é, por sua vez, especificado no estilo orientado para restrições.

Assim um estilo pode completar o outro na trajetória de design de um sistema. Por ser do tipo "extensional", o estilo orientado para restrições deve ser utilizado para descrever o comportamento observável de cada sistema enquanto que, por ser do tipo "intensional", o estilo orientado para recursos deve ser utilizado para descrever o refinamento de cada sistema. A alternância dos dois estilos pode prosseguir até a especificação final do sistema. Esta pode ser realizada em termos de agentes básicos predefinidos no estilo monolítico ou no estilo orientado para estados.

CAPÍTULO IX

ABORDAGEM PROPOSTA

O problema que se pretende resolver pode ser estabelecido, informalmente, do seguinte modo:

> "realize o design de um mecanismo lógico distribuído (entidades de protocolo) que, utilizando um serviço disponível (fornecido por um sistema existente ou supostamente existente) ofereça um serviço desejado."

Portanto, o problema pressupõe duas especificações formais, a do serviço disponível e a do serviço desejado, e pede uma terceira especificação formal, a de entidades de protocolo (Fig. 9.1).

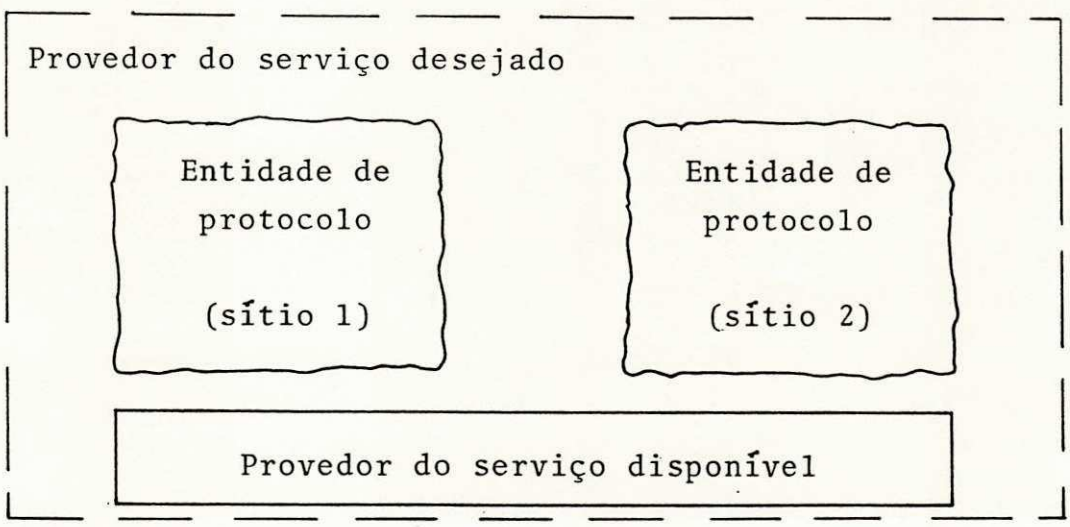


Figura 9.1 - Visualização do problema.

Por exemplo, seja um sistema distribuído constituído de dois sítios (sítio 1 e sítio 2). Dispõe-se de um serviço confiável de transmissão de dados entre esses sítios (serviço disponível SP) e deseja-se um serviço de comunicação do tipo Questão-Resposta, onde questões possam ser transmitidas do sítio 1 para o sítio 2 e as respostas correspondentes sejam transmitidas no sentido contrário. Uma nova questão só é transmitida após a transmissão da resposta correspondente à questão anterior (serviço desejado SJ).

O serviço SJ pode ser obtido através da ação conjunta de duas entidades de protocolo, E1 (no sítio 1) e E2 (no sítio 2), que executam o protocolo Questão-Resposta utilizando o serviço SP. A entidade E1 comunica-se diretamente com o usuário que formula a questão (usuário Questionador), enquanto que a entidade E2 comunica-se diretamente com o usuário que formula a resposta (usuário Respondedor) (Fig. 9.2).

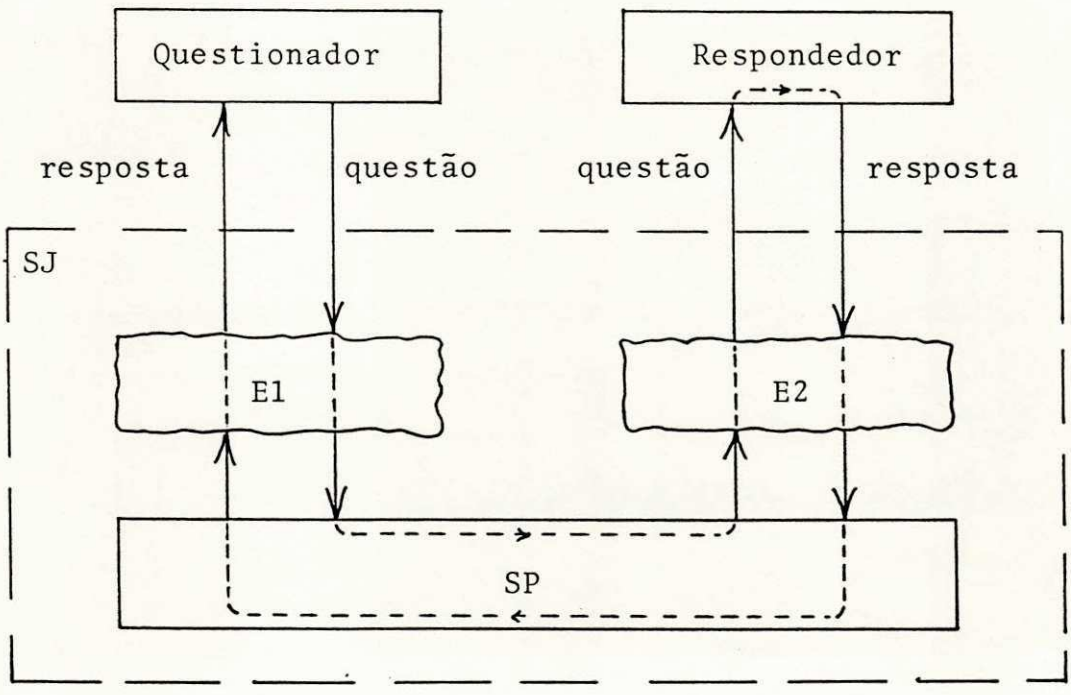


Figura 9.2 - Sistema Questão-Resposta.

Formalmente, o problema pode ser apresentado com o emprego de uma equação de contexto [NoEv 88] [Parr 89] [Riso 89] [Riso 90]:

$$(SP|(E1|E2))\backslash A \approx SJ \qquad \text{(eq. 9.1)}$$

onde SP e SJ representam o serviço disponível e o serviço desejado, respectivamente, E1 e E2 representam as entidades de protocolo e A representa o conjunto de portas para a comunicação das entidades de protocolo E1 e E2 com o provedor do serviço disponível SP. Essas portas são escondidas do observador através da operação de restrição (\A).

9.1. Etapas da abordagem

A abordagem proposta neste capítulo visa fornecer uma disciplina para o design de sistemas distribuídos e protocolos de comunicação. De acordo com essa disciplina a especificação das entidades de protocolo pode ser obtida, a partir das especificações do serviço desejado e do serviço disponível, em três etapas:

1ª) estruturação do serviço desejado em camadas,

2ª) especificação do protocolo correspondente a cada camada de serviço e

3ª) refinamento sucessivo das entidades de protocolo em cada camada.

9.1.1. Estruturações do serviço desejado

No caso de sistemas complexos, pode-se decompor o problema dado pela (eq. 9.1) em vários outros problemas com um grau menor de complexidade. Para tal são realizados dois passos:

Passo 1: decomponha o serviço desejado em camadas de serviço aninhadas (Fig. 9.3).

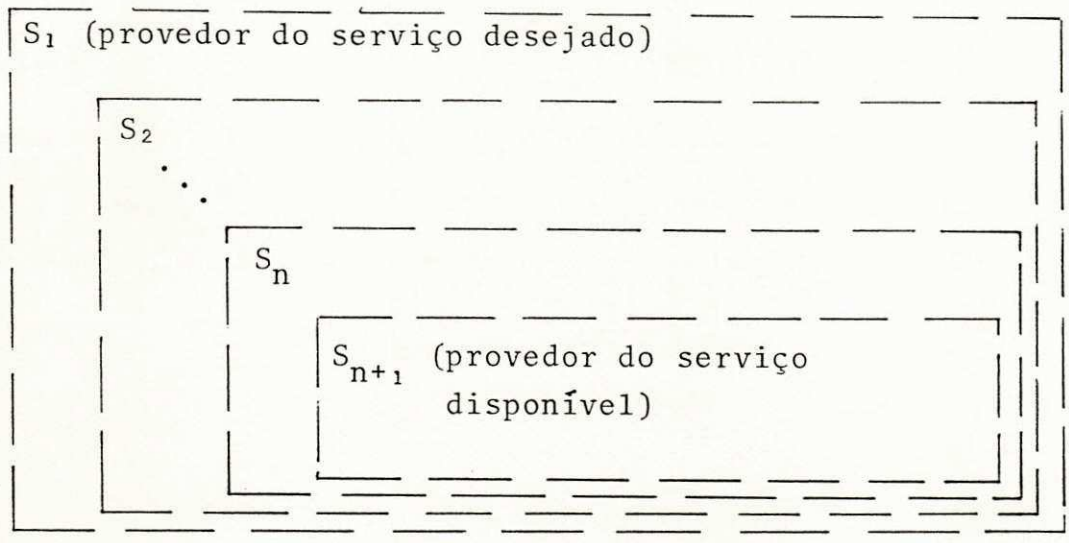


Figura 9.3 - Aninhamento das camadas de serviço.

O aninhamento das camadas de serviço é tal que a sofisticação dos serviços oferecidos diminui gradativamente, a partir do serviço desejado, até o ponto em que o serviço disponível é atingido.

Um dos exemplos mais interessantes de um sistema complexo, estruturado em camadas de serviços aninhados é o Modelo Básico de Referência OSI da ISO. Entretanto, outros sistemas complexos podem ser estruturados da mesma maneira. No caso de sistemas simples a decomposição em camadas não é necessária.

Passo 2: utilizando o estilo orientado para restrições, especifique cada um dos serviços obtidos no passo 1. Tais especificações de serviço devem seguir o modelo baseado em restrições locais (L1 e L2) e restrições fim-a-fim (F) apresentado na Fig. 9.4.

Devido à sua reduzida complexidade, o sistema Questão-Resposta não precisa ser decomposto em camadas. Portanto ao final do passo 2 obtêm-se as especificações de serviço, no estilo orientado para restrições, correspondentes a duas camadas de serviço aninhadas (Fig. 9.5).

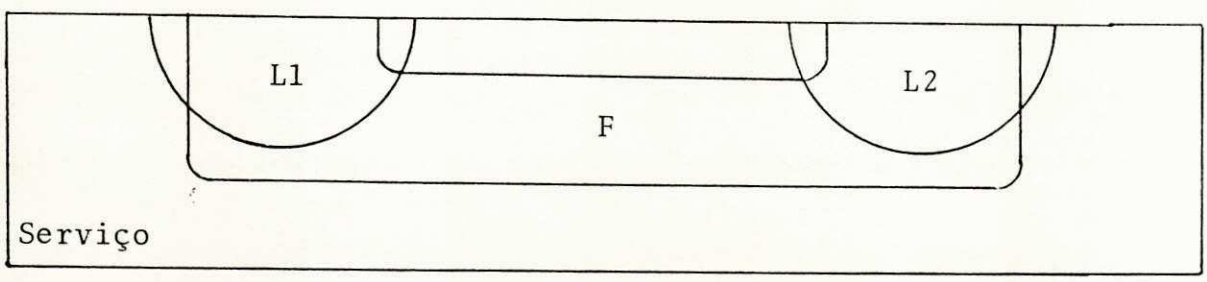


Figura 9.4 - Modelo para as especificações de serviço.

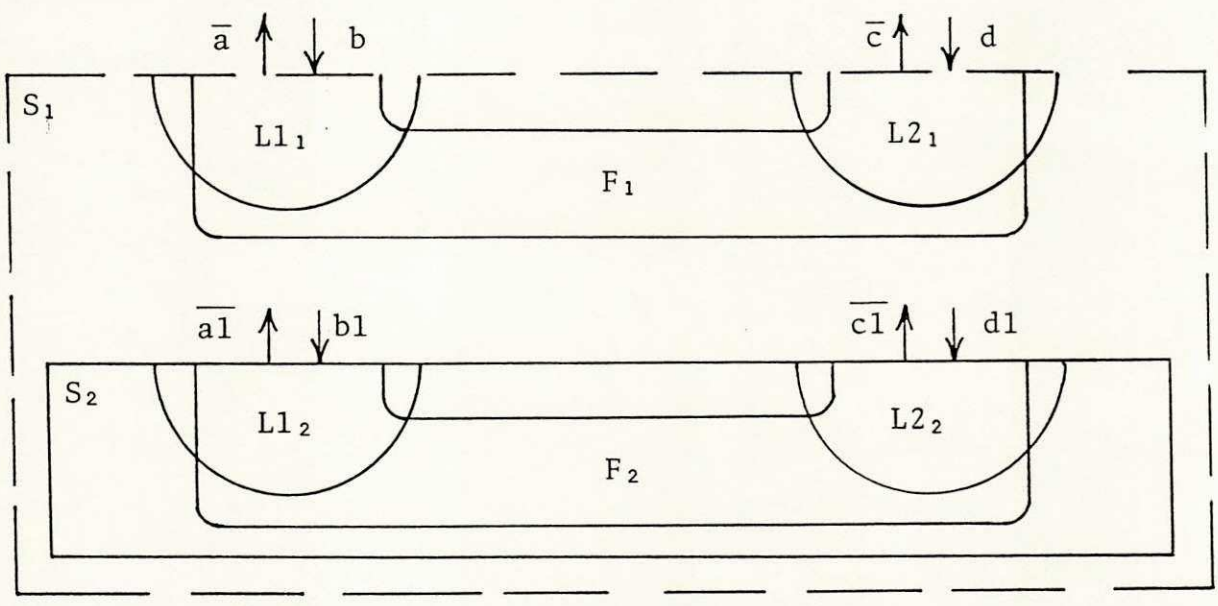


Figura 9.5 - Estruturação dos serviços S_1 e S_2.

Na Fig. 9.5 S_1 representa o provedor do serviço desejado e S_2 representa o provedor do serviço disponível.

S1 = SJ

S2 = SP

No estilo orientado para restrições, S_1 é definido por

$$S_1 = (L1_1 | L2_1) \&_M (F_1)$$

onde as restrições locais são definidas por

$$L1_1 \stackrel{def}{=} b.\bar{a}.L1_1$$

indicando que, no sítio do Questionador, o provedor do serviço desejado pode receber uma questão (em b) e entregar a resposta correspondente (em $\bar{a}$), e

$$L2_1 \stackrel{def}{=} \bar{c}.d.L2_1$$

indicando que, no sítio do Respondedor, o provedor do serviço desejado pode entregar uma questão (em $\bar{c}$) e receber a resposta correspondente (em d).

As restrições fim-a-fim podem ser estruturadas de acordo com os dois fluxos de informações (Fig. 9.6):

$$F_1 = (F1_1 | F2_1)$$

onde

$$F1_1 \stackrel{def}{=} b.\bar{c}.F1_1$$

indicando que uma questão pode ser recebida do Questionador (em b) e entregue ao Respondedor (em $\bar{c}$) e

$$F2_1 \stackrel{def}{=} d.a.F2_1$$

indicando que uma resposta pode ser recebida do Respondedor (em d) e entregue ao Questionador (em $\bar{a}$).

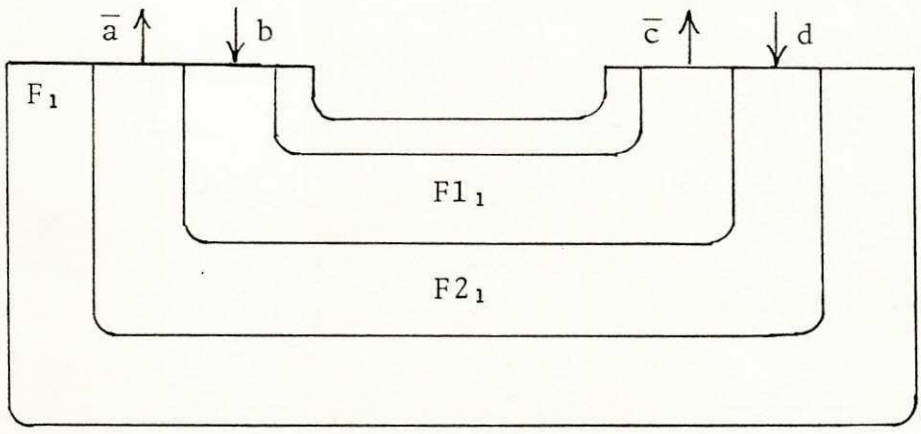


Figura 9.6 - Estrutura da restrição F_1.

M = {a,b,c,d} é o conjunto das portas compartilhadas por $(L1_1|L2_1)$ e (F_1).

No estilo orientado para restrições, S_2 é definido por

$$S_2 = (L1_2|L2_2) \&_N (F_2)$$

onde as restrições locais são definidas por

$$L1_2 \stackrel{def}{=} \overline{a1}.L1_2 + b1.L1_2$$

indicando que os eventos $\overline{a1}$ e b1 podem ocorrer em qualquer ordem (portanto, não se trata propriamente de uma restrição) e

$$L2_2 \stackrel{def}{=} \overline{c1}.L2_2 + d1.L2_2$$

também indicando que os eventos $\overline{c1}$ e d1 podem ocorrer em qualquer ordem (idem).

As restrições fim-a-fim podem ser estruturadas de acordo com os dois fluxos de dados (Fig. 9.7):

$$F_2 = (F1_2 | F2_2)$$

onde

$$F1_2 \stackrel{def}{=} b1.\overline{c1}.F1_2$$

indicando que um dado pode ser recebido no sítio do Questiomador (em b1) e entregue no sítio do Respondedor (em $\overline{c1}$) e

$$F2_2 \stackrel{def}{=} d1.\overline{a1}.F2_2$$

indicando que um dado pode ser recebido no sítio do Respondedor (em d1) e entregue no sítio do Questionador (em $\overline{a1}$).

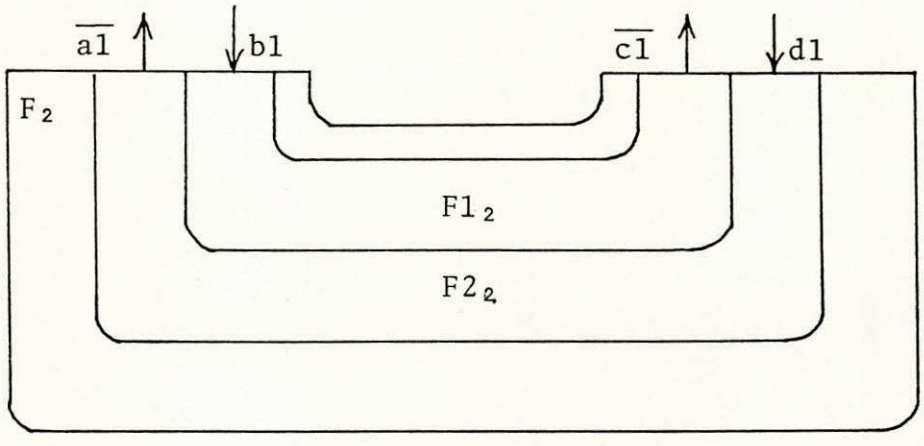


Figura 9.7 - Estrutura da restrição F_2.

N = {$\overline{a1}$, b1, $\overline{c1}$, d1} é o conjunto das portas compartilhadas por $(L1_2|L2_2)$ e (F_2).

Como se pode observar, a abordagem proposta parte das especificações de serviço para obter as especificações de protocolo, de acordo com uma consideração expressa em [ViLo 86].

Ao concluir-se a primeira etapa da abordagem, o produto obtido pode ser denominado "especificação estruturada do serviço desejado". Tal estruturação inclui o aninhamento em camadas à semelhança do Modelo Básico de Referência OSI da ISO (estruturação horizontal), onde cada camada está especificada no estilo orientado para restrições (estruturação vertical).

A especificação estruturada do serviço desejado é submetida à segunda etapa da abordagem.

9.1.2. Especificação do protocolo

Nesta etapa busca-se obter a especificação do par de entidades de protocolo correspondente a cada camada de serviço. Para isso cada camada passa a ser especificada no estilo orientado para recursos.

Esse problema pode ser apresentado, formalmente, através de um conjunto de equações de contexto (uma equação de contexto para cada camada de serviço) (Fig. 9.8):

$$(S_{i+1} \mid (E1_i \mid E2_i)) \setminus A_i \approx S_i, \quad i = 1, \ldots, n$$

onde S_1 corresponde ao serviço desejado e S_{n+1} corresponde ao serviço disponível.

