SOBRE O PROBLEMA DE MODELAGEM, USANDO GRAFOS DO TIPO PREDICADOS/TRANSIÇÕES, DE CONTROLE DE FLUXO EM REDES DE COMPUTADORES

Gerhard Schwarz

TESE SUBMETIDA AO CORPO DOCENTE DA COORDENAÇÃO DOS PROGRAMAS DE PÓS-GRADUAÇÃO DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS (D.Sc.) EM ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Aprovada por:

Prof.ª Sueli Mendes dos Santos
(Presidente)

Prof. Daniel A. Menascé

Dr. Pierre Jean Lavelle

Prof. João Lizardo R.H. de Araújo

Prof. Félix A.C. Mora-Camino

RIO DE JANEIRO, RJ - BRASIL

JULHO DE 1984

SCHWARZ, GERHARD

Sobre o Problema de Modelagem, Usando Grafos do Tipo Predicados/Transições, de Controle de Fluxo em Redes de Computadores (Rio de Janeiro) 1984.

XXIV,330p. ilustr. 29,7 cm (COPPE-UFRJ, D.Sc., Engenharia de Sistemas e Computação, 1984)

Bibliografia

Tese - Universidade Federal do Rio de Janeiro, Faculdade de Engenharia, COPPE.

1. Redes de Computadores. 2. Controle de Fluxo. 3. Modelagem. 4. Grafos tipo Predicados/Transições. I. COPPE/UFRJ II. Título (série).

BIOGRAFIA

Nome, data e local de nascimento:

Gerhard Schwarz, 23 de abril de 1941, Viena (Áustria)

Títulos acadêmicos (data; instituição (local) e departamento; grau obtido):

- 27/06/1961; Technologisches Gewerbemuseum (Viena, Áustria), Technische Bundeslehr- und Versuchsanstalt, Hoehere Abteilung fuer Starkstromtechnik; Engenheiro (ING.).
- 26/04/1974; Universidade Federal (Rio de Janeiro, Brasil), COPPE (Engenharia de Sistemas e Computação); Mestre em Ciências (M.Sc.).

Experiência profissional:

- Estágios (durante o estudo de engenharia) em várias firmas, fazendo trabalhos ligados à manutenção de equipamentos elétricos na construção civil (Rella & Co., Viena-Áustria), à mecânica (Alpine Montan, Donawitz-Áustria), ao laboratório de fábrica de comutadores (Kraus & Naimer, Viena-Áustria) e à operação de filmes, formato reduzido, num estabelecimento de formação popular (Volkshochschule Wiener Urania, Viena-Áustria).
- Siemens A.G., Departamento 'Energieversorgung' (abastecimento de energia elétrica), de 1961 até 1964, em Erlangen, Alemanha Ocidental. Trabalhando como engenheiro na concepção (projeto inicial, fabricação, ensaios finais, exploração comercial) das centrais elétricas industriais (principalmente as partes relacionadas ao comando, controle e medição) usando tecnologia analógica e digital para realizar todos os tipos de tele-operação.
- Siemens A.G., Departamento 'Entwicklung' (densenvolvimento do equipamento para turbinas a vapor), de 1964 até 1968, em Erlangen, Alemanha Ocidental. Trabalhando como pesquisador, junto à fábrica de turbinas, no desenvolvimento de um sistema de controle elétro-hidráulico e um sistema automatizado de teste do equipamento de segurança para turbinas. O trabalho consistiu em projetar, realizar, testar, documentar e iniciar a produção (em série) destes sistemas, seguido por um treinamento dos clientes potenciais destes equipamentos.
- CERN (Organização européia para pesquisa nuclear), departamento 'Proton Synchrotron', Projeto Telemanipulador, 1969-1972, em Genebra, Suiça. Este projeto, feito por uma equipe de três engenheiros (dois formados em mecânica e um em elétrica), consistiu na realização de todo equipamento de suporte (pontes, guindastes, telescópios) e de controle (alimentação, cabos, luzes, televisão, comutação automática) para um telemanipulador, baseado em servo-sistemas, previsto para a intervenção nos anéis de aceleração de prótons e nas áreas expostas à radiação.

- COPPE/UFRJ (Coordenação dos Programas de Pós-Graduação de Engenharia/Universidade Federal do Rio de Janeiro), Programa de Engenharia de Sistemas e Computação, desde 1972, no Rio de Janeiro, Brasil. A principal função foi, inicialmente, a chefia de um laboratório de automação e simulação de sistemas (1972-1978); as atividades ligadas a esta função foram o uso, extensão e manutenção de equipamentos existentes (computador analógico, minicomputador, periféricos, software), completada por aulas práticas em relação à computação analógica e simulação, assim como apoio para experiências e pesquisas.
- Desde fim 1974, depois a obtenção de M.Sc., sendo responsável pela Linha de Pesquisa "Computadores em Tempo Real", posteriormente (a partir de 1980) transformada em "Telemática", as atividades tinham se voltado para o ensino (graduação e pós-graduação) e pesquisa nas mencionadas áreas de Tempo Real e Telemática.
- Os interesses atuais de pesquisa estão orientados a Redes de Computadores, dando atenção especial à modelagem dos problemas dinâmicos (fluxo, acesso, desempenho) nestes sistemas distribuídos.

Publicações:

- Várias monografias didáticas relacionadas a diversos tópicos ligados a redes de computadores, tais como a descrição qualitativa, modelagem, teorias usadas, programação matemática, simulação, controle de fluxo e controle de acesso.
- Um livro-texto, usado na graduação do Instituto de Matemática, sobre os conceitos básicos de Computadores em Tempo Real.
- Vários relatórios técnicos sobre os problemas de modelagem dos problemas dinâmicos em redes de computadores.

AGRADECIMENTOS

À FINEP (Financiadora de Estudos e Projetos) e à UFRJ (Universidade Federal do Rio de Janeiro) ...

... pelo financiamento parcial desta pesquisa;

à COPPE (Coordenação dos Programas de Pós-Graduação de Engenharia) e, em especial, ao Programa de Sistemas e Computação ...

... pela liberação parcial de minhas tarefas profissionais para poder realizar esta pesquisa;

aos membros da Banca de Exame de Qualificação (P.J. Lavelle, J.L.R.H. de Araújo, N. Maculan Filho) e membros da Banca de Defesa de Tese de D.Sc. (S.M. dos Santos, D.A. Menascé, P.J. Lavelle, J.L.R.H. de Araújo, F.A.C. Mora-Camino) ...

... pela participação nestas Bancas;

à Prof.ª Sueli Mendes dos Santos (COPPE/UFRJ), Prof. João Lizardo R.H. de Araújo (COPPE/UFRJ) e Dr. Gernot Richter (GMD - Gesellschaft für Mathematik und Datenverarbeitung, Bonn, Alemanha Ocidental) ...

... pelas discussões construtivas e pelo apoio mostrado;

à Lilian Vicentini ...

... por dar o aspecto final da parte datilográfica;

às pessoas de minha roda de amizade ...

... por acreditarem em mim e por suportar meu mal humor, sobretudo na fase final da tese;

à minha família ...

... pelo estímulo e por confiar, sem restrições, nas minhas possibilidades.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

SOBRE O PROBLEMA DE MODELAGEM, USANDO GRAFOS DO TIPO PREDICADOS/TRANSIÇÕES, DE CONTROLE DE FLUXO EM REDES DE COMPUTADORES

Gerhard Schwarz

Julho, 1984

Orientador: Prof.ª Sueli Mendes dos Santos
Programa: Engenharia de Sistemas e Computação

Este estudo se divide em quatro partes. Na primeira, apresentam-se, sucintamente, os métodos de controle de fluxo em redes de computadores que queremos modelar. Na segunda parte, explica-se, formalmente e através de exemplos simples, a ferramenta principal de nossa modelagem, ou seja, o grafo do tipo predicados/transições ('PrT-Nets') que é, basicamente, um Petri-Net de primeira ordem.

Em seguida, a terceira e principal parte mostra como usar os 'PrT-Nets' para modelar o controle de fluxo. Foi empregado um procedimento progressivo, passo a passo, que, no primeiro passo, resultou num modelo que, basicamente, representa o controle 'end-to-end', ignorando, por enquanto, os vértices intermediários. Este modelo inicial serviu, depois, para incluir o problema mais importante, ou seja, a consideração destes vértices. Para obter isto, tivemos que "abrir" os vértices para poder tratar os diferentes canais de entrada e saída de uma maneira quase independente. Mostramos como se pode, usando a técnica de 'PrT-Nets', modelar filas em relação à administração do espaço disponível nas mesmas.

A inclusão, no modelo, dos problemas ACK ('acknowledgement') e NAK ('negative ACK'), trazia, como conseqüência direta, a modelagem de controle de fluxo em canais entre vértices vizinhos o que inclui, por exemplo, a questão de retransmissões de mensagens. Como estas também podem ocorrer por causa de um 'time-out', achamos por bem de incluir, no modelo, também os aspectos de tempos envolvidos se restringindo, porém, ao 'time-out', no nível entre vértices vizinhos, e ao 'check-time', no nível 'end-to-end'. O modelo final desta tese foi obtido com a consideração de mensagens do tipo multipacotes.

Finalmente, na parte conclusiva, indicou-se possíveis melhoramentos do modelo como, por exemplo, através da inclusão dos problemas de mensagens de controle, de ferramentas específicas (adaptação do crédito, alocação dinâmica do espaço nos 'buffers', roteamento distribuído e adaptativo), acesso em enlaces via satélite, etc.. Também foi chamada a atenção sobre a necessidade de sistemas computacionais para facilitar a modelagem em si e, como ponto mais importante, a investigação da dinâmica do sistema representado pelo modelo.

Abstract of Thesis presented to COPPE/UFRJ as partial fulfillment of the requirements for de degree of Doctor of Science (D.Sc.)

ABOUT THE PROBLEM OF MODELLING THE FLOW CONTROL IN COMPUTER NETWORKS BY USING PREDICATE/TRANSITION-NETS.

Gerhard Schwarz

July 1984

Chairwoman: Prof.ª Sueli Mendes dos Santos
Department: Systems Engineering and Computer Science

This study is divided into four parts. The first one briefly exposes those flow control methods used in computer networks which are going to be modelled. In the second part will be explained, formally and also using simply examples, the principal tool of our modelling: predicate/transition-nets (PrT-Nets) which are, basically, first order Petri-Nets.

Following this, the third and principal part shows how to use PrT-Nets for modelling the flow control. Using a progressive step-by-step proceeding, we got after the first step a model which basically represents an end-to-end control ignoring, however, the intermediate nodes. This first model served, afterwards, as a basis to include the most important problem, what means, the consideration of these intermediate nodes. To get this result, we had to "open" these nodes for beeing able to treat the different input- and outputchannels in a nearly independent way. We showed, using PrT-Nets, how one could model queues for what concernes the administration of their usable buffer space.

The inclusion of the problem concerning ACK (acknowledgement) and NAK (negative ACK) in the model brought, as a direct consequence, the modelling of flow control in channels between neighboring nodes which also includes, e.g., the question of retransmission of a message. Additional, as these retransmissions also may be caused by time-outs, we thought it reasonable to include in our model also the aspects of times envolved but limiting ourselves to the time-out, on an intervertice level, and the check-time, on an end-to-end level. The final model of this thesis was obtained by considering also multipacket--messages.

Finally, in the conclusive part, we indicated possible improvements of the model by considering, e.g., the different control messages, specific tools (adaptation of credit, dynamic allocation of buffers space, distributive and adaptative routing), access in satellite links, etc.. Also was made the alert about the necessity of computational systems to facilitate the modelling itself and, as the most important point, the investigation of dynamics of the modelled system.

Résumé de la thèse soutene à la COPPE/UFRJ comme exigence partielle pour l'obtention du grade de Docteur en Sciences (D.Sc.)

SUR LE PROBLÈME DE MODELAGE PAR LES RÉSEAUX DE TYPE PRÉDICATS/ /TRANSITIONS DU CONTRÔLE DE FLUX DANS LES RÉSEAUX D'ORDINATEURS

Gerhard Schwarz

Juillet 1984

President: Prof. Sueli Mendes dos Santos
Département: Génie des Systèmes et Informatique

Cette étude est divisée en quatre parties. La première expose brièvement les méthodes de contrôle de flux dans les réseaux d'ordinateurs que l'on veut modeler. Dans la deuxième partie on explique, formellement et par des exemples simples, l'outil principal de notre modelage: les réseaux de type prédicats/ /transitions ('PrT-Nets') que sont en fait des réseaux de Pétri de premier ordre.

La troisième partie, partie principal du travail, montre comment on peut utiliser les 'PrT-Nets' pour le modelage du contrôle de flux. Par l'utilisation d'un procédé progressif, mis en oeuvre pas a pas, nous avons réalisé tout d'abord un modèle qui représente simplement le contrôle 'end-to-end', mais qui ignore les noeuds intermédiaires. Ce premier modèle a constitué une base pour l'étude du problème principal concernant la considération des noeuds intermédiaires. Pour obtenir ce résultat, il est nécessaire "d'ouvrir" ces noeuds afin de traiter les différents canaux d'entrée et de sortie d'une manière pratiquement indépendente. Nous avons montré, en utilisant les 'PrT-Nets', comment on peut modeler les files d'attente en relation à l'administration de l'espace utile dans les 'buffers'.

L'incorporation du problème concernant ACK ('acknowledgement') et NAK ('negative ACK') dans le modèle a provoqué, comme une conséquences directe, le modelage du contrôle de flux dans les canaux entre des noeuds voisins, cette question incluant aussi le problème de la retransmission de messages. Additionellement, du fait que ces retransmissions peuvent aussi être causées par un 'time-out', le modèle considére aussi quelques uns des principaux aspects temporels du problème, à savoir: 'time-out' pour le niveau entre deux noeuds voisins, et 'check-time' pour le niveau 'end-to-end'. Le modèle final de cette thèse a été obtenue en considérant aussi les messages de type multipaquets.

Finalement, dans la conclusion, nous avons indiqué les améliorations possibles du modèle en considérant, par exemple, les différents messages de contrôle, les outils specifiques (l'adaptation de "crédit", l'allocation dynamique de l'espace dans les 'buffers', le routage distribué et adaptatif), l'accès aux liens de satellites, etc.. On a attiré l'attention sur la nécessité des systèmes d'ordinateurs avec le but de faciliter le modelage em lui même et aussi, comme objectif final, l'étude de la dynamique du système modelé.

Zusammenfassung der an der COPPE/UFRJ vorgelegten These als teilweise Erfüllung der Forderungen zur Erlangung des Doktorgrades in Wissenschaften (D.Sc.)

ÜBER DAS PROBLEM DER MODELLIERUNG DER FLUSSKONTROLLE IN RECHNERNETZWERKEN UNTER DER VERWENDUNG VON PRÄDIKAT/TRANSITIONS-NETZEN

Gerhard Schwarz

Juli 1984

Vorsitzende: Prof. Sueli Mendes dos Santos
Abeilung: System- und Informatikwissenschaften

Dieses Studium ist in vier Abschnitte aufgeteilt. Der erste Abschnitt illustriert in Kurzform jene Flußkontrollmethoden in Rechnernetzen die modelliert werden sollen. Im zweiten Teil wird auf formelle Art und unter Verwendung von einfachen Beispielen das Werkzeug unserer Modellierung erklärt: Es sind dies die Prädikat/Transitions-Netze (PrT-Netze) welche als Petri-Netze ersten Grades angesehen werden können.

Der dritte und hauptsächlichste Teil zeigt wie man die PrT-Netze für die Modellierung der Flußkontrolle verwenden kann. Unter Verwendung eines progressiven Vorgehens, Schritt für Schritt verwirklicht, erhielten wir am Anfang ein Modell das die 'end-to-end'-Kontrolle repräsentiert ohne aber die zwischenliegenden Knotenpunkte des Rechnernetzes zu berücksichtigen. Dieses erste Modell diente anschließend als Grundlage um das wichtigste Problem, das heißt, die Berücksichtigung dieser Knotenpunkte, studieren zu können. Um dieses Ergebnis zu erhalten, hatten wir diese Knotenpunkte zu "öffnen" um in der Lage zu sein die verschiedenen Ein- und Ausgangskanäle in einer annähernd unabhängigen Art zu behandeln. Wir zeigten mit Hilfe der PrT-Netze wie man Warteschlangen in Bezug auf die Verwaltung der zur Verfügung stehenden Speicherplätze modellieren kann.

Die Berücksichtigung des Problems der ACK (Empfangsbestätigung) und NAK (negative ACK) machte, als eine direkte Folge, die Modellierung der Flußkontrolle in Kanälen zwischen benachbarten Knotenpunkten nötig was auch, zum Beispiel, die Frage der Wiederholung der Übertragung einer Nachricht aufwirft. Zusätzlich, aufgrund der Möglichkeit daß diese Wiederholung auch von 'time-outs' verursacht werden kann, entschlossen wir uns auch diese zeitlichen Aspekte in das Modell mit einzubeziehen; wir beschränkten uns jedoch auf das 'time-out' auf dem Niveau zwischen den Knotenpunkten, und auf die 'check-time' in Bezug auf das Niveau der Endpunkte. Das endgültige Modell dieser These wurde durch die Berücksichtigung von Nachrichten die aus mehreren Paketen zusammengesetzt sind erlangt.

Im abschließenden Teil zeigten wir die Möglichkeit von Verbesserungen des Modelles auf; das betrifft, zum Beispiel, die verschiedenen Kontrollnachrichten, besondere Werkzeuge (Kreditanpassung, dynamische Verteilung der Speicherplätze, distributive und anpassungsfähige Kanalauswahl), Zugriff auf Satellitenkanäle, etc.. Wir machten auch auf die Notwendigkeit der Verwendung von Rechnersystemen aufmerksam um, einerseits, die Modellierung an sich zu erleichtern und, andererseits, als wichtigstes Ziel, die Möglichkeit einer Untersuchung der Dynamik des modellierten Systemes zu verwirklichen.

ÍNDICE

ÍNDICE DAS FIGURAS

C A P Í T U L O I

Introdução

Muito se tem falado, ultimamente, sobre redes de computadores, satélites domésticos, teleprocessamento, controle de processos distribuídos, etc. Podemos juntar todos estes termos numa só palavra, a saber, em TELEMÁTICA. Assim, esta expressão representa, em princípio, o conjunto de duas tecnologias: uma que é ligada à informática, e outra é aquela da comunicação, considerando, por enquanto, somente a comunicação no nível técnico.

Agora, refletindo um pouco mais sobre o significado da telemática, chegamos à conclusão que ela nada mais é que uma ferramenta que permite que sistemas, em geral, funcionem de uma maneira satisfatória. Não importa, se estes sistemas são tecnológicos, comerciais, educacionais, etc., o fator em comum é que sistemas, hoje em dia, quase sempre são grandes e complexos (no sentido distribuído, hierárquico e adaptativo) e que os seus componentes precisam se comunicar uns com os outros.

Teoricamente, precisaríamos especificar, agora, o que seria um componente de um sistema. Entretanto, como isto iria longe de mais, podemos nos contentar em afirmar que componentes de sistemas complexos (p.ex., subprocessos, agências bancárias, centros educacionais locais, terminais domésticos, etc.) têm que ter, quase obrigatoriamente, como base, algum sistema computacional.

Deste modo, chegamos à conclusão que, baseadas no fato que os componentes destes sistemas complexos têm que ter comunicação entre si, redes de computadores são, de fato, ferramentas indispensáveis para garantir o funcionamento satisfatório

em conjunto, ou seja, de sistema como um todo.

Sabendo agora para que servem REDES DE COMPUTADORES, ou seja, que elas têm que ser consideradas, basicamente, como sendo nada mais que utensílios em função de sistemas globais, podemos nos preocupar em aperfeiçoar, ao máximo possível, este ferramental. Conseqüentemente, uma vez que temos este sistema de suporte em condições perfeitas de funcionamento, podemos, eventualmente, chegar perto de um funcionamento ótimo do sistema em questão.

Como conseguir que este suporte, ou seja, as redes de computadores, funcionem com desempenho satisfatório, eis a questão crucial.

Em resposta a isto, podemos dizer que, mesmo querendo, não conseguiremos resolver esta questão como um todo (isto seria exagero demais de nossa parte) mas, ao menos, pretendemos contribuir, modestamente, com este trabalho.

Entretanto, antes de começar com este trabalho propriamente dito, tinhamos que estudar as várias idéias associadas a redes de computadores. Assim, depois de um estudo inicial, SCHWARZ (26), verificamos a utilidade de algumas teorias, notadamente a teoria de grafos e a de filas, SCHWARZ (27), para fins de modelagem. Algumas destas técnicas servem para criar arquiteturas adequadas, como foi estudado, por exemplo, por PALMA (19), outras servem para garantir o funcionamento em si, como, por exemplo, no caso em que se investiga as regras de comunicação, os chamados protocolos, como foi feito, por exemplo, no estudo de MUSCHELLACK (18).

Uma vez tendo conseguido um modelo da rede, ou de alguma parte física ou lógica dela, baseado em estudos mais qualitativos que quantitativos, poderíamos aplicar outras ferramentas para obter, agora sim, avaliações quantitativas. Entram aí, como foi resumido por SCHWARZ (28), técnicas de programação matemática e, também, métodos de simulação já que nem sempre é

possível fazer investigações quantitativas de uma maneira direta.

Resumindo vale dizer que, somente depois que a rede de computadores, usando as teorias e ferramentas mencionadas, foi concebida, podemos pensar na avaliação do desempenho do ponto de vista da dinâmica do sistema.

Lançada esta palavra chave, DINÂMICA DO SISTEMA, vimos-nos na obrigação de classificar as redes de computadores. Isto se deve, principalmente, ao fato de que no momento em que começamos a nos preocupar com a dinâmica de um sistema, introduzimos, ao lado da dimensão ESPAÇO (responsável pela distribuição geográfica ou física), uma segunda dimensão no problema que é a do TEMPO.

Estas duas dimensões nos forçaram, como já dissemos, a classificar as redes em aquelas com linhas de transmissão terrestres e outras que estão, por sua vez, baseadas em sistemas de rádio-difusão ou que possuem enlaces via satélites. Os estudos iniciais, feitos por SCHWARZ (30,31), justificaram esta classificação, reforçada ainda mais pelas filosofias bem distintas destas duas classes. Concluímos, porém, que, num futuro próximo, dever-se-ia juntar estas duas classes, o que resultaria, como foi sugerido por SCHWARZ (32), em redes MHT, ou seja, redes com meios heterogêneos de transmissão (chamadas de "mixed mediums", na literatura especializada).

Voltando ao nosso problema da dinâmica do sistema, vimos, durante todos estes estudos iniciais, que se deve, para alcançar um desempenho satisfatório, usar algum tipo de controle que permitiria corrigir os ERROS feitos na concepção da rede. Não se pode negar a existência destes erros, já que as informações usadas para realizar a rede são dados estatísticos, ou seja, médias, estimativas, extrapolações, etc. Mesmo admitindo, com reserva, que fossem dados corretos, o comportamento humano (que é, afinal representado pelo usuário das facilidades oferecidas) quase nunca pode ser previsto com segurança, sobretudo em casos como neste, quando, de repente,

alguma facilidade nova (a saber, as redes de computadores) lhe foi oferecida.

Estando obrigado a conviver com estes erros, precisa-se, assim, encontrar ferramentas para compensar ou, ao menos, tentar compensar, esta imprecisão da rede. Não querendo, entretanto, fazer especulações sobre a natureza destes erros (como, p.ex., capacidades das linhas insuficientes, tempos de processamento subestimados, etc.) podemos nos concentrar, desde já, em avaliar os efeitos (negativos, por sinal) destas deficiências na rede.

Estes efeitos se refletem, principalmente, nos dois parâmetros mais importantes para avaliar o desempenho de uma rede, ou seja, na vazão e no atraso em relação aos elementos que circulam na rede. Conseqüentemente, como estes dois parâmetros fazem parte do conceito FLUXO, e como os elementos que circulam em redes de computadores são INFORMAÇÕES (representadas por MENSAGENS), chegamos à conclusão que precisaríamos, para alcançar um desempenho satisfatório, um CONTROLE DE FLUXO (de informações).

O que é, então, este controle de fluxo? Como já foi relatado por SCHWARZ (30), existem, na literatura especializada, muitas definições sobre controle em si, sobre fluxo, etc. Usaremos, neste trabalho, o consenso das definições lançadas, entre outros, por KLEINROCK (15), POUZIN (22) e INOSE e SAITO (11) que diz, sobre o controle de fluxo, que é o conjunto de mecanismos — observando os limites disponíveis — que associa, de uma maneira eficiente, os recursos da rede às demandas dos usuários.

Tendo esta definição básica, podemos adaptá-la, sem grandes dificuldades, para fins de controle local ou global, ou, ainda, se incluíssemos também influências sobre os usuários, para controlar congestionamentos e acessos. De todas as maneiras, chegamos à conclusão que se precisa, basicamente, três ferramentas para permitir alguma eficiência do controle. Ter-se-ia, então, para usar as diferentes rotas ou caminhos à disposição, o

ROTEAMENTO; para usar eficientemente o espaço disponível nos computadores que compõem a rede, a ALOCAÇÃO DOS BUFFERS; e para evitar congestionamento e rejeições locais, a ADAPTAÇÃO de algum tipo de CRÉDITO. Incluindo ainda rádio-difusão ou enlaces via satélites, temos que usar, para obter acesso sem colisão, algum tipo de CONTROLE DE ACESSO.

Estudos qualitativos, feitos por SCHWARZ (30,31), serviram, entre outros, para conhecer os muitos métodos existentes e, também, para se ter uma idéia mais precisa sobre as vantagens e desvantagens de cada um deles. O que foi encontrado como sendo um dos fatores comum em todos estes métodos, é que eles usam, aparentemente, bastante eficientemente as técnicas disponíveis (como, p.ex., a teoria de filas, a de grafos, a programação matemática, etc.), mas que negligenciam, de um modo ou outro, as DECISÕES a serem feitas a cada momento. Não é suficiente, por exemplo, em nossa opinião, investigar, através da modelagem, o fluxo usando somente os diferentes parâmetros que representam as taxas de entrada, de serviço, etc., sem levar em consideração que, eventualmente, existem dezenas de usuários que competem, ao mesmo tempo, pelo mesmo recurso.

Precisa-se, então, uma ferramenta que permita modelar todos estes conflitos, concorrências, decisões, etc., fornecendo, assim, um complemento indispensável para se, primeiramente, estudar e modelar um sistema, e, depois, realizar a verdadeira rede. Conseqüentemente, pode-se, disto temos quase certeza, diminuir em muito a parcela que resolve estes problemas de uma maneira, digamos, empírica, a saber, na base de tentar e corrigir, chamada, muitas vezes, para encobrir este fato de tentativas, de "ganhar experiência".

Depois de algumas investigações, encontramos, na GNT ('general net theory'), uma ferramenta, chamada de técnica de 'PrT-Net' ('PREDICATE/TRANSITION-NET'; um tipo de Petri-Net de ordem avançada), que parecia preencher esta lacuna.

O trabalho presente se propõe, então, apresentar uma primeira tentativa de usar esta técnica de 'PrT-Net' para mo-

delar os problemas de controle de fluxo em redes de computadores do ponto de vista dos conflitos, concorrências e decisões a serem tomadas.

A estrutura deste trabalho reflete, bastante fielmente, a ordem cronológica de nossos estudos. Assim, o CAPÍTULO I está esclarecendo, em poucas palavras, tudo o que tínhamos estudado e investigado durante as primeiras fases deste projeto.

A partir dos estudos qualitativos feitos por SCHWARZ (30, 31,32), sobre o controle de fluxo e de acesso, decidimos separar alguns dos métodos de controle atualmente usados em redes de computadores. Em especial, usaremos, como base para a modelagem, uma das primeiras versões de controle 'end-to-end', aplicada, inicialmente, no ARPANET. A razão desta escolha se deve, principalmente, ao fato que este método representa, ainda hoje, uma base significativa para outros métodos de controle e, também, que permite a comparação de nosso modelo com um método existente e testado num ambiente real. Apresentaremos, então, no CAPÍTULO II deste trabalho, este método de uma maneira bem concisa, dando ênfase a alguns esboços representativos e auto-explicativos.

Escolhido o controle de fluxo a ser modelado e visando o uso da técnica de 'PrT-Net' , achamos aconselhável, por ser uma técnica nova, apresentar as bases desta técnica antes de usá-la para a modelagem de nossos problemas. Dedicamos, assim, o CAPÍTULO III a esta apresentação onde, em particular, mostraremos, primeiramente, a história do surgimento desta técnica, para depois, explicar, formalmente, os conceitos usados nela. O CAPÍTULO III termina com exemplos, bem simplificados, de como usar o 'PrT-Net' em geral.

Sentindo a necessidade de amadurecer mais ainda os conhecimentos desta nova técnica 'PrT-Net' , adaptamos o critério de "pequenos passos" na modelagem do método de controle; isto se mostrou, no decorrer deste trabalho, como sendo uma decisão acertada. Assim, modelamos, no CAPÍTULO IV, o controle 'end-

-to-end' bem simplificado, ou seja, considerando a subrede de comunicação como sendo um único recurso. Deste modo, nós nos sentimos bem mais à vontade em usar esta nova técnica de 'PrT--Net' por ter ganho mais confiança na sua utilidade.

Preparado assim, atacaremos, no CAPÍTULO V, o ponto mais importante deste trabalho, a saber, a inclusão dos vértices intermediários na modelagem. O problema principal desta inclusão se deve ao fato que não sabemos, a priori, quantas e quais dos vértices serão necessários num caminho entre algum par emissor e receptor de mensagens. Decidimos usar um tipo de modelo, tudo na base de 'PrT-Net', que permite, em vez da repetição gráfica dos vértices, aplicar uma indexação adequada nas inscrições usadas. Conseqüentemente, se tem, na representação gráfica, um único vértice intermediário, porém, sendo ele usado repetitivamente com indexação progressiva.

Este CAPÍTULO V é completado com a indicação, ainda que simplificada, das conseqüências em relação ao recebimento de um ACK ('acknowledgement') e NAK ('negative ACK') do vértice vizinho. Dissemos simplificada, porque usamos, por enquanto, somente o crédito UM entre vértices vizinhos e, também, não modelamos possíveis retransmissões consecutivas.

Tendo obtido, assim, um resultado básico, bastante representativo para nossos objetivos, podemos começar a desenvolver este modelo. Portanto, no CAPÍTULO VI, adicionamos algumas idéias sobre tempos envolvidos na transmissão de mensagens numa rede de computadores, se restringindo, porém, ao 'time-out' (entre vértices vizinhos) e ao 'check-time' (no nível 'end-to-end').

Finalmente, para terminar esta primeira fase da modelagem (o objetivo deste trabalho) decidimos, como sugere a versão número 1 do controle 'end-to-end' do ARPANET, modelar também a transmissão de mensagens do tipo multipacotes, o que, como se pode ver no CAPÍTULO VII, resulta numa adaptação bastante trivial dos modelos obtidos nos capítulos V e VI.

O CAPÍTULO VIII, chamado de "conclusões e sugestões", resume, por um lado, o que foi alcançado no trabalho presente, e, por outro lado, mostra uma multitude de possibilidades para continuar este estudo. Tem que ficar claro, que o modelo obtido neste trabalho é uma primeira tentativa de usar a técnica 'PrT-Net' para modelar o controle de fluxo em redes de computadores. O resultado pode ser considerado, não somente uma proposta (em favor de usar 'PrT-Nets') mas, principalmente, uma base para, usando todos estes conhecimentos adquiridos, partir para aplicações reais desta técnica, como, por exemplo, para modelar uma rede global de computadores a ser construída, ou ser colocada em funcionamento, realmente.

Como um dos complementos deste trabalho, anexamos, ainda, alguma BIBLIOGRAFIA, avisando, porém, que, para obter informações mais detalhadas, existam listas bibliográficas bastante completas nos trabalhos preliminares feitos por SCHWARZ (26,27, 28,30,31).

Também, como anexo, apresentaremos alguma NOMENCLATURA usada neste trabalho, facilitando assim a verificação de siglas, símbolos e expressões comumente usadas neste trabalho.

C A P Í T U L O II

Descrição qualitativa de um método para controlar o fluxo de mensagens em redes de computadores

II.1 - Introdução

Depois de ter estudado, de uma maneira introdutória, os problemas qualitativos de redes de computadores, SCHWARZ (26), as teorias principais, como, por exemplo, a teoria de filas e a de grafos, utilizadas para modelar vários problemas em relação a estas redes, SCHWARZ (27), e, finalmente, algumas ferramentas, tais como programação matemática e simulação, para poder analisar quantitativamente as redes, SCHWARZ (28), chegamos, finalmente, ao verdadeiro problema: CONTROLE DE FLUXO em redes de computadores.

Num estudo subseqüente a estes iniciais, SCHWARZ (30) tentou abordar o problema de uma maneira similar ao estudo básico. Em particular, foram estudados, de uma maneira qualitativa, os principais métodos utilizados para o controle de fluxo e, também, tentou-se isolar as ferramentas mais qualificadas para um ajuste dinâmico deste controle de fluxo.

Na primeira seção do estudo mencionado, encontram-se, basicamente, as principais definições encontradas na literatura. Tentou-se, também, homogeneizar um pouco estas definições para que possamos nos comunicar na base de uma mesma "linguagem".

Uma vez conseguida esta linguagem comum, examinaremos, nas seções subseqüentes deste estudo preliminar, métodos para o controle de fluxo em canais, para, depois, ampliar estes conceitos para serem utilizados em redes de computadores. O estudo foi prosseguido com explicações qua-

litativas das ferramentas principais, isto é, o crédito, a alocação dos buffers e o roteamento adaptativo, e foi concluído com uma lista, razoavelmente completa, de indicações bibliográficas.

O que precisamos fazer agora, é escolher aqueles trechos deste estudo que serão usados, diretamente, neste trabalho.

A nossa escolha caiu, sem muita justificativa formal, sobre um método para controlar o fluxo em canais, chamado por nós de "método simples com ACK, NAK e 'time-out'", e um outro para controlar o fluxo em redes, denominado de "controle 'end-to-end', tipo ARPANET, versão 1".

Descreveremos, então, na seção II.2, o controle de fluxo em canais o que, afinal, vai ser necessário para dominar os problemas na comunicação intervértices, e, na seção II.3, o método 'end-to-end' mencionado.

II.2 - Método simples de controle de fluxo em canais incluindo, ainda, ACK, NAK e 'time-out'.

Antes de apresentar este método, queremos mencionar uma técnica simples que é baseada na transmissão após um intervalo seguro. Calcula-se um tempo de intervalo entre o envio de dois pacotes, tal que um pacote sendo transmitido não perturbe o anteriormente enviado. Vale observar que o bom funcionamento depende deste tempo calculado e que, neste nível, não haverá nenhum teste ou confirmação sobre a consistência dos pacotes.

Outro extremo é o método que só transmite um novo pacote após ter recebido um ACK (confirmação) do pacote anterior. Este é o procedimento clássico de comunicações assíncronas, mas deve ser ressaltado que só é confirmada a chegada sem, contudo, testar a consistência do pacote. Estes testes podem, ou até devem, ser feitos num nível superior. A grande desvantagem deste método, além de não testar o pacote, é o uso ineficiente de

'bandwidth' ou faixa disponível.

Partindo destes dois métodos, podemos combiná-los e complementá-los, chegando, assim, a um método simples que usa um tipo de tempo seguro, o 'time-out', para verificar o tempo de transmissão e que complementa o ACK, através de algum teste, com um NAK (negação ou rejeição), provocando, assim, uma eventual retransmissão.

Para facilitar as investigações de diferentes métodos, visando ainda comparações posteriores, foi seguida, no estudo de SCHWARZ (30), uma divisão em itens, tais como: técnica, problemas principais, objetivos, utilidade e uso, vantagens e desvantagens, observações e referências. Como esta descrição sucinta é baseada neste estudo mencionado, usaremos esta mesma divisão em itens.

- Técnica: Transmitir um pacote por vez depois de ter recebido o ACK do pacote anterior;

  Retransmitir um pacote se, em vez de um ACK, chegou um NAK ('negative ACK') ou se ultrapassar o tempo máximo concedido, o 'time-out', para que chegue alguma resposta (ACK ou NAK) relativa ao pacote enviado (fig. II.1).

- Problemas principais: O 'bandwidth' é subutilizado por causa do comportamento assíncrono; a solução para este problema é encontrada nos métodos de crédito que permitem o uso da linha como 'buffer' ou 'pipe-line'.

  Temos, também, que decidir o que deve ser confirmado, isto é, se confirmarmos somente a chegada de um pacote, estamos obrigados a testar o pacote num nível superior que, depois, pede, eventualmente, uma retransmissão explícita. Também, não será garantido espaço no receptor para um próximo pacote, usando-se somente um ACK para chegadas.

  Métodos mais avançados testam, já neste nível mais baixo, a consistência do pacote, usando para este fim códigos especiais, adicionados ao conteúdo do pacote. Mesmo que pareça

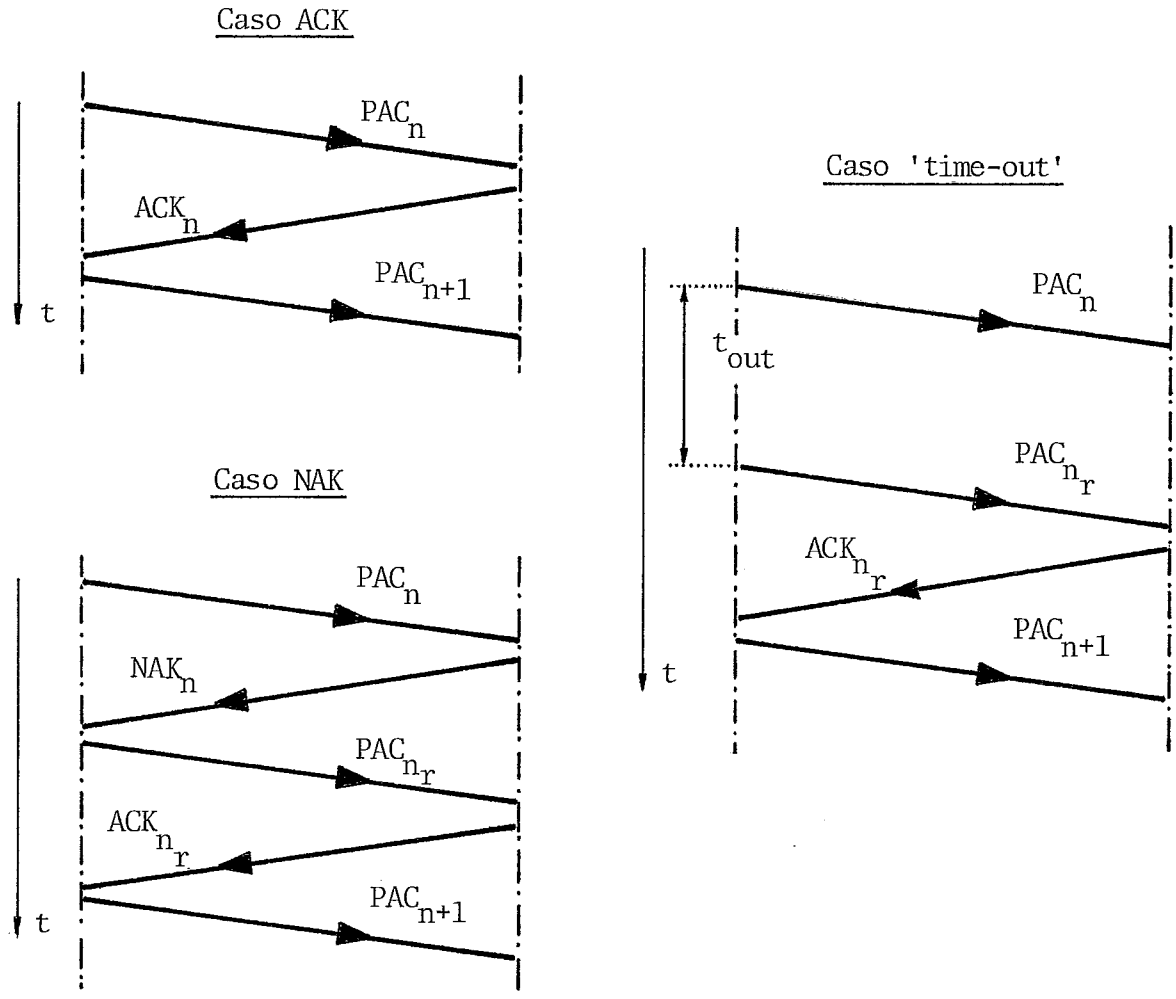


Fig. II.1: Transmissão depois de um ACK, ou retransmissão depois de um NAK ou 'time-out'.

uma técnica muito simples, temos que pensar na razão pela qual uma retransmissão possa ser solicitada e o que acontece caso um erro se repita.

Vejamos, primeiramente, o caso do NAK: o pacote já está incorreto no momento da emissão (p.ex., erros no processamento), ou o pacote é perturbado durante a transmissão (isto inclui os E/S, os modems, os repetidores e as próprias linhas). Dependendo destes pontos, temos diferentes probabilidades de repetição de erros. Normalmente, se o erro for aleatório, a retransmissão terá sucesso; mas, no caso de erro permanente, surge a pergunta de quantas vezes devemos retransmitir, isto é, como sair deste ciclo.

A segunda razão para uma retransmissão é um 'time-out', que pode acontecer quando a transmissão do pacote levou tempo demais, ou o sinal ACK está incorreto, nulo, perturbado na transmissão ou transmitido lentamente demais, ou o sinal NAK está incorreto, perturbado ou muito lento na transmissão. A princípio, vale o mesmo que o estudado no caso do NAK, isto é, a diferença nas nossas decisões depende dos erros serem aleatórios ou permanentes. Mas, no caso de 'time-out', temos, ainda, o problema de calcular um 'time-out' correto, isto é, se o 'time-out' é muito grande, perde-se a eficiência desta ferramenta e, se é pequeno demais, retransmite-se, eventualmente, sem razão e, neste caso, temos que distinguir entre um pacote transmitido (PAC_n) ou retransmitido (PAC_{n_r}).

- Objetivo: Proteção da linha e do receptor de excesso de pacotes, aplicando o procedimento clássico de linhas com transmissão assíncrona, diminuindo a probabilidade de que pacotes errados sejam enviados ao próximo vértice ou ao nível superior.

- Uso (utilidade): É um método útil em todas as ligações ponto-a-ponto, onde o uso ineficiente de 'bandwidth' é menos importante que a presença de um pacote correto.

  Nesta categoria, entram, por exemplo, o acesso de terminais ao 'host', a ligação do 'host' à subrede de comunicações, e o equivalente para sair da subrede, isto é, entre vértice/destinação e 'host'/destinação.

- Vantagens: O método básico é muito simples devido ao comportamento assíncrono e a garantia de pacotes corretos é bastante grande (depende, principalmente, do código cíclico usado).

- Desvantagens: Além do uso ineficiente de 'bandwidth' por não permitir o uso da linha como 'pipe-line', temos a sobrecarga para testar os pacotes (informação "não útil" no pacote e tempo de processamento maior) e o trabalho de distinguir entre pacotes originais ou retransmitidos.

- Observações adicionais: O ponto mais crítico deste método é o cálculo correto de 'time-out', que dependerá, neste caso, do tempo de transmissão do próprio pacote, do tempo de processamento no receptor e do tempo de transmissão do ACK ou NAK; sobre este último ponto, há, ainda, o seguinte a dizer: se existe tráfego em ambas as direções, os sinais ACK ou NAK podem ser transmitidos como pacotes de controle (isto é, isoladamente, seguindo ou não um controle de fluxo noutra direção), ou embutidos num pacote de informação (neste caso, incluído no controle de fluxo), técnica chamada, na literatura, 'piggy-back'.

- Referências: DAVIES et alii (04), SCHWARZ (30).

II.3 - Método de controle de fluxo numa rede de computadores, apresentando o controle 'end-to-end', tipo ARPANET.

O controle de fluxo em canais (pensamos aqui, sobretudo, em termos vértice-vértice ou intervértices) é a base de todo o controle de fluxo numa rede. Se o fluxo, dentro de um 'link' ou enlace, já não estiver bem controlado, havendo, também, interações com outros enlaces, o funcionamento da rede pode sofrer uma degradação considerável. Então, a questão principal será: "Como relacionar o controle de fluxo local (p. ex., dentro de um enlace) e o fluxo global (dentro da rede inteira)?" Conseqüentemente, é com esta questão que começam os verdadeiros problemas de controle de fluxo em redes de computadores.

Escolhemos, dentro dos três métodos distintos mais conhecidos ('end-to-end', datagrama e isarítmico), o método que nos parece o mais útil para nossos fins de modelagem, a saber, o controle 'end-to-end' do tipo ARPANET, versão 1.

Este tipo de controle foi o primeiro usado no ARPANET (nos anos 1971/72) e mostra, em princípio, a idéia de transmissão somente depois de ter recebido o RFNM associado ao pacote anterior (como o ACK, usado em canais, somente sendo adaptada a canais tipo 'multihop'). Vejamos esta técnica em seguida.

– Técnica: Usa-se a transmissão de uma mensagem por vez entre algum par de processos. Esta mensagem pode ser dividida em até oito pacotes, que podem, eventualmente, tomar caminhos diferentes. Uma nova mensagem pode somente ser enviada após ter chegado o RFNM ('request for next message') da mensagem anterior (isto é certamente dispersão de 'bandwidth'). E permitida a existência de muitos pares e, conseqüentemente, haverá transmissão simultânea entre diferentes pares de processos. Veja, então, na figura II.2, o esquema simplificado que é muito parecido com o esquema de controle em canais inter-vértices, e, na figura II.3, o mesmo esquema, mas mostrando a integração do 'check-time', uma espécie de 'time-out' no nível 'end-to-end'. Na figura II.4, mostramos este mesmo esquema simplificado, mas indicando a possibilidade de que a mensagem seja grande e, conseqüentemente, dividida em até oito pacotes. Finalmente, nas figuras II.5 e II.6, mostramos o esquema de uma maneira mais detalhada, chamado por KLEINROCK (14) de "anatomia da transmissão", uma vez por um único pacote (fig. II.5) e, noutra figura (fig. II.6), para uma mensagem do tipo multipacotes.

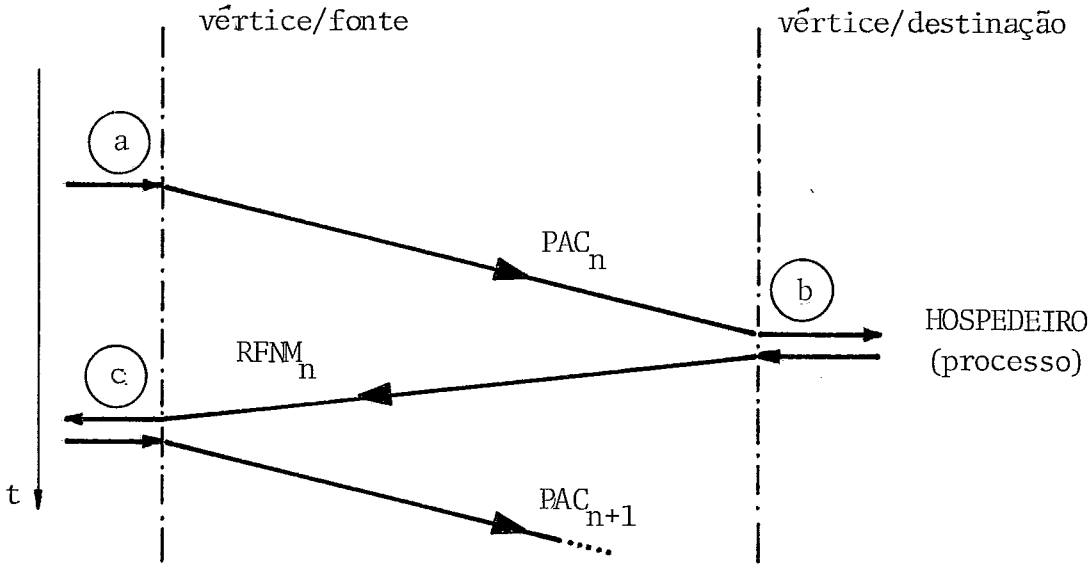


Fig. II.2: Esquema simplificado, sem detalhe, da seqüência de uma transmissão entre fonte e destinação.

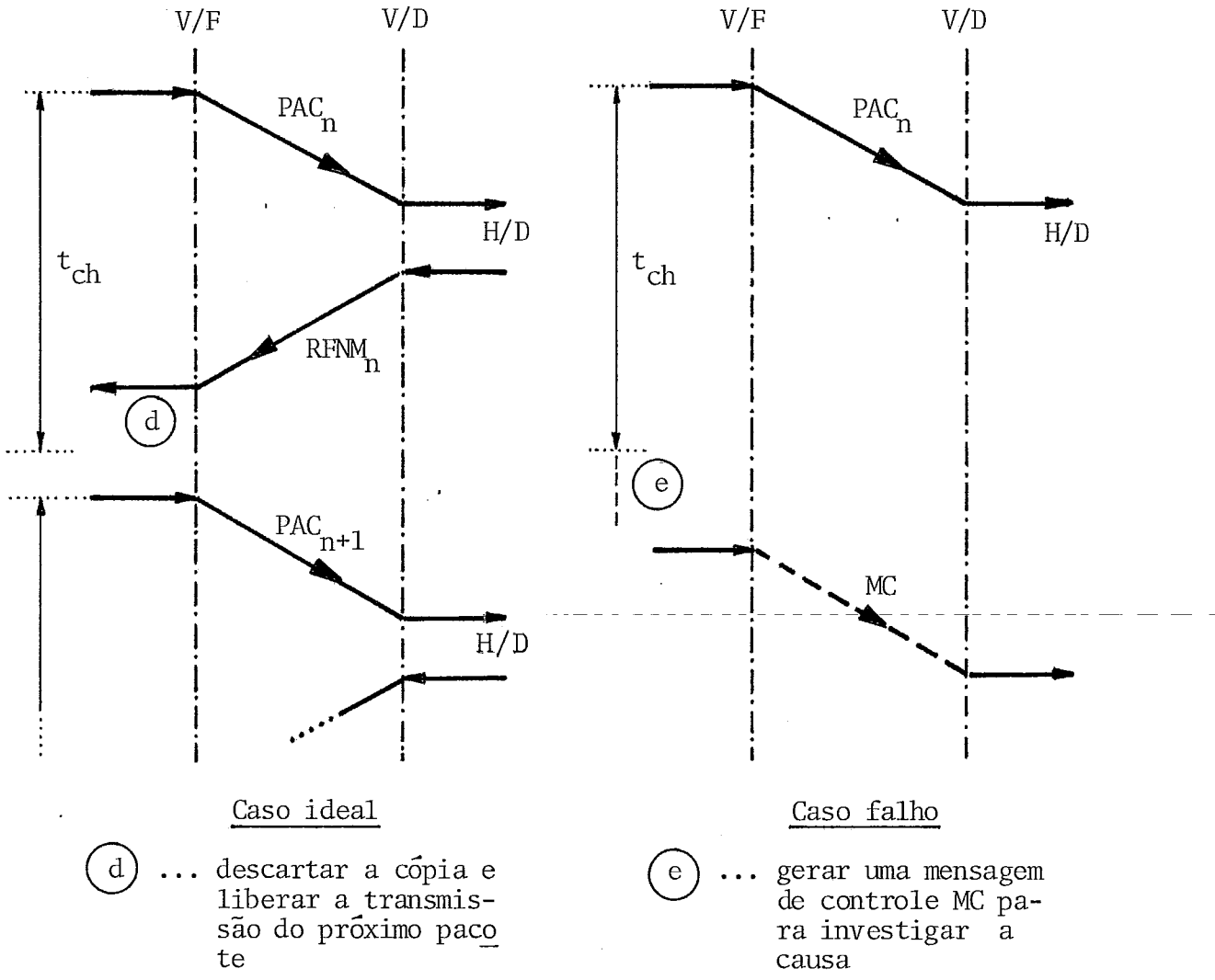
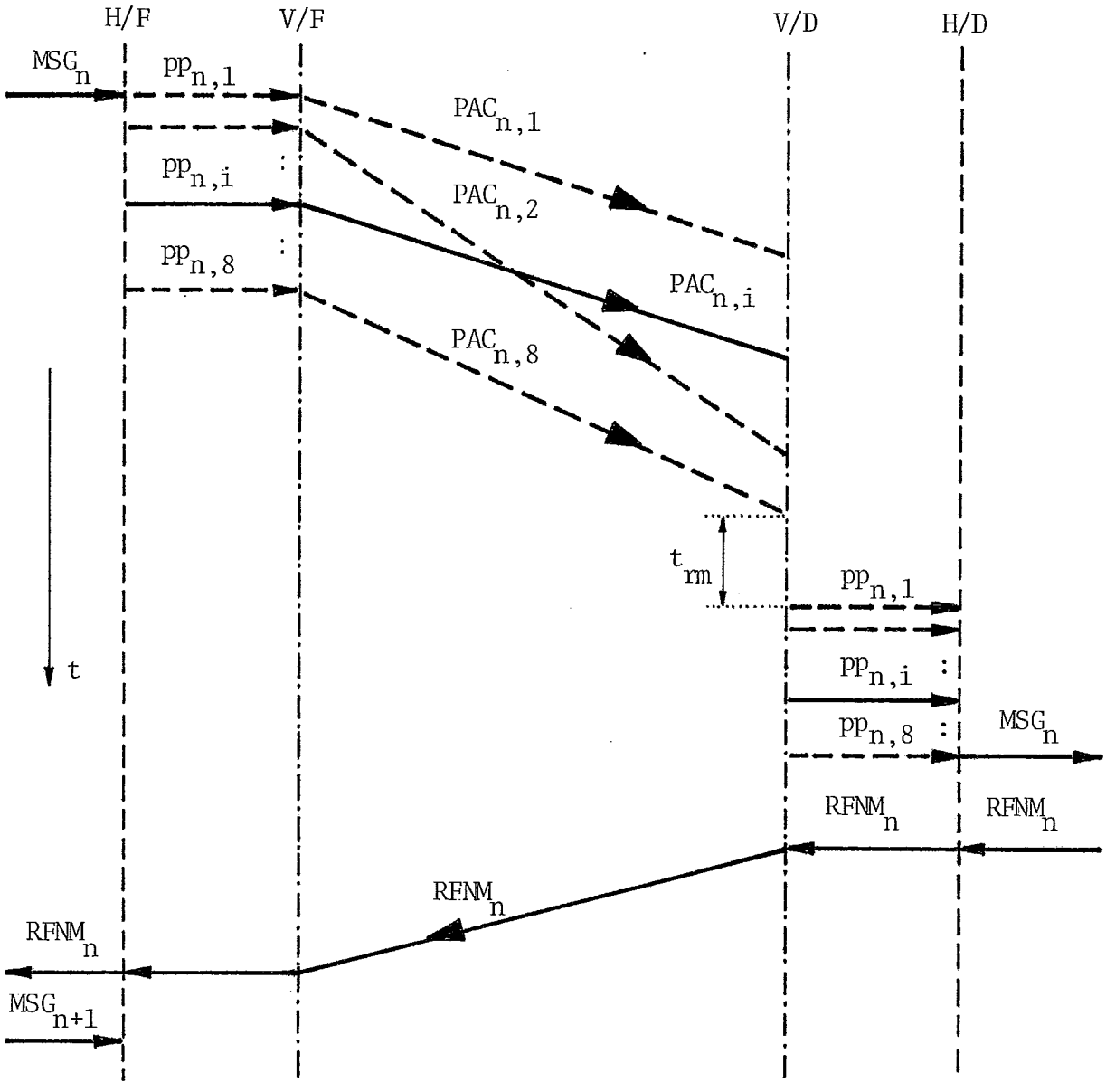


Fig. II.3: Esquema, sem detalhes, da transmissão 'end-to-end' de um pacote simples mostrando a integração do 'check-time', t_{ch}, envolvido.