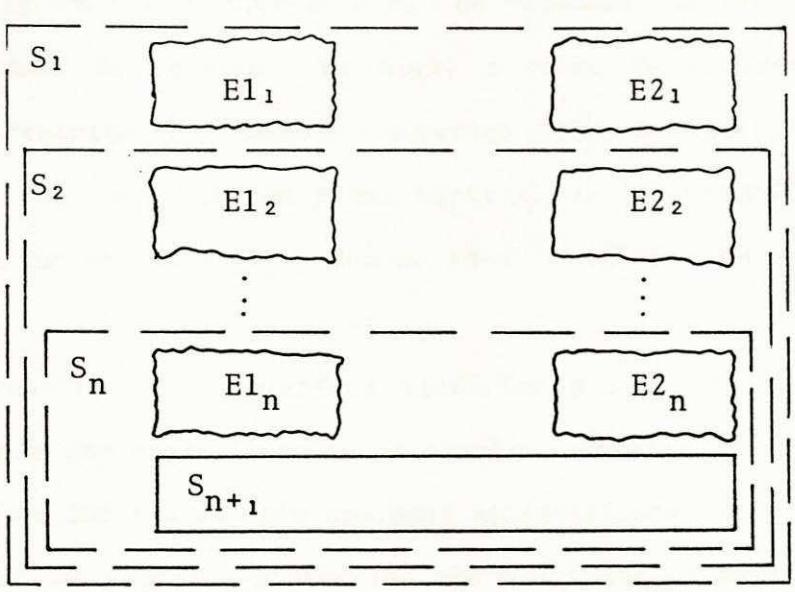


Figura 9.8 - Camadas de serviço e protocolos correspondentes.

Passo 1: Utilizando o estilo orientado para restrições especifique o par de entidades de protocolo em cada camada de serviço. A correção de cada par de entidades deve ser verificada através da equação de contexto correspondente (obtida a partir da eq. 9.2 onde i assume um valor particular i = k).

As especificações das entidades de protocolo devem seguir o modelo baseado em restrições de interface (superior e inferior) e restrições ao fluxo vertical de informações (ligando as duas interfaces) apresentado na Fig. 9.9.

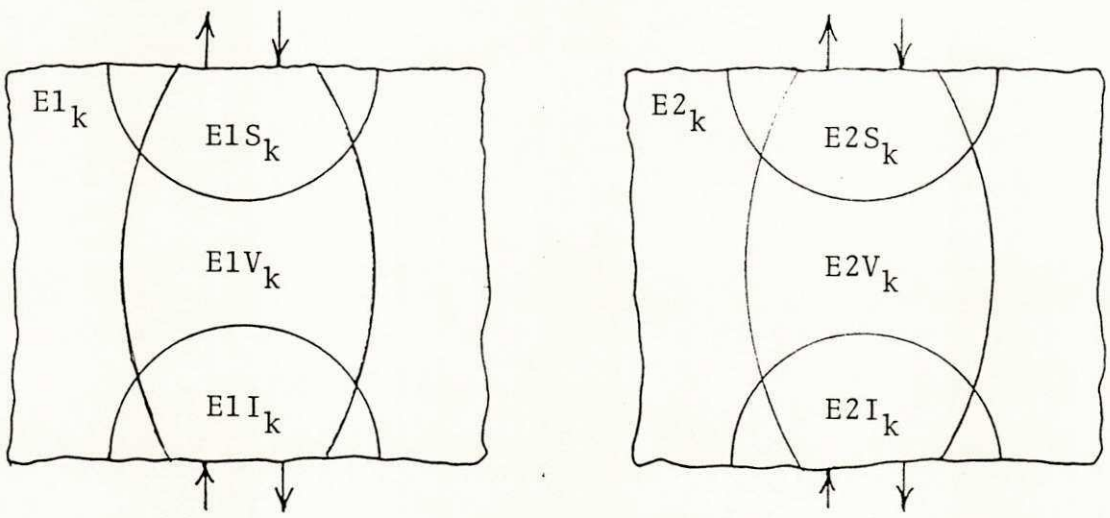


Figura 9.9 - Modelo para a especificação de entidades de protocolo.

Na Fig. 9.9 apresenta-se o par de entidades de protocolo $E1_k$ e $E2_k$ relativo à camada de serviço no nível i = k. Na entidade $E1_k$ pode-se identificar a restrição de interface superior $E1S_k$, a restrição de interface inferior $E1I_k$, e a restrição ao fluxo vertical de informações $E1V_k$. De modo correspondente, na entidade $E2_k$ podem-se identificar as restrições $E2S_k$, $E2I_k$ e $E2V_k$.

As restrições de interface (inferior e superior) são análogas às restrições locais das especificações de serviço, enquanto que as restrições ao fluxo vertical de informações são análogas às restrições fim-a-fim. Explorando essas analogias, as restrições atuantes nas entidades de protocolo $E1_k$ e $E2_k$ podem ser obtidas através de transformações das restrições atuantes nas especificações de serviço S_k e S_{k+1} (Fig. 9.10).

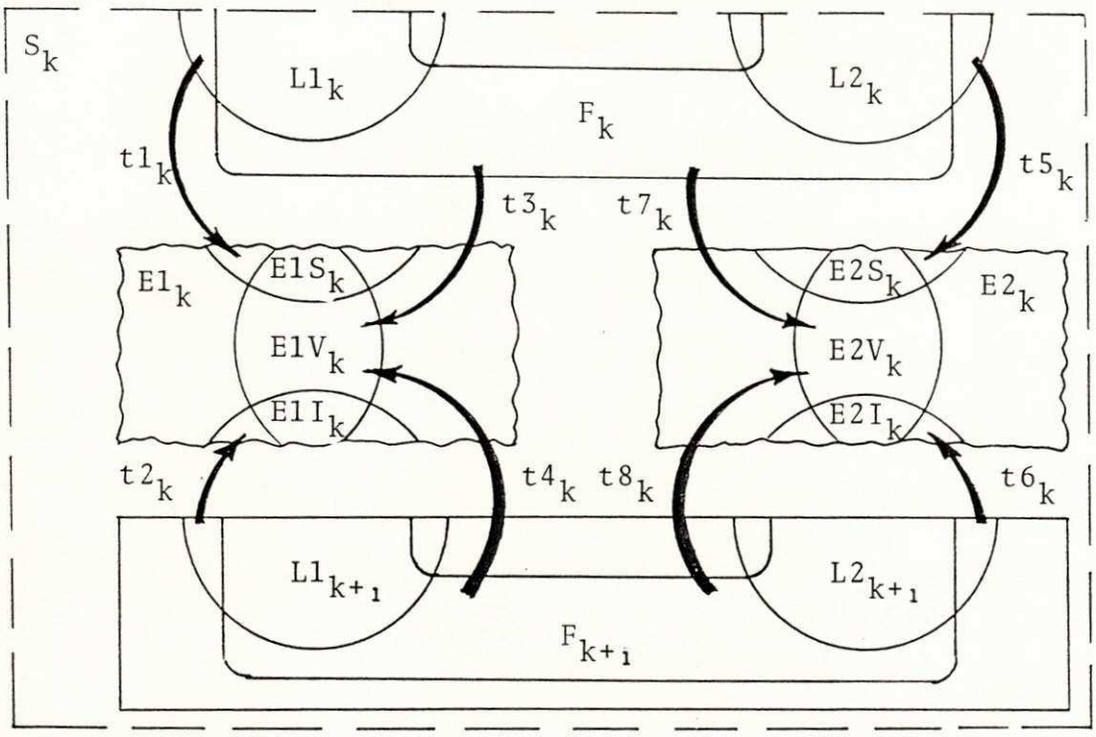


Figura 9.10 - Transformações de restrições.

Essas transformações podem envolver dois tipos de operações:

(a) **Complementação (−)**. A partir de um agente B obtém-se outro agente $\overline{B}$, com a mesma estrutura de comportamento de B, sendo que as portas de $\overline{B}$ são as portas de B complementadas.

Por exemplo, se

$$B \stackrel{def}{=} a.b.\tau.NIL + \overline{c}.B$$

então

$$\overline{B} \stackrel{def}{=} \overline{a}.\overline{b}.\tau.NIL + c.\overline{B}$$

(b) **Projeção** de um agente B sobre um conjunto de portas C. Obtém-se um agente

$$B' \stackrel{def}{=} p_c(B)$$

com a mesma estrutura de comportamento de B, onde foram apagadas todas as

portas que não pertencem ao conjunto C. Projeções são também definidas e utilizadas em [MeBo 83].

Por exemplo, se C = {a, $\bar{c}$, m},

$$p_c(B) \stackrel{def}{=} a.NIL + \bar{c}.p_c(B)$$
$$p_c(\bar{B}) \stackrel{def}{=} NIL + p_c(\bar{B})$$

Além dessas operações, os comportamentos dos dois agentes B1 e B2 podem ser intercalados. Nesse caso obtém-se um agente B3 que apresenta todos os eventos de B1 e B2.

Por exemplo, se

$$B1 \stackrel{def}{=} a1.a2.NIL$$
$$B2 \stackrel{def}{=} b1.b2.NIL$$

então uma intercalação de B1 com B2 pode resultar no agente (Fig. 9.11)

$$B3 \stackrel{def}{=} a1.b1.b2.a2.NIL$$

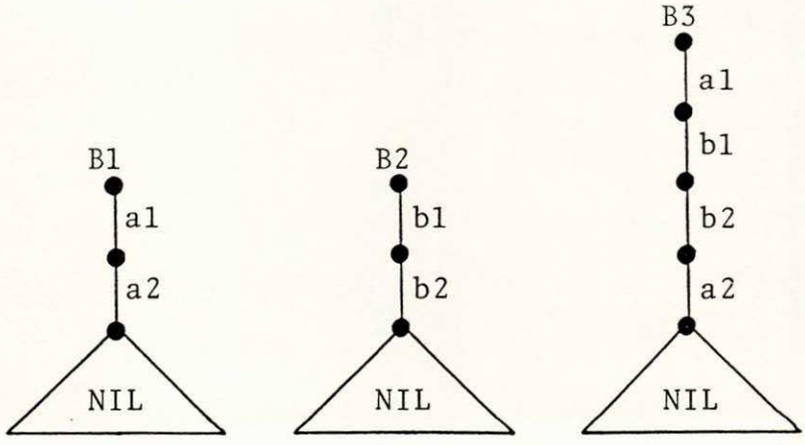


Figura 9.11 - Árvores de sincronização B1, B2 e B3.

Na prática, as transformações aplicadas às restrições das especificações de serviço são guiadas pelas características que se deseja impor às restrições das entidades de protocolo. Tais características já se encontram, de algum modo, presentes nas especificações de serviço. As transformações visam isolar essas características e transportá-las para a especificação das entidades de protocolo.

Obtenção das entidades de protocolo do sítio 1

As restrições atuantes na entidade de protocolo $E1_k$ (sítio 1, camada i = k) são obtidas com o emprego das transformações $t1_k$ até $t4_k$ (Fig. 9.10).

A restrição de interface superior $E1S_k$ é obtida, a partir da restrição local $L1_k$ (do serviço S_k), com o emprego da transformação $t1_k$. Essa transformação simplesmente dá a $E1S_k$ o comportamento de $L1_k$, uma vez que é através da comunicação direta com a entidade $E1_k$ que uma entidade da camada k-1 pode utilizar o serviço S_k no sítio 1.

A restrição de interface inferior $E1I_k$ é obtida, a partir da restrição local $L1_{k+1}$ (do serviço S_{k+1}), com o emprego da transformação $t2_k$. Essa transformação envolve a complementação de $L1_{k+1}$, uma vez que a entidade $E1_k$ e o provedor do serviço S_{k+1} comunicam-se através de portas complementares.

A restrição ao fluxo vertical de informações $E1V_k$ é obtida, a partir das restrições fim-a-fim F_k (do serviço S_k) e F_{k+1} (do serviço S_{k+1}), com o emprego das transformações $t3_k$ e $t4_k$. Essas transformações envolvem as projeções de F_k e F_{k+1} sobre o conjunto A_k constituído de todas as portas da entidade $E1_k$. A transformação $t4_k$ envolve ainda a complementação de F_{k+1}.

Por exemplo, a restrição de interface superior E1S da entidade E1 do sistema Questão-Resposta (Fig. 9.12) é obtida, a partir da restrição local $L1_1$ (do serviço S_1), com o emprego da transformação t1:

$$E1S = L1_1$$

isto é,

$$E1S \stackrel{def}{=} b.\bar{a}.E1S$$

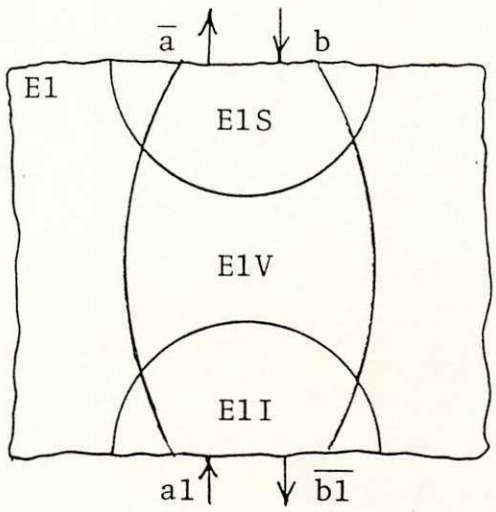


Figura 9.12 - Estrutura da entidade de protocolo E1.

A restrição de interface inferior E1I, da mesma entidade, é obtida a partir da restrição local $L1_2$ (do serviço S_2) com o emprego da transformação t2. No presente caso essa transformação poderia envolver apenas a complementação de $L1_2$

$$\overline{L1}_2 \stackrel{def}{=} a1.\overline{L1}_2 + \overline{b1}.\overline{L1}_2$$

Entretanto, a escolha indeterminística presente na expressão de comportamento $\overline{L1}_2$ resulta em um comportamento complexo para a entidade E1. Para simplificar o comportamento da entidade E1 tal escolha é suprimida. O comportamento resultante deve representar o envio de uma questão e, em seguida, a recepção da resposta correspondente:

$$E1I \stackrel{def}{=} \overline{b1}.a1.E1I$$

A restrição ao fluxo vertical de informações E1V, da entidade E1, pode ser estruturada considerando independentemente os aspectos relativos ao fluxo das questões (restrição E1VQ) e os aspectos relativos ao fluxo das respostas (restrição E1VR) (Fig. 9.13):

$$E1V = (E1VQ|E1VR)$$

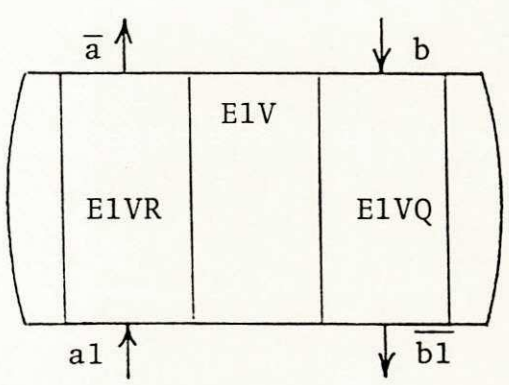


Figura 9.13 - Estrutura da restrição E1V.

A restrição E1VQ é obtida a partir das projeções

$$p_A(F1_1) \stackrel{def}{=} b.p_A(F1_1)$$

$$p_A(\overline{F1}_2) \stackrel{def}{=} \overline{b1}.p_A(\overline{F1}_2)$$

onde $A = \{\overline{a}, b, a1, \overline{b1}\}$ é o conjunto de portas da entidade E1.

A intercalação de $p_A(F1_1)$ com $p_A(\overline{F1}_2)$ fornece

$$\boxed{E1VQ \stackrel{def}{=} b.\overline{b1}.E1VQ}$$

De modo semelhante, a restrição E1VR é obtida intercalando-se as projeções

$$p_A(F2_1) \stackrel{def}{=} \overline{a}.p_A(F2_1)$$

$$p_A(\overline{F2}_2) \stackrel{def}{=} a1.p_A(\overline{F2}_2)$$

Essa intercalação fornece

$$\boxed{E1VR \stackrel{def}{=} a1.\overline{a}.E1VR}$$

Obtenção das entidades de protocolo do sítio 2

As restrições atuantes na entidade de protocolo $E2_k$ (sítio 2, camada i = k) são obtidas com o emprego das transformações $t5_k$ até $t8_k$. Essas transformações são análogas àquelas empregadas para a obtenção das restrições atuantes na entidade de protocolo $E1_k$.

A restrição de interface superior $E2S_k$ é obtida, a partir da restrição local $L2_k$, com o emprego da transformação $t5_k$.

A restrição de interface inferior $E2I_k$ é obtida, a partir da restrição local $L2_{k+1}$, com o emprego da transformação $t6_k$.

A restrição ao fluxo vertical de informações $E2V_k$ é obtida, a partir das restrições fim-a-fim F_k e F_{k+1}, com o emprego das transformações $t7_k$ e $t8_k$.

Por exemplo, a restrição de interface superior E2S da entidade E2, do sistema Questão-Resposta (Fig. 9.14) é obtida a partir da restrição local

$L2_1$ (do serviço S_1), com o emprego da transformação t5:

$$E2S = L2$$

isto é,

$$E2S \stackrel{def}{=} c.d.E2S$$

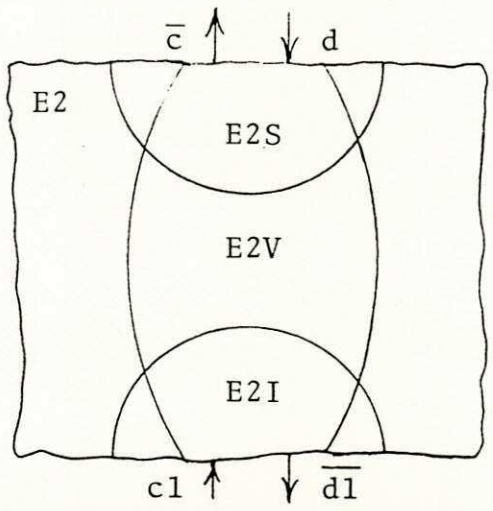


Figura 9.14 - Estrutura da entidade de protocolo E2.

A restrição de interface inferior E2I, da mesma entidade, é obtida a partir da restrição local $L2_2$ (do serviço S_2), com o emprego da transformação t6. Neste caso aplica-se o mesmo raciocínio empregado para a obtenção da restrição de interface inferior da entidade E1.

Para simplificar o comportamento da entidade E2, a escolha indeterminística presente na expressão de comportamento

$$\overline{L2_2} \stackrel{def}{=} c1.\overline{L2_2} + \overline{d1}.\overline{L2_2}$$

é suprimida. O comportamento resultante deve representar a recepção de uma questão e, em seguida, o envio da resposta correspondente:

$$E2I \stackrel{def}{=} c1.\overline{d1}.E2I$$

De modo semelhante ao que foi feito para a entidade de protocolo E1, a restrição ao fluxo vertical de informações da entidade E2 pode ser estruturada considerando independentemente os aspectos relativos ao fluxo das questões (restrição E2VQ) e os aspectos relativos ao fluxo das respostas (restrição E2VR) (Fig. 9.15):

$$E2V = (E2VQ|E2VR)$$

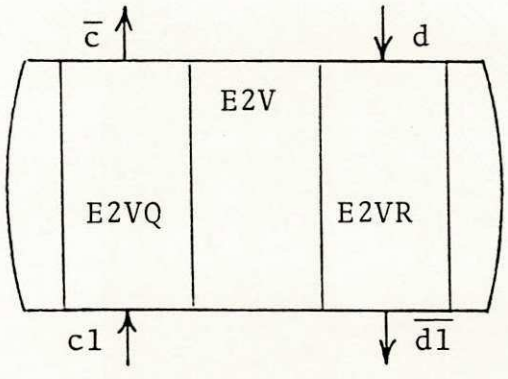


Figura 9.15 - Estrutura da restrição E2V.

A restrição E2VQ é obtida a partir das projeções

$$p_B(F1_1) \stackrel{def}{=} \overline{c}.p_B(F1_1)$$

$$p_B(\overline{F1}_2) \stackrel{def}{=} c1.p_B(\overline{F1}_2)$$

onde $B = \{\overline{c}, d, c1, \overline{d1}\}$ é o conjunto de portas da entidade E2.

A intercalação de $p_B(F1_1)$ com $p_B(\overline{F1}_2)$ fornece

$$\boxed{E2VQ \stackrel{def}{=} c1.\overline{c}.E2VQ}$$

De modo semelhante, a restrição E2VR é obtida intercalando-se as projeções

$$p_B(F2_1) \stackrel{def}{=} d.p_B(F2_1)$$

$$p_B(\overline{F2}_2) \stackrel{def}{=} \overline{d1}.p_B(\overline{F2}_2)$$

Essa intercalação fornece

$$\boxed{E2VR \stackrel{def}{=} d.\overline{d1}.E2VR}$$

Passo 2: verifique a correção global do sistema (Fig. 9.16) utilizando a equação do contexto (eq. 9.1) onde

$$E1 = (E1_1|\ldots|E1_n)\backslash A_1,\ldots,A_n$$

$$E2 = (E2_1|\ldots|E2_n)\backslash B_1,\ldots,B_n$$

Cada conjunto A_i, $i = 1,\ldots,n$ é constituído das portas utilizadas para a comunicação da entidade de protocolo $E1_i$ com o provedor do serviço S_{i+1}. Cada conjunto B_i, $i = 1,\ldots,n$ é constituído das portas utilizadas para a comunicação da entidade de protocolo $E2_i$ com o mesmo provedor de serviço.

No caso do sistema Questão-Resposta há apenas uma camada de protocolo. Portanto, a verificação da correção dessa camada corresponde à verificação da correção global do sistema.