- Problemas principais: A ligação entre os hospedeiros dos processos é feita através da criação de dois enlaces (lembrar, que um enlace é uma conexão lógica unidirecional), mas ela não é feita de uma maneira explícita, isto é, os recursos (linhas físicas,'buffers',etc) não foram reservados de antemão. Isto gerou inconvenientes, tais como 'lockups', e estimulou um controle diferente (como, apresentado por KLEINROCK (14), nas versões dois e três).



Obs.: – Somente é permitida, nesta versão, a transmissão de UMA mensagem por vez entre qualquer par de usuários;

– Os PACOTES individuais têm tratamento distinto (p. ex., caminhos diferentes, retransmissões intermediárias, etc.), podendo chegar, no V/D, fora da ordem original;

– t_{rm} indica o tempo necessário para remontagem.

Fig. II.4: Esquema simplificado, sem os detalhes dos vértices intermediários, da transmissão de uma mensagem do tipo multipacote.

Então, parcialmente baseado no fato de não haver reserva antecipada neste método, há um grupo de problemas, que são os diferentes 'lockups' mencionados:

O problema de 'direct store-and-forward lockup' (não há mais espaço disponível para passar os pacotes adiante) pode ser resolvido dividindo o espaço disponível num vértice em buffers reservados para diferentes filas de saída, além de dividir em buffers para 'store-and-forward' e buffers para 're-assembly'.

Já o problema de 'indirect store-and-forward lockup', que acontecerá quando todos os pacotes viajam numa só direção, somente pode ser evitado (sem grandes custos acidionais) usando mensagens de controle.

O problema mais sério deste método foi o 'reassembly lockup', que poderia acontecer por causa de multipacotes, que podem tomar diferentes enlaces para chegar ao vértice/ /destinação ou, ainda, por causa de comunicação de vários 'hosts' com a mesma destinação. Como KLEINROCK (14) mostrou num outro método (a versão 2), isto pode ser resolvido reservando o espaço nos buffers antes de enviar mensagens.

- Observações adicionais: O seqüecionamento das mensagens é correto (não se deve confundir este seqüecionamento com o dos pacotes, que devem ser colocados em ordem no vértice/ /destinação e assim, ao mesmo tempo, evitar duplicatas).

O ponto mencionado, o de não precisar alocar, antecipamento, os recursos da subrede, pode se tornar desvantajoso no caso em que o tráfego esteja pesado e que não haja bastante recursos disponíveis (isto, p.ex., pode gerar um bloqueio perpétuo). A alocação do espaço do buffer pode aliviar este problema. Relativo ao buffer, não se deve esquecer que, similar ao método simples de ACK em canais, no vértice/fonte deverá ser guardada uma cópia da mensagem até que chegue a confirmação, o RFNM, o que implica num gerenciamento do espaço no buffer de uma maneira adequada.

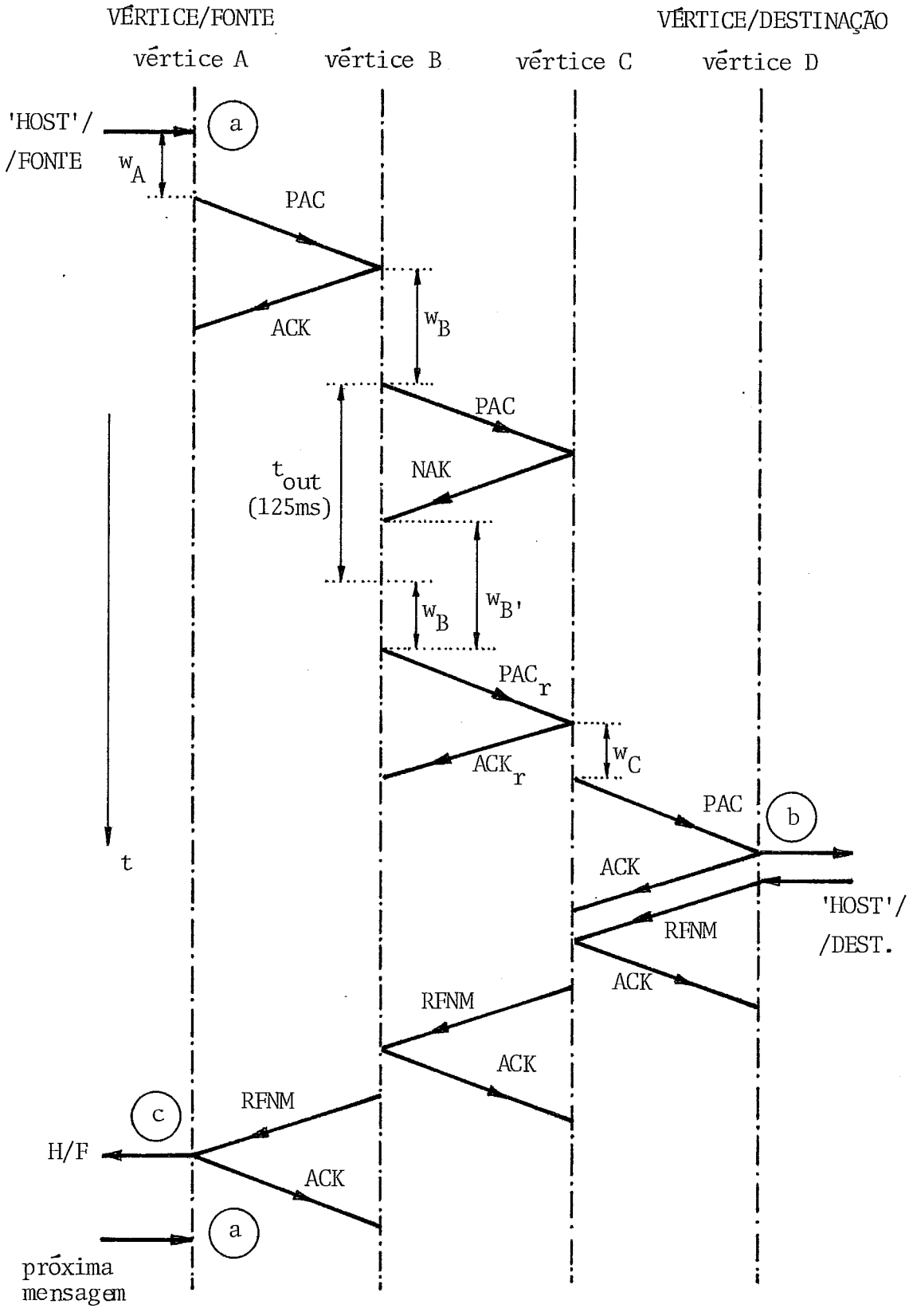


Fig. II.5: Anatomia de uma transmissão de um ÚNICO PACOTE, mostrando detalhes tais como PAC (pacote), ACK ('acknowledgement'), NAK ('negative ACK'), 'time-out', RFNM ('request for next message') e os tempos de espera nas filas (w_I).

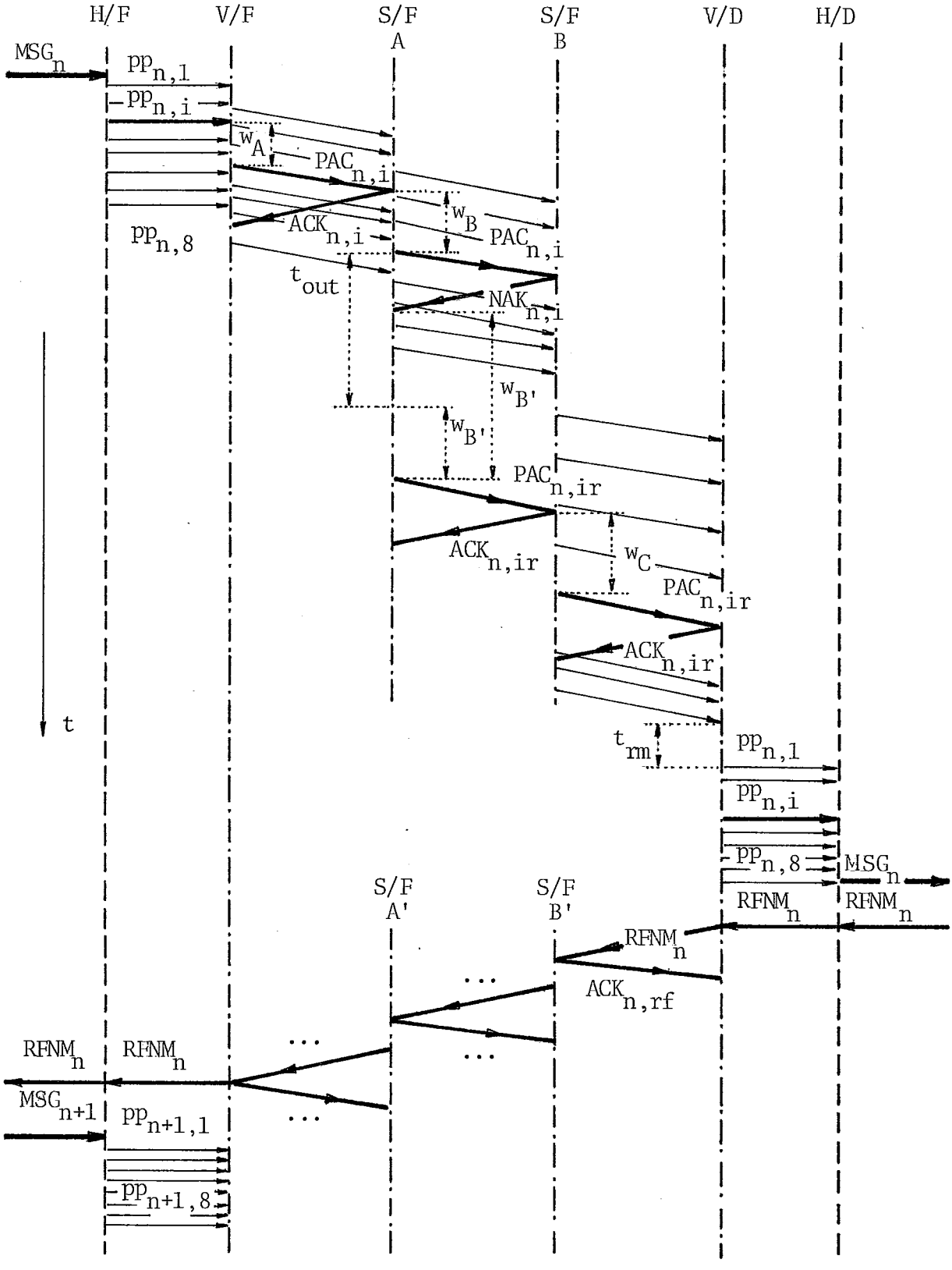


Fig. II.6: Anatomia da transmissão de mensagens do tipo MULTI-PACOTES mostrando os detalhes, tais como PAC (pacote), ACK ('acknowledgement'), NAK ('negative ACK'), 'time-out', RFNM ('request for next message'), w_I (tempo de espera nas filas) e t_{rm} (tempo para remontagem), somente para um pacote ($PAC_{n,i}$).

Outro ponto a observar é que também precisa-se um teste especial de software para verificar não somente os erros de transmissão, mas, também, os erros que possam ocorrer nos próprios vértices. Em outras palavras, precisa-se de testes num nível superior.

— Referências: KLEINROCK (14), DAVIES et alii (04), SCHWARZ (30).

II.4 - Resumo do capítulo II

Neste capítulo II, apresentamos, de uma maneira bem resumida e usando bastante o recurso de esboços auto-explicativos, os métodos de controle de fluxo que serão a base para nossas tentativas de modelagem. Queremos alertar, que este capítulo somente foi introduzido neste trabalho para servir como revisão dos métodos e, também, para facilitar as passagens no texto deste trabalho que fazem referência a estes métodos de controle de fluxo em redes de computadores.

C A P Í T U L O III

Uma introdução a GNT ('General Net Theory') e grafos do tipo PrT ('Predicate/Transition-Nets').

III.1 - Introdução

(Observação: uma versão preliminar deste capítulo foi publicada como relatório técnico, SCHWARZ (33)).

Já mencionamos em um outro trabalho de SCHWARZ (27) a necessidade de usar modelos matemáticos para representar os métodos empregados para resolver problemas de controle, a formação de uma arquitetura adequada, para aumentar a segurança, etc. Neste mesmo trabalho tínhamos mostrado, de um modo geral, como as teorias de filas e de grafos poderiam ser utilizadas.

No início de nossos estudos sobre o controle de fluxo e de acesso, explicamos como estas teorias podem ser usadas para modelar algumas das ferramentas e métodos descritos qualitativamente por SCHWARZ (30,31,32). Mas, ao mesmo tempo, avisamos que estas teorias são mais úteis para modelar o fluxo do que as influências. E é neste ponto onde poderíamos, a nosso ver, precisar de uma outra ferramenta, uma outra teoria. Encontramos esta ferramenta na GNT ('General Net Theory'), que, se baseando na idéia de Petri-Net, desenvolveu "grafos de nível superior" como, por exemplo, o grafo PrT ou 'PrT-Net' ('Predicate/Transition-Net').

Vamos então, neste capítulo III, mostrar a utilidade desta ferramenta. Em particular, começamos (na seção III.1) a dar uma introdução histórica à GNT, para termos uma idéia global como se chegou a 'PrT-Net'. Depois de uma definição global deste 'PrT-Net', explicaremos (na seção III.2) os componentes, utili

zando um exemplo da literatura (R/W, 'reader/writer'). Para melhorar ainda mais a compreensão, discutimos ainda um outro exemplo da literatura, um simples banco de dados distribuídos, mas já visando uma interpretação relativa ao nosso problema de redes.

III.1.a - Observações gerais

Baseados em observações sobre insuficiências das teorias utilizadas, queremos introduzir neste capítulo III a idéia de grafos do tipo PrT ('PrT-Nets') e mostrar, nos capítulos subseqüentes, como este grafo poderia ser usado para modelagem.

A nossa idéia é que se faz o seguinte:

- Usa-se a GNT ('General Net Theory'), em especial o 'PrT-Net', para modelar os problemas dinâmicos numa rede (conflitos, concorrências, etc.);
- Indica-se como e onde, neste modelo, podem ser usadas as teorias chamadas de clássicas. Sabe-se que as teorias clássicas têm seus méritos, só que colocam muito em hipóteses iniciais e que tratam as influências mútuas de uma maneira, a nosso ver, demasiadamente generalizada.

Reunimos então em seguida o desenvolvimento desta teoria, a GNT. Depois, apontando as deficiências e dificuldades iniciais, mostraremos como o poder de modelagem e "descritividade" aumentou através da cadeia de desenvolvimento formada por Petri-Net ('Place/Transition Net'), sistemas C/E ('Condition/Event-Systems') e, finalmente, o grafo PrT. Terminaremos, esta seção, com alguns tópicos de destaque, seguido pelas preocupações típicas da GNT.

III.1.b - História da GNT ('General Net Theory')

III.1.b.1 - Geral

A adaptação de uma teoria altamente especializada, represen-

tada, por exemplo, pelos grafos de transição ('transition nets'), para uma grande escala de aplicações como, por exemplo, encontrada na ciência da computação, necessitou grandes esforços. Para familiarizar os cientistas com estes esforços e progressos, houve um curso internacional avançado sobre a teoria de grafos e suas aplicações, editado por BRAUER (01), introduzindo assim os conceitos e as ferramentas matemáticas desta "General Net Theory".

A teoria inicial, usando grafos de transição, foi também chamada de 'Special Net Theory', sobretudo porque está orientada ao fluxo de recursos contáveis em redes, isto é, estruturas similares a grafos em que vários fluxos são coordenados (sincronizados) e também ramificados e reunidos. Esta teoria serve, entre outros, para a representação gráfica, para o tratamento numérico de gargalos e bloqueios perpétuos, e também para o tratamento relativo à segurança e conflitos.

De um modo geral, podemos dizer que o Petri-Net (nome comumente usado para os grafos de tipo lugares/transições ('Stellen/Transitionen'), descritos resumidamente por PETERSON (20) e SCHWARZ (27) é uma ferramenta para projetar e validar "sistemas". Esta ferramenta permite ao usuário, numa fase inicial do projeto, obter uma melhor compreensão, e assim também confiança, na consistência lógica de algum mecanismo de controle, sobretudo em sistemas do tipo distribuído.

Agora, como podemos usar esta ferramenta?

Podem-se usar símbolos (por exemplo, T-elementos □ e S-elementos ◯) que representam unidades de funcionamento tais como os acontecimentos de processos ou de fenômenos, ou tais como a existência de estados ou condições, respectivamente.

A relação causal, direcionada, entre estas unidades de funcionamento, é representada por flechas (→) onde a ponta da flecha indica a direção do fluxo da informação (as flechas não representam uma outra unidade de funcionamento, mas meramen

te a relação entre unidades, conforme observação feita por RICHTER (23) neste sentido).

Cada um destes símbolos (□○→) pode receber indicações do tipo:

- Imagens, esboços, gráficos, etc.;
- Linguagens formais, fórmulas matemáticas, etc.;
- Linguagens naturais, descrições verbais, etc.;

A semântica destas incrições é deduzida (por exemplo, por abstração) de um modelo inicial, como, por exemplo, um sistema C/E. Essas inscrições descrevem: acontecimentos, relações, objetivos, recursos reais ou abstratos, atividades, etc., e valem para cada instante da aplicação.

Apesar de sua utilidade em algumas aplicações, esta ferramenta básica tem suas limitações práticas. O tratamento ocorre em um nível muito "baixo" e detalhado demais (compare, por exemplo, as tentativas de modelar os protocolos de transmissão do nível 2 por MUSCHELLACK (18) que usou este famoso 'token game') e não se mostra útil para sistemas maiores.

Para evitar estas limitações, foram desenvolvidas diversas extensões que resultaram na teoria geral de redes (ou grafos), tradução um pouco infeliz do nome original que é de 'General Net Theory'. Esta teoria (usaremos a partir de agora somente a sigla GNT) trata, entre outros, dos seguintes tópicos:

- As relações entre grafos ('nets');
- As operações e funções relativas a determinadas classes de grafos;
- As transformações de grafos;
- O 'net-morphism'.

Este último ponto, o 'net morphism', é um dos tópicos mais importantes para nós, pois permite "diminuir" ou "ampliar" a

estrutura de grafos. Existem, para este fim, funções de mapeamento de um 'net' para um outro, respeitando conectividade e orientação, permitindo assim adaptar os modelos às nossas necessidades. Se temos, por exemplo, uma estrutura N_1, podemos aplicar este tipo de mapeamento, f: $N_1 \rightarrow N_2$, para obter uma estrutura N_2 que é mais "conveniente".

Estes fatores, em conjunto com a possibilidade de ter inscrições de nossa escolha, torna a GNT extremamente interessante para nós porque representa, provavelmente, a ferramenta mais indicada para modelar os conflitos, influências, dependências, etc., num problema como o nosso, que trata de controle de fluxo e de acesso, de uma maneira dinâmica, numa rede de computadores do tipo MHT ("Meios Heterogêneos de Transmissão"). Estas são, por natureza, estruturas distribuídas, grandes e complexas, mas que têm a virtude de uma possível classificação dos problemas o que, eventualmente, permite a aplicação do 'net-morphism'.

Após estas considerações, descreveremos nos próximos itens desta seção, a evolução histórica, os tópicos de destaque e as preocupações típicas da GNT.

III.1.b.2 - Evolução histórica

Para ter alguma idéia sobre a criação desta GNT, queremos dar, em seguida, uma breve enumeração das marcas históricas, especificando assim um pouco mais a cadeia de desenvolvimento formada por:

- Grafos de lugares/transições ou Petri-Nets;
- Derivações de Petri-Nets, resultando em grafos com inscrições;
- Sistemas de condições/eventos;
- Grafos do tipo predicado/transição.

Marcas históricas, resumidas por GENRICH et alia (08) e SCHWARZ (34):

1962: C.A. Petri introduziu um tipo de grafo chamado grafo de lugares/transições ('place/transition-net'). Este serviu como ferramenta para desenvolver uma tentativa de estudar, de uma maneira flexível, CONCORRÊNCIA e FLUXO DE INFORMAÇÕES em sistemas organizacionais.

1968/69/70: A.W. Holt et alii., R.M. Shapiro e H. Saint, S.S. Patil. Em vários estudos (ver bibliografia apresentada por GENRICH e LAUTENBACH (08)) foi demonstrado como usar na prática estes grafos, agora chamado de Petri-Nets, para modelar a arquitetura ('design') de sistemas.

1970: A.W. Holt e F. Commoner. Este grupo estimulou o uso de Petri-Nets para projetos mais ambiciosos. Entretanto, foi sentido rapidamente que se tem que trabalhar com sistemas grandes demais e, ainda, num nível de detalhamento não aceitável. Assim sendo, muita gente abandonou a idéia de trabalhar com Petri-Nets. Mas, felizmente, existíam ainda outros grupos que tentaram usar derivações e extensões da idéia geral para uma eventual adaptação aos objetivos específicos. Exemplos disto são grafos coloridos, grafos com inscrições relativas ao tempo, com diferentes tipos de lugares, etc.

1973: C.A. Petri propôs "interconectar" os vários modelos usados, trabalhando com algum tipo de transformação (que, porém, garante o significado original) destes grafos com inscrições ('inscribed nets').

1975: As inscrições podem ser texto, linguagens, gráficos, etc. A sua semântica é deduzida através de complementação e abstração, a partir de uma interpretação básica de grafos, chamada de sistemas tipo condição/evento ('condition/event-system').

1977: C.A. Petri chamou este desenvolvimento de GNT ('General Net Theory'), indicando a sua origem na 'special net theory' (como, por exemplo, o antigo 'token game' em 'place/transition-nets' era chamado).

1979: Curso avançado sobre a GNT, indicando, entre outros, uma tentativa com inscrições bastante sofisticadas, o chamado 'PrT-Net' ('predicate/transition-net'). Esta ferramenta combina e complementa, a partir da idéia de um simples Petri-Net, as evoluções individuais de H.J. Genrich e G. Thieler-Mevissen (estrutura 'enlogy'), K. Lautenbach, M. Schiffers, H. Wedde ('tokens' coloridos), R.M. Shapiro (condições complexas e transmissões). Os autores de 'PrT-Net' (H.J. Genrich, K. Lautenbach) afirmam que este tipo de grafo adiciona uma nova dimensão à complexidade e ao poder de modelagem de Petri-Nets, a saber, o tratamento formal de indivíduos e, também, das suas propriedades e relações.

1979/80/81: Aplicação desta última ferramenta, o 'PrT-Net', para analisar "Banco de Dados Distribuídos", como feito por GENRICH et alii (06,08) e VOSS (39).

1981/82: Um grupo em Darmstadt, Alemanha Ocidental, (Burkhardt, Eckert e Prinoth) usou o 'PrT-Net' para a modelagem de protocolos seguindo a norma ISO ('International Standards Organization') sobre OSI ('open systems interconnection'), para aplicações no GMD-IDF, ('Gesellschaft für Mathematik und Datenverarbeitung - Institut für Datenfernverarbeitung'), G. Richter (GMD - Birlinghofen), Alemanha Ocidental, está usando a IML ('Information Management Language') como linguagem de inscrições em 'PrT-Nets', no contexto de comunicações relativas a escritórios.

1982/83: Tentativa de usar o 'PrT-Net' para complementar o modelo para o controle de fluxo e de acesso em redes de computadores, tipo MHT, ponto de vista influências e dependências mútuas (Gerhard Schwarz, COPPE/UFRJ, Rio de Janeiro).

Este último ítem será mostrado neste trabalho, sob forma de uma proposta inicial; num futuro trabalho devemos, obviamente, investigar mais profundamente o conjunto destes modelos, visando um comportamento harmonioso de um conjunto de teorias, formado pelas teorias de filas, de grafos, de decisões e da GNT.

III.1.b.3 - Tópicos de destaque

Lembrando as idéias lançadas na seção III.1.b.1, surgem, em relação à GNT, várias questões. Elas são, segundo PETRI (21), questões de natureza conceitual e formal e serão mostradas, de uma maneira concisa, em seguida:

- Definição parcial de um grafo. Este ponto se faz necessário se a estrutura N_1 (lembrar de $f : N_1 \rightarrow N_2$) não está completamente especificada. Conseqüentemente, a função f também não o é. Assim precisamos algo para, por exemplo, extrapolar a função f para obter uma estrutura N_2 "razoavelmente" representativa;

- Existência eterna. Este conceito aparece quando temos que trabalhar com, por exemplo, uma cadeia infinita. Aí devemos nos concentrar ou em observar somente uma parte da cadeia, ou em achar partes "repetitivas" às quais possamos aplicar uma única função;

- Horizonte de interesse ou de preocupação ('scope of concern'). Cabe a nós definir se observarmos um sistema inteiro ou somente uma parte dele. Isto se faz comumente quando se tem sistemas grandes e complexos. Aqui, na GNT, surge adicionalmente mais uma pergunta, ou seja, "O sistema sob observação começa com uma condição ou com uma transição?";

- Extensão do horizonte de interesse. Em vez de dar atenção aos elementos de um conjunto X, podemos dar atenção a um subconjunto de X, o que é equivalente a um agrupamento de alguns dos elementos. Este ponto é importante, porque dele vem o objetivo principal de 'net-morphism': AUMENTAR ou DIMINUIR o horizonte de interesse;

- Definições formais dos elementos da expressão $f : N_1 \rightarrow N_2$. A função f, uma vez definida, é chamada de processo, N_2 é chamado de sistema e N_1 é o domínio de processo f. No caso em que N_1 não tem inscrições, não é mais chamado de domínio, mas somente de estrutura de domínio de um (possível) processo;

- Duração de estados e de transições. Um estado certamente tem uma duração diferente de zero. Mas o que pensar em relação às transições? Pode-se dizer que a duração de uma transição é não-zero ou não-instantânea, ou, no outro extremo, é zero (veja a semelhança em sistemas de 'polling' estudada por VASCONCELLOS ([38])). Pode-se também dizer que elas, as transições, são momentos de incerteza entre as situações bem definidas de estados. Ou, ainda, pode-se determinar que o conceito de duração não se aplica às transições.

Como cada um destes pontos de vista sobre a duração contém algum aspecto importante, em determinadas situações ou visto de diferentes ângulos, a GNT conclui o seguinte:

A duração de transição tem algo a ver com a lógica de mudanças de valores (reais); assim define-se que transições são ENTIDADES AUTÔNOMAS, e não somente relações entre condições.