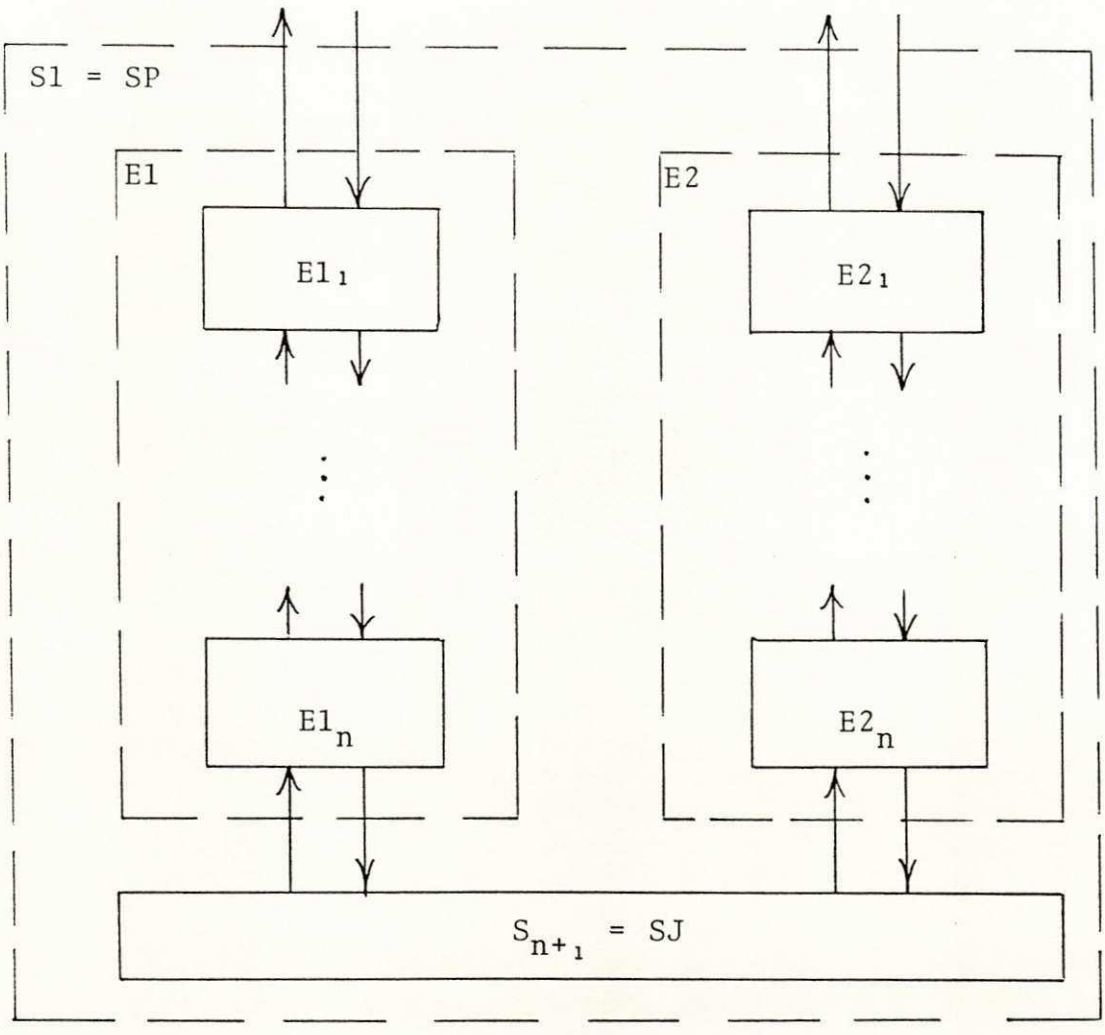


Figura 9.16 - Visão global do sistema.

A expansão da composição $(S_2|(E_1|E_2))\backslash\{a1,\ b1,\ c1,\ d1\}$ pode ser representada pelo agente

$$S \stackrel{def}{=} b.\tau.\tau.\bar{c}.d.\tau.\tau.\bar{a}.S$$

de modo que

$$(S_2|(E_1|E_2))\backslash\{a1,\ b1,\ c1,\ d1\} \sim S$$

A representação dos eventos internos (τ), na expressão de comportamento do agente S, pode ser suprimida com o emprego das propriedades da equivalência de observação ($\approx$). Nesse caso obtém-se o agente

$$S' \stackrel{def}{=} b.\bar{c}.d.\bar{a}.S'$$

de modo que

$$S \approx S'$$

Mas S' pode representar a expansão do agente S_1 (serviço desejado Questão-Resposta), de modo que

$$S' \sim S_1$$

Então

$$(S_2|(E_1|E_2))\backslash\{a1,\ b1,\ c1,\ d1\} \sim S \approx S' \sim S1$$

Como a relação ≈ inclui a relação ~ (isto é, ≈ é mais abrangente que ~),

$$(S_2|(E_1|E_2))\backslash\{a1,\ b1,\ c1,\ d1\} \approx S_1$$

o que prova a correção da especificação das entidades de protocolo E1 e E2.

Ao concluir-se a segunda etapa da abordagem, cada camada, que antes estava especificada no estilo orientado para restrições, agora está especificada também no estilo orientado para recursos. Cada recurso, por sua vez, está especificado no estilo orientado para restrições. O resultado obtido pode ser denominado "design dos protocolos correspondentes às camadas de serviço do sistema". Esse resultado é submetido à terceira etapa da abordagem.

9.1.3 Refinamentos sucessivos

Cada entidade de protocolo pode ser submetida a um processo de refinamentos sucessivos. Um refinamento consiste na substituição de uma caixa preta por várias outras que se comunicam para realizar as tarefas da primeira. A aplicação repetida de tais refinamentos pode ser representada na forma de

uma árvore de refinamentos que reflete a estruturação hierárquica dos resultados (Fig. 9.17).

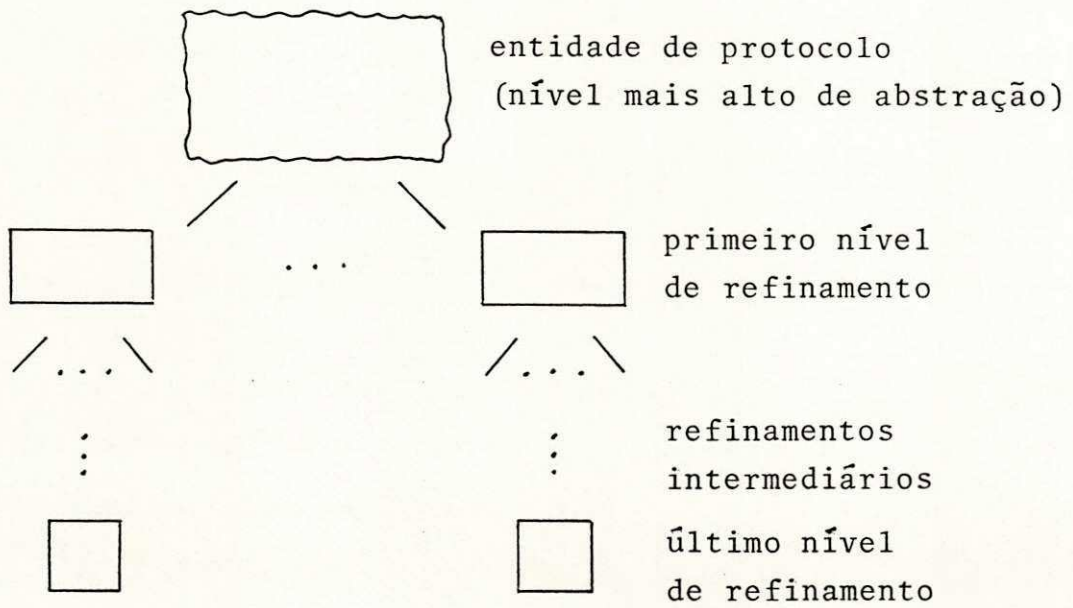


Figura 9.17 - Refinamentos sucessivos de uma entidade de protocolo.

Passo único: realize o refinamento progressivo de cada entidade de protocolo até que um grau conveniente de detalhamento (a critério do projetista) seja alcançado. No caso da disponibilidade de agentes básicos predefinidos, a especificação final das entidades pode ser constituída somente desses agentes básicos. Cada refinamento deve ser submetido à prova de correção.

No caso do sistema Questão-Resposta, as entidades de protocolo E1 e E2 são intercambiáveis com as entidades de protocolo E1' e E2', respectivamente (Fig. 9.18).

Para a obtenção de E1' as restrições E1S, E1I, E1VQ e E1VR foram identificadas como Ciclo1, Ciclo2, Ciclo3 e Ciclo4, respectivamente

$$E1' = (Ciclo1 | Ciclo3) \&_A (Ciclo2 | Ciclo4)$$

onde

$$A = \{\overline{a},\ b,\ a1,\ \overline{b1}\}$$

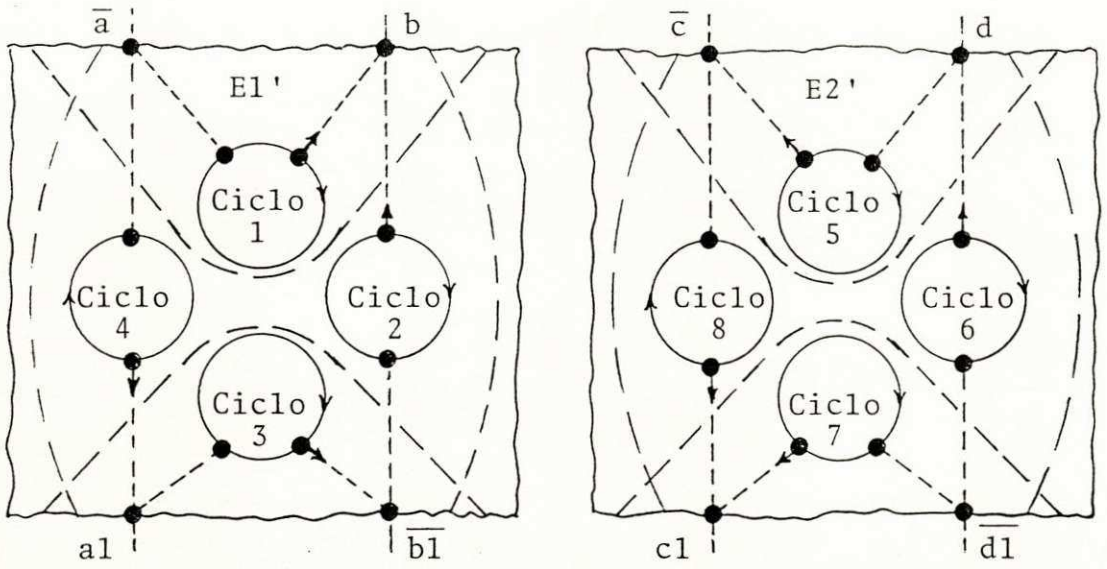


Figura 9.18 - Restrições das entidades E1 e E2 representadas como agentes básicos.

Para a obtenção de E2' as restrições E2S, E2I, E2VQ e E2VR foram identificadas como Ciclo5, Ciclo6, Ciclo7 e Ciclo8, respectivamente.

$$E2' = (Ciclo5 | Ciclo7) \&_B (Ciclo6 | Ciclo8)$$

onde

$$B = \{\bar{c},\ d,\ c1,\ \bar{d1}\}$$

Tal estruturação sugere que as entidades de protocolo E1 e E2 podem ser refinadas, de modo a apresentarem estruturas constituídas somente desses agentes básicos interligados. O comportamento cíclico dessas entidades pode ser obtido, por exemplo, como mostra a Fig. 9.19.

A prova da correção dos refinamentos de E1 e E2 é realizada a seguir com a utilização da técnica de bissimulação.

Definindo

$$E1ref = (C1 | C2 | C3 | C4) \setminus \{m, n, o, p\}$$

$$E2ref = (C5 | C6 | C7 | C8) \setminus \{q, r, s, t\}$$

e expandindo E1ref e E2ref obtém-se

$$E1ref \sim b.\tau.\overline{b1}.\tau.a1.\tau.\overline{a}.\tau.E1ref$$

$$E2ref \sim c1.\tau.\overline{c}.\tau.d.\tau.\overline{d1}.\tau.E2ref$$

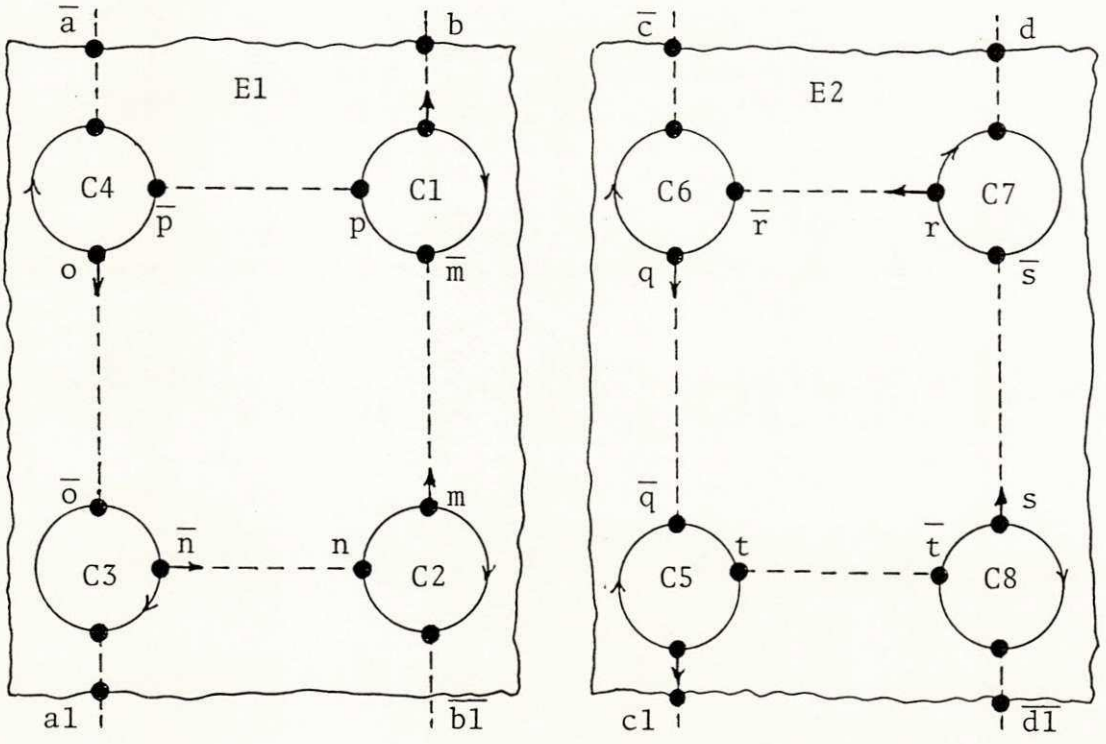


Figura 9.19 - Refinamentos de E1 e E2.

A Fig. 9.20 mostra que existe uma relação de bissimulação entre E1 e E1ref e uma outra relação de bissimulação entre E2 e E2ref. Desse modo,

$$E1 \approx E1ref$$

$$E2 \approx E2ref$$

Como os refinamentos de E1 e E2 estão apresentados em termos de agentes básicos predefinidos, completou-se a tarefa de especificação dessas entidades de protocolo.

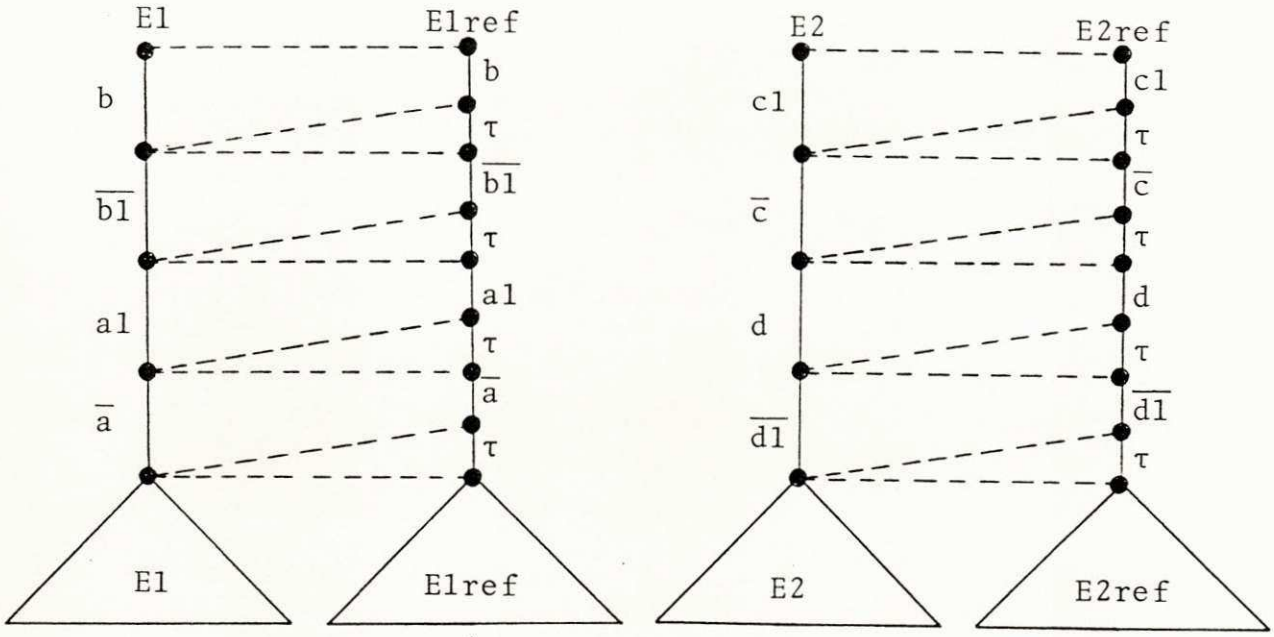


Figura 9.20 - Bissimulações entre E1 e E1ref e entre E2 e E2ref.

9.2. Aplicação: o protocolo do bit alternante

Para ilustrar a abordagem proposta, nesta seção mostra-se como essa abordagem pode ser aplicada a um problema simples mas não trivial.

Problema (informal):

"Especifique as entidades de protocolo que oferecem um serviço de comunicação confiável a dois usuários - um Produtor de Dados e um Consumidor de Dados -, de modo que o Consumidor possa realizar, explicitamente, o controle do fluxo de dados através do envio de confirmações para o Produtor. Considere disponível um serviço de comunicação que pode perder mensagens em ambos os sentidos de transmissão".

Formalização do problema:

A arquitetura correspondente ao problema é apresentada na fig. 9.21. Nessa figura SC representa o serviço confiável desejado, SD representa o serviço disponível e EP e EC representam as entidades de protocolo no sítio do

Produtor e no sítio do Consumidor, respectivamente.

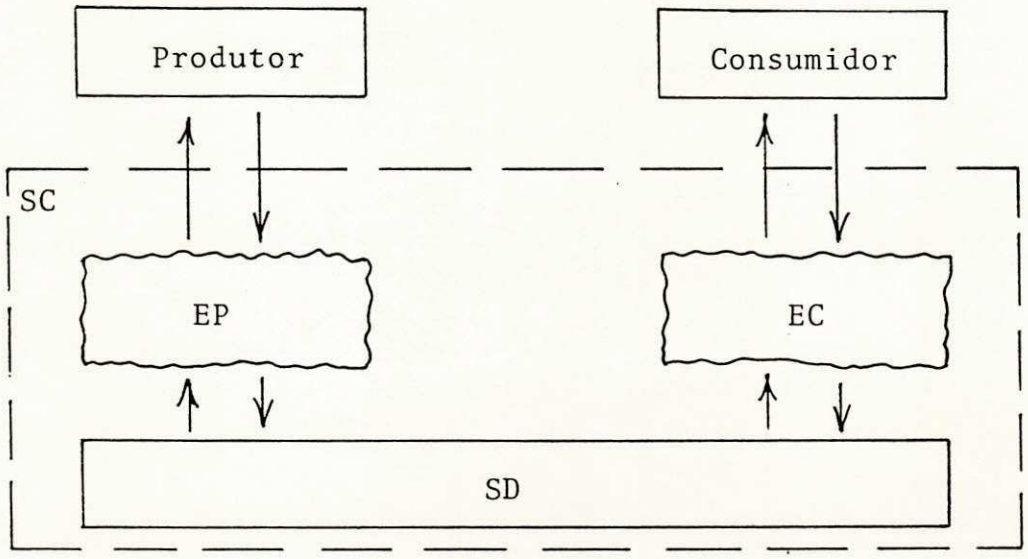


Figura 9.21 - EP e EC comunicam-se utilizando SD.

(a) Especificação formal de SC:

O serviço confiável SC pode ser descrito no estilo orientado para restrições como na secção 8.2:

$$SC = (L1 \mid L2) \;\&_c\; (F1 \mid F2)$$

onde

$$L1 \stackrel{def}{=} b.\bar{a}.L1$$

$$L2 \stackrel{def}{=} \bar{c}.d.L2$$

são as restrições locais no sítio do Produtor e no sítio do Consumidor, respectivamente,

$$F1 \stackrel{def}{=} b.\bar{c}.F1$$

$$F2 \stackrel{def}{=} d.\bar{a}.F2$$

são as restrições fim-a-fim e C = {$\bar{a}$, b, $\bar{c}$, d} é o conjunto das portas compartilhadas por (L1|L2) e (F1|F2) (Fig. 8.2).

(b) Especificação formal de SD

Como o serviço disponível SD não é confiável, as entidades de protocolo EP e EC devem comportar-se de modo a recuperarem as mensagens perdidas (dados e confirmações). Para tal, empregam o protocolo do bit alternante. A versão original desse protocolo foi apresentada em [BaSc 69]. Desde então várias versões têm sido utilizadas para ilustrar metodologias e linguagens de especificação [Boch 78] [LoRi 88] [Parr 89] [Miln 89].

Na arquitetura do serviço SD (Fig. 9.22), quatro portas ocupam-se com a transmissão de dados: f0 e $\overline{g0}$ para a transmissão de dados com o bit 0, e f1 e $\overline{g1}$ para a transmissão de dados com o bit 1.