Respostas a esta questão, relativa à duração, surgirão da teoria de concorrência, mas não em termos de momentos e durações (que não estão bem definidos no sentido operacional). Trataremos os "tempos" como sendo leituras de relógios que, por sua vez, serão tratados como qualquer outro componente de um sistema, ou seja, sujeito a mal-funcionamento e, eventualmente, destruição.

Assim, a GNT, com sua FLEXIBILIDADE, serve para obtenção de APLICAÇÕES REAIS e não somente para algum desenvolvimento teórico.

III.1.b.4 - Preocupações típicas da GNT.

Partindo destes tópicos de destaque, apresentados na seção III.1.b.3, podemos agora enumerar as maiores preocupações da GNT. Elas são relacionadas com, por exemplo, níveis conceituais, concorrência, limitação dos recursos, conceitos relevantes, imprecisões, etc.

Vamos, em seguida, falar um pouco sobre cada um destes pontos, sendo o mais importante deles o item sobre os níveis conceituais, já que as definições relativas a estes níveis são os maiores responsáveis para obter sucesso na modelagem. Dedicamos, então, o item β desta seção a estes níveis conceituais.

Mas antes de apresentar estas considerações, falaremos, no item α, sobre problemas, digamos, ligados à "realidade física". São estas as limitações dos recursos e as imprecisões na "interligação" do mundo físico com o de um modelo. O item α menciona, então, estes problemas.

Mencionamos também a palavra "concorrência" e aí devemos especificar que queremos falar da concorrência ligada à dependência parcial de acontecimentos. É necessário conhecer, por exemplo, as propriedades de concorrência, sobretudo em sistemas SEM controle central ou SEM observabilidade global. Surgem aí problemas ligados ao fato que concorrência não é transitiva. Daí são derivadas questões do tipo: "Efeitos devem ser (ou podem ser) os mesmos em estruturas topológicas diferentes?", ou: "A definição de uma determinada topologia de um grafo influenciará a questão de concorrência?"

Esta é mais uma, entre outras, das maiores preocupações da GNT, e entramos então, agora nestes itens mencionados acima.

III.1.b.4.α - Preocupação com a realidade física.

Limitação de todos os recursos.

Este é um ponto importantíssimo pois pode influenciar drasticamente no comportamento de qualquer sistema. Podemos, mas somente inicialmente, estudar os sistemas admitindo recursos infinitos (por exemplo, num sistema de filas com 'buffers' infinitos) para facilitar a compreensão básica, mas qualquer teoria realística deve incluir, mais cedo ou tarde, o conceito da limitação de recursos.

Deve-se, então, começar a definir o que são recursos. Temos

aí, ao lado de recursos clássicos, tais como energia, materiais, 'man-power', etc., recursos não tão óbvios tais como a dupla tempo e espaço, e também recursos quase puramente abstratos. Nesta última categoria entra, por exemplo, a INFORMAÇÃO, recursos COMPUTACIONAIS, a AUSÊNCIA de entidades, fenômenos, ruídos, etc...

Ao mesmo tempo que se define o que é um recurso, deve-se também pensar para que ele serve. Isto é relativamente fácil de determinar no caso em que se trata de recursos clássicos. Já no caso de recursos abstratos, como, por exemplo, a ausência de um recurso (falta de ruído), temos que especificar, artificialmente, qual a influência, a vantagem, o objetivo em relação a outros recursos. Isto fica mais evidente ainda no caso em que se trata do recurso INFORMAÇÃO, que deve ter tratamento especial porque está sempre em algum lugar mas nunca em toda parte. Então, a informação pode ser considerada como sendo um pré-requisito para o uso de outros recursos, ou pode ser usada para resolver conflitos, ou pode ser necessária para determinar o horizonte de observação, etc... Enfim, temos que estudar um pouco mais sobre a dinâmica da informação para poder responder às questões levantadas.

Imprecisões de medidas.

Já foi mencionada que a GNT se preocupa em ser uma teoria realística. Assim ela se propôs também a poder modelar as imprecisões existentes em, por exemplo, medições (precisão do sensor limitada, conversão A/D com um número finito de bits, etc.), em transmissões (erros devidos a interferências, ruídos básicos, etc.), em cálculos (precisão simples, dupla, etc.), e em outros aspectos.

Veja como exemplo a imprecisão de medidas. Este fato ocorre não somente devido às deficiências dos métodos de medição (como mencionadas acima), mas também porque o processo de medição é um procedimento complicado de fluxo de informações influenciado pelo ruído. Assim temos não somente modelos de um determinado

nível conceitual (por exemplo, o nível "fluxo-influência", representado, na figura III.1, como sendo o nível 3), mas também os modelos de um nível adjacente (por exemplo, um que é mais "baixo" como o par "condições-transições", nível 2 na figura III.1) que, por causa de uma teoria realística, devem ser interrelacionados entre si.

Este ponto, muitas vezes negligenciado, é um dos objetivos fortes da GNT. Ela, a GNT, permite estudar sistemas de ponto de vista realístico e, conseqüentemente, também sob um aspecto mais seguro.

III.1.b.4.β - Conceitos nos níveis conceituais.

Níveis conceituais.

O problema tratado neste item é de encontrar os conceitos mais relevantes em cada um dos níveis conceituais. Aí temos que parar um instante e definir o que é um nível conceitual. Podemos, para este fim, usar uma seqüência informal de níveis proposta por PETRI (21). Estes podem ser agrupados como sendo níveis básicos, níveis relativos à ciência da computação, e níveis de algumas aplicações (fig. III.1).

Agora, a escolha dos diferentes conceitos, em cada nível, se faz de uma maneira tal que eles fiquem compreensíveis e precisos em relação aos níveis adjacentes. Devemos, então, também mencionar o problema da interconexão entre vários níveis conceituais. Como queremos usar o 'net-morphism' para aumentar ou diminuir o horizonte de observação, precisamos bem definir as diferentes categorias de grafos, os diferentes mapeamentos, as relações, etc., para poder "passar" de um nível para outro SEM modificar a estrutura, o conteúdo e o conceito do sistema em questão.

Deve-se também não esquecer o fato de que as ferramentas clássicas se tornam menos e menos úteis na medida em que subimos nesta seqüência de níveis. Assim a descrição se torna menos pre

cisa e menos explicativa o que, às vezes, é até desejado. Isto acontece, por exemplo, no nível n que trata de grandezas "vagas" tais como interesses e lucros. Isto pode, eventualmente, ser comparado com o conceito de controle hierárquico em sistemas complexos, mencionado por SCHWARZ (29).

Descrição formal.

É agora, neste item, que temos que introduzir algum formalismo para poder descrever os conceitos, descritos na figura III.1, de uma maneira formal, aplicando assim o primeiro passo de uma modelagem.

Das muitas definições que existem, apresentadas por GENRICH et alia (07), escolhemos uma que é simples e que é uma das mais usadas na literatura. Ela se baseia numa tripla formada de lugares, transições e relações, e pode ser vista na figura III.2.

Para explicitar esta definição podemos mencionar como exemplo um multígrafo direcionado, sem vértices isolados, que tem a seguinte tripla: (arcos, vértices, relação incidente). Ou como contra-exemplo, um grafo bipartite não segue esta definição, porque S e T tem papeis diferentes, e assim um 'net' (S,T;F) nem sempre é isomórfico a um 'net' (T,S;F).

Neste contexto é importante mencionar o seguinte: associação de pares de conceito (como, por exemplo, descritos na figura III.1) à dupla S e T (por exemplo, substâncias químicas sendo o S e a reação química sendo o T) levanta os seguintes pontos essenciais:

- ela, a associação, contém uma parte essencial de INTERPRETAÇÃO;
- ela é pré-requisito para a INTERCONEXÃO entre níveis adjacentes;
- ela é pré-requisito para a TRANSFERÊNCIA de conhecimentos estruturais entre ciências diferentes.

níveis	Conceitos típicos classificados informalmente em estados S; ◯	transições T; □
n	interesses, lucros, etc. (de grupos, indivíduos, ...)	restrições (naturais, legais, econômicas, ...)
n-1	canais (para recursos, mensagens, etc...) papel (representado por pessoas, artefatos, ...)	agências ou estações (instituições, escritórios, ...) atividades (relativas a cada papel)
n-2	confiabilidade banco de dados protocolos . . .	desempenho arquitetura sistemas operacionais . . .
4	palavras hardware	instruções funções
3	lugares ('places') fluxo	transferências influências
2	condições sincronismo	transições 'enlogy'
1	ocorrências	ordenação (parcial) relativa ao tempo
0	concorrência	acontecimentos

Fig. III.1: Uma possível seqüência de níveis conceituais, agrupados em nível adicional (n), de aplicação (n-1), de ciência da computação (n-2 até 4), e níveis básicos (3 até 0), PETRI (21).

Procedimento.

Tendo a descrição formal e a idéia de uma associação, podemos proceder da seguinte maneira:

- Escolher a tripla. Por exemplo, um conjunto de agências (T) se comunica (F) através de canais (S); ou, eventos (T) ocorram, seguindo regras estabelecidas (F), se certas condições (S) estão satisfeitas; ou, haverá uma reação química (T), se as substâncias químicas (S) são submetidas a certas leis de natureza (F), etc. (Veja outros exemplos na figura 10 do trabalho de PETRI (21)).

Tendo observado este procedimento, fizemos o primeiro grande passo para a modelagem. Pode-se, então, estudar certos fenômenos através do modelo associado, ou melhor, através da GNT e assim, o que provou ser uma grande vantagem adicional, compreender também outros sistemas simplesmente por causa de uma "estrutura do modelo" igual ou similar.

Uma tupla, (S,T,F) é chamado 'net', se, e somente se, S e T são conjuntos, não vazios e disjuntos, com relação F entre eles tal que cada elemento de S e T aparece na relação F:

Net (S,T,F) : $\leftrightarrow$

1.	$S \cap T = \emptyset$
2.	$S \cup T \neq \emptyset$
3.	$F \subseteq (S \times T) \cup (T \times S)$
4.	$dom(F) \cup cod(F) = S \cup T$

Fig. III.2: A forma (S,T,F) de um 'net',PETRI (21).

III.1.c - Observações finais sobre a seção III.1.

Para terminar esta seção, queremos avisar que esta GNT ainda

está em evolução e que nós, como uma tentativa, queremos estudar a validade de algumas partes desta teoria para nosso objetivo. Para alguém mais interessado na GNT em si há, no trabalho editado por BRAUER (01), algo bastante completo sobre o estado atual desta teoria em 1980 onde, ainda, pode ser vista a estrutura geral da GNT (veja isto numa parte desta publicação, em particular, na figura 14 do trabalho de PETRI (21)).

Escolhemos, então, para serem utilizados nas próximas seções deste capítulo, subtópicos que achamos úteis para nosso problema de modelagem de problemas dinâmicos em redes de computadores. São eles os grafos tipo predicado/transição, referenciado como 'PrT-Nets', tendo como base os sistemas condição/evento, chamado de sistema C/E.

III.2 - Um grafo do tipo predicado/transição ('predicate/transition-Net', 'PrT-Net').

Depois da história da GNT, apresentada na seção III.1, devemos agora, nesta seção, fazer algo para definir mais formalmente o que é um 'PrT-Net'. Usamos para este fim publicações relacionadas a Petri-Net, BRAUER (01), sistemas C/E ('condition-event-systems') e grafos do tipo P/T ('place/transition-nets'), para chegar às explicações sobre 'PrT-Nets', GENRICH et alii (06,08) e RICHTER (23).

Dispondo desta bibliografia básica e também de um "dicionário" dos termos mais importantes, editado por GENRICH et alia (07), podemos agora apresentar uma definição formal de um 'PrT-Net'. Isto será feito na seção III.2.a, servindo-se principalmente das definições dadas por GENRICH e LAUTENBACH (08) e RICHTER (23).

Como sentimos que estas definições formais, sozinhas, não são o suficiente para bem compreender o papel de um 'PrT-Net', usamos a "cadeia" de desenvolvimento (C/E, P/T e PrT), junto com alguns exemplos, para explicar melhor como se chegou a um 'PrT-Net' e o que representa este tipo de grafo. Isto permite,

depois, mostrar e especificar mais facilmente os diferentes pontos e componentes desta definição formal. Então, mostraremos, na seção III.2.b, a representação gráfica de um sistema C/E para continuar depois, na seção III.2.c, com a representação (gráfica e a matriz de incidência) de um grafo P/T.

Depois disto podemos voltar ao 'PrT-Net' e explicaremos, agora mais detalhadamente, os componentes de um 'PrT-Net' usando, sempre que for conveniente, exemplos já tirados de uma aplicação em redes de computadores (seção III.2.d).

Surge, quase automaticamente, a necessidade de explicar algo a respeito da identidade dos usuários. Falaremos sobre isto na seção III.2.e.

Também, da mesma maneira como em grafos P/T, temos em 'PrT--Nets' a necessidade de representar este "grafo" dentro de um sistema de computadores. Isto pode ser feito usando também aqui a matriz de incidência (seção III.2.f), só que, desta vez, com argumentos bem mais complexos.

Todas estas explicações sobre sistemas C/E, grafos P/T e PrT, junto com pequenos exemplos de aplicação, achamos necessário para que se possa depois, nos capítulos subseqüentes, prosseguir com a tentativa de modelagem dos problemas dinâmicos dentro de redes de computadores.

III.2.a - Definição formal de um 'PrT-Net'.

Já mencionado várias vezes, chegou agora o momento de apresentar a definição formal de um 'PrT-Net'. Esta definição é baseada naquelas de GENRICH e LAUTENBACH (08) e RICHTER (23).

Definição: Um grafo tipo predicado/transição ('PrT-Net', 'predicate/transition-net') consiste dos seguintes elementos essenciais:

1) Um grafo direcionado, $N = (S,T;F)$, onde

- S representa o conjunto de predicados;
- T é o conjunto de transições (esquema de acontecimentos);
- F é a relação causal entre estes dois conjuntos.

2) Uma estrutura matemática Σ, consistindo de tipos de indivíduos (objetos abstratos) em conjunto com os respectivos operadores e relações.

3) Uma inscrição nos elementos S, incluindo:
   - Um limite superior de multiplicidade ('upper limit of multiplicity', ulm);
   - A informação de quantos argumentos o elemento S (o predicado) tem;
   - Para cada um destes argumentos, o domínio de validade (isto é, o tipo dentro da estrutura matemática Σ ao que foi associado o objeto individual deste argumento).

4) Uma inscrição nos arcos com expressões para indicar operações em relação às tuplas de variáveis.

5) Uma inscrição dos elementos T (as transições) usando expressões lógicas baseadas em variáveis e na estrutura matemática Σ.

6) Uma regra de transições para 'PrT-Nets'.

RICHTER (23) fez uma distinção entre 'PrT-Nets' e 'PrT--Systems'. Esta discriminação é baseada no fato que este tipo de grafos ('Netze') representa, intuitivamente, alguma estrutura básica, estática ou de suporte, e que sistemas representam mais um determinado "estado", ou até uma seqüência de estados (≙ dinâmica de um sistema), encima desta estrutura. Assim, dependendo de uma determinada situação inicial, pode-se, baseado no mesmo suporte ou estrutura, criar até diferentes sistemas.

Assim sendo, pode-se então definir:

Um sistema PrT ('PrT-system') é um 'PrT-Net' que contém:

7) Uma marcação inicial M_o nos elementos S (predicados) com operações relativas às tuplas de indivíduos (dentro de seus domínios pré-definidos).

Como já mencionado no início desta seção III.2, queremos agora mostrar como se chegou a estas definições para depois explicitar melhor, junto com exemplos aplicativos, estas definições formais.

III.2.b - Explicação de um sistema do tipo C/E através do exemplo R/W ('reader/writer').

Sendo isto um exemplo bem elaborado por GENRICH e LAUTENBACH (08) e comumente usado na literatura, decidimos apresentar este sistema R/W neste trabalho; esta decisão é ainda mais justificável, porque mostra muito bem o compromisso a ser feito entre:

- DIMINUIÇÃO da complexidade na estrutura da rede,
- MAIOR COMPLEXIDADE nas inscrições, ou seja, nos itens.

Este segundo ponto é o "preço" pago para poder trabalhar com uma estrutura simplificada.

Vamos então explicar, bastante detalhadamente, este modelo e convém se lembrar que esta seção é um complemento para compreender como se chegou às definições formais dadas na seção III. .2.a.

III.2.b.1 - Representação gráfica de um sistema tipo C/E ('condition/event-system')

Seja inicialmente algum sistema que pode ser representado por elementos S (lugares) e T (transições), ligados por flechas que indicam a relação entre estes elementos. Podemos ainda dizer que o sistema consiste de dois subsistemas que têm uma estrutura e que têm alguma interferência mútua. Observe esta estrutura na figura III.3.

Observando a figura III.3 podemos ainda complementá-la com duas outras definições úteis: os 'tokens' presentes nos predicados Hi, Hl e R são pré-condições para as transições *1*i, *1*1, 2i e 21, respectivamente. Vemos ainda que as transições *1*i e *1*1 podem ocorrer simultaneamente ou concorrentemente, já que não têm pré- ou pós-condições em comum. Ao contrário disto, transições 2i e 21 estão em conflito, o que quer dizer, que somente uma delas pode acontecer.

Um exemplo de possíveis transições, a partir do estado mostrado na figura III.3, pode ser visto na figura III.4. Convém se lembrando que o 'TOKEN' em sistemas C/E representa somente a satisfação de uma condição e que não tem nenhum atributo quantitativo.

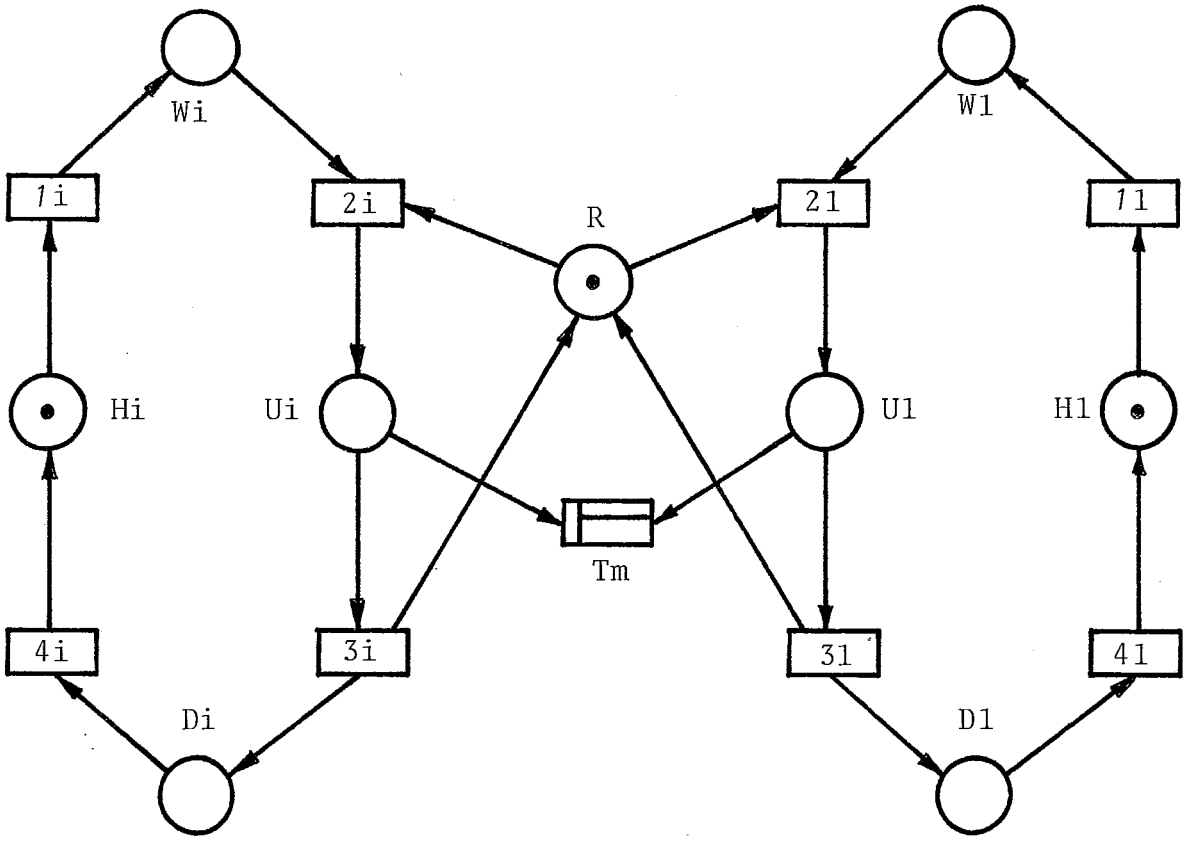
Interessante ainda é a observação da transição representada pelo símbolo [⊟] . Este tipo de transição NUNCA tem a chance de acontecer, isto é, ela significa uma transição "morta", ou, como chamada na literatura, um 'fact' (ao contrário das transições que podem ocorrer e que são chamadas de eventos).

A utilidade destes 'facts' é que eles podem representar estados imagináveis que em realidade representam estados teoricamente impossíveis de serem alcançados. Em nosso caso, o símbolo [⊟] representa a expressão matemática relativa aos predicados Ui e Ul: $\neg$ (Ui $\wedge$ Ul), e é uma afirmação constante ('invariant assertion') que expressa a exclusão mútua das condições Ui e Ul.

Temos então, para o uso em sistemas C/E, duas classes de estruturas lógicas, a de eventos e outra a de transição morta, que classificam TODOS os estados imagináveis.

III.2.c - Explicação de um grafo do tipo P/T ('place/transition-net').

Tomando a mesma estrutura da figura III.3, podemos transformá-la numa representação de um grafo do tipo P/T, ou grafo ins-



onde:

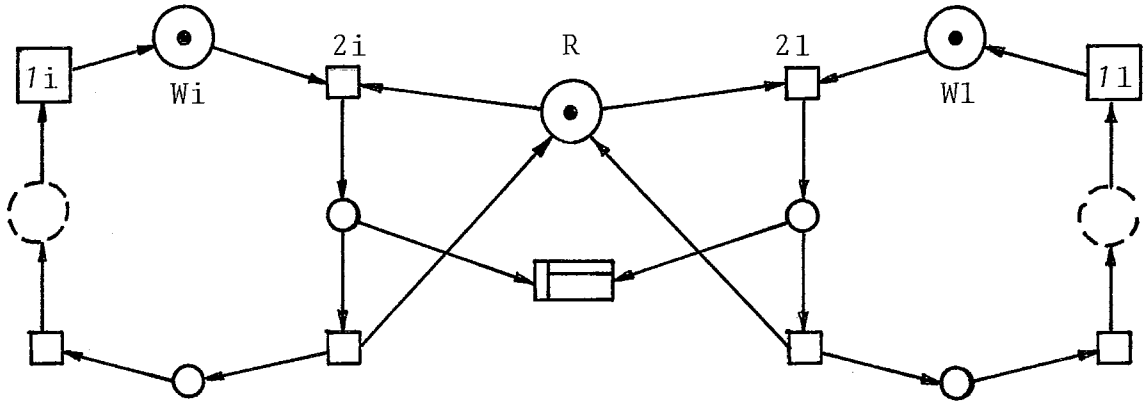
S = {Hi,Wi,Ui,Di;H1,W1,U1,D1;R}, conjunto de predicados;

T = {1i,2i,3i,4i;11,21,31,41}, conjunto de transições;

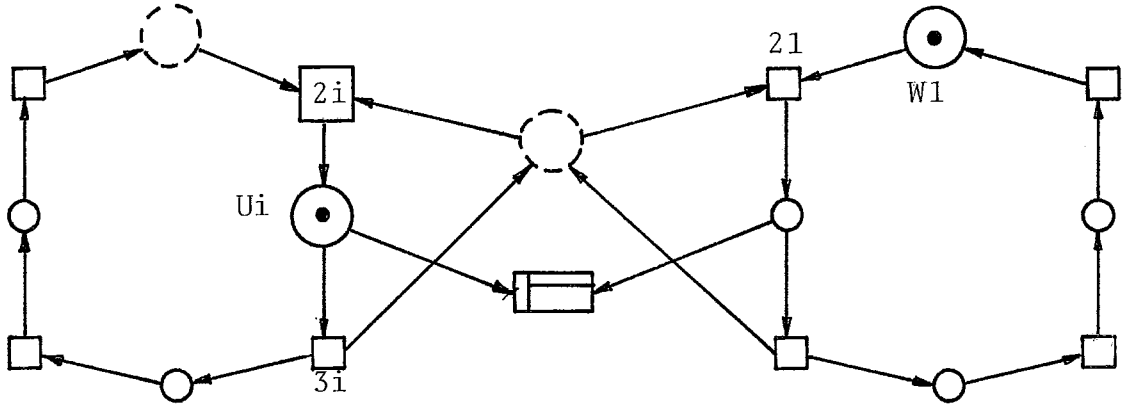
F = conjunto de arcos, representando a relação entre os elementos de S e T;

• = 'token', indicando a satisfação de uma condição, presente em algum dos predicados.

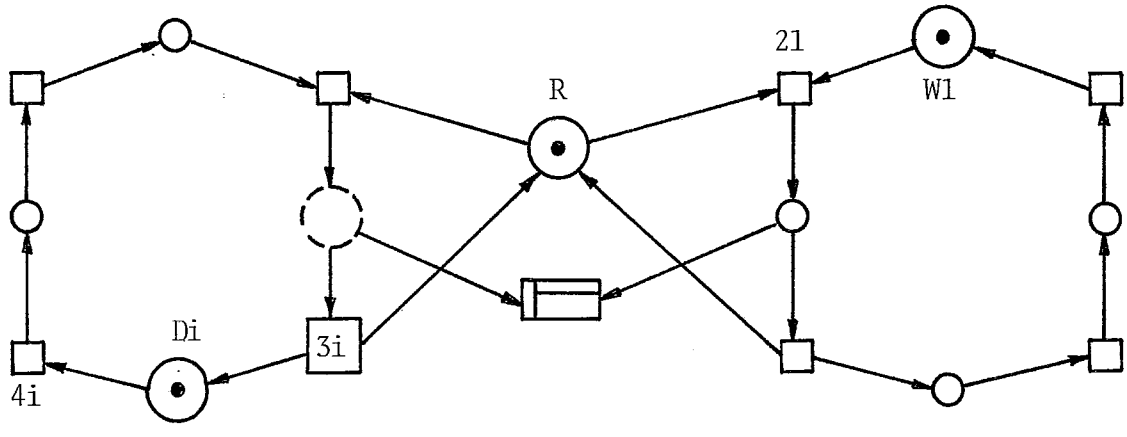
Fig. III.3: Representação gráfica de um 'net' tipo C/E ('condition/event-system'), GENRICH et alia (08), com rotulação (identificação) dos elementos dos conjuntos S e T, e indicação de uma transição morta (Tm).



(a) Vistas as pré-condições na fig. III.3, as transições 1i e 11 podem ocorrer concorrentemente. Feitas estas transições, temos um novo conjunto de pré-condições, desta vez para as transições 2i e 21.



(b) Visto o conflito entre 2i e 21, foi dada, segundo alguma regra, vantagem à transição 2i. Há, depois das transições, um novo conjunto de pré-condições para 3i e 21.



(c) Depois da transição 3i, vemos que se formou um conjunto de pré-condições para a transição 4i e, de novo, para a transição 21 que, sendo finalmente sem conflito, pode ocorrer simultaneamente com 4i.

Fig. III.4: Uma possível seqüência de estados para a estrutura da fig. III.3, representando um sistema C/E.

crito, simplesmente usando atributos quantitativos.

III.2.c.1 - Representação gráfica de um grafo do tipo P/T.

A diferença fundamental é que um lugar pode ter mais que um 'token' (veja R na fig. III.5), representando assim quantidades inteiras, não-negativas e não mais somente a satisfação de alguma condição. Adicionalmente pode-se usar alguma indicação relativa à multiplicidade máxima de um predicado (veja Hl na fig. III.5), forçando assim à obediência a regras estritas de transição (no caso em que não há multiplicidades associadas, a regra de transição é chamada de fraca).

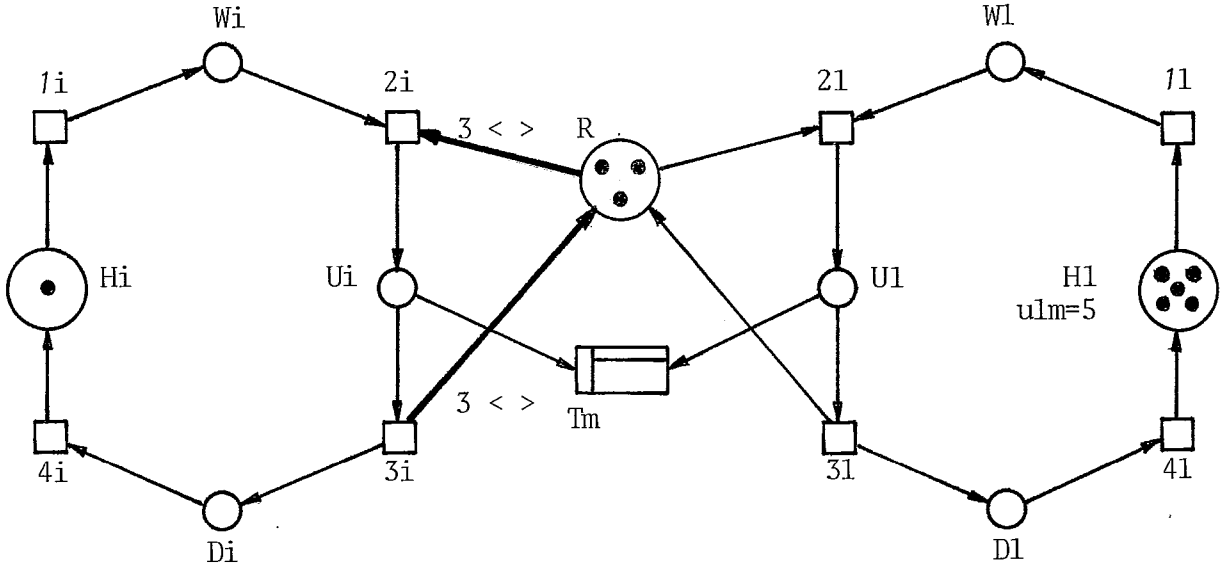


Fig. III.5: Sistema simples de um conjunto de leitoras, impressoras e recursos, onde os 3 'tokens' no predicado R indicam o uso simultâneo por 3 componentes (ver as siglas na fig. III.3).

Outra diferença é que mais que um 'token' pode ser removido de um lugar; isto é indicado por inteiros, rotulando assim os arcos relativos ao coeficiente ou "peso" (os arcos entre R-2i e entre 3i-R na fig. III.5).

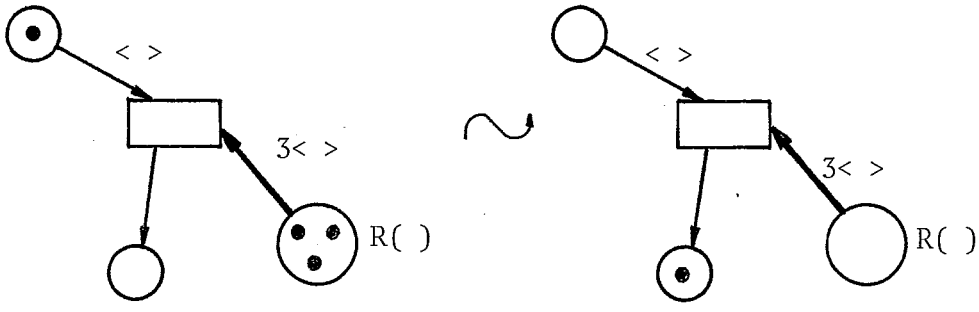
Interpretando a fig. III.5, podemos ver que temos agora cinco leitoras (Hl) que usam, de modo compartilhado, algum recurso e que temos também uma impressora (Hi) que pode usar, em modo exclusivo, este mesmo recurso. A presença de três 'tokens' em R

indica que três componentes podem usar este recurso simultaneamente.

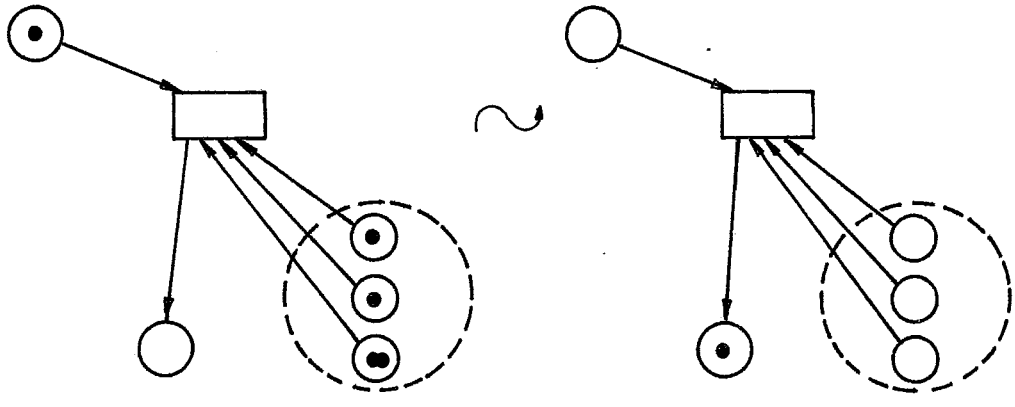
É importante rotular bem os arcos, no que diz respeito a seus coeficientes (pesos), e os predicados, em relação à multiplicidade (ulm), para garantir os modos compartilhado e exclusivo, e ao mesmo tempo evitar que a transição morta possa ocorrer.

Veja a rotulação de um arco na fig. III.6 e uma possível seqüência de estados na figura III.7.

A representação:



significa:



onde: a multiplicidade de 3 arcos é equivalente à ligação de "três" predicados a mesma transição.

Fig. III.6: O significado da rotulação de um arco.

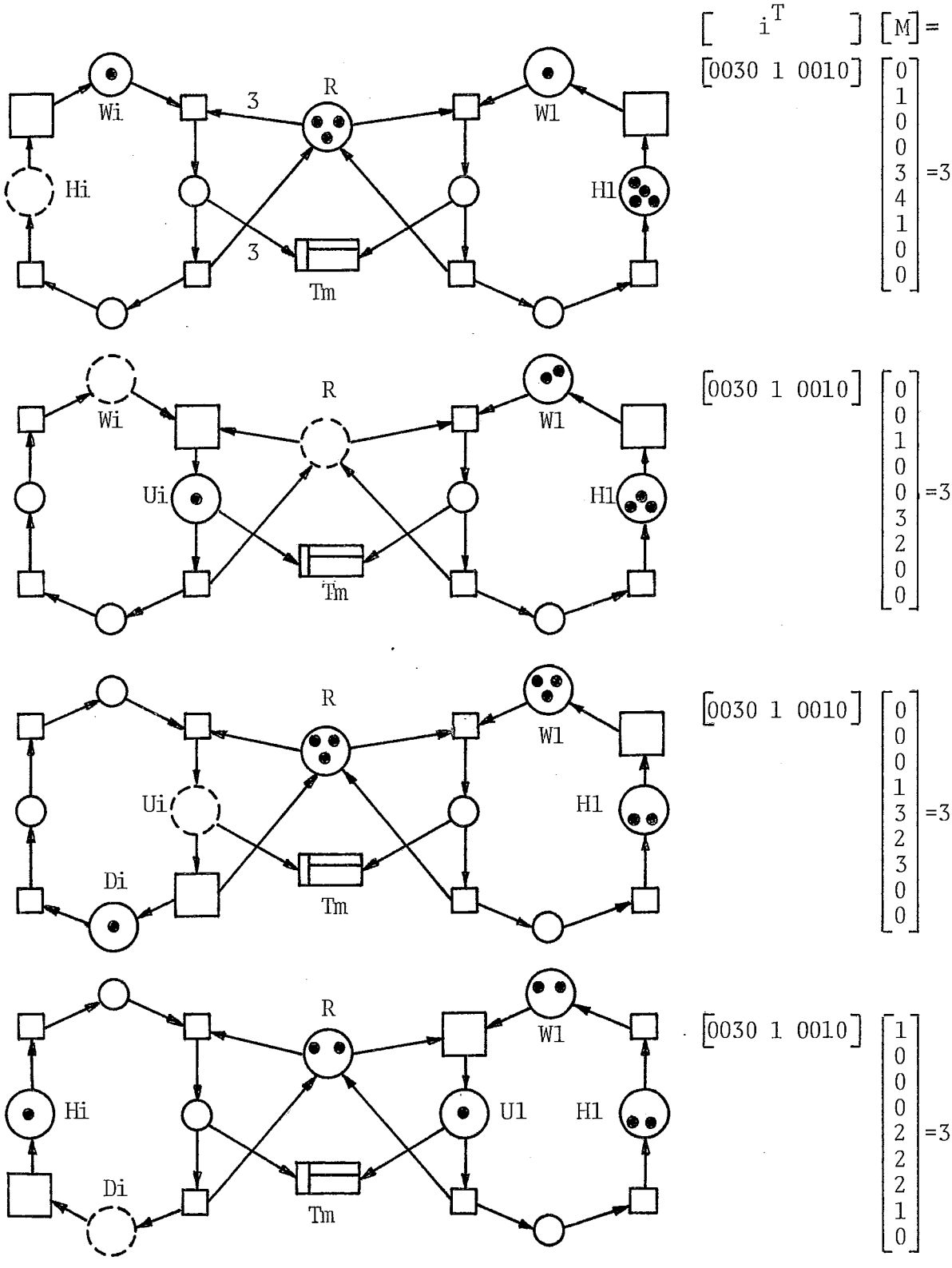


Fig. III.7: Uma possível seqüência de estados (dada a estrutura da fig. III.5), com garantia de uso exclusivo do recurso (R) pela impressora (Hi) e do não-acontecimento de uma transição morta (Tm). Também mostrado o teste "produto interno" (veja a seção III.2.c.2). As siglas usadas são as mesmas que na fig. III.3.

III.2.c.2 - A matriz de incidência para grafos do tipo P/T.

No momento em que permitimos multiplicidades diferentes nos diversos predicados, devemos também assegurar que estas multiplicidades (na literatura às vezes chamadas de capacidades) associadas são consistentes com a marcação inicial. Neste caso poderíamos dizer que o sistema é seguro com respeito às multiplicidades.

Para garantir isto, ou melhor, para testar isto, pode-se usar o método de S-invariantes, que é baseado na matriz de incidência C de um grafo P/T. Vejamos primeiro, na fig. III.8.a, a matriz de incidência de estrutura mostrada na fig. III.5. Junto com esta matriz mostramos o vetor da marcação inicial, M_o, e também um vetor i de polinômios, que é chamada a S-invariante do grafo P/T no caso em que garante $C^T.i = 0$. Isto quer dizer, que a combinação linear de linhas da matriz de incidência, C, usando os coeficientes correspondentes do vetor i, resulta na linha zero.

O exemplo em nosso caso (veja a figura III.8.b) mostra isto.

$$C^T.i = \left[\begin{array}{cccc|c|cccc} -1 & 1 & 0 & 0 & 0 & & & & \\ 0 & -1 & 1 & 0 & -3 & & & & \\ 0 & 0 & -1 & 1 & 3 & & \multicolumn{4}{c}{\emptyset} & \\ 1 & 0 & 0 & -1 & 0 & & & & \\ \hline & & & & 0 & -1 & 1 & 0 & 0 \\ & & & & -1 & 0 & -1 & 1 & 0 \\ & & \multicolumn{2}{c}{\emptyset} & 1 & 0 & 0 & -1 & 1 \\ & & & & 0 & 1 & 0 & 0 & -1 \end{array}\right] \times \left[\begin{array}{c} 0 \\ 0 \\ 3 \\ 0 \\ \hline 1 \\ \hline 0 \\ 0 \\ 1 \\ 0 \end{array}\right] = \emptyset$$

Fig. III.8.b: Exemplo do teste da consistência da marcação usando a matriz de incidência e uma S-invariante.

Outro teste é relacionado às marcações, a saber, que o produto interno de um vetor M pelo vetor i deve ser igual ao produto interno do vetor das marcações iniciais, M_o, por este mesmo vetor i. Temos então a relação

$$i^T.M = i^T.M_o \qquad \text{(III.1)}$$

que, aplicada à estrutura da fig. III.5 (para a marcação inicial) e, por exemplo, para a marcação da última seqüência mostrada na fig. III.7, dará o mesmo resultado, isto é, 3.

Com estes dois testes, $C^T.i = \emptyset$ e $i^T.M = i^T.M_o$, podemos verificar se uma possível seqüência, derivada através de simulações, está correta.

	1i	2i	3i	4i	11	21	31	41	M_o	i
Hi	-1			1					1	0
Wi	1	-1								0
Ui		1	-1							3
Di			1	-1						0
R		-3	3			-1	1		3	1
H1					-1			1	5	0
W1					1	-1				0
U1						1	-1			1
D1							1	-1		0

onde:

n ... indica a multiplicidade (peso) de um arco, partindo de uma transição (coluna) t até um predicado (linha) s dos conjuntos T e S, respectivamente;
-n ... indica a multiplicidade no sentido inverso;
$\emptyset$ ... indica, que não há conexão entre eles (os zeros estão, nesta figura, suprimidos).

Fig. III.8.a: Matriz de incidência junto com o vetor da marcação inicial M_o e um vetor i ($\hat{=}$ S-invariante) da estrutura mostrada na fig. III.5.

III.2.d - Explicações detalhadas dos componentes de um 'PrT-Net'

Achamos que as explicações de sistemas C/E e de grafos P/T fornecem agora uma base suficientemente clara para que possamos explicar, detalhadamente, os componentes de um 'PrT-Net', enumerados, de uma maneira formal, na seção III.2.a.

III.2.d.1 - Um grafo direcionado N = (S,T;F).

Este é o elemento básico de um 'PrT-Net', fornecendo a base para a aplicação e utilização de outros conceitos adicionais.

Podemos então dizer, como já vimos, de uma maneira informal, na fig. III.2, o seguinte:

A tripla, N = (S,T;F), é chamado de grafo direcionado ('directed net') se, e somente se,

$$\left.\begin{array}{l} S \cap T = \emptyset \\ S \cup T \neq \emptyset \\ F \subseteq (S \times T) \cup (T \times S) \\ \mathrm{dom}\,(F) \cup \mathrm{cod}\,(F) = S \cup T \end{array}\right\} \quad \text{(III.2)}$$

e onde os elementos têm os seguintes significados:

S = $\{s_1, s_2, \ldots\}$; é o conjunto de predicados ($\hat{=}$ lugares de 1ª ordem, 'first-order places'), usando o símbolo circular, ◯ , para representações gráficas;

T = $\{t_1, t_2, \ldots\}$; é o conjunto de transições (de 1ª ordem), indicadas pelo símbolo retangular, ▭ ;

F = $\{f_1, f_2, \ldots\}$; é o conjunto de arcos de N ou a relação causal de fluxo de informações ('flow relation'), indicado por flechas, → .

Agora, observando o dicionário de conceitos básicos da GNT, compilado por GENRICH et alia (07), vemos que existem definições adicionais relativas ao fluxo, às fontes, às destinações, às

condições, etc...

Tendo em vista 'PrT-Nets', adicionamos, então, ainda os seguintes pontos, mencionados por GENRICH e LAUTENBACH (08):

Para um dado grafo, $N = (S,T,F)$, chamamos:

$X := S \cup T$ o conjunto de (S ou T)-elementos do grafo N; (III.3)

Tendo as definições das eq(III.2) e (III.3), podemos dizer ainda o seguinte:

Para um elemento $x \in X$, temos:

$$\left.\begin{array}{l} .x := \{y \mid (y,x) \in F\} \text{ , chamado de pré-conjunto de x;} \\ x. := \{y \mid (x,y) \in F\} \text{ , chamado de pós-conjunto de x.} \end{array}\right\} \quad \text{(III.4)}$$

Estas duas últimas definições estão intimamente ligadas à possibilidade de ocorrência, ou não, de uma transição e, conseqüentemente, serão tratadas mais detalhadamente no decorrer deste texto.

III.2.d.2 - Uma estrutura matemática Σ.

Esta estrutura é constituída de alguns tipos de indivíduos (objetos abstratos), junto com alguns operadores e relações. Formou-se, então, um conjunto estruturado da seguinte forma:

$$\left.\begin{array}{l} \Sigma = (S \ ; \ 0 \ ; \ R) \\ \quad = (S \ ; \ op_1, \ op_2, \ \ldots, \ op_m \ ; \ R_1, \ R_2, \ \ldots, \ R_n) \end{array}\right\} \quad \text{(III.5)}$$

onde:

S .... objetos abstratos, p. ex.: S_1, S_2, S_3, IN, GH, ...

op_i .... operadores, p.ex.: + - * / ...

R_i .... relações, p. ex.: = > < $\leq$ ...

III.2.d.3 - As inscrições em relação aos predicados S.

Seguindo as definições dadas por RICHTER (23), pode-se dizer que "predicado" é um lugar onde se encontram determinados objetos (p.ex., objetos de informação). Os objetos que estão presentes, num dado momento, neste predicado formam a extensão do predicado. Assim temos o fato que S-elementos em 'PrT-Nets' são predicados com extensão variável.

Podemos então, tendo em vista diferentes conjuntos de objetos, definir os predicados de uma maneira tal que eles representem exatamente nossos objetivos de modelagem. Assim teremos predicados com um, dois, três, etc ..., conjuntos diferentes de objetos, também chamados de predicados de grau n (RICHTER (23) chamou estes predicados de 'n-stellige Praedikate').

Veja os exemplos mostrados na fig. III.9, que podem ser explicados da seguinte maneira:

MAN(m,g) ..... Predicado de grau 2 que representa, segundo RICHTER (24) o fato que "tem uma notícia sobre mudanças no mercado de trabalho do tipo m (m ε Mart) em relação ao objeto g(g ε Geg)".

CAMINHO(s,r,fc) .. Predicado de grau 3 que representa, segundo SCHWARZ (35), a informação sobre o feixe de todos os caminhos possíveis ('Wegbuendel') entre quaisquer usuários s e r (s ε Us, r ε Ur, r ≠ s)".

W(u,m) ..... Predicado de grau 2 indicando que o "usuário u (u ε U) deseja usar um determinado recurso em modo m (m ε M), p.ex. em modo exclusivo ou compartilhado", GENRICH et alia (08).

Existem, ainda, predicados de grau UM, como, por exemplo, no caso:

AAK(k) ..... Indica, como disse RICHTER (23), que "um ope

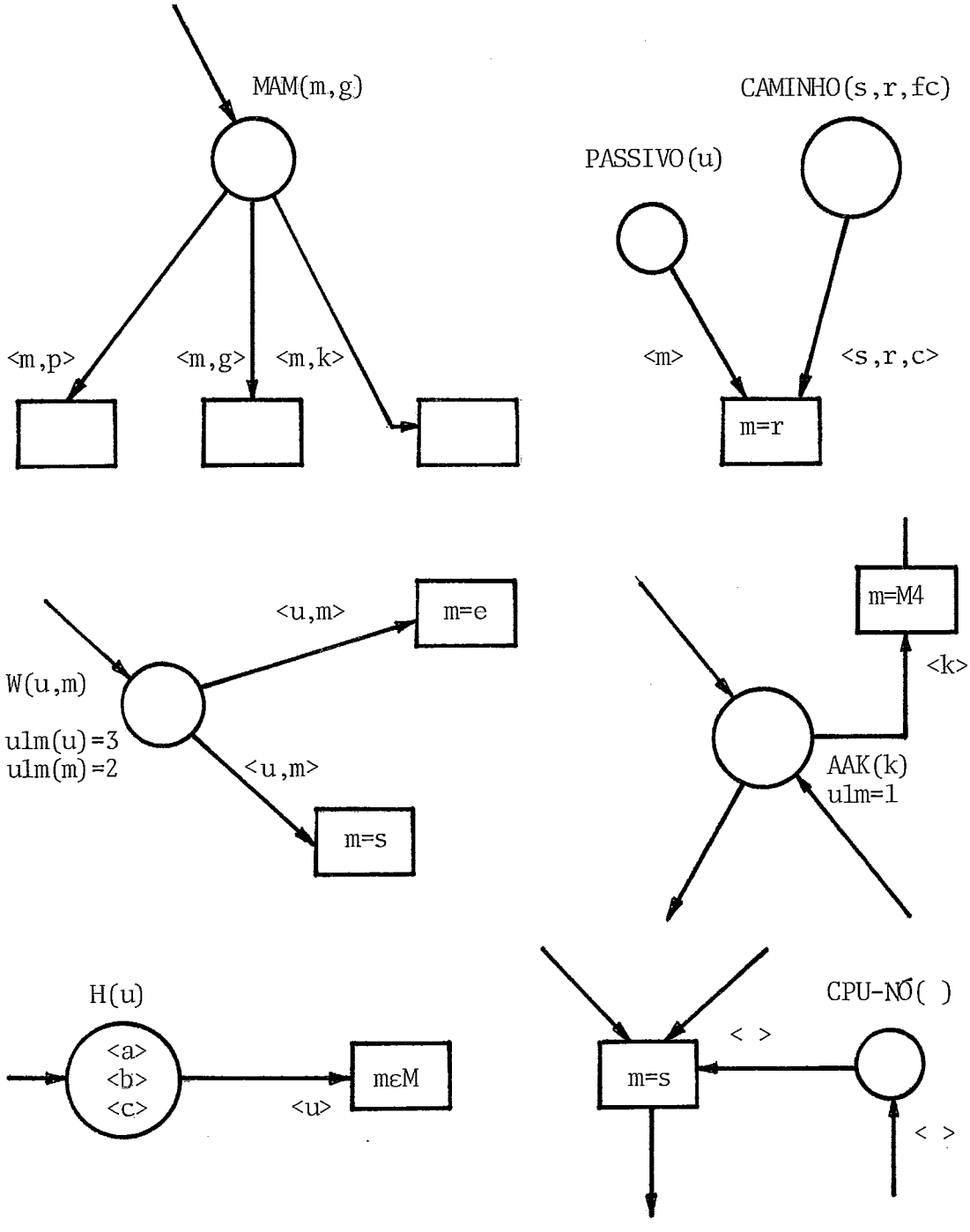


Fig. III.9: Exemplos de representação gráfica de predicados com os associados graus e limitações, oriundas dos trabalhos de RICHTER (23), GENRICH e LAUTENBACH (08) e SCHWARZ (35).

rário k (k ε Akr) está registrado como estando disponível para aceitar um trabalho'.

H(u) ..... Um usuário u (u ε U) está presente no sistema.

Assim como temos predicado de grau n, existem também predicados de grau zero. Estes indicam somente uma situação sem, portanto, entrar em detalhes desta situação. Então, no caso em que se quer indicar, por exemplo, que algum processamento foi feito, pode-se usar a seguinte notação:

CPU-NÓ( ) ..... Predicado de grau zero; sendo isto o equivalente aos 'tokens' em Petri-Nets que também indicam somente condições sem entrar na forma destas condições.

Lembrando a introdução (seção III.1), sabemos que queremos modelar sistemas físicos e reais que estão, portanto, sujeitos às restrições relacionadas a determinadas situações físicas (p. ex., limitações do espaço na memória de um computador, limitação da capacidade de uma linha de transmissão, número restrito de caminhos entre dois pontos numa rede, etc...).

Temos então um limite superior de multiplicidade ('upper limit of multiplicity', ulm) para cada um dos conjuntos de objetos num predicado. Isto quer dizer que, por exemplo, no caso de um predicado de grau três, temos três diferentes ulm's, um para cada conjunto, por exemplo:

CAMINHO(s,r,fc) .. com ulm(s)=x, ulm(r)=y, ulm(fc)=z; isto pode indicar que somente um determinado número (x) de usuários tem o direito de emitir para um número restrito de receptores (y), só permitindo o uso dos z-melhores caminhos.

Assim pode-se explicar, eventualmente, o uso da palavra "capacidade" na literatura que, no caso de predicados de grau

um, significa o equivalente a ulm, mas que no caso de predicados com um grau maior que um deveria ser relacionado a todos ulm's. Como limite superior "natural" pode ser considerada a cardinalidade de produtos dos ulm's de cada um dos conjuntos num predicado (por exemplo, no caso do predicado CAMINHO(s,r,fc), temos |ulm(s) x ulm(r) x ulm(fc)|).

II.2.d.4 - As inscrições nos arcos.

A inscrição dos arcos em 'PrT-Nets' (não necessariamente o equivalente à rotulação de arcos em grafos) permite, lembrando que os f_i's representam a relação causal entre elementos S e T, a indicação de como, p. ex., um determinado usuário quer usar um determinado recurso.

A inscrição consiste então de alguma expressão denotando uma operação sobre as n-tuplas de variáveis, onde n indica -aridade ('-arity') ou grau de um predicado ('n-stelliges Praedikat') conectado a este arco.

A tupla "zero" indica um predicado sem argumento (um lugar ou 'place' comum) e a notação usada para representar este fato, no predicado e no arco adjacente, é o símbolo especial <>, compatível com o formato usado para outras inscrições, como p. ex. <n,m> ou <s,t,r>, etc. (obs.: antigamente se usava, em vez do símbolo <>, o 'token' (•), no predicado e outro símbolo especial, ¢, no arco adjacente).

Veja alguns exemplos na fig. III.10 e observe, ainda, o seguinte: como as variáveis somente têm validade em torno de uma transição, é desejável colocar, numa representação gráfica, a descrição das variáveis "perto" da transição em questão.

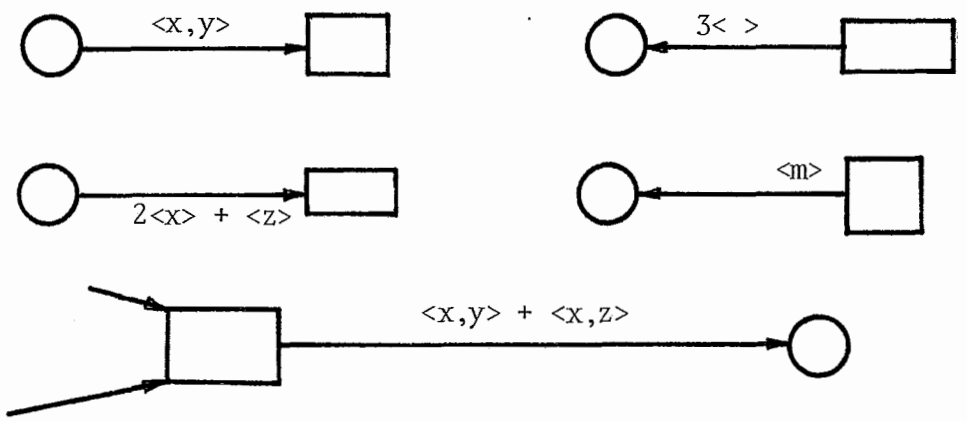


Fig. III.10: Exemplos de possíveis inscrições de arcos.