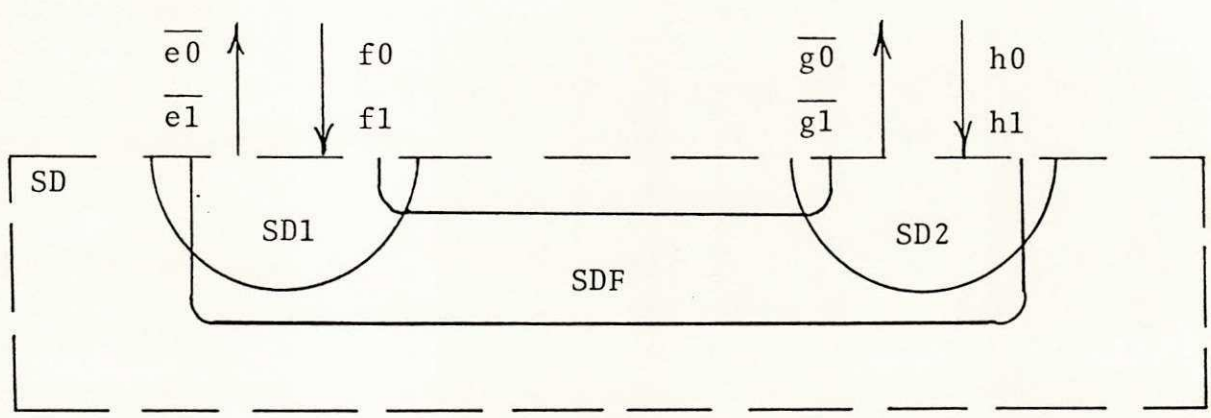


Figura 9.22 - Restrições de SD.

Igualmente, quatro portas ocupam-se com a transmissão de confirmações: h0 e $\overline{e0}$ para a transmissão das confirmações com o bit 0 e h1 e

$\overline{e1}$ para a transmissão das confirmações com o bit 1.

Empregando o estilo orientado para restrições,

$$SD = (SD1 | SD2) \&_D (SDF)$$

onde

$$D = \{\overline{e0},\ \overline{e1},\ f0,\ f1,\ \overline{g0},\ \overline{g1},\ h0,\ h1\}$$

SD1 é a restrição local de SD no sítio do Produtor. Ela pode ser estruturada como a composição de duas restrições: a primeira relacionada à recepção de dados e à entrega de confirmações com o bit 0 (restrição SP0), e a segunda relacionada à recepção de dados e à entrega de confirmações com o bit 1 (restrição (SP1):

$$SD1 = (SP0 | SP1)$$

onde

$$SP0 \stackrel{def}{=} f0.(\overline{e0}.SP0 + SP0)$$

indica que SP0 pode receber do Produtor um dado com o bit 0 (em f0) e, após tal evento, comportar-se indeterministicamente: na primeira alternativa pode entregar a esse usuário uma confirmação com o bit 0 (em $\overline{e0}$) e voltar recursivamente, na segunda alternativa volta recursivamente para receber o mesmo dado.

$$SP1 \stackrel{def}{=} f1.(\overline{e1}.SP1 + SP1)$$

indica um comportamento semelhante ao de SP0, mas utilizando as portas de comunicação f1 e $\overline{e1}$, correspondentes à recepção de dados e à entrega de confirmações com o bit 1 (Fig. 9.23).

SD2 é a restrição local de SD no sítio do Consumidor. Ela também pode ser estruturada como a composição de duas restrições: a primeira

relacionada à entrega de dados e à recepção de confirmações com o bit 0 (restrição SC0), e a segunda relacionada à entrega de dados e à recepção de confirmações com o bit 1 (restrição SC1):

$$SD2 = (SC0 | SC1)$$

onde

$$SC0 \stackrel{def}{=} \overline{g0}.h0.SC0$$

indica que SC0 pode entregar ao Consumidor um dado com o bit 0 (em $\overline{g0}$) e, após tal evento, pode receber do mesmo usuário uma confirmação com o bit 0 (em h0) e voltar recursivamente.

$$SC1 \stackrel{def}{=} \overline{g1}.h1.SC1$$

indica um comportamento semelhante ao de SC0, mas utilizando as portas de comunicação $\overline{g1}$ e h1, correspondentes à entrega de dados e à recepção de confirmações com o bit 1 (Fig. 9.24).

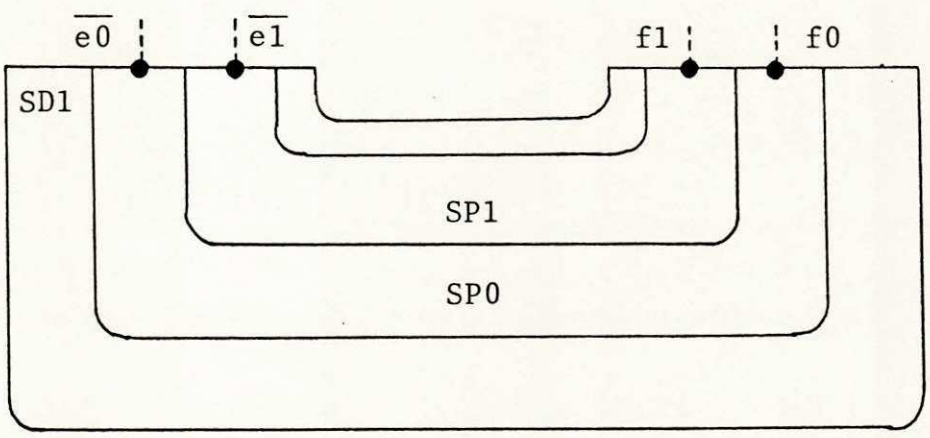


Figura 9.23 - Estrutura da restrição SD1.

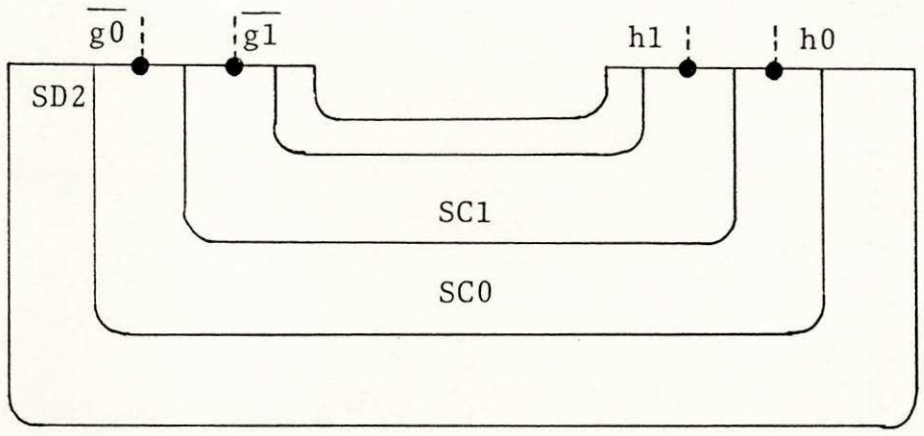


Figura 9.24 - Estrutura da restrição SD2.

SDF é a restrição fim-a-fim de SD. Ela pode ser estruturada como a composição de quatro restrições: a primeira relacionada à transmissão de dados com o bit 0 (restrição FD0); a segunda relacionada à transmissão de confirmações com o bit 0 (restrição FC0); a terceira relacionada à transmissão de dados com o bit 1 (restrição FD1); e a quarta relacionada à transmissão de confirmações com o bit 1 (restrição FC1):

$$SDF = (FD0 | FC0 | FD1 | FC1)$$

onde

$$FD0 \stackrel{def}{=} f0.(\tau.\overline{g0}.FD0 + \tau.FD0)$$

indica que FD0 pode receber do Produtor um dado com o bit 0 (em f0) e, após tal evento, comportar-se indeterministicamente: na primeira alternativa a transmissão é realizada (τ ocorre), o dado pode ser entregue ao Consumidor (em $\overline{g0}$) e FD0 volta recursivamente; na segunda alternativa, o dado é perdido (τ ocorre) e FD0 volta recursivamente.

$$FC0 \stackrel{def}{=} h0.(\tau.\overline{e0}.FC0 + \tau.FC0)$$

indica que FC0 pode receber do Consumidor uma confirmação com o bit 0 (em h0) e, após tal evento, comportar-se indeterministicamente: na primeira alternativa a transmissão é realizada (τ ocorre), a confirmação pode ser entregue ao Produtor (em $\overline{e0}$) e FC0 volta recursivamente; na segunda alternativa, a confirmação é perdida (τ ocorre) e FC0 volta recursivamente.

$$FD1 \stackrel{def}{=} f1.(\tau.\overline{g1}.FD1 + \tau.FD1)$$

indica um comportamento semelhante ao de FD0, mas utilizando as portas de comunicação f1 e $\overline{g1}$, correspondentes à transmissão de dados com o bit 1.

$$FC1 \stackrel{def}{=} h1.(\tau.\overline{e1}.FC1 + \tau.FC1)$$

indica um comportamento semelhante ao de FC0, mas utilizando as portas de comunicação h1 e $\overline{e1}$, correspondentes à transmissão de confirmações com o bit 1.

O problema proposto no início desta seção pode ser estabelecido formalmente: encontre EP e EC tais que satisfaçam a equação de contexto

$$(SD|(EP|EC))\backslash D \approx SC \qquad \text{(eq. 9.2)}$$

A composição das entidades EP e EC com o provedor do serviço SD permitirá descrever o serviço SC no estilo orientado para recursos.

9.2.1. Especificação da entidade de protocolo EP

As restrições atuantes na entidade EP (Fig. 9.25) podem ser obtidas através de complementações, projeções e intercalações das restrições impostas às especificações de serviço SC e SD, de acordo com o passo 1 da segunda etapa da abordagem proposta (seção 9.1.2).

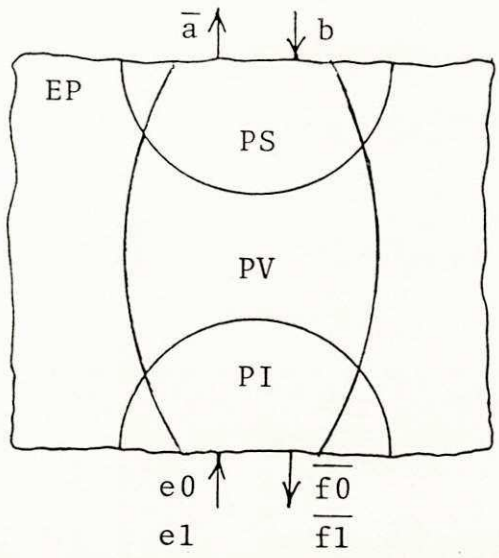


Figura 9.25 - Restrições da entidade EP.

Empregando o estilo orientado para restrições,

$$EP = (PS|PI) \&_C (PV)$$

onde

$$C = \{\bar{a},\ b,\ e0,\ e1,\ \overline{f0},\ \overline{f1}\}$$

PS é a restrição atuante na interface superior da entidade EP. Ela tem o comportamento da restrição local L1, do serviço SC, no sítio do Produtor:

$$PS = L1 \quad .$$

isto é,

$$\boxed{PS \stackrel{def}{=} b.\bar{a}.PS}$$

PI é a restrição atuante na interface inferior da entidade EP. Como a entidade EP recebe as confirmações na interface PI, torna-se natural associar a essa interface a capacidade para a detecção de perdas de mensagens.

Tal detecção é realizada por temporização:

Para representar os eventos de temporização define-se o agente

$$T \stackrel{def}{=} \tau.T$$

indicando que T pode realizar um tal evento (τ ocorre) e voltar recursivamente.

Após realizar um evento τ de temporização, a restrição PI deverá adotar o comportamento que corresponde à detecção da perda de uma mensagem.

As complementações das restrições locais SP0 e SP1 (ambas no sítio do Produtor) da especificação do serviço disponível SD são denotadas por $\overline{SP0}$ e $\overline{SP1}$ e definidas por

$$\overline{SP0} \stackrel{def}{=} \overline{f0}.(e0.\overline{SP0} + \overline{SP0})$$
$$\overline{SP1} \stackrel{def}{=} \overline{f1}.(e1.\overline{SP1} + \overline{SP1})$$

A intercalação I0 de T com $\overline{SP0}$ deve fornecer um comportamento onde ocorre um evento τ de temporização, se a confirmação com o bit 0 não é recebida a tempo em e0 (Fig. 9.26):

$$I0 \stackrel{def}{=} \overline{f0}.(e0.I0 + \tau.I0)$$

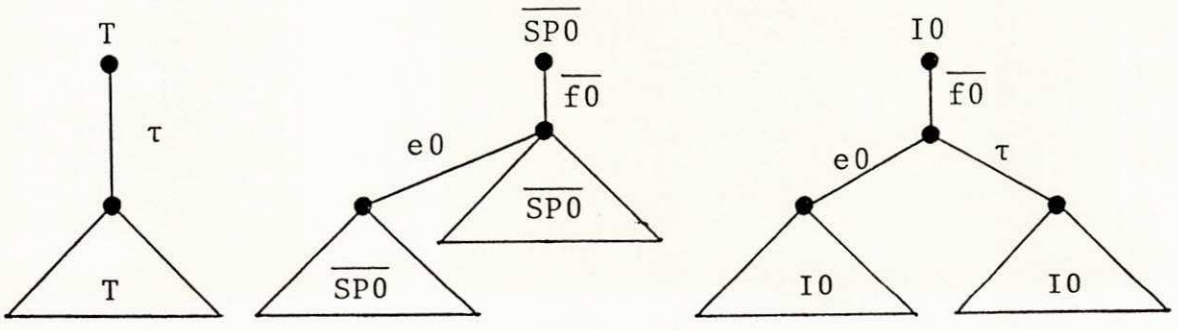


Figura 9.26 - Árvores de sincronização T, $\overline{SP0}$ e I0.

De modo semelhante, a intercalação I1 de T com $\overline{SP1}$ deve fornecer um comportamento onde ocorre um evento τ de temporização, se a confirmação com o bit 1 não é recebida a tempo em e1 (Fig. 9.27):

$$I1 \stackrel{def}{=} \overline{f1}.(e1.I1 + \tau.I1)$$

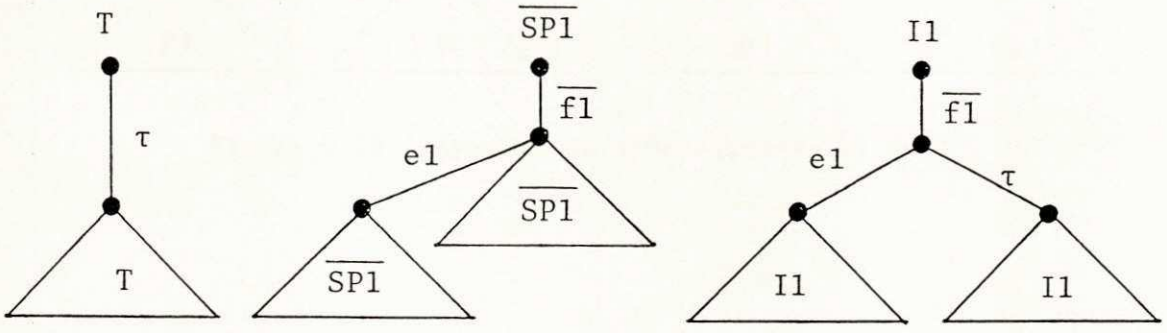


Figura 9.27 - Árvores de sincronização T, $\overline{SP1}$ e I1.

A restrição PI pode ser obtida, através da intercalação de I0 e I1, impondo-se que, após a recepção de uma confirmação com o bit 0 (ou com o bit 1), o próximo dado a enviar deva ter o bit 1 (ou o bit 0) (Fig. 9.28):

$$PI \stackrel{def}{=} \overline{f0}.(e0.PI' + \tau.PI)$$
$$PI' \stackrel{def}{=} \overline{f1}.(e1.PI + \tau.PI')$$

PV é a restrição ao fluxo vertical de dados e confirmações através da entidade de protocolo EP. Essa restrição pode ser estruturada considerando duas restrições: a primeira relacionada ao fluxo de dados (restrição PVD) e a segunda relacionada ao fluxo de confirmações (restrição PVC) (Fig. 9.29):

$$PV = (PVD \mid PVC)$$

A restrição PVD envolve a ocorrência de eventos exclusivamente nas portas pertencentes ao conjunto BF = {b, $\overline{f0}$, $\overline{f1}$}.

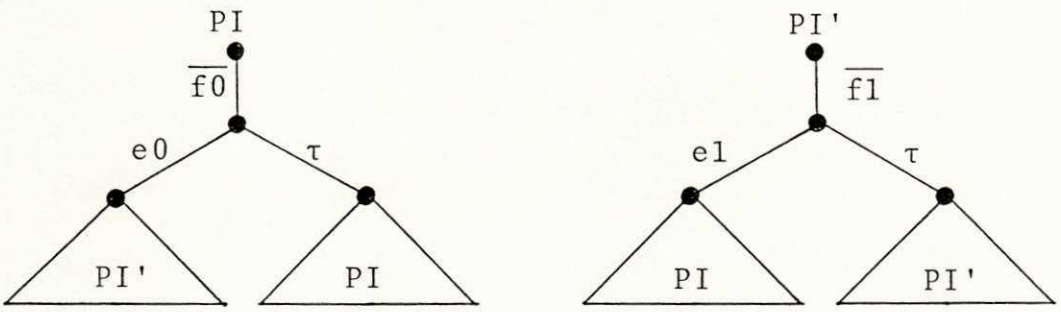


Figura 9.28 - Árvores de sincronização PI e PI'.

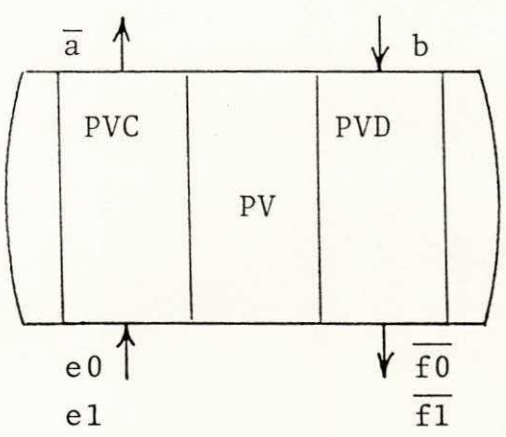


Figura 9.29 - Estrutura da restrição PV.

A projeção da restrição fim-a-fim F1 (do serviço SC) sobre o conjunto BF apaga, na expressão do comportamento de F1, os eventos que não pertencem a BF (Fig. 9.30):

$$p_{BF}(F1) \stackrel{def}{=} b.p_{BF}(F1)$$

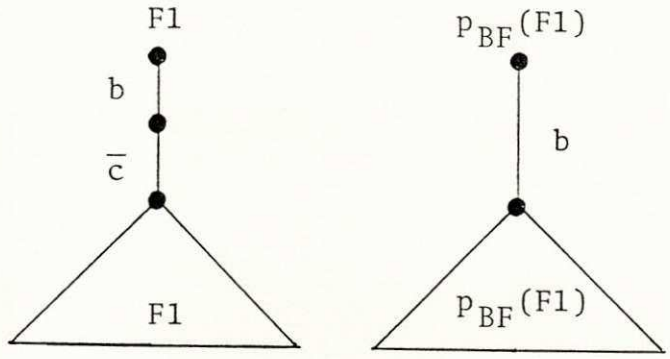


Figura 9.30 - Árvores de sincronização F1 e p_{BF} (F1).

A complementação das restrições fim-a-fim FD0 e FD1, atuantes na especificação do serviço disponível SD, são denotadas por $\overline{FD0}$ e $\overline{FD1}$.

$$\overline{FD0} \stackrel{def}{=} \overline{f0}.(\tau.g0.\overline{FD0} + \tau.\overline{FD0})$$

$$\overline{FD1} \stackrel{def}{=} \overline{f1}.(\tau.g1.\overline{FD1} + \tau.\overline{FD1})$$

A projeção de $\overline{FD0}$ sobre o conjunto BF apaga, na expressão do comportamento de $\overline{FD0}$, os eventos que não pertencem a BF (Fig. 9.31):

$$p_{BF}(\overline{FD0}) \stackrel{def}{=} f0.(p_{BF}(\overline{FD0}) + p_{BF}(\overline{FD0}))$$

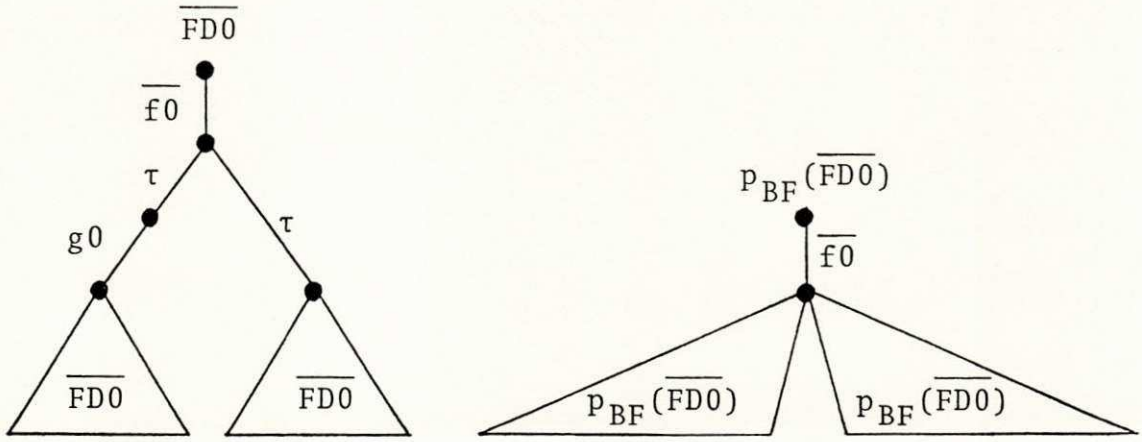


Figura 9.31 - Árvores de sincronização $\overline{FD0}$ e $p_{BF}(\overline{FD0})$.