III.2.d.5 - As inscrições nas transições.

Uma inscrição, que pode existir em algumas das transições, representa uma fórmula lógica usando para este fim as operações e relações da estrutura matemática Σ, descrita na seção III.2.d.2.

As variáveis, que se encontram "livres" numa destas fórmulas de transição, têm que estar presentes, obrigatoriamente, numa aresta adjacente (fig. III.11(a)).

Se uma fórmula num retângulo (isto é, numa transição) tem a forma $v = t \wedge \ldots$ (onde v representa uma variável e t um termo), todas as ocorrências da variável v nos arcos adjacentes a esta transição podem ser substituídas por cópias do termo t (veja isto na fig. III.11(b)).

III.2.d.6 - Regras de transição para 'PrT-Nets'.

Cada elemento do conjunto de transições, T, representa uma classe de mudanças, possíveis e indivisíveis, das marcações relativas aos predicados adjacentes. Tais mudanças consistem em movimentações de cópias de itens, a saber, a remoção de itens de lugares (○→□) e a adição de itens a lugares (□→○), sempre de acordo com as expressões indicadas nos arcos.

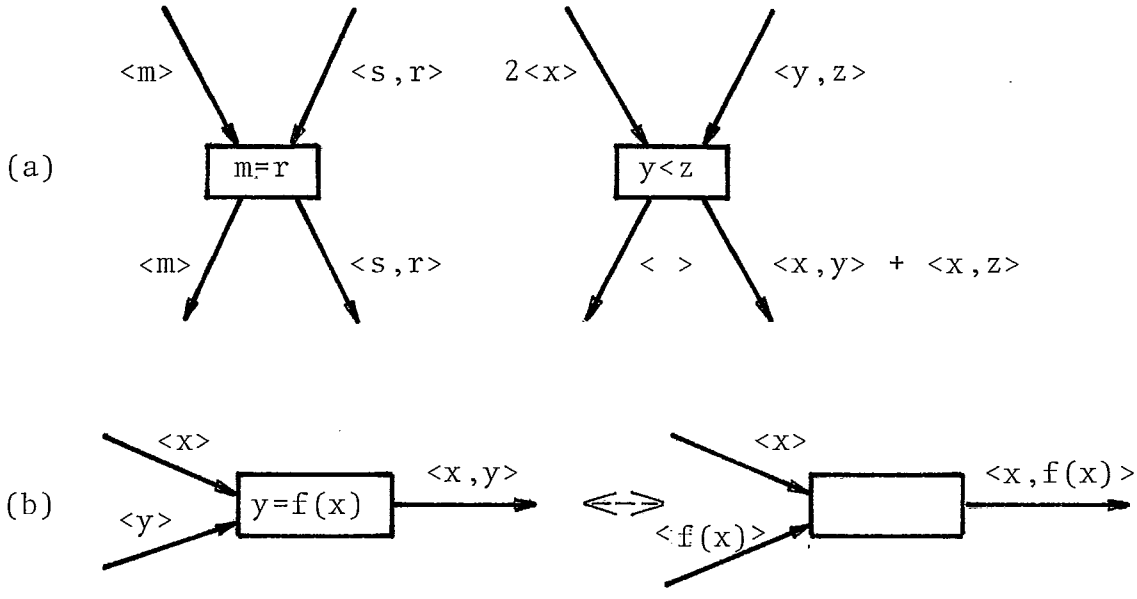


Fig. III.11: Exemplos de possíveis inscrições nas transições, destacando a inscrição obrigatória nas arestas adjacentes (a), e a possível substituição de termos (b).

Agora, para uma possível associação de indivíduos às variáveis que satisfazem a fórmula inscrita na transição, deveria acontecer que todos os predicados da entrada contivessem bastante cópias dos itens em questão, e que os predicados da saída não tivessem a sua capacidade ultrapassada quando se adicionam os mencionados itens. Mostraremos um exemplo disto, utilizando o símbolo ⌒ para expressar a regra de transição.

Exemplo (fig. III.12), tirado do trabalho de GENRICH e LAUTENBACH (08):

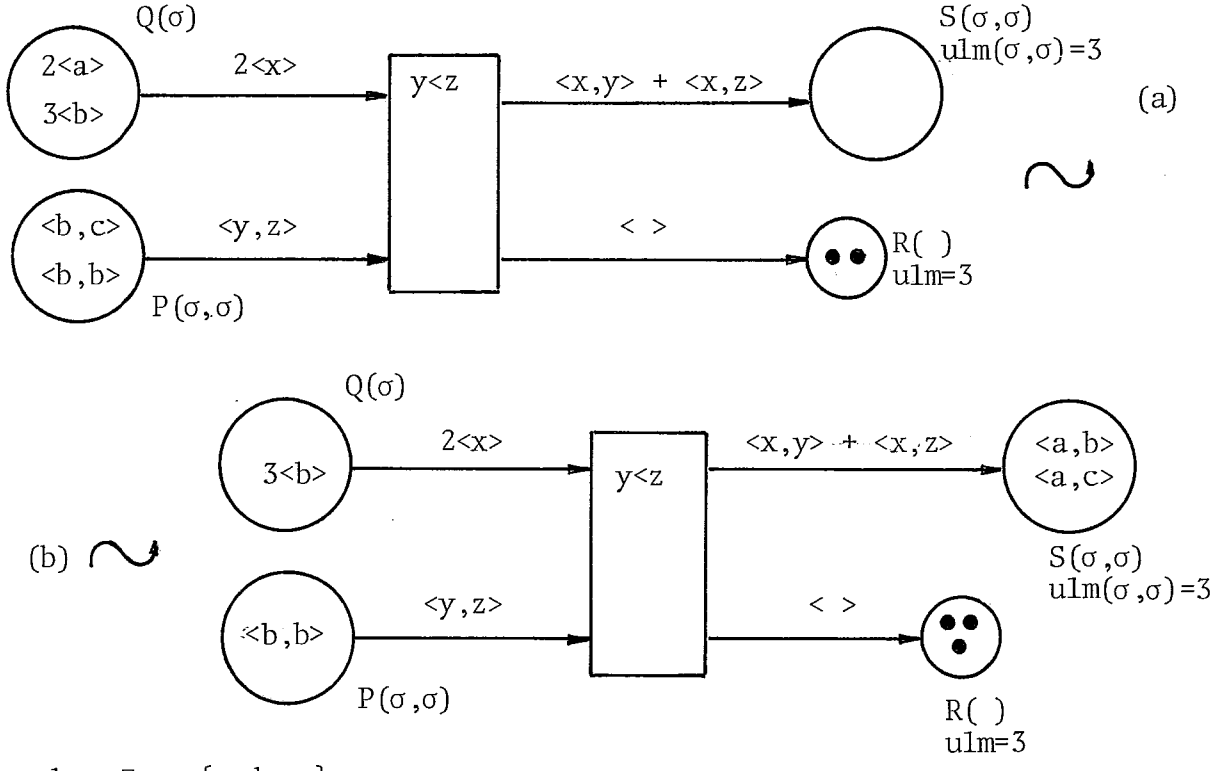
Seja dada uma estrutura ({s,b,c} ; < := ordem alfabética); Haverá uma multiplicidade dos conjuntos nos predicados de saída, no valor de ulm = 3.

Podem-se, então, ignorando por enquanto as marcações iniciais da figura III.12, imaginar as mais variadas associações das variáveis x, y, z aos indivíduos a, b, c. Para limitar um pouco as possibilidades podemos observar, já, a ordem alfabética den-

tro das tuplas com 2 elementos. Assim, combinando as tuplas com um elemento com aqueles de dois elementos, obtemos:

no.	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
x	a	a	a	a	a	a	b	b	b	b	b	b	c	c	c	c	c	c
y	a	a	a	b	b	c	a	a	a	b	b	c	a	a	a	b	b	c
z	a	b	c	b	c	c	a	b	c	b	c	c	a	b	c	b	c	c

Agora, aplicando um pouco mais de restrições, ou seja, considerando, além das inscrições nos arcos, também a inscrição na transição (y < z), vemos que somente nove destas combinações (ou seja, 2, e, 5, 8, 9, 11, 14, 15 e 17) satisfazem as inscrições e a fórmula sem, contudo, respeitar ainda as marcações iniciais. Se consideramos estas, ficamos com somente duas opções, a saber, 5 e 11.



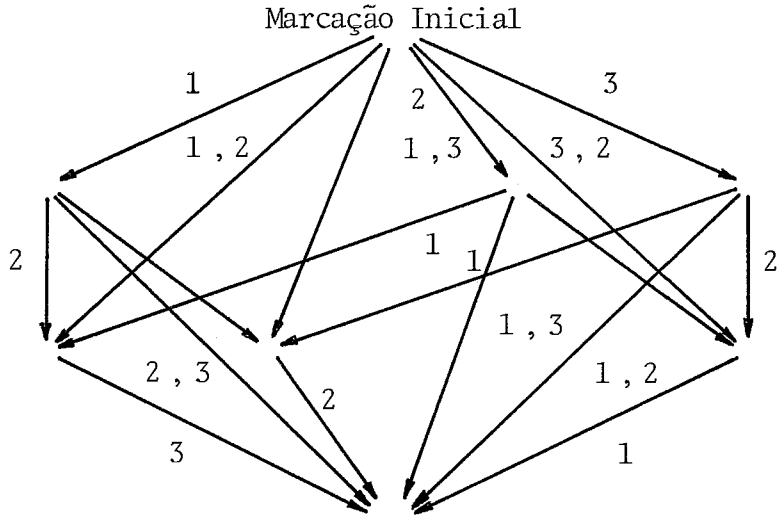
onde: Σ : {a,b,c} = σ

< := ordenação alfabética

Fig. III.12: Exemplo, tirado do trabalho de GENRICH et alia (08) de uma possível transição, mostrando o resultado (b) a partir de uma marcação inicial (a).

Este resultado teria surgido, também, se, desde o início, tivéssemos considerado as marcações iniciais, (RICHTER (25)), o que daria quatro possibilidades, ou seja, 4, 5, 10 e 11, e destas, aplicando a fórmula y < z, ficariam, novamente, somente as opções 5 e 11.

Criou-se, então, um conflito entre estas duas possibilidades, 5 e 11, e, por alguma decisão, desconhecida por nós, se obtém a marcação mostrada na figura III.12.



onde: Um vértice representa um caso (marcação de todos os elementos S);
Um arco direcionado representa um ou mais eventos (acontecimentos em paralelo estão separados por uma vírgula).

Fig. III.13: Exemplo de possível seqüência de passos entre uma marcação inicial e final, usando um grafo de casos, apresentado por RICHTER (23).

III.2.d.7 - A marcação inicial M_o.

Já foram mencionadas várias vezes as palavras marcação, marcação inicial e marcação subseqüente; cabe agora aqui explicar melhor o que foi dito na definição de um sistema PrT o que é bem diferente de um 'PrT-Net' (vide seção III.2.a).

Uma marcação nos predicados de um 'PrT-Net' representa um determinado estado dentro de uma possível seqüência. Este modelo de uma situação, isto é, a representação por uma marcação associada, se chama em NT ('Netztheorie') um caso ('case' (ingl.) ou 'Fall' (alem.)). Agora, dependendo dos instantes da observação, podemos definir uma marcação inicial (M_0) ou final. A marcação final (estado objeto) pode-se obter, a partir de um dado M_0, num número determinado de passos. No exemplo, mostrado na fig. III.12, obtemos uma marcação "final" (fig. III.12(b)), num único passo.

Esta seqüência de passos entre um M_0 e a marcação objetiva pode ser visualizada com a ajuda de um grafo de casos 'Fallgraph') e, podemos mostrar um exemplo disto na fig. III.13, conforme trabalho de RICHTER (23).

III.2.e - A identidade de vários usuários.

III.2.e.1 - Representação de uma maneira desdobrada.

Vimos, na fig. III.5, onde estão indicadas, entre outros, cinco leitoras, que o esquema utilizado não permite a identificação de uma leitora particular. Para garantir isto, podemos, por exemplo, representar a parte relativa às leitoras de uma maneira desdobrada ('unfolded'). Veja esta tentativa na fig. III.14.

Como este tipo de representação não é muito prático, foi inventado um sistema que tenta, através de inscrições nos predicados, transições e arcos, guardar a identidade dos usuários e, ao mesmo tempo, apresentar uma estrutura não grande demais. Vejamos isto na seção III.2.e.2.

III.2.e.2 - O uso de inscrições.

Mostraremos, nesta seção, como usar as inscrições para garantir a identidade dos usuários e, ainda, como determinadas inscrições podem indicar o modo de usar recursos.

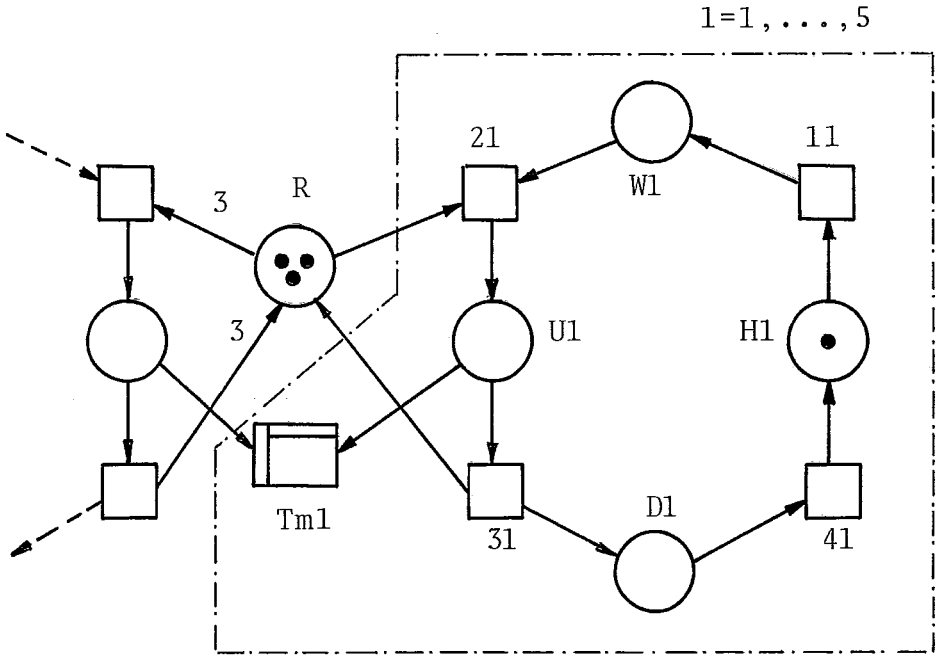


Fig. III.14: Representação de parte das leitoras da fig. III.5 de uma maneira particular, garantindo a identidade das leitoras, GENRICH et alia (08).

III.2.e.2.α - O sistema e a estrutura original.

Temos uma comunidade de usuários de algum recurso, denotados por:

$$U = \{a,b,c\}.$$

Cada um destes usuários pode usar o recurso de algum modo permitido (isto é, compartilhado ou exclusivo), sendo o limite superior para o uso compartilhado também indicado. Temos então:

$$M = \{s,e\} \; ; \quad \text{com limite} \quad N = 2 \quad \text{para o modo "e"}$$

Então, sendo este um exemplo bem similar àquele da fig. III.3 ou da fig. III.5, podemos usar as mesmas idéias para modelá-lo. Veja o resultado deste esforço na fig. III.15.

Vendo esta figura, pode-se compreender como seria pouco prático usar este tipo de modelagem, só imaginando-se como esta estrutura ficaria no caso de, por exemplo, 100 ou 200 usuários.

Entra, então, aqui, a idéia de usar inscrições para guardar a identidade dos usuários, modos, etc. numa versão, digamos, "condensada" desta figura III.15.

III.2.e.2.β - Os passos para condensar a estrutura original.

1) A relação dos predicados com o recurso.

O primeiro passo é verificar a possibilidade de criar uma estrutura condensada a partir de um estudo atento dos predicados e suas relações com o recurso R:

Então, usando predicados de diferentes graus, temos:

H(u) ↔ O usuário u (u ε U, isto é, a,b ou c) não tem (ainda) envolvimento com o recurso;

W(u,m) ↔ O usuário u deseja usar o recurso em modo m (isto é, exclusivo ou compartilhado);

U(u,m) ↔ O usuário u usa o recurso em modo m;

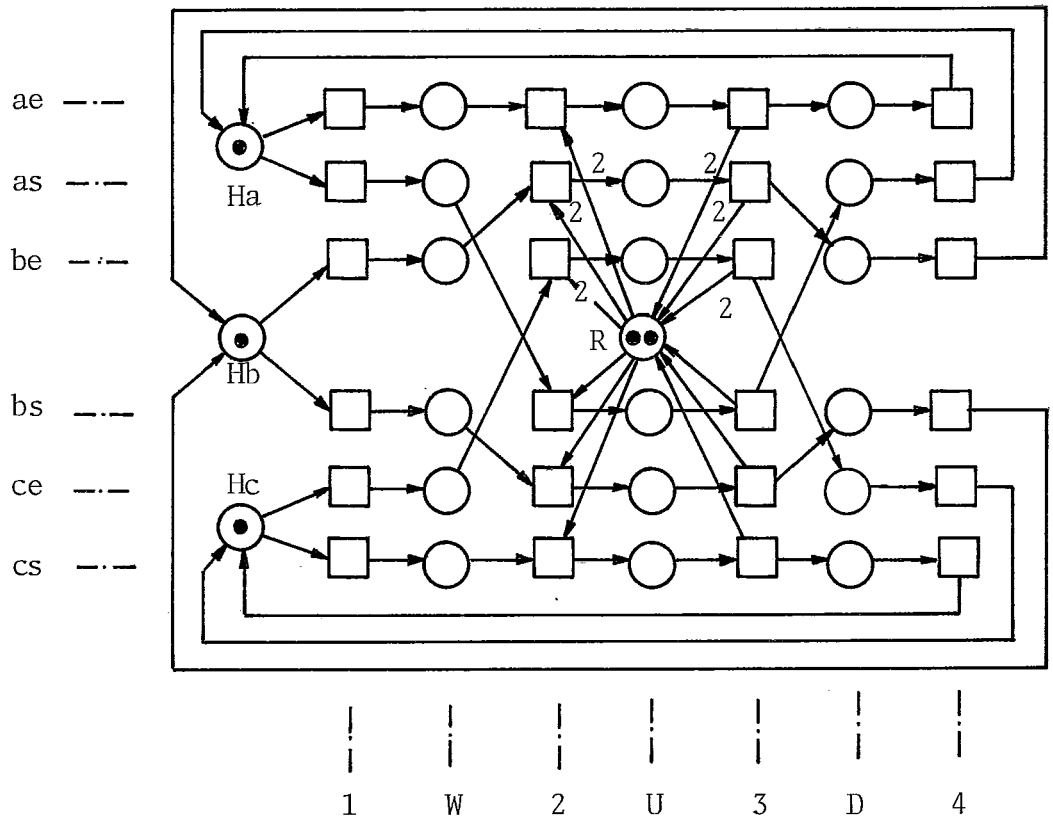
D(u,m) ↔ O usuário u terminou o uso do recurso em modo m;

R( ) .... Número de vezes que o recurso está ainda disponível para o uso compartilhado.

Assim já podemos criar uma estrutura bem mais simples que a da fig. III.15.

2) Modos e identidades.

O próximo passo é se preocupar com os diferentes modos de uso pelos usuários e, também, com a garantia da identidade de cada um deles. Vejamos isto então ponto a ponto:



onde: H, W, U, D ..... predicados
a, b, c ..... usuários
e, s ..... modo esclusivo ou compartilhado
R ..... recurso

Fig. III.15: Modelo completo de uma comunidade de três usuários que querem usar, em modo compartilhado (até 2) ou exclusivo, um único recurso, GENRICH et alia (08).

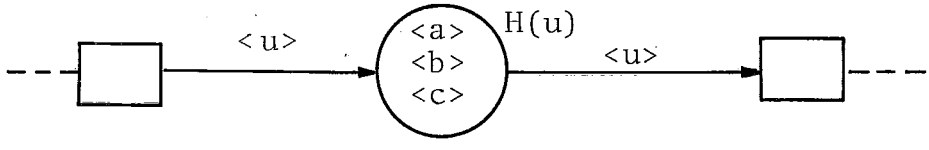


Fig. III.16: O predicado H(u) e a identidade dos usuários.

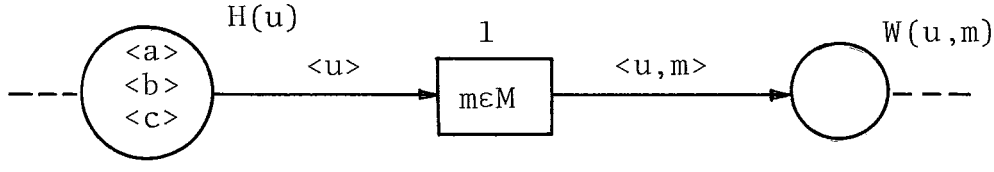


Fig. III.17: Transição 1 e a escolha de um modo.

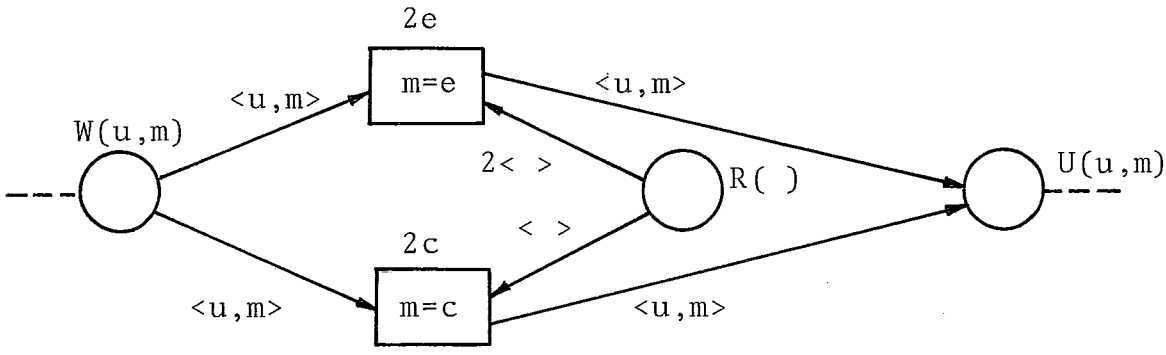


Fig. III.18: Bifurcação dependendo do modo escolhido.

- O predicado H(U) tem que conter a identidade dos usuários e os arcos incidentes têm que mostrar isto (fig. III.16).

- Na passagem de H(u) → W(u,m) o usuário fixa em que modo ele quer usar o recurso, isto é, que na transição 1 tem que ser escolhido algum m ε M, o que deve ser refletido também no arco (fig. III.17).

- Tem que ser observado que a substituição das variáveis deve ser consistente: todas as variáveis serão substituídas por símbolos individuais e todas as ocorrências destas variáveis são substituídas pelo mesmo símbolo (satisfazendo, obviamente, a fórmula lógica inscrita na transição).

- Dependendo do modo escolhido, tem que se passar, para atingir o predicado U(u,m) através de transições diferentes, isto é, do tipo "exclusivo ou compartilhado". Isto se faz necessário porque a multiplicidade dos arcos entre o recurso e as mencionadas transições é diferente (fig. III.18). O símbolo < > na figura III.18.c indica a presença de um lugar "comum", isto é, que somente há alguma condição satisfeita, (através da presença de 'tokens'), ou não (ausência do 'token').

III.2.e.2.γ - Estrutura equivalente.

Mostramos, como se pode investigar, passo a passo, alguma estrutura e as respectivas inscrições. Veja, então na figura III.19 uma estrutura equivalente à da figura III.15, mas, esta vez, COM inscrições relativas à identidade e ao modo.

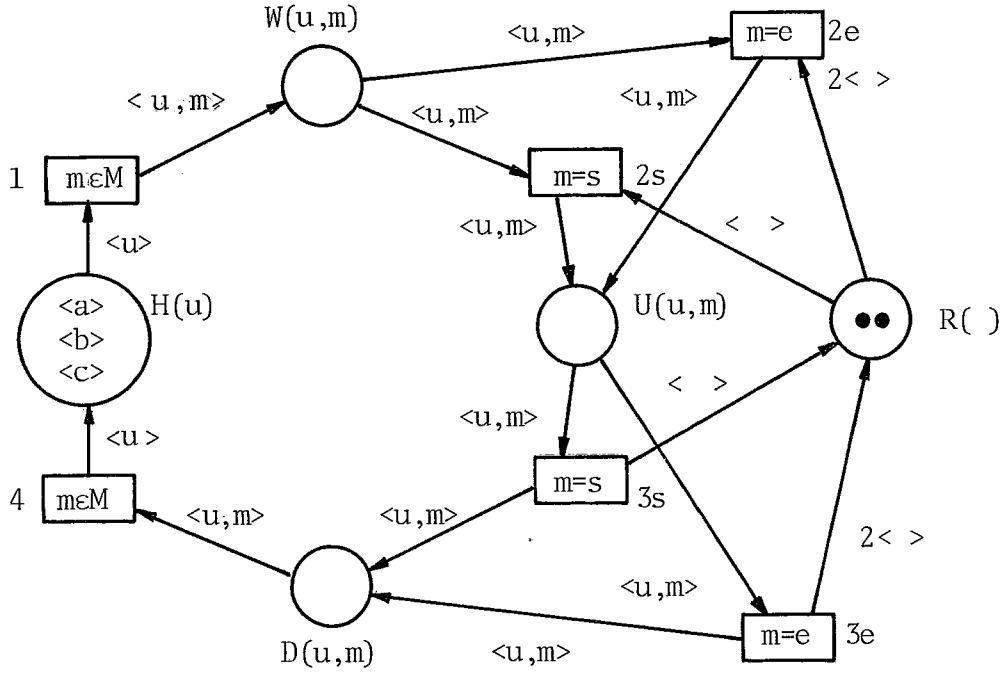


Fig. III.19: Modelo (equivalente à figura III.15) com estrutura simplificada e com inscrições relativas à identidade e ao modo, GENRICH et alia (08).

O que falta ainda, nesta figura III.19, é a indicação da transição morta, ou melhor, o esquema de transições mortas.

Sabemos, que o recurso pode ser compartilhado por, no máximo, dois componentes. Agora, em modo exclusivo, o predicado U (u,m) não pode conter dois pares de inscrições onde um deles tem um "e" (= exclusivo) na posição "modo". Então, comparando esta observação com o exemplo da figura III.5, podemos também formular este conceito para o exemplo dado na figura III.19 (veja este esquema na figura III.20).

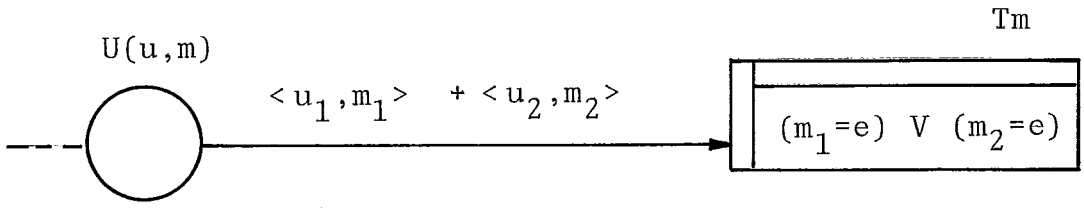


Fig. III.20: O esquema de transições mortas em relação à figura III.19.

III.2.e.2.δ - Os últimos ajustes.

A última consideração, para este exemplo, seria, eventualmente, a tentativa de "unir" as transições 2e e 2s, já que as ligações aos predicados são bem similares, isto é, somente diferem em relação às inscrições. Então, poder-se-ia pensar em colocar a decisão relativa ao modo (m ε M ... m = e ou m = s) já dentro da primeira transição e criar, para a conexão com o recurso R( ), uma função adicional do seguinte tipo:

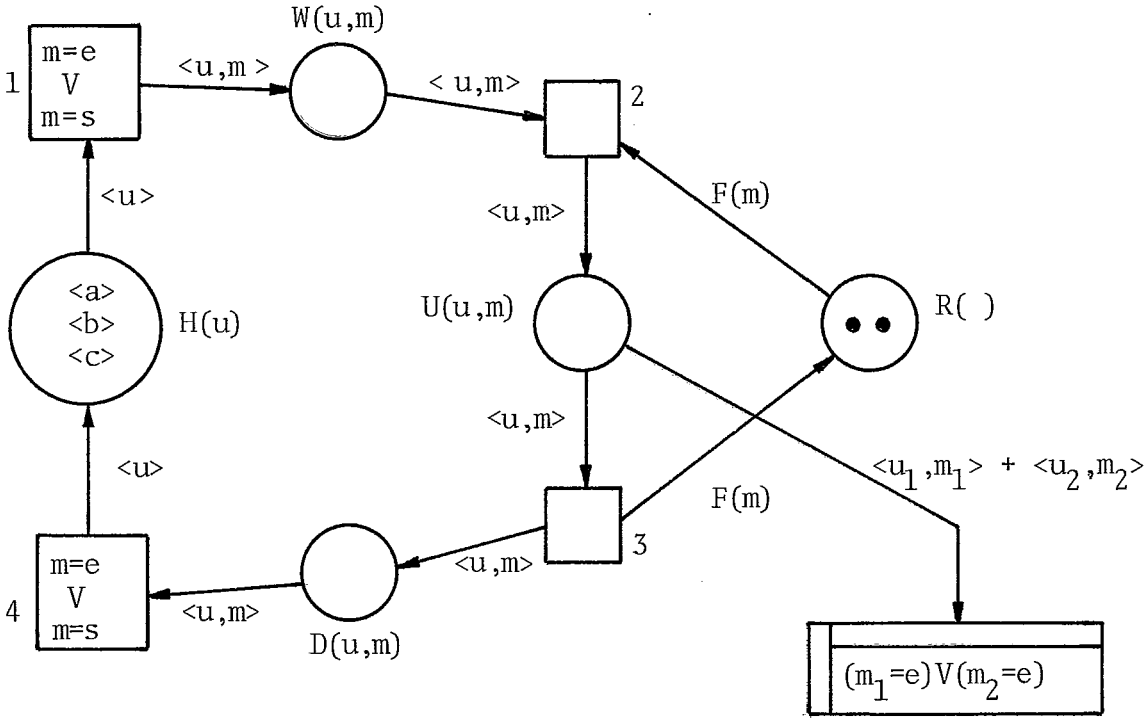
$$F(m) := \begin{cases} 1 <> \text{ , se } m=s \\ 2 <> \text{ , se } m=e \end{cases}$$

Desta maneira teríamos compactada ao máximo a estrutura da figura III.15, obviamente, com o preço pago de haver como inscrição agora expressões mais complexas.

Este tipo de compactação, embora mostrada por GENRICH e LAUTENBACH (08), não é permitido dentro das regras formais (em relação a 'PrT-Nets') que dizem que não deve haver inscrições "variáveis" nos arcos. Fica, então, somente a idéia registrada (veja o resultado na figura III.21) sem, contudo, usá-la no decorrer deste trabalho.

Observando este resultado final, junto com a versão inicial (fig. III.15) e a outra, intermediária (fig. III.19), devemos agora "sentir" o problema de modelagem. Isto quer dizer, que a versão intermediária (fig. III.19) facilitou em muito a compreensão do sistema em questão; isto porque nem a estrutura, nem as inscrições estão complexas demais. Já a última versão (fig. III.21)

transfere a complexidade da estrutura para as inscrições, onde é até necessária uma tabela "extra" para explicar algumas funções usadas como inscrições (isto, provavelmente, é uma das razões pela qual é excluída da modelagem formal na base de 'PrT--Nets').



onde: $F(m) := \begin{cases} 1 <> \text{ , se } m = s \\ 2 <> \text{ , se } m = e \end{cases}$

Fig. III.21: Representação do sistema da fig. III.15 de modo mais compacto: com inscrições relativas à identificação dos usuários e ao modo como usar o recurso, GENRICH et alia (08).

III.2.f - Metodologia para chegar ao modelo gráfico e à matriz de incidência para 'PrT-Nets'.

Da mesma maneira como mostramos a matriz de incidência para grafos P/T, podemos investigar aquela para 'PrT-Nets'.

Observamos que um 'PrT-Net' pode ser considerado como sendo um esquema de camadas de grafos (ordinários) do tipo P/T. Então, o vetor i, mostrado na relação da equação (III.1), $i^T M = i^T M_o$, é representativo de uma destas camadas, portanto, não é "cons-

tante" em relação a 'PrT-Net' como um todo.

A única indicação constante é o vetor das marcações iniciais, M_o. O que temos que fazer agora, é encontrar as marcações válidas para uma determinada camada, junto com as variáveis do vetor i, para encontrar as S-invariantes da camada em questão. Mostraremos isto através de um exemplo, adaptado àquele usado por GENRICH et alii(06).

III.2.f.1 - O sistema.

Seja o sistema da figura III.24. A idéia atrás deste esquema é a seguinte: existem n usuários (por exemplo, 'database managers'), representados por seus identificadores dentro de um conjunto U. Estes usuários podem transmitir mensagens uns aos outros. Estas mensagens estão representadas como elementos de um conjunto N.

A estrutura da fig. III.24 criou-se através da definição de determinados protocolos (≙ regras de comunicação entre usuários). Durante esta criação deve-se sempre comparar o funcionamento previsto com as inscrições colocadas nesta representação gráfica.

III.2.f.2 - O funcionamento e os protocolos básicos.

Como já foi dito, os usuários podem transmitir mensagens uns aos outros. Será permitido que somente um usuário por vez transmita mensagens, mas pode enviar para todos os usuários simultaneamente. Não será permitido enviar mensagens a si mesmo.

Se há mais que uma mensagem para o mesmo destino, esta segunda mensagem somente deve ser transmitida depois de ter recebido o ACK ('acknowledgment') da mensagem anterior.

Os ACK's são considerados como sendo mensagens de controle e não sofrem nenhuma restrição em relação ao paralelismo, multiplicidade, conflitos, etc.

III.2.f.3 - Definições iniciais.

Primeiramente, devemos associar a cada predicado um certo significado ou objetivo. Obtemos assim uma lista de predicados, escolhendo, da melhor maneira possível, nomes auto-explicativos como, por exemplo, INATivo, EMISsor, CONTrolar, PERMissão, FONTe, PRONTo, PROCessar, ACK, etc...

O passo seguinte é a definição dos argumentos destes predicados, já que os predicados devem ser aplicados a estes objetos ou argumentos. Veja então, na fig. III.22, as definições em relação ao nosso exemplo.

Com a definição destes argumentos podemos então fixar os predicados da lista mencionada de uma maneira tal que representarão exatamente o objetivo da modelagem; veja isto na figura III.23.

U = $\{u_1, u_2, \ldots, u_n\}$; usuários presentes no sistema.

Tipo = {i,c}; o tipo da mensagem que pode ser uma mensagem de INFORMAÇÃO ou uma de CONTROLE.

EndE = EE = $\{e_1, e_2, \ldots, e_n\}$; endereço do emissor onde, potencialmente, cada usuário pode ser emissor.

EndR = ER = $\{r_1, r_2, \ldots, r_n\}$; endereço do receptor. Cada usuário pode se tornar receptor, exceto aquele que emite mensagens de informação neste instante.

Fig. III.22: Definição dos argumentos para os predicados.

INAT(u) ... Predicado de grau 1, representando um usuário u (u ε U) que está num estado inativo.

EMIS(u) ... Predicado de grau 1; representa um usuário u (u ε U) que se tornou emissor.

CONT(u) ... O usuário receptor u (u ε U) controla o recebimento e o processamento de uma mensagem a ele destinada.

PERM( ) ... Um predicado de grau zero; representa uma permissão no sentido que somente um usuário poder-se-ia tornar emissor.

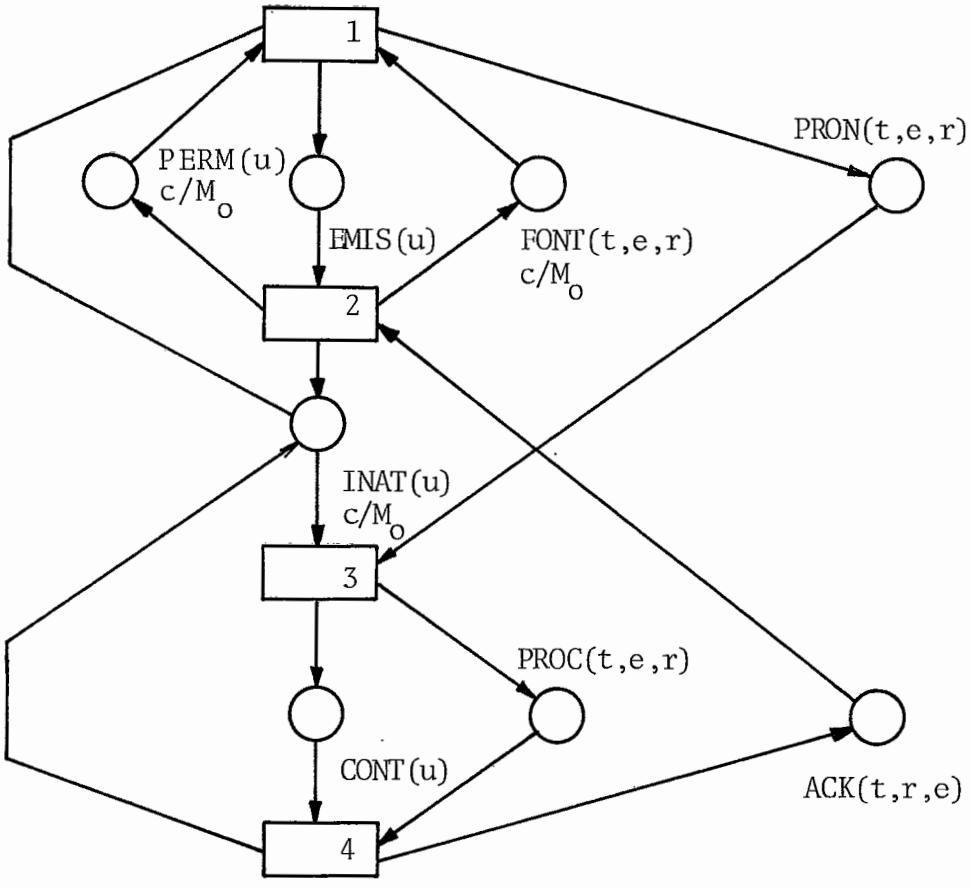
FONT(t,e,r) ... Predicado de grau 3; representa a existência de todas as mensagens do tipo t (t ε Tipo) numa fonte. A mensagem pode conter como endereços todos os usuários: especificando e (e ε EndE) para o endereço da emissão e r (r ε EndR) para o endereço da recepção.

PRON(t,e,r) ... Predicado de grau 3; representa as mensagens do tipo t (t ε Tipo), prontas, associadas a um único emissor ativo, e (e ε EndE, ulm(e)=1), porém para todos os outros usuários, r (r ε EndR, ulm(r) = = |U|-e).

PROC(t,e,r) ... Predicado de grau 3; representa algum processamento da mensagem do tipo t (t ε Tipo), vindo de e (e ε EndE, ulm(e)=1), sendo recebida e controlada por r (r ε EndR, ulm(r)=1).

ACK(t,r,e) ... Predicado de grau 3; representa uma mensagem do tipo confirmação afirmativa ('acknowledgement') t (t ε Tipo), no sentido que a confirmação da mensagem recebida em r (r ε EndR, ulm=1) permite o eventual envio de uma próxima mensagem do mesmo emissor e (e ε EndE, ulm=1).

Fig. III.23: Definição dos predicados, seus argumentos e objetivos da modelagem.



onde tem as seguintes marcações iniciais:

M_o(PERM()) = <>

M_o(INAT(u)) = U = $\{u_1, u_2, \ldots, u_n\}$

M_o(FONT(t,e,r)) = N ; N := (EndE x EndR) - id;

Fig. III.24: Estrutura inicial de um conjunto de usuários que podem enviar mensagens um ao outro, usando as definições das figuras III.22 e III.23.

II.2.f.4 - A modelagem do funcionamento de sistema.

O funcionamento dinâmico é baseado na seqüência de disparo das transições. Veja como este conceito funciona em nosso exemplo e observe ao mesmo tempo como as inscrições devem obedecer às definições formais dadas na seção III.2.d.

1) O disparo da *transição 1* é possibilitada porque existem:

- Usuários (agentes) no predicado INAT(u);
- Uma permissão (um 'token') disponível no predicado PERM( );
- Mensagens de informação presentes em FONT(t,e,r), isto é, um agrupamento abstrato neste predicado do tipo 'pool'.

Assim são iniciadas as seguintes ações:

- O agente em questão, representado pela tupla <x> (x ε U), sai do estado INAT(u) para se colocar na posição de emissor, representado pelo predicado EMIS(u);
- As mensagens {<t,e,r>|t=i, r ε EndR ∧ r ≠ e}, i.e., todas as mensagens relativas ao agente x (x ε U) a serem destinadas a outros agentes (excluindo mensagens a si mesmo), passam da posição FONT(t,e,r) para o predicado seguinte, PRON(t,e,r), onde este predicado pode ser considerado como sendo o lugar das mensagens "ativadas ou prontas". Observe que a marcação de PRON(t,e,r) é incrementada por N_e na medida em que a marcação de FONT(t,e,r) é decrementada por este mesmo subconjunto, podendo N_e ser definido como N_e := {<t,e,EndR>|r (r ε EndR) ≠ e}.
- O único 'token' <> presente em PERM( ) sai deste predicado, implicando assim que somente um único usuário pode transmitir por vez.

Mostrando o modelo parcial (fig. III.25) em relação a isto, pode-se observar a consistência das inscrições usadas em predicados, transições e arcos.

2) Tendo agora mensagens N_e, i.e., associadas ao usuário x (x ε U), no predicado PRON(t,e,r), e um dos receptores potenciais, digamos y (y ε U, y ≠ x), destas mensagens à disposição em INAT(u), vemos que a *transição 3*, que é a "próxima", pode disparar, provocando os seguintes acontecimentos:

   - O agente y (y ε U), o receptor em questão, "aceita" a mensagem a ele dirigida, representada pela tupla <t,e,r>, e providencia o deslocamento do PRON(t,e,r) para outro predicado de grau três, PROC(t,e,r), que representa "uma mensagem de informação do usuário x, (x = e, e ε EndE), para o receptor y, (y = r, r ε EndR ∧ y ≠ x)".
   - Fazendo isto, o agente y em questão muda de estado e passa para o estado de controle, representado pelo predicado de grau um, CONT(u).

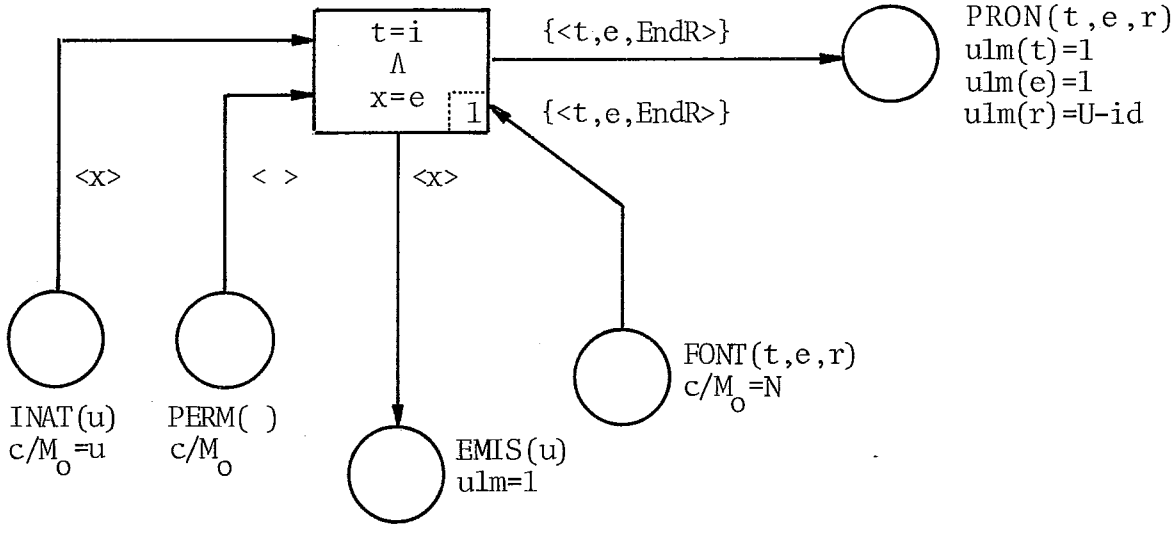
   Veja isto no modelo parcial da fig. III.26.

3) Depois de algum processamento (verificação, decodificação, etc,), feito num outro nível, a saber, irrelevante para os nossos objetivos de modelagem, obtemos o resultado deste processamento que compõe agora uma das pré-condições para disparar a *transição 4*. Assim teremos as seguintes ações:

   - O usuário y (y ε U), responsável pela recepção e pelo processamento da mensagem em questão, pode confirmar a mensagem tendo com isto terminado a sua atividade. Ele, então, retorna ao estado inativo (esperando lá para, eventualmente, receber uma outra mensagem, ou se tornar, ele mesmo, emissor de uma mensagem de informação).
   - A mensagem de informação foi "absorvida" em PROC(t,e,r) e provocou a formação de uma mensagem do tipo controle, apresentada pela tupla <t,r,e>, t = c, invertendo os campos de endereçamento.
   - Observe ainda que somente depois do disparo da transição 4 pode ser enviada uma eventual próxima mensagem do mesmo emissor x (x ε U) ao mesmo receptor y (y ε U). Isto porque a passagem pela transição 4 resultou na disponibilidade de

y (y ε U) como pré-condição para a transição 3 e na liberação do "lugar de processamento" (indicado por ulm(r)=1 , no predicado PROC(t,e,r)), sendo este último uma pós-condição desta mesma transição 3.

Observe este fato no modelo parcial em torno das transições 3 e 4, mostrado na fig. III.27.



onde: {<t,e,EndR>} representa o subconjunto N_e.

Fig. III.25: Modelo parcial em torno da transição 1 da estrutura da fig. III.24.

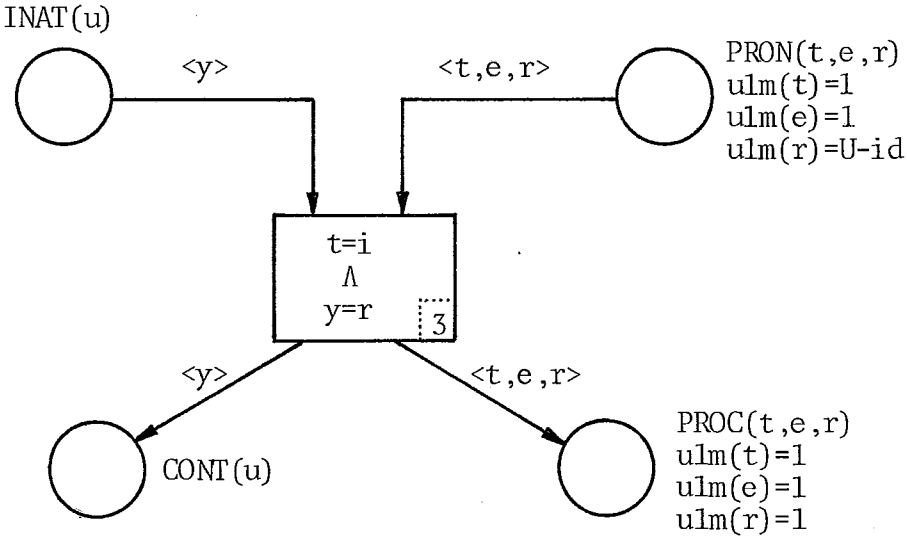


Fig. III.26: Modelo parcial em torno da transição 3 da estrutura da fig. III.24.

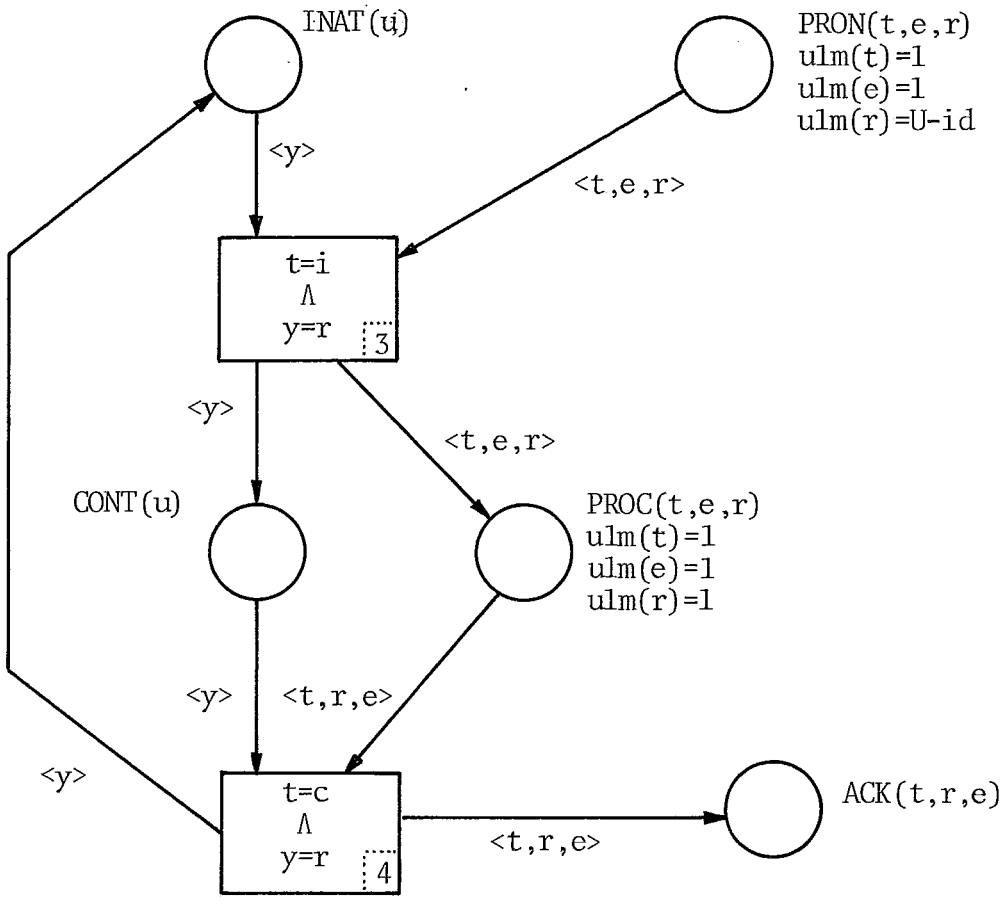
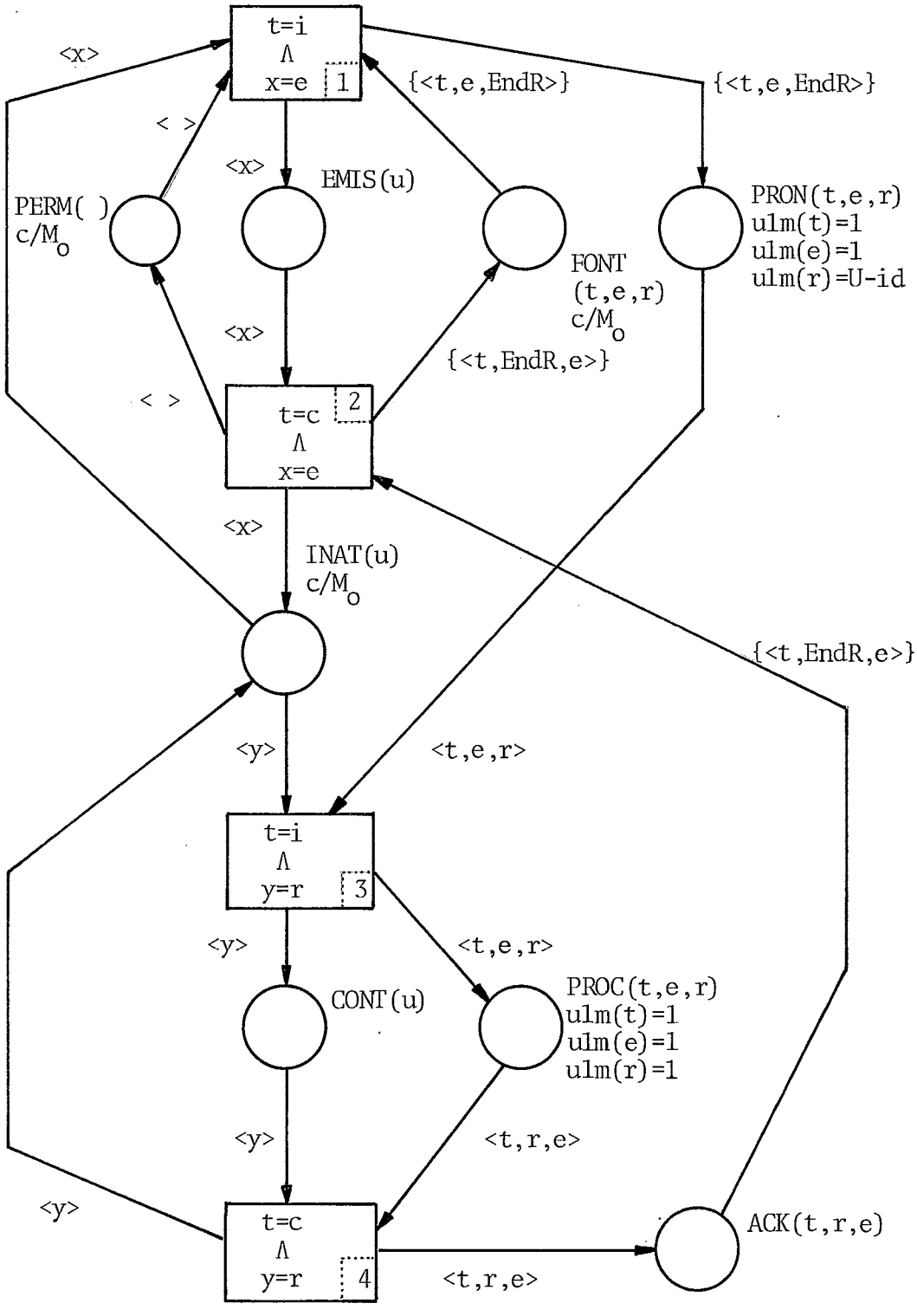


Fig. III.27: Modelo parcial em torno das transições 3 e 4 da estrutura da fig. III.24.