A intercalação J0 de $p_{BF}(F1)$ com $p_{BF}(\overline{FD0})$ deve fornecer um comportamento onde o evento b precede a primeira ocorrência de $\overline{f0}$, na sequência de tentativas para a transmissão de um dado com o bit 0 (Fig. 9.32):

$$J0 \stackrel{def}{=} b.J0'$$

$$J0' \stackrel{def}{=} f0.(J0 + J0)$$

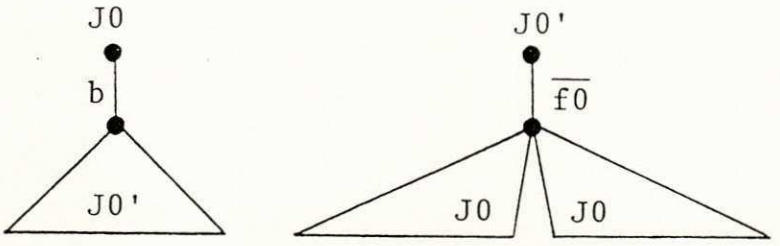


Figura 9.32 - Árvores de sincronização J0 e J0'.

De modo semelhante, a intercalação J1 de $p_{FB}(F1)$ com $p_{FB}(\overline{FD1})$ deve fornecer um comportamento onde o evento b precede a primeira ocorrência de $\overline{f1}$, na sequência de tentativas para a transmissão de um dado com o bit 1 (Fig. 9.33):

$$J1 \stackrel{def}{=} b.J1'$$

$$J1' \stackrel{def}{=} \overline{f1}.(J1 + J1)$$

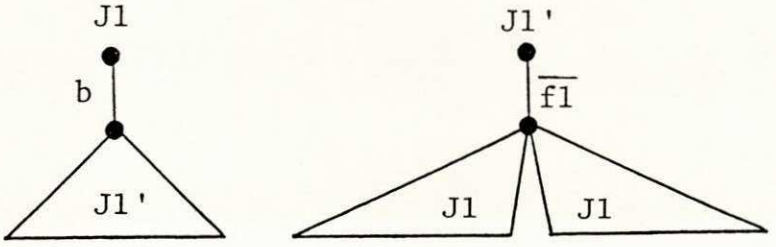


Figura 9.33 - Árvores de sincronização J1 e J1'.

A restrição ao fluxo de dados PVD pode ser obtida, através da intercalação de J0 e J1, impondo-se que o sucesso da transmissão de um dado com o bit 0 (ou com o bit 1) seja seguido pela tentativa de transmissão de um dado com o bit 1 (ou com o bit 0) (Fig. 9.34):

$$
\begin{aligned}
PVD &\stackrel{def}{=} b.PVD' \\
PVD' &\stackrel{def}{=} \overline{f0}.(b.PVD'' + PVD') \\
PVD'' &\stackrel{def}{=} \overline{f1}.(b.PVD' + PVD'')
\end{aligned}
$$

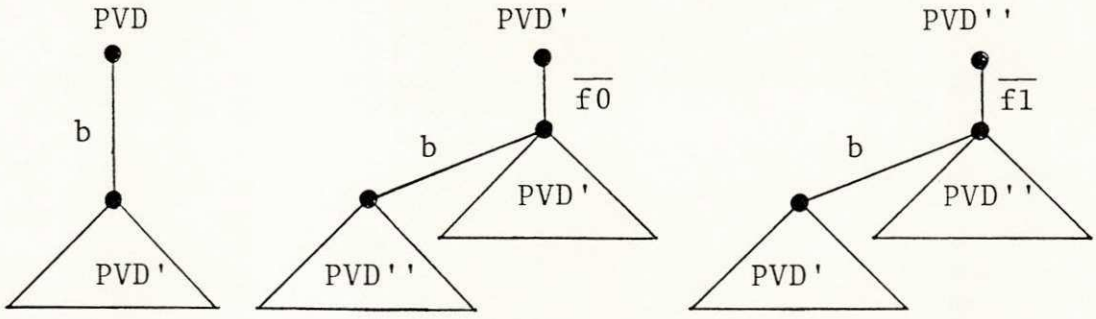


Figura 9.34 - Árvores de sincronização PVD, PVD' e PVD''.

A primeira equação que define PVD corresponde a um comportamento inicial do agente PVD (apenas o evento b ocorre). Após essa inicialização PVD comporta-se alternadamente como PVD' (para tentar transmitir um dado com o bit 0) ou com PVD'' (para tentar transmitir um dado com o bit 1).

A restrição PVC envolve a ocorrência de eventos exclusivamente nas portas pertencentes ao conjunto

$$EA = \{e0,\ e1,\ \overline{a}\}$$

A projeção da restrição fim-a-fim F2 (do serviço SC) sobre o conjunto EA apaga, na expressão do comportamento de F2, os eventos que não pertencem a EA (Fig. 9.39):

$$p_{EA}(F2) \stackrel{def}{=} \overline{a}.p_{EA}(F2)$$

As complementações das restrições fim-a-fim FC0 e FC1, atuantes na especificação do serviço disponível SD, são denotadas por $\overline{FC0}$ e $\overline{FC1}$.

$$\overline{FC0} \stackrel{def}{=} \overline{h0}.(\tau.e0.\overline{FC0} + \tau.\overline{FC0})$$
$$\overline{FC1} \stackrel{def}{=} \overline{h1}.(\tau.e1.\overline{FC1} + \tau.\overline{FC1})$$

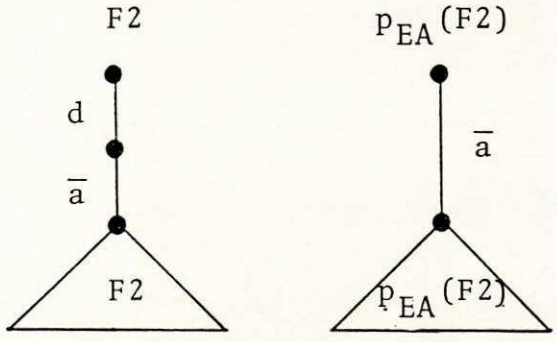


Figura 9.35 - Árvores de sincronização F2 e $p_{EA}(F2)$.

A projeção de $\overline{FC0}$ sobre o conjunto EA apaga, na expressão do comportamento de $\overline{FC0}$, os eventos que não pertencem a EA (Fig. 9.36):

$$p_{EA}(\overline{FC0}) \stackrel{def}{=} e0.p_{EA}(\overline{FC0}) + p_{EA}(\overline{FC0})$$

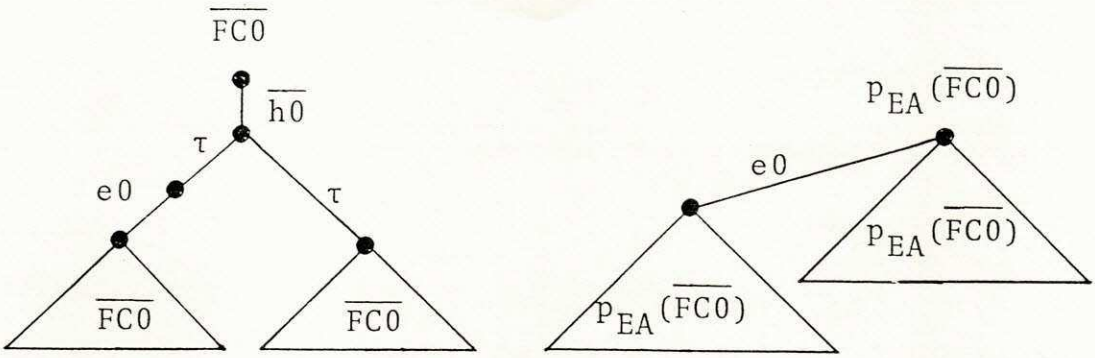


Figura 9.36 - Árvores de sincronização $\overline{FC0}$ e $p_{EA}(\overline{FC0})$.

A intercalação K0 de $p_{EA}(F2)$ com $p_{EA}(\overline{FC0})$ deve fornecer um comportamento onde a chegada de uma confirmação com o bit 0 (em e0) precede o oferecimento dessa confirmação ao Produtor (em $\overline{a}$) (Fig. 9.37):

$$K0 \stackrel{def}{=} e0.\overline{a}.K0$$

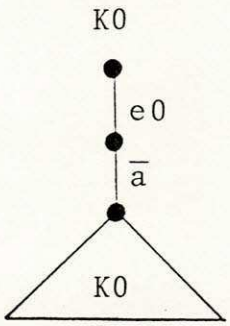


Figura 9.37 - Árvore de sincronização K0.

De modo semelhante, a intercalação K1 de $p_{EA}(F1)$ com $p_{EA}(\overline{FC1})$, deve fornecer um comportamento onde a chegada de uma confirmação com o bit 1 (em e1) precede o oferecimento dessa confirmação ao Produtor (em $\overline{a}$) (Fig. 9.38):

$$K1 \stackrel{def}{=} e1.\overline{a}.K1$$

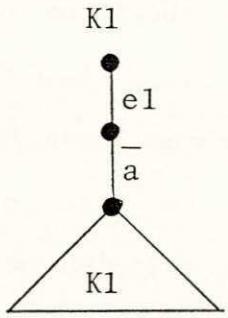


Figura 9.38 - Árvore de sincronização K1.

A restrição ao fluxo vertical de confirmações PVC pode ser obtida através da intercalação de K0 e K1. Como a alternância das transmissões (de dados e confirmações) com o bit 0 e com o bit 1 é imposta pela seleção que a restrição da interface inferior PI realiza sobre as restrições fim-a-fim do serviço disponível SD, a restrição PVC tem apenas que garantir a entrega das confirmações ao Produtor (Fig. 9.39):

$$PVC \stackrel{def}{=} e0.\overline{a}.PVC + e1.\overline{a}.PVC$$

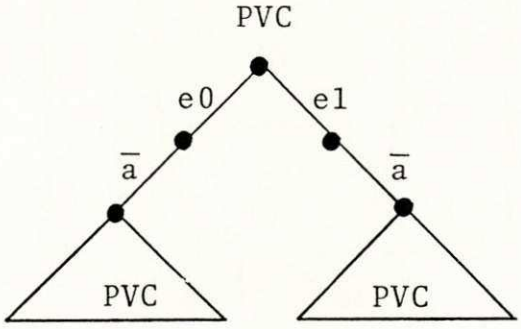


Figura 9.39 - Árvore de sincronização PVC.

9.2.2. Especificação da entidade de protocolo EC

As restrições atuantes na entidade EC (Fig. 9.40) podem ser obtidas de modo semelhante ao que foi utilizado para a obtenção das restrições atuantes na entidade EP (seção 9.3.1).

Empregando o estilo orientado para restrições

$$EC = (CS|CI)\ \&_D\ (CV)$$

onde

$$D = \{\bar{c},\ d,\ g0,\ g1,\ \overline{h0},\ \overline{h1}\}$$

CS é a restrição atuante na interface superior da entidade EC. Ela tem o comportamento da restrição local L2, do serviço SC, no sítio do Produtor:

$$CS = L2$$

isto é,

$$CS \stackrel{def}{=} \bar{c}.d.CS$$

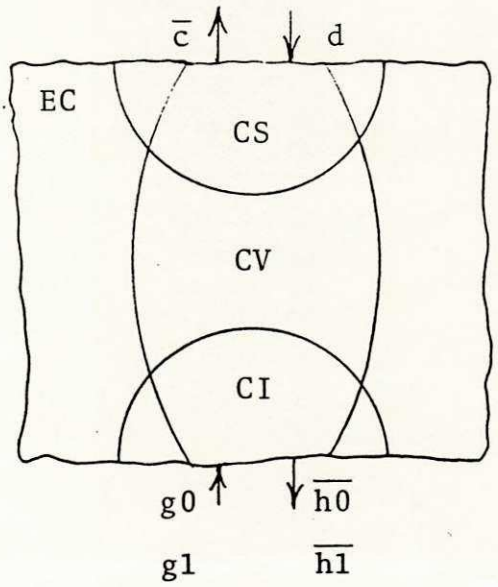


Figura 9.40 - Restrições da entidade EC.

CI é a restrição atuante na interface inferior da entidade EC. Ela pode ser obtida a partir das restrições locais SC0 e SC1 (ambas no sítio do Consumidor) da especificação do serviço disponível SD.

As complementações das restrições SC0 e SC1 são denotadas por $\overline{SC0}$ e $\overline{SC1}$ (Fig. 9.41):

$$\overline{SC0} \stackrel{def}{=} g0.\overline{h0}.\overline{SC0}$$

$$\overline{SC1} \stackrel{def}{=} g1.\overline{h1}.\overline{SC1}$$

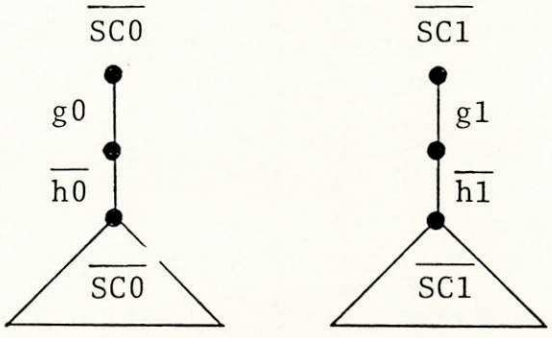


Figura 9.41 - Árvores de sincronização $\overline{SC0}$ e $\overline{SC1}$.

A restrição CI pode ser obtida através da intercalação de $\overline{SC0}$ e $\overline{SC1}$, impondo-se que após a chegada de um dado (com o bit 0 ou com o bit 1) deve ser enviada a confirmação correspondente (com o bit 0 ou com o bit 1) (Fig. 9.42):

$$CI \stackrel{def}{=} g0.\overline{h0}.CI + g1.\overline{h1}.CI$$

CV é a restrição ao fluxo vertical de dados e confirmações através da entidade de protocolo EC. Essa restrição pode ser estruturada considerando duas restrições: a primeira relacionada ao fluxo de dados (restrição CVD) e a segunda relacionada ao fluxo de confirmações (restrição CVC) (Fig. 9.43):

$$CV = (CVD|CVC)$$

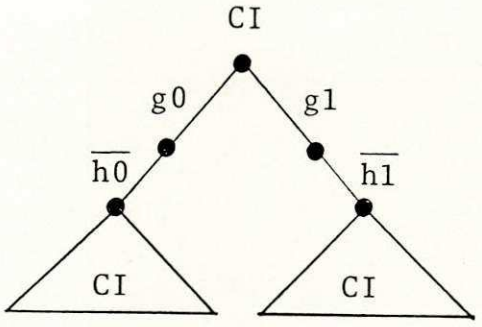


Figura 9.42 - Árvore de sincronização CI.

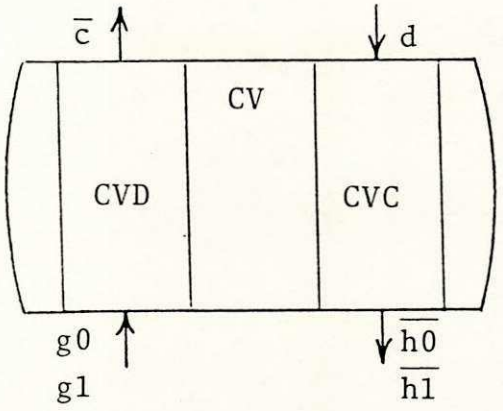


Figura 9.43 - Estrutura da restrição CV.

A restrição CVD envolve a ocorrência de eventos exclusivamente nas portas pertencentes ao conjunto GC = {g0, g1, $\overline{c}$}.

A projeção da restrição fim-a-fim F1 (do serviço SC) sobre o conjunto GC apaga, na expressão do comportamento de F1, os eventos que não pertencem a BF (Fig. 9.44):

$$p_{GC}(F1) \stackrel{def}{=} \overline{c}.p_{GC}(F1)$$

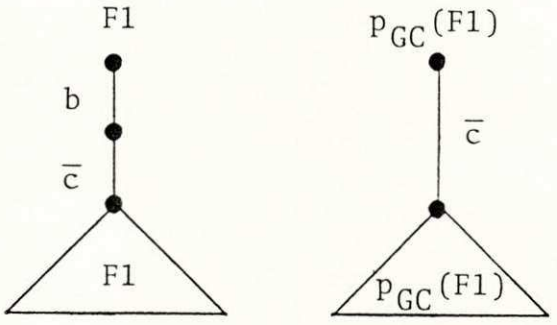


Figura 9.44 - Árvores de sincronização F1 e $p_{GC}(F1)$.

As complementações das restrições fim-a-fim FD0 e FD1, atuantes na especificação do serviço disponível SD, são denotadas por $\overline{FD0}$ e $\overline{FD1}$ (seção 9.2).

A projeção de $\overline{FD0}$ sobre o conjunto GC apaga, na expressão do comportamento de $\overline{FD0}$, os eventos que não pertencem a GC (Fig. 9.45):

$$p_{GC}(\overline{FD0}) \stackrel{def}{=} g0.p_{GC}(\overline{FD0}) + p_{GC}(\overline{FD0})$$

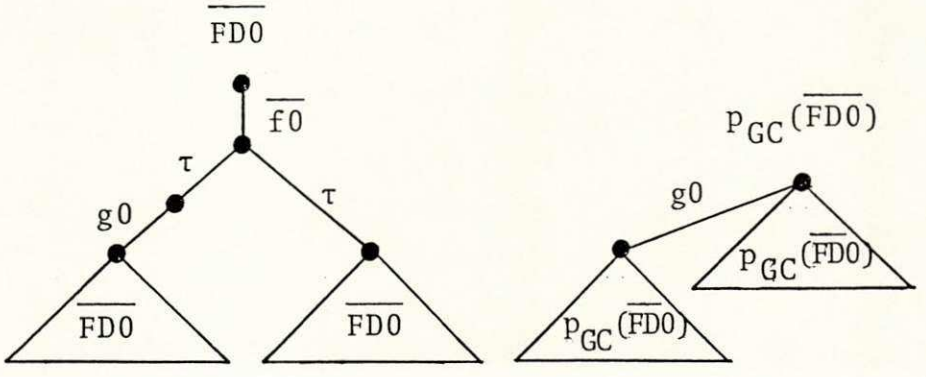


Figura 9.45 - Árvores de sincronização $\overline{FD0}$ e $p_{GC}(\overline{FD0})$.

A intercalação M0 de $p_{GC}(F1)$ com $p_{GC}(\overline{FD0})$ deve fornecer um comportamento onde o evento g0 (recepção de um dado com o bit 0) precede a ocorrência de $\overline{c}$ (entrega desse dado ao Consumidor) (Fig. 9.46):

$$M0 = \stackrel{def}{=} g0.\overline{c}.M0 + M0$$

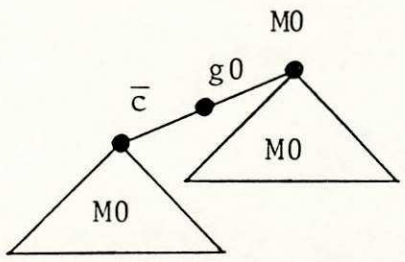


Figura 9.46 - Árvore de sincronização M0.

A projeção de $\overline{FD1}$ sobre o conjunto GC apaga, na expressão do comportamento de $\overline{FD1}$, os eventos que não pertencem a GC (Fig. 9.47):

$$p_{GC}(\overline{FD1}) \stackrel{def}{=} g1.p_{GC}(\overline{FD1}) + p_{GC}(\overline{FD1})$$

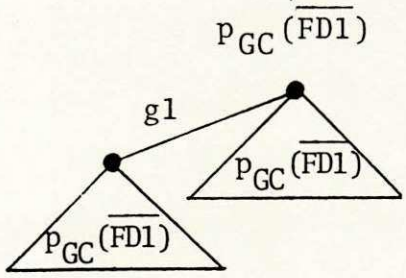


Figura 9.47 - Árvore de sincronização $p_{GC}(\overline{FD1})$.

A intercalação M1 de $p_{GC}(F1)$ com $p_{GC}(\overline{FD1})$ deve fornecer um

comportamento onde o evento g1 (recepção de um dado com o bit 1) precede a ocorrência de $\bar{c}$ (entrega desse dado ao Consumidor) (Fig. 9.48):

$$M1 \stackrel{def}{=} g1.\bar{c}.M1 + M1$$

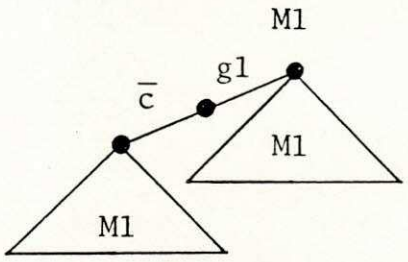


Figura 9.48 - Árvore de sincronização M1.

A restrição ao fluxo de dados CVD pode ser obtida através da intercalação de M0 com M1, impondo-se que o primeiro dado a ser recebido tenha o bit 0 e que um dado seja entregue ao Consumidor uma só vez (Fig. 9.49):

$$CVD \stackrel{def}{=} g0.\bar{c}.CVD'$$
$$CVD' \stackrel{def}{=} g1.\bar{c}.CVD'' + g0.CVD'$$
$$CVD'' \stackrel{def}{=} g0.\bar{c}.CVD' + g1.CVD''$$

A primeira equação que define CVD corresponde a um comportamento inicial do agente CVD (um dado com o bit 0 é recebido em g0 e entregue ao Consumidor em $\bar{c}$). Após essa inicialização CVD comporta-se alternadamente como CVD' (para esperar a chegada de um dado com o bit 1 e, no caso deste chegar, entregá-lo ao Consumidor) ou como CVD" (para esperar a chegada de um dado com o bit 0 e, no caso deste chegar, entregá-lo ao Consumidor).

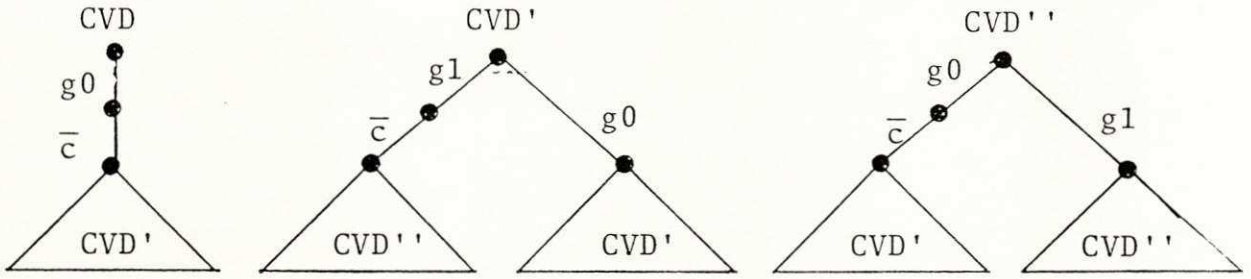


Figura 9.49 - Árvores de sincronização CVD, CVD' e CVD".

A restrição CVC envolve a ocorrência de eventos exclusivamente nas portas pertencentes ao conjunto DH = {d, $\overline{h0}$, $\overline{h1}$}.

A projeção da restrição fim-a-fim F2 (do serviço SC) sobre o conjunto DH apaga, na expressão do comportamento de F2, os eventos que não pertencem a DH (Fig. 9.50):

$$p_{DH}(F2) \stackrel{def}{=} \overline{a}.p_{DH}(F2)$$

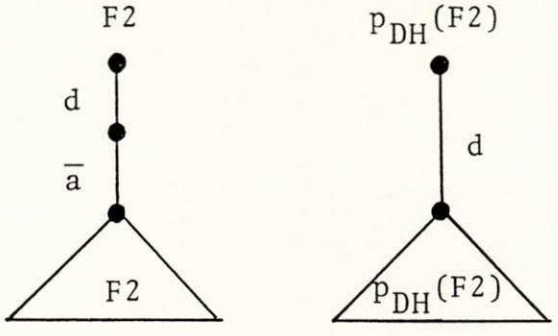


Figura 9.50 - Árvores de sincronização F2 e $p_{DH}(F2)$.

As complementações das restrições fim-a-fim FC0 e FC1, atuantes na especificação do serviço disponível SD, são denotadas por $\overline{FC0}$ e $\overline{FC1}$ (seção 9.2).

A projeção de $\overline{FC0}$ sobre o conjunto DH apaga, na expressão do comportamento de $\overline{FC0}$, os eventos que não pertencem a DH (Fig. 9.51):

$$p_{DH}(\overline{FC0}) \stackrel{def}{=} \overline{h0}.(P_{DH}(\overline{FC0}) + p_{DH}(\overline{FC0}))$$

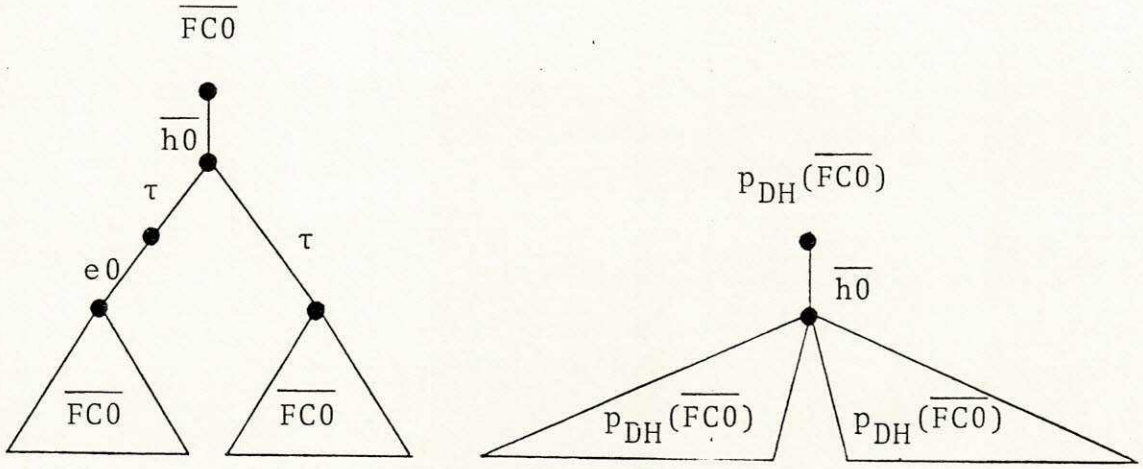


Figura 9.51 - Árvores de sincronização $\overline{FC0}$ e $p_{DH}(\overline{FC0})$.

A projeção de $\overline{FC1}$ sobre o conjunto DH apaga, na expressão do comportamento de $\overline{FC1}$, os eventos que não pertencem a DH (Fig. 9.52):

$$p_{DH}(\overline{FC1}) \stackrel{def}{=} \overline{h1}.(p_{DH}(\overline{FC1}) + p_{DH}(\overline{FC1}))$$

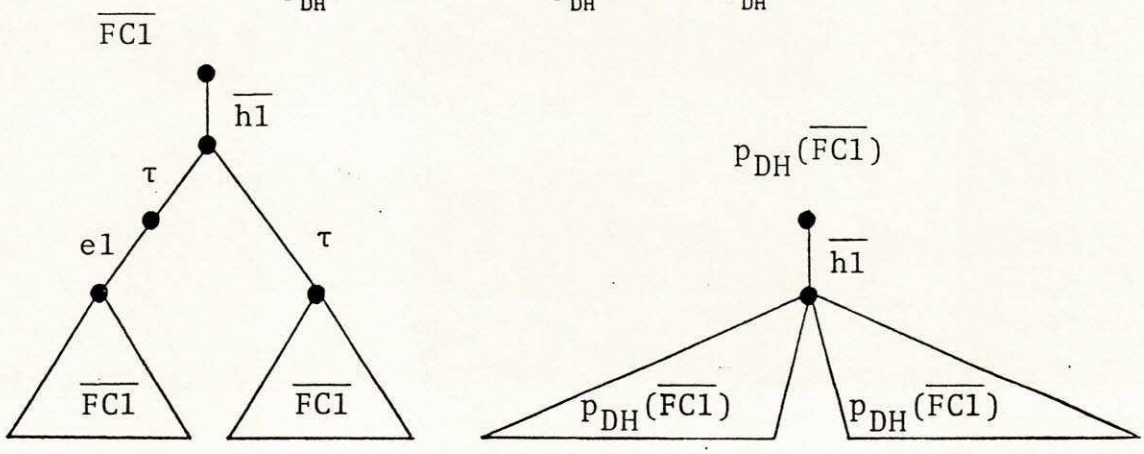


Figura 9.52 - Árvores de sincronização $\overline{FC1}$ e $p_{DH}(\overline{FC1})$.

A intercalação N1 de $p_{DH}(F2)$ com $p_{DH}(\overline{FC0})$ deve fornecer um comportamento onde o evento $\overline{h0}$ (entrega de uma confirmação com o bit 0 ao provedor do serviço disponível SD) é precedido pela ocorrência do evento d (recepção dessa confirmação do Consumidor) (Fig. 9.53):

$$N1 \stackrel{def}{=} d.\overline{h0}.(N1 + N1)$$

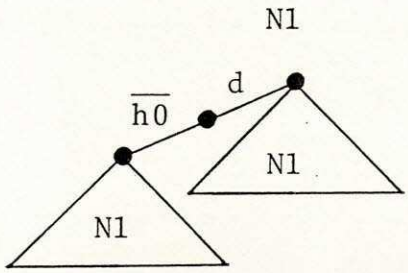


Figura 9.53 - Árvore de sincronização N1.

A intercalação N2 de $p_{DH}(F2)$ com $p_{DH}(\overline{FC1})$ deve fornecer um comportamento onde o evento $\overline{h1}$ (entrega de uma confirmação com o bit 1 ao provedor do serviço disponível SD) é precedido pela ocorrência do evento d (recepção dessa confirmação do Consumidor) (Fig. 9.54):

$$N2 \stackrel{def}{=} d.\overline{h1}.(N2 + N2)$$

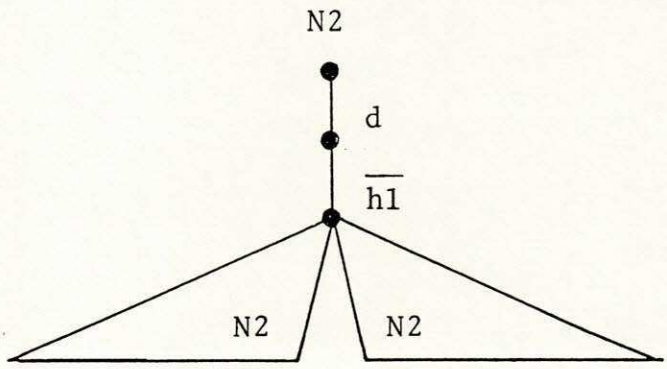


Figura 9.54 - Árvore de sincronização N2.

A restrição ao fluxo de confirmações CVC pode ser obtida, através da intercalação de N1 com N2, impondo-se que a primeira confirmação a ser enviada tenha o bit 0 e que a cada nova confirmação recebida do Consumidor seja atribuído um novo bit (Fig. 9.55):

$$CVC \stackrel{def}{=} d.\overline{h0}.CVC'$$

$$CVC' \stackrel{def}{=} d.\overline{h1}.CVC'' + \overline{h0}.CVC'$$

$$CVC'' \stackrel{def}{=} d.\overline{h0}.CVC' + \overline{h1}.CVC''$$

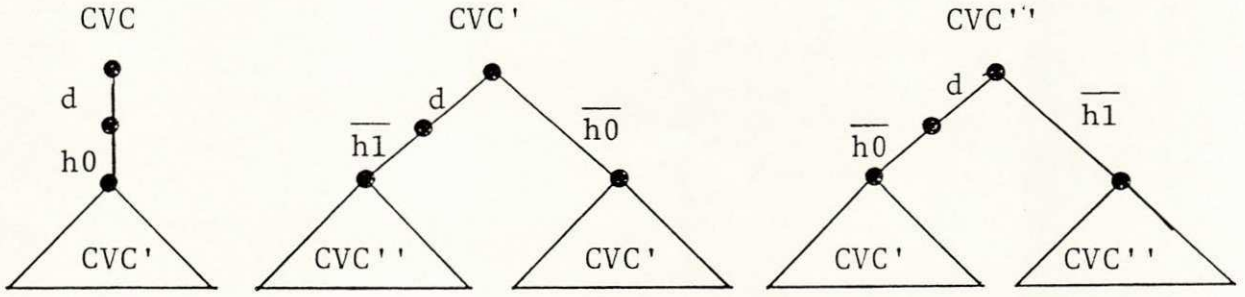


Figura 9.55 - Árvores de sincronização CVC, CVC' e CVC".

A primeira equação que define CVC corresponde a um comportamento inicial do agente CVC (uma confirmação é recebida do Consumidor em d e entregue ao provedor do serviço disponível SD em $\overline{h0}$). Após essa inicialização CVC comporta-se alternadamente como CVC' (para receber uma nova confirmação em d e enviá-la com o bit 1 ou para reenviar uma confirmação com o bit 0) ou como CVC" (para receber uma nova confirmação em d e enviá-la com o bit 0 ou para reenviar uma confirmação com o bit 1).

9.3. Validação do design do protocolo do bit alternante

A correção da especificação das entidades de protocolo EP e EC pode ser provada mostrando-se que a equação de contexto (eq. 9.2) é satisfeita.

Realizando a composição do serviço disponível SD com as entidades de protocolo EP e EC, e restringindo o conjunto D das portas de comunicação entre esses agentes, obtém-se o agente composto

$$AC = (SD|EP|EC)\backslash D$$

onde o serviço disponível SD (Fig. 9.22) é definido por

$SD = (SD1|SD2)\&_D(SDF)$

$SD1 = (SP0|SP1)$ (restrição local no sítio do Produtor)

$SP0 \stackrel{def}{=} f0.(\overline{e0}.SP0 + SP0)$

$SP1 \stackrel{def}{=} f1.(\overline{e1}.SP1 + SP1)$

$SD2 = (SC0|SC1)$ (restrição local no sítio do Consumidor)

$SC0 \stackrel{def}{=} \overline{g0}.h0.SC0$

$SC1 \stackrel{def}{=} \overline{g1}.h1.SC1$

$SDF = (FD0|FC0|FD1|FC1)$ (restrição fim-a-fim)

$FD0 \stackrel{def}{=} f0.(\tau.\overline{g0}.FD0 + \tau.FD0)$

$FC0 \stackrel{def}{=} h0.(\tau.\overline{e0}.FC0 + \tau.FC0)$

$FD1 \stackrel{def}{=} f1.(\tau.\overline{g1}.FD1 + \tau.FD1)$

$FC1 \stackrel{def}{=} h1.(\tau.\overline{e1}.FC1 + \tau.FC1)$

a entidade de protocolo EP (Fig. 9.25) é definida por

$EP = (PS|PI)\ \&_C\ (PV)$

$PS \stackrel{def}{=} b.\overline{a}.PS$ (restrição da interface superior)

$PI \stackrel{def}{=} \overline{f0}.(e0.PI' + \tau.PI)$ (restrição da interface inferior)

$PI' \stackrel{def}{=} \overline{f1}.(e1.PI + \tau.PI')$

$PV = (PVD|PVC)$ (restrição ao fluxo vertical de informações)

$PVD \stackrel{def}{=} b.PVD'$

$PVD' \stackrel{def}{=} \overline{f0}.(b.PVD'' + PVD')$

$PVD'' \stackrel{def}{=} \overline{f1}.(b.PVD' + PVD'')$

$PVC \stackrel{def}{=} e0.\overline{a}.PVC + e1.\overline{a}.PVC$

e a entidade de protocolo EC (Fig. 9.40) é definida por

$EC = (CS|CI)\ \&_D\ (CV)$

$CS \stackrel{def}{=} \overline{c}.d.CS$ (restrição da interface superior)

$CI \stackrel{def}{=} g0.\overline{h0}.CI + g1.\overline{h1}.CI$ (restrição da interface inferior)

$CV = (CVD|CVC)$ (restrição ao fluxo vertical de informações)

$CVD \stackrel{def}{=} g0.\overline{c}.CVD'$

$CVD' \stackrel{def}{=} g1.\overline{c}.CVD'' + g0.CVD'$

$CVD'' \stackrel{def}{=} g0.\overline{c}.CVD' + g1.CVD''$

$CVC \stackrel{def}{=} d.\overline{h0}.CVC'$

$CVC' \stackrel{def}{=} d.\overline{h1}.CVC'' + \overline{h0}.CVC'$

$CVC'' \stackrel{def}{=} d.\overline{g0}.CVC' + \overline{h1}.CVC''$

Os conjuntos de portas compartilhadas são

$C = \{\overline{a},\ b,\ e0,\ e1,\ \overline{f0},\ \overline{f1}\}$

$D = \{\overline{e0},\ \overline{e1},\ f0,\ f1,\ \overline{g0},\ \overline{g1},\ h0,\ h1\}$

A seguir prova-se que o agente composto AC é equivalente quanto à observação ao agente provedor do serviço desejado SC, isto é,

$$AC \approx SC$$

Para obter tal prova, inicialmente aplica-se a Lei de Expansão (seção 7.3) a AC. Obtém-se

$$AC \sim b.AC1$$

indicando que AC pode receber um dado do usuário Produtor na porta b e, a partir desse evento, passar a comportar-se como AC1. Continuando com as aplicações da Lei de Expansão, obtém-se

$$AC1 \sim \tau.(\tau.\tau.AC2 + \tau.AC2')$$

indicando que pode ocorrer uma sincronização da entidade de protocolo EP com o provedor do serviço disponível SD na porta f0, para que seja enviado um dado com o bit 0 (como f0 está restrita, o evento τ ocorre).

A escolha indeterminística que se segue apresenta duas alternativas que dependem do comportamento não confiável do provedor do serviço disponível SD.

A primeira alternativa indica que SD resolve a escolha indeterminística de modo que o dado seja entregue à entidade de protocolo EC. O primeiro evento τ corresponde à escolha efetuada e o segundo corresponde à entrega do dado na porta restrita g0. Nesse caso o sistema passa a comportar-se como AC2.

A segunda alternativa indica que SD resolve a escolha

indeterminística de modo que o dado é perdido (τ ocorre). Nesse caso o sistema passa a comportar-se como AC2'.

Neste ponto a análise prossegue com o comportamento identificado por AC2, ficando o comportamento identificado por AC2' para ser analisado posteriormente (I). Continuando com as aplicacões da Lei de Expansão, obtém-se

$$AC2 \sim \overline{c}.d.\tau.(\tau.\tau.AC3 + \tau.AC3')$$

indicando que a entidade de protocolo EC entrega o dado com o bit 0 ao usuário Consumidor na porta $\overline{c}$ e recebe deste a confirmação correspondente na porta d. Essa confirmação é entregue ao provedor do serviço disponível SD na porta restrita h0 (τ ocorre) para ser transmitida.

Novamente, a escolha indeterminística que se segue apresenta duas alternativas que dependem do comportamento não confiável do provedor do serviço disponível SD.

A primeira alternativa indica que SD resolve a escolha indeterminística de modo que a confirmação seja entregue à entidade de protocolo EP. O primeiro evento τ corresponde à escolha efetuada e o segundo corresponde à entrega da confirmação na porta restrita e0. Nesse caso o sistema passa a comportar-se como AC3.

A segunda alternativa indica que SD resolve a escolha indeterminística de modo que a confirmação é perdida (τ ocorre). Nesse caso o sistema passa a comportar-se como AC3'.

De modo semelhante ao que foi feito anteriormente, neste ponto a análise prossegue com o comportamento identificado por AC3, ficando o comportamento identificado por AC3' para ser analisado posteriormente (II). Continuando com as aplicações da Lei de Expansão, obtém-se

$$AC3 \sim \overline{a}.AC4$$

indicando que a entidade de protocolo EP entrega a confirmação ao usuário Produtor na porta $\overline{a}$. Em seguida o sistema passa a comportar-se como AC4 para receber um novo dado e transmiti-lo com o bit 1 de sequenciamento.

Os agentes AC e AC4 têm o mesmo comportamento observável, uma vez que o observador não distingue as transmissões de dados e confirmações com o bit 0 das transmissões de dados e confirmações com o bit 1.

(I) O agente AC2' corresponde à descrição do comportamento do sistema após a perda de um dado com o bit 0 de sequenciamento.

$$AC2' \sim \tau.\tau.\ (\tau.\tau.AC2 + \tau.AC2')$$

indicando que, na entidade de protocolo EP, a espera pela confirmação é limitada por um evento de temporização (τ ocorre). Após esse evento, a entidade de protocolo EP repete a entrega do dado com o bit 0 ao provedor do serviço disponível SD (τ ocorre), para que seja transmitido à entidade de protocolo EC.

O comportamento que se segue reflete o caráter não confiável do provedor do serviço disponível SD. Considerando que após um número finito de tentativas de transmissão o dado finalmente chega ao destino, o sistema passa eventualmente a comportar-se como AC2.

(II) O agente AC3' corresponde à descrição do comportamento do sistema após a perda de uma confirmação com o bit 0 de sequenciamento.

$$AC3' \sim \tau.\tau.(\tau.\tau.AC4' + \tau.AC3')$$

indicando a ocorrência do evento de temporização (τ ocorre) e o reenvio do dado com o bit 0 (τ ocorre).

Na escolha que se segue, a primeira alternativa indica que há sucesso na transmissão do dado ($\tau.\tau$ ocorre) e que o comportamento posterior do sistema é identificado por AC4'. A segunda alternativa indica o insucesso dessa transmissão (τ ocorre) e o sistema simplesmente volta a comportar-se como AC3'.

Considerando a primeira alternativa,

$$AC4' \sim \tau.(\tau.\tau.AC5 + \tau.AC3')$$

indicando que, desta vez o dado não é entregue ao usuário Consumidor para evitar duplicações, mas a confirmação é enviada (τ ocorre).

Novamente, o comportamento que se segue reflete o caráter não confiável do provedor do serviço disponível SD. No comportamento do agente AC5 considera-se que, após um número finito de tentativas, a confirmação chega finalmente ao destino e o sistema passa a comportar-se como AC4.

Colocando lado a lado as árvores de sincronização correspondentes às expansões de SC e de AC, pode-se estabelecer uma associação entre os nós dessas duas árvores, isto é, entre os estados de SC e de AC, como mostra a Fig. 9.56.

Como os pares de nós dessa associação constituem uma relação de bissimulação, completa-se a validação do design do protocolo de bit alternante.

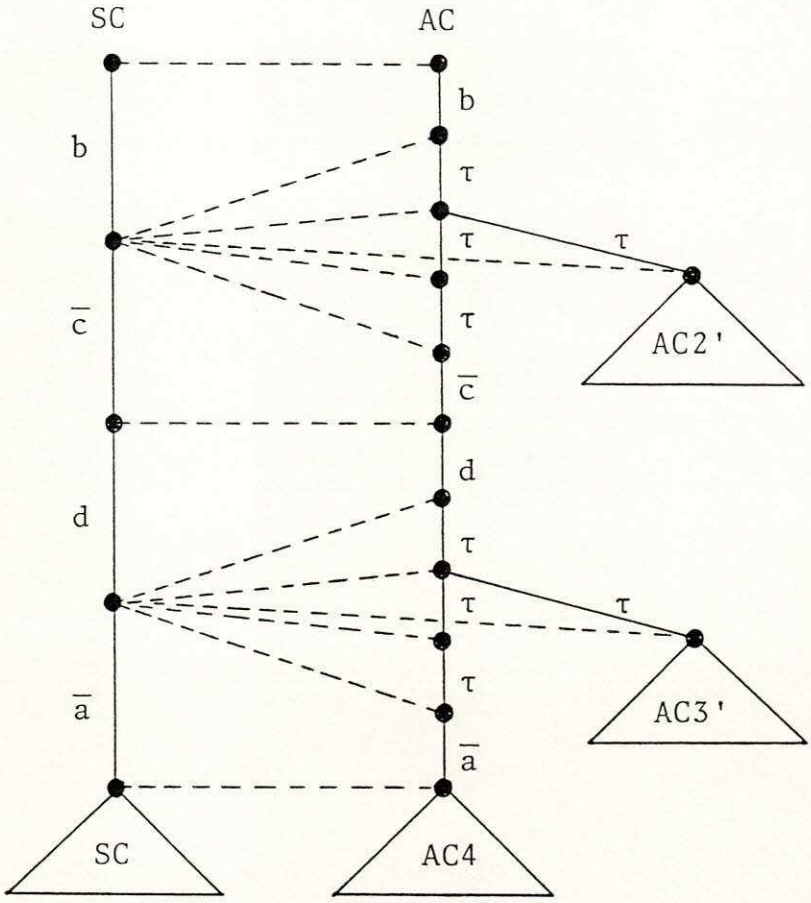


Figura 9.56 - Associação entre os estados de SC e AC.

9.4. Avaliação da abordagem proposta

A abordagem proposta neste trabalho situa-se entre a abordagem clássica e a abordagem transformacional. Do mesmo modo que a abordagem clássica ela exige do projetista a realização de uma prova de correção após cada refinamento do sistema que está sendo desenvolvido, e assim como a abordagem transformacional ela pocura fornecer ao projetista um procedimento parcialmente padronizado para o desenvolvimento desse sistema.

Entretanto, diferentemente de ambas, a abordagem proposta tem como preocupação fundamental a estruturação do sistema. Tal estruturação é obtida, ao longo dos refinamentos sucessivos, explorando-se as características de diferentes estilos de especificação.

Inicialmente, um sistema distribuído é especificado considerando o

serviço que pode oferecer. Essa especificação de serviço é realizada com o emprego do estilo orientado para restrições, que permite estruturar a descrição do sistema sem revelar-lhe a composição interna.

Para o refinamento do sistema é empregado o estilo orientado para recursos, que permite definir cada um dos componentes do sistema (as entidades de protocolo e o provedor do serviço subjacente) no estilo orientado para restrições. A abordagem proposta estabelece então um procedimento parcialmente padronizado para a realização da especificação das entidades de protocolo.

A abordagem proposta considera um modelo para as especificações de serviço e um outro modelo para as especificações das entidades de protocolo. Explorando as analogias que existem entre os tipos de restrições atuantes nesses modelos, ela permite obter as restrições atuantes nas entidades de protocolo a partir da transformação das restrições atuantes nas especificações de serviço.

A transformação das restrições envolve operações bem caracterizadas como projeções de restrições sobre um conjunto de eventos e complementações de restrições, mas envolve também procedimentos que dependem do sistema particular em desenvolvimento. Esses procedimentos incluem a intercalação de comportamentos e exigem a participação atenta do projetista para que expressões de comportamento corretas e expressas de modo adequado sejam obtidas em cada caso.

O conjunto de procedimentos derivados da abordagem proposta constitui uma metodologia de design que procura atender aos princípios de ortogonalidade, generalidade e flexibilidade.

O princípio de ortogonalidade (segundo o qual aspectos independentes devem ser especificados independentemente) é atendido na medida em que o estilo orientado para restrições permite isolar os aspectos locais dos aspectos fim-a-fim (nas especificações de serviço) e os aspectos de interface dos aspectos de fluxo vertical de informações (nas especificações das entidades de protocolo).

O princípio da generalidade (segundo o qual devem ser buscadas construções gerais reutilizáveis) é atendido na medida em que são adotados modelos gerais (para a construção das especificações de serviço e para a construção das entidades de protocolo), na medida em que são definidos procedimentos gerais para a obtenção das restrições atuantes nas entidades de protocolo e, acima de tudo, na medida em que são utilizados agentes básicos predefinidos para a descrição final dos sistemas.

O princípio de flexibilidade (segundo o qual não se deve dificultar a realização de extensões futuras) é atendido na medida em que as especificações de serviço e as especificações das entidades de protocolo podem ser submetidas a modificações e inclusões facilmente, graças à estruturação fornecida pelo estilo orientado para restrições.

A abordagem proposta para a especificação de sistemas distribuídos encaixa-se na concepção de um sistema organizado em camadas à semelhança do Modelo Básico de Referência para a Interconexão de Sistemas Abertos (Modelo OSI) da ISO. Assim, após a estruturação dos serviços em camadas, o protocolo correspondente a cada camada pode ser obtido através da transformação das restrições atuantes nas especificações de serviço (serviço oferecido pelas entidades de protocolo e serviço utilizado por essas entidades).

O serviço que cada camada do sistema oferece é especificado com a utilização do estilo orientado para restrições. Nessas especificações distinguem-se os aspectos locais e os aspectos fim-a-fim de cada camada.

Ao refinar-se um desses serviços utiliza-se o estilo orientado para recursos. Nesse caso os recursos são as entidades de protocolo da camada e o provedor do serviço subjacente. Cada recurso é especificado com a utilização do estilo orientado para restrições.

Cada entidade de protocolo é refinada progressivamente, alternando-se o estilo orientado para restrições com o estilo orientado para recursos. A versão final de cada entidade de protocolo (e portanto, do sistema todo) pode ser apresentada em termos de agentes básicos predefinidos. Tais

agentes básicos podem ser descritos no estilo monolítico e/ou no estilo orientado para estados, visando facilitar a tarefa de implementação do sistema.

CAPÍTULO X

CONCLUSÕES

O esforço realizado no sentido de estender a aplicabilidade da abordagem transformacional obteve sucesso parcial: levou à concepção de um novo tipo de abordagem, preso ainda à abordagem clássica, mas buscando os elementos (isto é, as regras de transformação) que caracterizam a abordagem transformacional.

Esse trabalho teve início com o estudo de vários aspectos gerais relacionados com os sistemas distribuídos e com as redes de computadores. Em seguida foi dirigido para o estudo das questões ligadas ao desenvolvimento de tais sistemas. Nessa fase, uma atenção especial foi dedicada às etapas de especificação e validação de protocolos de comunicação.

Relativamente a tais etapas de desenvolvimento foi realizado o estudo de um conjunto significativo de TDFs. Esse estudo resultou em um interesse maior por CCS, devido às características de simplicidade, poder de análise e o embasamento matemático dessa álgebra.

As investigações subsequentes foram encaminhadas no sentido da definição de procedimentos sistemáticos para o desenvolvimento de protocolos, considerando CCS como linguagem de especificação. Tais investigações tiveram início com o estudo da abordagem clássica. O estudo desse tipo de abordagem (baseado no refinamento progressivo das especificações com provas de correção a posteriori) levou à proposição de uma metodologia na qual o especificador, partindo de uma especificação de serviço, desenvolve passo-a-passo a especificação do protocolo correspondente.

De acordo com tal metodologia cada refinamento de um sistema se dá pela substituição de uma caixa preta (um agente CCS) por um conjunto de outras

caixas pretas, que compostas e tendo as suas intercomunicações escondidas do observador, guardam uma relação de equivalência de observação (≈) com a primeira. Uma característica dessa metodologia é a possibilidade do emprego de agentes básicos predefinidos para a apresentação da versão final do sistema. A utilização de tais agentes tem como objetivo tornar mais fácil a tarefa de implementação.

A abordagem clássica apóia-se consideravelmente na criatividade do especificador, o que dificulta o seu emprego na produção automatizada de sistemas.

A fim de estabelecer uma disciplina mais dirigida para a automatização, o trabalho de pesquisa voltou-se para a abordagem transformacional. O estudo das peculiaridades desse outro tipo de abordagem levou à proposição de uma segunda metodologia na qual o especificador (homem ou máquina) realiza o desenvolvimento de um sistema através da aplicação de regras de transformação às especificações desse sistema. Tais regras produzem resultados sempre corretos se aplicadas no domínio para o qual foram definidas. Por isso dispensam a prova de correção a posteriori.

Seguindo a abordagem transformacional, a metodologia proposta caracteriza-se pela algoritmização do processo de obtenção das entidades de protocolo. Os algoritmos propostos levam à descrição de duas dessas entidades a partir da especificação do serviço que elas devem oferecer.

As pesquisas dirigidas para a concepção de metodologias que adotam a abordagem transformacional vêm obtendo resultados muito importantes, a nível internacional. Entretanto, pode-se dizer que esse tipo de abordagem apresenta muitas limitações para o desenvolvimento de sistemas complexos.

O grande problema desse tipo de abordagem é que os algoritmos obtidos normalmente só podem ser aplicados a casos simples. Espera-se que a intensa pesquisa, que vem sendo desenvolvida nessa área, resulte em algoritmos poderosos, capazes de serem aplicados em situações práticas.

Para superar as limitações da abordagem transformacional, neste

trabalho buscou-se introduzir, nesse tipo de abordagem, novos elementos. Desse modo chegou-se à proposição de um novo tipo de abordagem.

A abordagem proposta distingue-se das abordagens anteriores (clássica e transformacional) por adotar uma estratégia baseada na exploração do potencial descritivo dos estilos de especificação. Essa preocupação com estilos tem como objetivo impor um alto grau de estruturação às especificações. Nesse sentido, o estilo orientado para restrições é especialmente visado.

Contudo, para permitir o estilo orientado para restrições, CCS precisou ser estendido. As extensões realizadas neste trabalho referem-se ao operador de conjunção ($\&_A$). Do modo como foi proposto por Robin Milner, o operador $\&_A$ expressava "three-way synchronization" (permitindo que dois agentes recebessem simultaneamente um mesmo evento de sincronização oferecido pelo ambiente). Do modo como é proposto neste trabalho, o operador $\&_A$ pode expressar "multi-way synchronization" (permitindo que dois ou mais agentes recebam ou ofereçam, simultaneamente, um mesmo evento de sincronização com o ambiente ou com outro agente).

A nova abordagem permite que as restrições atuantes nas entidades de protocolo sejam obtidas como resultados da manipulação de dois conjuntos de restrições. O primeiro conjunto refere-se às restrições definidas para a especificação do serviço que o protocolo deve oferecer (restrições do serviço desejado), e o segundo conjunto refere-se às restrições definidas para a especificação do serviço que as entidades de protocolo podem utilizar (restrições do serviço disponível).

Essa abordagem levou à proposição de uma metodologia em que as manipulações das restrições envolvem projeções de restrições sobre conjuntos de portas (para o apagamento de portas), complementações de restrições (para transformar portas oferecedoras em portas receptoras e vice-versa) e intercalações de restrições (para gerar um comportamento a partir de dois outros). A metodologia não é totalmente automatizável porque, por exemplo, nos

casos de intercalação de comportamentos, a participação criativa do especificador torna-se fundamental.

Somente em alguns casos as restrições atuantes nas entidades de protocolo são obtidas através de intercalações de comportamentos. Nesses casos os comportamentos intercalados fornecem o conjunto completo dos eventos nos quais as restrições podem participar. Esses comportamentos fornecem também algumas sugestões sobre a estruturação (em termos de sequenciamento de ações, escolhas indeterminísticas, recursividades etc.) das restrições.

Apesar dessas facilidades, cada sistema exige um procedimento diferente para a realização das intercalações. É nesse momento que o especificador intervém para definir expressões de comportamento corretas, expressas da forma mais conveniente possível.

Como a intervenção do especificador pode se dar equivocadamente, faz-se necessária a prova de correção das especificações a posteriori. De modo semelhante ao que é feito na metodologia correspondente à abordagem clássica, essa prova consiste em demonstrar que a composição das entidades de protocolo com o provedor do serviço disponível (uma vez ocultadas as portas de sincronização das entidades com esse provedor de serviço) é equivalente em observação ($\approx$) ao provedor do serviço desejado. Como na abordagem proposta o problema da especificação de protocolos é definido, formalmente, através de uma equação de contexto, provar a correção da especificação de um protocolo significa provar que ela satisfaz a equação de contexto que define o problema.

A abordagem proposta neste trabalho vai além da idéia de que as especificações de protocolo devem ser precedidas e condicionadas por especificações de serviço. Ela sugere uma sequência de atividades de desenvolvimento que se apóia nessa precedência e expõe claramente como de fato se dá esse condicionamento.

Sobretudo, a abordagem proposta explora tal condicionamento, tomando-o como base para a realização da especificação de protocolos a partir de especificações de serviço. As restrições que, conjuntamente, definem as

entidades de protocolo são obtidas como transformações bem caracterizadas das restrições que definem as especificações do serviço desejado e do serviço disponível.

Essa abordagem disciplina a tarefa do projetista, apresentando indicativos de como deve ser realizada a especificação global do sistema e de como deve ser realizado o refinamento progressivo dessa especificação. Seguindo tais indicativos, as especificações resultam estruturadas e, na sua versão final, podem assumir uma forma adequada à fase de implementação.

10.1. Sugestões para a continuação desta linha de pesquisa

O presente trabalho dá origem a muitas sugestões de pesquisa. Essas sugestões situam-se principalmente em áreas de estudo ligadas ao desenvolvimento de metodologias que visam o uso sistemático de técnicas de descrição formal (TDFs).

Para organizar a apresentação dessas sugestões, quatro direções principais de pesquisa são consideradas:

1) realização de extensões a CCS,

2) concepção e construção de ferramentas,

3) definição e análise de estilos de especificação e

4) estudo de abordagens para o desenvolvimento de sistemas.

Realização de extensões a CCS

Para dar continuidade às extensões de CCS, a "multi-way synchronization" que, no presente trabalho, ficou limitada ao CCS Básico, poderia ser estendida ao CCS Completo. Nesse caso, os agentes especiais (produtos estendidos) deveriam ser especificados com a utilização de variáveis associadas às portas de comunicação desses agentes.

Por outro lado, uma das limitações de CCS está relacionada à ausência dessa TDF para a formalização da representação dos dados. Por isso justifica-se um esforço de pesquisa para prover CCS desses recursos.

Tal esforço pode ser dirigido para a anexação, com possivelmente alguma compatibilização, de uma TDF já desenvolvida que seja voltada para os aspectos de dados. Outra possibilidade é buscar a criação de uma TDF voltada para os aspectos de dados, que possa ser utilizada em harmonia com as características de CCS.

Em ambos os casos seria importante desenvolver uma metodologia que permitisse o uso desse CCS estendido de acordo com princípios de design bem estabelecidos.

No que diz respeito aos aspectos de controle de sistemas, CCS poderia ser estendido através da introdução da noção de intervalo de tempo entre a ocorrência de ações assim como da noção de duração de ações. Outra possibilidade seria a criação de meios para expressar escolhas indeterminísticas entre comportamentos com indicações de que algumas alternativas têm mais chance de ocorrer do que outras.

Concepção e construção de ferramentas

O especificador de sistemas pode valer-se do uso de ferramentas automáticas para obter benefícios como maior produtividade, maior legibilidade, maior segurança etc.. Essas ferramentas podem ser utilizadas tanto nas tarefas de síntese como nas tarefas de análise.

No caso de síntese de sistemas, a disponibilidade de editores voltados para as características de CCS como símbolos, formas de apresentação etc., seria bem-vinda. Tais editores poderiam oferecer facilidades para a escolha de estilos de especificação e de abordagens de desenvolvimento, assim como realizar a simplificação de especificações e produzir sinais de alerta em casos de erro. A disponibilidade de recursos gráficos e animados seriam de interesse.

Os algoritmos propostos para a abordagem transformacional poderiam ser implementados, de forma a constituírem ferramentas de auxílio ao especificador. Esses algoritmos poderiam ser modificados para terem um domínio

de aplicações mais amplo e levarem a resultados mais otimizados. Uma ferramenta mais completa poderia ser obtida através da integração e da implementação desses algoritmos.

A análise de especificações com a finalidade de prova de propriedades é uma tarefa árdua quando realizada com recursos exclusivamente manuais. Para o alívio e a segurança do especificador, essas tarefas poderiam ser automatizadas no todo ou em parte. Também aqui poderiam ser utilizados recursos gráficos e animados.

Definição e análise de estilos de especificação

No presente trabalho a investigação relacionada com os estilos de especificação em CCS concentra-se em quatro estilos básicos: monolítico, orientado para recursos, orientado para restrições e orientado para estados. Entretanto, estilos mistos poderiam ser definidos, assim como estilos novos. Nessa linha de pesquisa caberia investigar as propriedades desses estilos e os modos como eles poderiam ser empregados para a descrição de sistemas.

Cabe ainda investigar como esses estilos se apresentam em outras TDFs. Além disso, que extensões são necessárias a essas TDFs para que elas possam admitir tais estilos.

Outra possibilidade é investigar o efeito do uso de estilos de especificação sobre a representação dos dados. Tanto no caso de especificação informal como no caso de especificação formal desses dados.

Estudo de abordagens para o desenvolvimento de sistemas

Três abordagens são apresentadas neste trabalho: a clássica, a transformacional e a proposta. Nessa linha de pesquisa podem ser investigadas variações sobre as abordagens apresentadas, que poderiam ser caracterizadas como novos tipos de abordagem.

A abordagem clássica e a abordagem transformacional situam-se em extremos opostos no que diz respeito à participação criativa que é concedida

ao projetista. A abordagem proposta situa-se em um ponto intermediário. Outros pontos intermediários poderiam ser estabelecidos para a definição de novas abordagens.

Dando continuidade às investigações sobre a abordagem proposta, um esforço poderia ser realizado para que as transformações das restrições pudessem ser estabelecidas em termos de procedimentos automatizáveis. Provavelmente, novas operações (além de complementação e projeção) precisariam ser definidas. Essas novas operações estabeleceriam, por exemplo, diferentes formas de intercalação de eventos.

A abordagem proposta privilegia a estruturação das especificações em um desenvolvimento top-down. Outras abordagens poderiam privilegiar, por exemplo, a facilidade de realização de provas de correção ou o uso de agentes básicos predefinidos em um desenvolvimento bottom-up.

Neste trabalho as três abordagens são estudadas no contexto de CCS. entretanto, elas podem ser reelaboradas para dar apoio à utilização de outras TDFs.

O uso de ferramentas automáticas é conveniente em qualquer uma das abordagens apresentadas. No caso da abordagem clássica as ferramentas poderiam auxiliar nas tarefas de editoração e de prova de correção. No caso da abordagem transformacional as ferramentas poderiam ser construídas a partir dos algoritmos disponíveis, ou a partir de algoritmos mais abrangentes, que implementam as regras de transformação. No caso da abordagem proposta, as ferramentas poderiam realizar alguns tipos de transformação e provas de correção, assim como a editoração baseada em estilos.

REFERÊNCIAS BIBLIOGRÁFICAS

[AmNa 76] American National Standards Institute;"American national standard programming language PL/I", New York, 1976.

[Ansa 82] Ansart, J.P.; "GENEPI/A: a protocol independent system for testing protocol implementation". In: Proceedings of the 2nd International Symposium on Protocol Specification, Testing, and Verification, pp. 539-554, 1982.

[BaSc 69] Bartlett, K.; Scantlebury, R. and Wilkinson, W.; "A note on realiable full-duplex transmission over half-duplex links", Communications of ACM, V. 12, N. 5, pp. 260-261, 1969.

[BiLi 70] Birkhoff G. e Lipson, J.D.; "Heterogeneous algebras", Journal of Combinatorial Theory, V. 8, pp. 115-133, 1970.

[Boch 75] Bochmann, G.v.; "Logical verification and implementation of protocols", publication 190, Département d'Informatique, Université de Montréal, Canadá, 1975.

[Boch 78] Bochmann, G.v.; "Finite state description of communication protocols", Computer Networks and ISDN Systems, Vol. 2, N. 4/5, pp. 361-372, 1978.

[Boch 80] Bochmann, G.v.; "A general transition model for protocols and communication services", IEEE Transactions on Communications, Vol. 28, N. 4, pp. 643-650, 1980.

[Boch 83] Bochmann, G.v.; "Concepts for distributed systems design", Springer-Verlag, Berlim, 1983.

[Boch 87] Bochmann, G.v.; "Specifications of a simplified transport protocol using different formal description techniques", Technical Report, Université de Montréal, 1987.

[Boch 90] Bochmann, G.v.; "Protocol specification for OSI", Computer Networks and ISDN Systems, V. 18, pp. 167-184, 1990.

[BoGe 76] Bochmann, G.v. e Gecsei, J.; "A unified method for the specification and verification of protocols". In: Proceedings of the IFIP Congress, pp. 229-234, 1976.

[BoGo 86] Bochmann, G.v. and Gotzhein, R.; "Deriving protocol specifications from service specifications", publication 562, Université de Montréal, Canadá, 1986.

[BoSu 80] Bochmann, G.v.; e Sunshine, C.A.; "Formal methods in communication protocol design", IEEE Transactions on Communications, V 28, N. 4, pp. 624-631, 1980.

[BoVe 87] Bochmann, G.v.; and Verjus, J.P.; "Some comments on 'transition-oriented' versus 'structured' specification of distributed algorithms and protocols", IEEE Transactions on Software Engineering, Vol. 13, pp. 501-505, 1987.

[BrHo 84] Brookes, S. D.; Hoare, C.A.R. e Roscoe, A.W.; "A theory of communicating sequential process", Journal of the ACM, V. 31, N. 3, pp. 560-599, 1984.

[Brin 86] Brinksma, E.; "A tutorial on LOTOS". In: Proceedings of the 5th International Symposium on Protocol Specification, Testing, and Verification, pp. 171-194, 1986.

[BrJo 78] Brand, D. e Joyner Jr.; W.H.; "Verification of protocols using symbolic execution", Computer Networks, N. 2, pp. 351-360, 1978.

[BrKa 85] Brinksma, E. e Karjoth, G.; "A specification of the OSI transport service in LOTOS". In: Proceedings of the 4th International Symposium on Protocol Specification, Testing, and Verification, pp. 227-251, 1985.

[BrSc 87] Brinksma, E.; Scollo, G. e Steenbergen, C.; "LOTOS specification, their implementations and their tests", Technical Report, Twente University of Technology, The Netherlands, 1987.

[CCIT 84] CCITT Recommendation X.200; "Reference model of open systems interconnection for CCITT applications", Genebra, 1984.

[ChAo 87] Chang. C. K.; Aoyama, M. e Jiang, T.M.; "Design methods for distributed software systems". In: Proceedings of the National Computer Conference, pp. 477-483, 1987.

[ClPa 89] Cleaveland, R.; Parrow, J. e Steffen, B.; "A semantics - based verification tool for finite - state systems". In: Proceedings of the 9th International Symposium on Protocol Specification, Testing, and Verification, 1989.

[CoDe 87] Courtiat, J.-P.; Dembinski, P.; Groz, R. e Jard, C.; "Estelle: un langage ISO pour les algorithmes distribués et les protocoles", Technique et Science Informatiques, V. 2, pp. 89-102, 1987.

[Colm 85] Colmerauer, A.; "Prolog in 10 figures", Communications of the ACM, Vol. 28, N. 12, pp. 1296-1310, 1985.

[Corn 81] Cornafion (nom collectif); "Systèmes informatiques répartis", Dunod, Paris, 1981.

[Dant 80] Danthine, A.A.S.; "Protocol representation with finite-state models", IEEE Transactions on Communications, Vol. 28, N. 4, pp. 632-642, 1980.

[DaZi 83] Day, J.D. e Zimmermann, H.; "The OSI reference model", Proceedings of IEEE, V. 71, N. 12, pp. 1334-1340, 1983.

[DiCh 83] Dickson, G.J. and Chazal, P.E. de; "Status of CCITT description techniques and application to protocol specification", In: Procedings of the IEEE, Vol. 71, N. 12, pp. 1346-1355, 1983.

[EhMa 85] Ehrig, H. and Mahr, B.; "Fundamentals of algebraic specification 1 - equations and initial semantics", Springer-Verlag, 1985.

[Eijk 89] Eijk, P.v.; "Tools for LOTOS specification style transformation". In: Proceedings of the 3rd International Conference on Formal Description Techniques, 1990.

[FaLi 87] Favreau, J.P. e Linn Jr.; R.J.; "Automatic generation of test scenario skeletons from protocol specifications written in Stelle". In: Proceedings of the 6th International Symposium on Protocol Specification, Testing, and Verification, pp. 191-202, 1987.

[FaSc 83] Faro, A. e Scollo, G.; "SDL and CCS based description of communicating entities". In: Proceedings of the European Teleinformatics Conference, pp. 577-586, 1983.

[GhJa 85] Ghezzi, C. e Jazayeri, M.; "Conceitos de linguagens de programação", Tradução de Paulo A. S. Veloso, Editora Campus, Rio de Janeiro, 1985.

[GiMa 86] Giozza, W.F.; Marinho de Araújo, J. F.; Moura, J.A.B. e Sauvé, J.P.; "Redes locais de computadores - tecnologia e aplicações", McGraw-Hill, 1986.

[Groz 83] Groz, R.; "Description de l'algorithme de Mossiere - Tchwente - Verjus avec le langage de description FDT de l'ISO et du CCITT", Technical Report, CNET/EVP, Lannion, France, 1983.

[GuPe 83] Gustavsson, R. and Pehrson, B,; "The power of some formal models of distributed computing". In: Proceedings of the 3rd Protocol Specification, Testing, and Verification, pp. 77-86, 1983.

[Henn 88] Hennessy, M.; "Algebraic theory of process", MIT Press, London, 1988.

[HoLo 85] Hopelain, D. and Loesh, B.; "Automated development methodologies - overview and conclusions", Data Processing, Vol. 27, N. 2, pp. 43-45, 1985.

[Holz 82] Holzmann, G.J.; "Algebraic validation methods: a comparison of three methods". In: Proceedings of the 2nd International Workshop on Protocol Specification, Testing, and Verification, Idyllwild, 1982.

[Hung 86] Hung, C.C.; "CCS used as a proof-assistant tool". In: Proceedings of the 5th Protocol Specification, Testing, and Verification, pp. 387-398, 1986.

[ISO 83] ISO IS 7498; "Information processing systems - open systems interconnection - basic reference model", 1983.

[ISO 86] ISO DIS 8571; "Information processing systems - open sistems interconnection - file transfer, access and management", 1986.

[ISO 88] ISO/IEC JTC1/SC21 N3067; "Specification styles for structuring of OSI Formal Descriptions", Information Retrieval, Transfer and Management for OSI, september, 19, 1988.

[ISO 88a] ISO IS 9074; "Information processing systems - open systems interconnection - ESTELLE - A formal description technique based on an extended state transition model", 1988.

[ISO 89] ISO IS 8807; "Information processing systems - open systems interconnection - LOTOS - Formal description technique based on the temporal ordering of observational behaviour", 1989.

[JeWi 75] Jensen, K. and Wirth , N.; "Pascal user manual and report", New York, Springer-Verlag, 1975.

[Just 88] Justo, G.R.R.; "Ambiente de programação distribuída com configuração dinâmica de processos", Tese de mestrado em informática, UFPE, 1988.

[KaHi 90] Kant, C.; Higashino, T. and Bochmann, G.v.; "Deriving protocol specifications from service specifications written in Basic LOTOS", 1990.

[Kell 76] Keller, R.M.; "Formal verification of parallel programs", Communications of the Association for Computing Machinery, V. 19, N. 7, pp. 371-384, 1976.

[KhBo 89] Khendek, F.; Bochmann, G.v. and Kant, C.; "New results on deriving protocol specifications from services specifications", publication 692, Université de Montréal, Canadá, 1989.

[Lamp 80] Lamport, L.; "Sometime' is sometimes 'Not Never': on the temporal logic programs". In: Proceedings of the 7th Annual ACM Symposium on Principles of Programming Languages, Las Vegas, pp. 174-185, 1980.

[Lang 90] Langerak, R.; "Decomposition of functionality: a correctness preserving LOTOS transformation". In: Proceedings of the 10th International Symposium on Protocol Specification, Testing, and Verification, Ottawa, pp. 203-218, 1990.

[LeMo 73] Le Moli, G.; "A theory of colloquies". In: Proceedings of the 1st European Workshop on Computer Networks, Arles, pp. 57-64, 1973.

[Li 86] Li, D.-H.; "Top-down and step-wise refinement of protocol specifications", Tese de doutorado, University of London, 1986.

[LiMa 88] Li, D.-H. e Maibaum, T.S.E.; "A top-down step-wise refinement methodology for protocol specification", Lecture Notes in Computer Science, V. 335, Springer-Verlag, pp. 197-221, 1988.

[Lini 83] Linington, P.F.; "Fundamentals of the layer service definitions and protocol specifications", Proceedings of the IEEE, V. 71, N. 12, pp. 1341-1345, 1983.

[LiNi 83] Linn Jr.; R. J. e Nightingale, J.S.; "Testing OSI protocols at the National Bureau of Standards", Proceedings of the IEEE, V. 71, N. 12, pp. 1431-1434, 1983.

[Linn 86] Linn Jr.; R.J.; "Testing to assure interworking of implementations of OSI/ISO protocols", Computer Networks and ISDN Systems, V. 11, pp. 277-286, 1986.

[Linn 86a] Linn Jr.; R.J.; "The features and facilities of Estelle". In: Proceedings of the 5th International Symposium on Protocol Specification, Testing, and Verification, pp. 271-296, 1986.

[LoNe 88] Lopes de Souza, E.; Neuman de Souza, J. e Celestino Jr.;J.; "Especificação formal de protocolos de comunicação através de refinamentos sucessivos". In: Anais do 8º Congresso da Sociedade Brasileira de Computação, Rio de Janeiro, pp. 113-124, 1988.

[Lope] Lopes de Souza, W.; "Introdução à especificação e validação de protocolos de comunicação", Trabalho realizado junto ao IRO/Université de Montréal.

[Lope 86] Lopes de Souza, W.; "Uma proposta para o desenvolvimento sistemático do software de comunicação da rede local baseada no processador preferencial", Relatório Técnico do ITEEL, Campina Grande, PB, 1986.

[Lope 87] Lopes de Souza, W.; "LOTOS: uma técnica para a descrição formal de serviço e protocolos de comunicação". In: Anais do 5º Simpósio Brasileiro de Redes de Computadores, São Paulo, pp. 121-144, 1987.

[Lope 88] Lopes de Souza, W.; "An approach for developing communication software for local area networks", Relatório Técnico 03/88, Universidade Federal da Paraíba, Campina Grande, 1988.

[Lope 89] Lopes de Souza, W.; "Uma metodologia, baseada em linguagens, ferramentas e ambientes, para o projeto de sistemas distribuídos", nonografia apresentada para o concurso de Professor Titular, em Computação, junto à Universidade Federal de Santa Catarina (UFSC), 1989.

[LoRi 88] Lopes de Souza, W.; Riso, B.G. e Monteiro Filho, A.F.; "Especificação e verificação em CCS de sistemas de comunicação". In: Anais do 6º Simpósio Brasileiro de Redes de Computadores, Belo Horizonte, 1988.

[LoRi 88a] Lopes de Souza, W. e Riso, B.G.; "Using CCS for protocol specification by step-wise refinements". In: Proceedings of the 2nd International Symposium on Interoperable Information Systems, ISIIS' 88, Tóquio, pp. 135-142, 1988.

[LoSt 88] Lopes de Souza, W. e Stiubiener, S.; "Especificação, verificação e testes de protocolos", Relatório Técnico Nº RT-01/88, GRC/UFPB, 1988.

[McAb 65] McCarthy, J.; Abrahams, P.W.; Edwards, D.J.; Hart, T.P. and Levin, M.I.; "LISP 1-5 Programmer's Manual", Cambridge, The MIT Press, 1965.

[MeBo 83] Merlin, P. and Bochmann, G.v.; "On the construction of submodule specifications and communication protocols". In: ACM Transactions on Programming Languages and Systems, New York, V. 5, N. 1, pp. 1-25, 1983.

[Merl 76] Merlin, P.M.; "A methodology for the design and implementation of communication protocols", IEEE Transactions on Communications, V. 24, N. 6, pp. 614-621, 1976.

[Merl 79] Merlin, P.M.; "Specification and verification of protocols", IEEE Transactions on Communications, V. 17, N. 11, pp. 1671-1680, 1979.

[Miln 80] Milner, A.J.R.G.; "A calculus of communicating systems", Lecture Notes in Computer Science, V. 92, Springer-Verlag, 1980.

[Miln 86] Milner, A.J.R.G.; "Process constructors and interpretations". In: Kugler, H.-J. (ed.). Proceedings of the information processing 86. Amsterdam: North-Holland, pp. 507-514, 1986.

[Miln 89] Milner, A.J.R.G.; "Communication and concurrency", Prentice-Hall International (UK) Ltda.; 1989.

[MoSa 86] Moura, J.A.B.; Sauvé, J.P. Giozza, W.F. e Marinho de Araújo, J.F.; "Redes locais de computadores - protocolos de alto nível e avaliação de desempenho", McGraw-Hill, 1986.

[Nico 87] Nicola, R. de, "Extensional equivalences for transition systems", Acta Informatica, V. 24, pp. 211-237, 1987.

[NoEv 88] Norris, M.T.; Everett, R.P.; Martin, G.A.R.; Shields, M.W.; "Method for the synthesis of interactive system specifications", Information and Software Technology, Vol. 30, N. 7, pp. 438-442, 1988.

[NoNe 87] Norris, M.T.; Newman, J.P. e James, P.; "A step-by-step guide to using formal methods", British Telecommunications Engineering, V. 5, pp. 308-311, 1987.

[NoYe 85] Nounou, N. and Yemini, Y.; "Algebraic specification-based performance analysis of communication protocols". In: Proceedings of the 4th Protocol Specification, Testing, and Verification, pp. 541-560, 1985.

[ObLo 87] Obaid, A. and Logrippo, L.; "An atomic calculus of communicating systems". In: Proceeding of the 7th Protocol Specification, Testing and Verification, Zurich, 1987.

[ONnLi 89] O'Neal, M.R.; Lively, W. e Sheppard, S.; "Software function allocation methodology", Software-Practice and Experience, V. 19, N. 8, pp. 775-786, 1989.

[PaGu 85] Parrow, J. and Gustavsson, R.; "Modelling distributed systems in a extension of CCS with infinite experiments and temporal logic". In: Proceedings of the 4th Protocol Specification, Testing, and Verification, pp. 309-348, 1985.

[Park 81] Park, D.; "Concurrency and automata on infinite sequences", Lecture Notes in Computer Science, Vol. 104, Goos, G. and Hartmanis, J. (ed.), pp. 167-183, 1981.

[Parr 89] Parrow, J.; "Submodule construction as equation solving in CCS", Theoretical Computer Science, Vol. 68, North-Holland, pp. 175-202, 1989.

[PiLo 90] Pires, L.F. and Lopes de Souza, W.; "Step-wise refinement design example using LOTOS". In: Proceedings of the 3rd International Conference on Formal Description Techniques (FORTE'90), Madrid, 1990.

[Plet 86] Pletat, U.; "Algebraic specifications of abstract data types and CCS: an operational junction". In: Proceedings of the 6th International Workshop on Protocol Specification, Testing, and Verification, pp. 10-13/10-24, 1986.

[Rayn 82] Rayner, D.; "A system for testing protocol implementations". In: Proceedings of the 2nd International Symposium on Protocol Specification, Testing, and Verification, pp. 539-553, 1982.

[Rayn 86] Rayner, D.; "Progress on standardizing OSI conformance testing", Computer Standards and Interfaces, N. 5, pp. 317-334, 1986.

[RiLo 90] Riso, B.G. e Lopes de Souza, W.; "Especificação algébrica de processos da camada de aplicação do RM OSI/ISO". In: Anais do 8º Simpósio Brasileiro de Redes de Computadores, 1990.

[RiLo 91] Riso, B.G. e Lopes de Souza, W.; "Uma abordagem para a especificação estruturada, em CCS, de sistemas distribuídos", aceito para publicação na Revista Brasileira de Computação.

[Riso 89] Riso, B.G.; "Uma metodologia para a especificação 'top-down' de protocolos", seminário apresentado na UFPB, setembro 1989.

[Riso 90a] Riso, B.G. e Lopes de Souza, W.; "Uma abordagem para a especificação estruturada, em CCS, de sistemas distribuídos", monografia, UFPB, 1990.

[Riso 89a] Riso, B.G.; "Especificação algébrica de processos computacionais", seminário apresentado na UFPB, dezembro 1989.

[Riso 89b] Riso, B.G.; "Utilização da metodologia m para a especificação estruturada de um protocolo de comunicação", monografia, UFPB, 1989.

[RoSa 82] Rockstrom, A. and Saracco, R.; "SDL-CCITT specification and description language", IEEE Transactions on Communications, V. COM-30, N. 6, pp. 1310-1318, 1982.

[SaBo 82] Sarikaya, B. e Bochmann, G.v.; "Some experience with test sequence generation for protocols". In: Proceedings of the 2nd International Symposium on Protocol Specification, Testing, and Verification, pp. 555-567, 1982.

[SaBo 84] Sarikaya, B. e Bochmann, G.v.; "Synchronization and specification issues in protocol testing", IEEE Transactions on Communications, V. COM-32, N. 4, pp. 389-395, 1984.

[SaBo 87] Sarikaya, B.; Bochmann, G.v. e Cerny, E.; "A test design methodology for protocol testing", IEEE Transactions on Software Engineering, V. SE-13, N. 5, pp. 518-531, 1987.

[Sari 87] Sarikaya, B.; "Recent developments in protocol testing". In: Anais do 5º Simpósio Brasileiro de Redes de Computadores, pp. 372-392, 1987.

[SaTi 87] Saracco, R. and Tilanus, P.A.J.; "CCITT SDL: overview of the language and its applications", Computer Networks and ISDN Systems, Vol. 13, pp. 65-74, 1987.

[Schi 80] Schindler, S.; "Algebraic and model specification techniques". In: Peoceedings of the 13th Hawaii International Conference on System Sciences, pp. 103-111, 1980.

[ScMe 81] Schwartz, R.L. and Melliar-Smith, P.M.; "Temporal logic specification of distributed systems". In: Proceedings of the 2nd International Conference on Distributed Systems, INRIA, Paris, pp. 446-454, 1981.

[ScRo 80] Schultz, G.D.; Rose, D.B.; West, C.H. and Gray, J.P.; "Executable description and validation of SNA", IEEE Transactions on Communications, Vol. COM-28, N. 4, pp. 661-677, 1980.

[Shie 87] Shields, M.W.; "An introduction to automata theory", Blackwell Scientific Publications. Oxford, 1987.

[Shie 89] Shields, M.W.; "Implicit system specification and the interface equation", The Computer Journal, V. 32, N. 5, pp. 399-412, 1989.

[Soar 86] Soares, L.F.G.; "Redes locais", Editora Campus, Rio de Janeiro, 1986.

[SPBr 87] SPECS Consortium e Bruijning, J.; "Evaluation and Integration of specification languages", Computer Networks and ISDN Systems, V. 13, pp. 75-89, 1987.

[Sten 76] Stenning, N.V.; "A data transfer protocol", Computer Networks and ISDN Systems, Vol. 1, N. 2, pp. 99-110, 1976.

[StGo 89] Stroup, T.; Gotz, N. e Mendler, M.; "Stepwise refinement of layered protocols by formal program development". In: Proceedings of the 9th International Symposium on Protocol Specification, Testing, and Verification, 1989.

[SuDa 78] Sunshine, C.A. e Dalal, Y.K.; "Connection management in transport protocols", Computer Networks, V. 2, pp. 454-473, 1978.

[Suns 78] Sunshine, C.A.; "Survey of protocol definition and verification techniques", Computer Networks, V. 2, N. 4/5, pp. 346-350, 1978.

[Suns 82] Sunshine, C.A.; "Protocol specification, testing, and verification", IEEE Transactions on Communications, V. COM-30, N. 12, p. 2485, 1982.

[Tane 89] Tanenbaum, A.S.; "Computer networks", 2nd edition, Prentice-Hall International, Inc.; 1989.

[Taro 86] Tarouco, L.M.R.; "Redes de computadores locais e de longa distância", McGraw-Hill, 1986.

[TeLi 78] Teng, A.Y. and Liu, M.T.; "A formal model for automatic implementation and logical validation of network communication protocol". In: Proceedings of the Computer Network Sumposium, NBS, pp. 1271-1275, 1978.

[USDD 80] United States Department of Defense; "Reference manual for the Ada programming language", Proposed Standard Document, 1980.

[ViLo 86] Vissers, C.A. and Logrippo, L.; "The importance of the service concept in the design of data communications protocols". In: Proceedings of the 5th Protocol Specification, Testing, and Verification, pp. 3-17, 1986.

[ViSc 88] Vissers, C.A.; Scollo, G. and Sinderen, M.v.; "Architecture and specification style in formal descriptions of distributed systems". In: Proceedings of Protocol Specification, Testing, and Verification VIII, S. Aggarwal (ed.), North-Holland, Amsterdam, 1988.

[ViSc 89] Vissers, C.A.; Scollo, G.; Sinderen, M.v. and Brinksma, E.; "On the use of specification styles in the design of distributed systems". In: Proceedings of TAPSOFT' 89, Barcelona, 1989.

[ViTe 83] Vissers, C.A.; Tenney, R.L. e Bochmann, G.v.; "Formal description techniques", Proceedings of the IEEE, V. 71, N. 12, pp. 1356-1364, 1983.

[Webe 85] Weber, K.C.; "A padronização de protocolos como um dos instrumentos da política nacional de informática". In: Anais do 3º SimpÓsio Brasileiro de Redes de Computadores, pp. II.1 - II.9, 1985.

[West 78] West, C.H.; "An automated technique of communications protocol validation", IEEE Transactions on Communications, Vol. 26, N. 8, pp. 1271-1275, 1978.

[Zafi 78] Zafiropulo, P.; "Protocol validation by duologue matrix analysis", IEEE Transactions on Communications, Vol. 26, N. 8, pp. 1187-1194, 1978.

[ZaWe 80] Zafiropulo, P.; West, C.H.; Rudin.; H.; Cowan, D.D. and Brand, D.; "Towards analyzing and synthesizing protocols", IEEE Transactions on Communications, Vol. COM-28, N. 4, pp. 651-661, 1980.