4) Para disparar a *transição 2* (isto é, terminar as emissões do usuário em questão e dar essa possibilidade a um eventual outro usuário) precisamos nos lembrar que isto somente deve acontecer se todas as mensagens do emissor em questão, colocadas em PRON(t,e,r), foram enviadas e confirmadas. Assim acontece o seguinte:

   - Todas as confirmações, reunidas em ACK(t,r,e), permitem a satisfação da pré-condição para a transição 2 e, conseqüentemente, a consistência com a inscrição do arco adjacente, {<t,EndR,e>}. Assim se obtêm novamente "acesso" ao predicado FONT(t,e,r) e ao mesmo tempo podem-se "eliminar eventuais cópias das mensagens originais".



onde: Os significados de predicados e argumentos se encontram nas figuras III.22 e III.23, respectivamente; valores de M_o ... ver fig. III.24.

Fig. III.28: Exemplo de um conjunto de usuários (p.ex.,'database managers') que podem enviar mensagens um ao outro, obedecendo um determinado protocolo de transmissão.

- Usuário x (x ε U) tinha que esperar somente estes ACK's para poder novamente entrar no estado INAT(u) e assim se colocar na situação de poder competir por uma nova série de emissões (dependendo das regras de prioridade dentro de INAT(u)), ou para receber informações de outros usuários.
- O 'token' que representa a permissão para uma emissão e, sendo o único, também a garantia para a existência de somente um emissor, volta ao predicado inicial, PERM( ), permitindo assim o início de um novo "ciclo".

Veja agora estes fatos, junto com os das outras transições, na fig. III.28. Esta figura representa, para nossos objetivos de modelagem, o estágio final do modelo.

III.2.f.5 - A matriz de incidência e as marcações.

Em paralelo à compreensão do funcionamento e à modelagem, devemos investigar as marcações, em especial as S-invariantes em relação às várias camadas nesta estrutura. Mostraremos, inicialmente, a matriz de incidência na fig. III.29 e a marcação básica na fig. III.30, para, depois, investigar as marcações subseqüentes.

Tendo então a matriz de incidência e a marcação inicial, pode-se agora, usando a equação (III.1): $i^T M = i^T M_o$, verificar, ou fixar, as marcações seguintes. Algumas delas podem ser consideradas como sendo invariantes e, conseqüentemente, serão chamadas de S-invariantes.

Veja, por exemplo, um vetor i_1 da forma como mostrado na fig. III.31. Ele indica que sempre, sob qualquer marcação, todos os usuários estão em qualquer estado, isto é,

$$M(INAT) + M(EMIS) + M(CONT) = U ,$$

o que, aplicando valores, dará

$$|M(INAT)| + |M(EMIS)| + |M(CONT| = n,$$

o número total dos usuários.

Uma idéia similar pode ser aplicada para as mensagens; veja o vetor i_2 nesta mesma figura III.31. Neste caso, ele indica que todas as mensagens estão sempre em alguma parte; veja:

$$M(FONT) + M(PRON) + M(PROC) + M(ACK) = N .$$

	t=i ∧ x=e 1	t=c ∧ x=e 2	t=i ∧ y=r 3	t=c ∧ y=r 4
INATU(u)	-<x>	<x>	-<y>	<y>
EMIS(u)	<x>	-<x>		
CONT(u)			<y>	-<y>
PERM( )	-< >	< >		
FONT(t,e,r)	-{<t,e,r>}	{<t,r,e>}		
PRON(t,e,r)	{<t,e,r>}		-<t,e,r>	
PROC(t,e,r)			<t,e,r>	-<t,r,e>
ACK(t,r,e)		-{<t,r,e>}		<t,r,e>

Fig. III.29: A matriz de incidência da estrutura mostrada na fig. III.28.

	INAT	EMIS	CONT	PERM	FONT	PRON	PROC	ACK
M_o^T =	U	-	-	<>	N	-	-	-

Fig. III.30: Uma possível marcação inicial M_o para a estrutura da fig. III.28.

	INAT	EMIS	CONT	PERM	PONT	PONT	PROC	ACK
i_1^T =	1	1	1	-	-	-	-	-
i_2^T =	-	-	-	-	1	1	1	1
i_3^T =	-	-	<t,e,r>	-	-	-	-<r>	-

Fig. III.31: Alguns exemplos de S-invariantes para a estrutura da fig. III.28.

Ou, para mostrar ainda um outro exemplo, podemos relacionar o par de predicados CONT/PROC, dando o vetor i_3 na figura III. .31. Este vetor, formalmente, indica

$$\langle t,e,r\rangle \cdot M(CONT) - r \cdot M(PROC) = 0$$

com a conseqüência que

$$M(CONT) = 0 \leftrightarrow M(PROC) = 0$$

e, ainda, que

$$\forall r : \left[M(CONT) = r \leftrightarrow \exists e : M(PROC) = \langle t,e,r\rangle\right].$$

Esta formulação diz, então, que as marcações de CONT(u) e PROC(u) são intimamente relacionadas: NÃO pode "haver" um usuário u se NÃO há uma mensagem para este mesmo usuário.

Dependendo da natureza do problema, podemos fixar, ou investigar, ainda outras S-invariantes (como, por exemplo, o relacionamento de predicados EMIS/PERM, EMIS/FONT, PERM/FONT, ou até uma quádrupla CONT/PERM/PRON/ACK). Isto foi mostrado por GENRICH et alii (06), mas para esta seção explicativa de nosso trabalho, os exemplos são o suficiente para compreender a idéia por trás das S-invariantes em grafos do tipo predicado/transição.

III.2.g - Observações finais sobre a seção III.2.

Foi apresentada, nesta seção III.2, uma definição global de um 'PrT-Net'. Para explicar melhor o uso desta definição global, escolhemos um sistema simples de R/W ('reader/writer') e, assim, deve ter ficado bem claro como, e em que contexto, se pode usar os diversos componentes de um 'PrT-Net'.

Mostramos também, no mesmo exemplo, como se deve escolher o meio termo entre modelos extremos: um é o modelo com estrutura complexa, mas sem inscrições, e outro é a representação estrutural simples mas que coloca a complexidade dentro das inscrições.

Para explicar melhor como se chega ao modelo gráfico, a matriz de incidência, e aos vetores "invariantes" em relação a 'PrT-Nets', escolhemos como exemplo um sistema simples de enviar mensagens, mostrando ao mesmo tempo, mais uma vez, como proceder na modelagem.

III.3 - Resumo do capítulo III.

Neste capítulo apresentamos uma ferramenta poderosa, o 'PrT--Net' ('predicate/transition-net'), para modelar acontecimentos dinâmicos em estruturas distribuídas.

Começamos com a história da GNT ('General Net Theory') dentro da qual se enquadra, também, o 'PrT-Net'. Reservando uma seção inteira para explicá-lo, conseguimos mostrar, depois de uma definição formal, os componentes, regras e aplicações em relação a esta ferramenta.

Mais ainda, este capítulo serviu, além de explicar o 'PrT--Net', para alertar sobre as dificuldades de modelagem, isto é, encontrar o equilíbrio certo entre a ciência e a arte. Este problema se adiciona às dificuldades do problema em si (p.ex., p controle de fluxo e de acesso em redes com meios heterogêneos de transmissão) e nos força, assim, a proceder com certos cui-

dados.

Vamos, nos próximos capítulos, tentar, usando a estratégia de "pequenos passos", como usado por SCHWARZ (34, 35, 36, 37), alcançar o nosso objetivo.

C A P Í T U L O IV

Modelo simplificado, usando 'PrT-Nets', da transmissão de um pacote simples em redes de computadores.

IV.1 - Introdução

(Observação: uma versão preliminar deste capítulo foi publicada como relatório técnico, SCHWARZ (34)).

Neste capítulo podemos, finalmente, começar a modelar alguns dos problemas em redes de computadores. Em particular, usando como base as descrições qualitativas expostas no capítulo II, pensamos em modelar a transmissão de um pacote simples. Este tipo de abordagem, isto é, modelando um sistema bem simplificado, serve, principalmente, para "aprender" como usar a técnica de 'PrT-Nets'.

Consideramos, então, uma subrede de comunicação, como definida por KLEINROCK (14) e SCHWARZ (30), e as possibilidades de acesso à esta subrede. Numa primeira análise, para não complicar demais as investigações iniciais, escolheremos o tipo de acesso que envolve um único 'host', isto é, uma ligação ponto-a-ponto entre aquele 'host'/fonte e um vértice/fonte associado.

O próximo passo é a delimitação da investigação. Isto pode ser feito usando o esquema simplificado como mostrado no capítulo II (figura II.2), isto é, uma seqüência de transmissão entre pares de processos, para um único pacote, e ignorando, por enquanto, os vértices intermediários.

Como os esboços do capítulo II mostram somente a seqüência para um único par H/F - H/D, cabe agora investigar o comportamento do sistema em relação ao funcionamento simultâneo de mui-

tos pares. Isto é, afinal, o objetivo de uma rede.

Mostraremos, então, neste capítulo IV, a maneira como usar 'PrT-Nets' para modelar a transmissão de mensagens pequenas, ou seja, mensagens que "cabem" no formato de um único pacote, referenciadas, no decorrer deste capítulo, como "pacotes simples".

Começamos o estudo com definições e observações iniciais (seções IV.2.a até IV.2.d) para, depois, na seção IV.3, mostrar as fases da modelagem. Finalmente, na seção IV.4, juntamos as diferentes fases da modelagem para obter um primeiro modelo que servirá, em capítulos subseqüentes, como ponto de partida para conseguir modelos bem mais completos e representativos.

IV.2 - Usando 'PrT-Nets' para modelar uma versão simplificada do controle 'end-to-end'.

Sabendo já bastante a respeito de 'PrT-Nets', podemos realmente nos lançar à modelagem, começando com definições e observações iniciais. Como estas se referem já ao sistema a ser modelado, vale lembrar que as definições necessárias em relação à ferramenta usada, o 'PrT-Net', se encontram no capítulo III, que tem, na seção III.2.a, a definição formal e, na seção III.2.d, a explicação dos componentes que compõem um 'PrT-Net'.

IV.2.a - Observações iniciais sobre o sistema.

As nossas investigações, de uma forma geral, se situam num nível lógico. Portanto, suponhamos que a rede, no nível físico, está funcionando perfeitamente. Mas mesmo assim, deve-se ficar consciente que um erro num nível inferior sempre tem as suas implicações em relação aos níveis superiores. Se, por exemplo, uma linha de comunicação se rompe, deve haver meios de "saber" disto e tomar providências. Este tipo de informação pode chegar ao nível superior, através de mensagens explícitas (por exemplo, "linha interrompida") ou via um funcionamento fora das normas estabelecidas (por exemplo, falta de uma confirmação, atraso

excessivo, etc.).

Dissemos no início que escolhemos um tipo de acesso chamado de ponto-a-ponto. Mas temos que simplificar ainda mais a respeito deste tipo de acesso. Observando o conjunto envolvido num acesso, {HOST-LINHA-VÉRTICE}, pode-se dizer que o ponto de atrito, em relação a outros conjuntos, está situado no vértice/fonte. Portanto, podemos, para uma primeira investigação, assumir que existe, para cada V/F, somente um conjunto. Estes conjuntos são organizados similarmente, com respeito ao funcionamento, mas a velocidade de execução pode ser bem diferente para cada um.

Mais tarde, na continuação das investigações iniciais, podemos "abrir" este conjunto, a partir de agora chamado de USUÁRIO, e investigar as questões em relação a pontos tais como: cada 'host' pode representar vários processos; ou, vários 'hosts' podem competir pelo mesmo V/F; ou ainda, as linhas, entre estes 'hosts' e o V/F associado, podem funcionar de diferentes modos, velocidades, etc.

O 'PrT-Net', a ser construído, apresentará o resultado de mapeamento de n 'P/T-Nets' isomórficos, cada um representando o funcionamento entre um único par de usuários, isto é, um processo (usuário) quer se comunicar, através da rede, com um outro processo. Reunindo todos estes usuários num único 'PrT-Net', temos que, devido ao seu comportamento desigual e para garantir a identificação, fazer as distinções através do uso de INSCRIÇÕES e MARCAÇÕES.

Assim, pode-se dizer que usamos esta nova técnica para modelar esta organização simplificada de redes de computadores. A técnica empregada adiciona às vantagens de Petri-Nets no sentido descritivo e analítico, uma nova dimensão: o tratamento formal dos indivíduos (elementos, componentes, etc.) e das suas, sempre se modificando dinamicamente, propriedades e relações. E, ainda, o uso desta técnica permite evitar o tratamento de uma estrutura grande, como a de redes, num nível não aceitável

de detalhes.

IV.2.b - Definições iniciais.

Lembrando as definições e o exemplo mostrado no capítulo III e em SCHWARZ (33), vemos que devemos indicar os conjuntos necessários para formar o grafo direcionado, N = (S,T;F), porque este será a base para a representação do funcionamento de nossa rede de computadores.

IV.2.b.1 - Definição dos ARGUMENTOS.

Existem, relacionado ao conjunto de predicados, vários ARGUMENTOS básicos, a serem definidos na tabela da figura IV.1. Adicionalmente a estas definições podemos observar ainda vários pontos, esclarecidos a seguir.

O conjunto U de símbolos individuais u_i, que representam os identificadores dos usuários, é limitado em W; isto é o equivalente, neste exemplo, ao número de todos os conjuntos {host--linha-vértice}. Cada um destes conjuntos pode, em princípio, mandar e receber mensagens.

Assim, chegamos às mensagens. Se indicassemos como argumento, para os predicados envolvidos com mensagens, o símbolo M, $M = \{m_1, m_2, \ldots, m_N\}$, não usaríamos toda a potencialidade da modelagem, já que não permitiremos a identificação individual dos emissores e/ou receptores. Para atacar este problema "quebramos" o argumento M em dois outros: um é o símbolo EndE, ou abreviando mais, EE, para indicar o endereço do emissor, onde cada usuário pode se tornar emissor; e outro é o símbolo GR, que indica o grupo de receptores, endereçado pelo emissor em questão. Então, os pares <s,gr> $\{<s,gr> | s \in EE, gr \in GR\}$, indicariam o conjunto das mensagens de um usuário s para vários receptores, agrupados em gr e algum par <s,r>, $\{<s,r> | s \in EE, r \in ER\}$, é uma destas mensagens, sob restrição que $r \neq s$.

O terceiro grupo de argumentos está ligado aos caminhos que

$U = \{u_1, u_2, \ldots, u_W\}$;

usuários conectados em torno da rede de computadores em questão. Usaremos o símbolo geral u em vez de u_i.

$EndE = EE = \{s_1, s_2, \ldots, s_W\}$;

cada usuário pode se tornar emissor; a indicação "endereço do emissor" é contida numa parte da mensagem.

$EndR = ER = \{r_1, r_2, \ldots, r_W\}$;

cada usuário pode se tornar receptor, tendo como restrição que o receptor deve ser diferente do emissor. Então, para um dado r ($r \varepsilon ER$) temos |U-id| como número de receptores possíveis.

$GR = \{gr_1, gr_2, \ldots, gr_W\}$;

cada emissor terá vários receptores r_i possíveis, sendo estes agrupados em gr ($gr \varepsilon GR$).

$P = \{p_i | p_i = f(s,r);\ i = 1, 2, \ldots, n\}$;

indica a parte adicional à mensagem para formar um pacote (no sentido da subrede de comunicação). Esta parte consiste, p. ex., em endereços, controles, códigos para controle de erros, 'flags', etc.

$CV = \{cv_{ij} | cv_{ij} = \text{caminho virtual entre vértices i para j},\ i = 1, 2, \ldots, W,\ j = 1, 2, \ldots, W\}$;

indica o conjunto de todos os caminhos virtuais possíveis dentro da rede.

$FC = \{fc_{ij} | fc_{ij} = f(cv_{ij}, s, r) = \text{agrupamento de caminhos virtuais possíveis entre vértices i para j},\ i = 1, 2, \ldots, W,\ j = 1, 2, \ldots, W\}$;

entre cada par de vértices haverá um grupo de caminhos possíveis (feixe de caminhos, 'Wegbündel'), não sendo considerados os caminhos para si mesmo ($i \neq j$).

$C = \{c_{ij} | c_{ij} = sel(fc_{ij}) = \text{determinado caminho escolhido entre vértices i para j},\ i = 1, 2, \ldots, W,\ j = 1, 2, \ldots, W\}$;

são aqueles caminhos que foram escolhidos como ligação entre dois vértices i para j, $i \neq j$. O critério da escolha pode ser relacionado a propriedades tais como: velocidade, distância, custo, etc.

Fig. IV.1: Definição dos argumentos enumerados na tabela da fig. IV.2.

levam de um vértice/fonte a um vértice/destinação. Embora, no esquema da figura II.2, não apareçam os diferentes caminhos possíveis entre V/F e V/D, achamos conveniente modelar, já agora, de um modo bem simplificado, a influência da escolha de caminhos. Num modelo subseqüente, que representará a figura II.5, isto pode já servir como ponto de partida na modelagem. Temos então o argumento FC, feixe de caminhos ('Wegbündel') que reune todos os caminhos possíveis entre dois vértices.

Outro argumento, em conseqüência direta, é o do caminho escolhido, C. Como há sempre um feixe de caminhos entre dois pontos i e j, $i \neq j$, podemos escolher um destes caminhos. Dependendo de algum critério usado (por exemplo, o critério de custo, ou de distância, ou de velocidade, etc.), pode-se ordenar os caminhos e escolher, por exemplo, o caminho mais rápido ou, no caso da não-disponibilidade deste caminho, o caminho que ocupa o segundo, o terceiro, etc., lugar nesta ordenação. A escolha poderia ser feita dentro de uma transição, digamos, por exemplo, naquela entre os predicados FEIXE/POSSÍVEL e CAMINHO/ /ESCOLHIDO.

IV.2.b.2 - Definição dos PREDICADOS.

Tendo definidos os argumentos básicos, podemos, agora, nos dedicar aos predicados. Vamos agrupar, na tabela da figura IV.2, os predicados básicos, ou seja, aqueles que têm importância vital em relação ao modelo global (deixamos, assim, as definições dos predicados auxiliares para as ocasiões em que necessitamos deles).

Temos, então, um grupo de predicados relacionados aos estados de um usuário; escolhendo, sempre que possível, nomes auto-explicativos, podemos citar: PASSIVO, ATIVADO para TRANSMITIR, TRANSMITINDO, ESPERANDO por CONFIRMAÇÃO, COMPARANDO, ATIVADO para RECEBER, DESPACHANDO, etc.

Outro grupo está relacionado às mensagens. Aí pode-se definir, entre outros, POOL/MENSAGEM, LISTA DE MENSAGENS, FONTE, PRONTA, MENSAGEM em TRÂNSITO, DESTINAÇÃO, ACEITA, LISTA DE RE-

CEPTORES, etc.

Podemos mencionar um outro grupo principal: aquele ligado aos caminhos onde se pode enumerar inicialmente: CAMINHOS/VIRTUAIS, FEIXE/POSSÍVEL, CAMINHO/ESCOLHIDO.

Só para indicar o que pode ser considerado como sendo um predicado auxiliar, pode-se relacionar aqueles que representam, entre outros, as SEG.VIAS das mensagens (chamada na literatura de "cópias"), as CONFIRMAÇÕES, as MENSAGENS de CONTROLE, o POOL/BUFFER, etc.

Mas temos que fazer ainda algumas observações em relação aos predicados: tendo definido os predicados, deve-se ter cuidado em ser o mais realista possível. Então, deveríamos colocar limites superiores de multiplicidade (ulm's) que representem as limitações (físicas, abstratas, etc.) dentro das diversas filas, 'buffers', etc. Veja, por exemplo, a indicação de uma limitação, ulm = %U, no predicado PEDIDO(u) na figura IV.3. Agora, para não complicar demais este primeiro passo de nossa investigação, devemos evitar a colocação imediata destes limites de multiplicidade. Mas convém, para futuras investigações, não esquecer o fato de que estamos lidando com limitações físicas, inerente ao mundo real. Assim, devemos, mais tarde, colocar um determinado ulm para os diferentes predicados, sempre tendo em vista a consistência do modelo.

IV.2.b.3 - Outras definições.

Obtidas as definições dos argumentos básicos e dos mais importantes predicados, podemos verificar os conjuntos restantes; assim, os outros dois conjuntos básicos de N = (S,T;F):

o das TRANSIÇÕES, $T = \{t_1, t_2, \ldots\}$, e

o das RELAÇÕES DE FLUXO, $F = \{f_1, f_2, \ldots\}$,

seguem as definições (sobre conjuntos, inscrições, etc.), dadas no capítulo III, a saber, que não temos que definir objetos adicionalmente neste ponto das investigações.

U S U Á R I O S :

ATIV/RECEBER(u) ... grau 1; representa uma ativação de um usuário, u_j ($u_j \in U$, $u_j \neq u_i$), para se tornar receptor potencial de uma mensagem.

ATIV/TRANSM(u) ... grau 1; representa um usuário, u_i ($u_i \in U$), que foi ativado, através do crédito em questão, para iniciar uma transmissão da mensagem <s,r>.

COMPARANDO(u) ... grau 1; recebida a confirmação, o usuário, u_i ($u_i \in U$), deve proceder com uma comparação entre a cópia da mensagem empacotada, <s,r,p>, e a confirmação recebida, <mc,r,s>. Se esta foi a última mensagem do grupo <s,gr>, o usuário u_i está pronto para retornar ao estado passivo.

DESPACHANDO(u) ... grau 1; feita a verificação, pode-se, no caso de uma situação correta, despachar a mensagem recebida, <s,r>, para o processo em questão. O usuário, u_j ($u_j \in U$), está assim preparado para retornar ao estado passivo.

ESP/CONF(u) ... grau 1; uma vez feita a transmissão, o usuário, u_i ($u_i \in U$), espera uma confirmação da mensagem enviada (obs.: neste exemplo simples estamos lidando somente com confirmações positivas, isto é, com RFNM's).

FEITO(u) ... grau 1; indica que o pedido colocado no predicado PEDIDO foi não somente aceito, mas também executado ou feito.

continua .....

Obs.: Quando não há dúvidas sobre a identificação, usaremos, sempre que possível, os símbolos sem indexação, ou seja, u, r, s, etc., em vez de u_i, r_i, s_i, etc.

Fig. IV.2: Definição dos mais importantes predicados, com seus argumentos, agrupados segundo as suas "naturezas", e seus objetivos na modelagem.

PASSIVO(u) ... grau 1; representa um usuário u ($u \in U$), que está num estado passivo ou inativo.

PEDIDO(u) ... grau 1; representa o pedido (ou uma permissão dada) de um processo (usuário) para transmitir uma ou mais mensagens.

PREP/TRANSM(u) ... grau 1; representa um usuário, u_i ($u_i \in U$), que já está ativado e que está se preparando para uma transmissão, isto é, que pode, dependendo de um eventual crédito, passar para o estado transmitir.

REJEITO(u) ... grau 1; indica que o pedido, por alguma razão, for rejeitado (obs.: este predicado não está modelado, por enquanto).

TRANSMITIR(u) ... grau 1; o usuário, u_i ($u_i \in U$), está, efetivamente, transmitindo a mensagem em questão, representada pela tupla <s,r>.

VERIFICANDO(u) ... grau 1; representa o fato que um usuário, u_j ($u_j \in U$, $u_j \neq u_i$), tem que verificar a mensagem com respeito a erros de transmissão, inconsistências, etc.

C A M I N H O S :

CAMINHO/ESCOLHIDO(s,r,c) ... grau 3; representa o caminho entre s e r realmente escolhido dentro do feixe, FC, disponível. O critério da escolha pode ser, p.ex., o caminho mais rápido, etc., mas é, por enquanto, irrelevante neste modelo.

CAMINHOS/VIRTUAIS(s,r,cv) ... grau 3; representa todos os caminhos virtuais, cv (cv ε CV) entre quaisquer vértices/fonte e vértices/destinação, indicado pelos endereços s(s ε EE) e r(r ε ER).

FEIXE/POSSÍVEL(s,r,fc) ... grau 3; ele representa o agrupamento de todos os caminhos possíveis fc(fc ε FC), num feixe de caminho ('Wegbündel'), entre os usuários s(s ε EE), que podem se tornar emissor, e aqueles que serão os receptores r, (r ε ER), de uma mensagem <s,r>.

Fig. IV.2 (continuação): Definição dos predicados.....

M E N S A G E N S :

ACEITA(s,r,b) ... grau 3; passada a verificação, a mensagem, <s,r>, já sem os complementos de pacotagem, está virtualmente aceita, isto é, pode ser despachada em direção ao host/destinação ou ao processo/receptor. Este fato será modelado pelo retorno da mensagem, <s,r>, ao 'pool', onde b é 'buffer'.

DESTINAÇÃO(s,r,p,b) ... grau 4; representa que a mensagem empacotada <s,r,p>, chegou à sua destinação, incluindo ainda a representação da necessidade de haver 'buffers', b, à disposição para receber a mensagem em questão.

FONTE(s,r) ... grau 2 ; representa a mensagem individual, <s,r> (s ε EE, r ε ER, r ≠ s), colocada numa fonte que precede o estágio da transmissão propriamente dita.

LISTM(s,gr) ... grau 2; representa um predicado auxiliar, onde se pode listar as mensagens do usuário, s(s ε EE), para um grupo de receptores, r,(r ε gr ε GR), preparando assim as mensagens a serem colocadas em FONTE.

LISTR(s,gr) ... grau 2; similar ao predicado LISTM, só que serve para verificar se todas as mensagens, <s,r> (s ε EE, r ε gr ε GR), foram enviadas. Se isto é o fato, o usuário s pode retornar ao estado passivo, e o processo/emissor receberia a indicação "feito".

M-em-TRÂNSITO(s,r,p) ... grau 3; este predicado representa o fato que uma mensagem, <s,r,p>, realmente está no caminho, isto é, que está em trânsito. Como a própria transmissão é feita é, neste nível de modelagem, irrelevante.

POOL/MENSAGEM(s,gr) ... grau 2; representa o fato que todas as mensagens, <s,gr> (s ε EE, gr ε GR), estão no início agrupadas, abstratamente, num 'pool'.

PRONTA(s,r,p) ... grau 3; representa a mensagem individual <s,r>, já preparada para ser enviada, isto é, com os respectivos campos de endereços e controle preenchidos. A mensagem empacotada estará, assim, pronta para a transmissão, representado pela tripla <s,r,p>, onde p representa esta parte adicional à mensagem.

Fig. IV.2 (continuação,final): Definição dos predicados ....

IV.2.c - As marcações iniciais.

Antes de entrar na fixação da marcação inicial, $M_o \in [M_o]$, queremos acrescentar algo sobre marcações em si.

Segundo a definição dada por GENRICH et alia (07), é denotado por [M] o conjunto de todas as marcações que são alcançáveis, a partir de uma dada marcação, num número finito de passos (para frente ou para trás). Se considerarmos somente passos para frente, o símbolo usado é [M> .

No caso das marcações iniciais vale este mesmo conceito, e temos assim $[M_o]$, a 'full marking class', e com $[M_o>$, a 'forward marking class'.

Agora, no que diz respeito aos predicados definidos na figura IV.2, podemos pensar que cada USUÁRIO tem que estar em algum estado como, por exemplo, representados pelos predicados PASSIVO, ATIV/TRANSM, TRANSMITIR, ATIV/RECEBER, etc. Desta idéia pode-se tirar a conclusão que, no "início" do funcionamento, todos os usuários estão num estado passivo, modelado pelo predicado PASSIVO (u). Assim,

$$M_o(\text{PASSIVO}(u)) := W = \Sigma\, u_i \mid u_i \in U,\ i = 1, 2, \ldots, W. \qquad \text{(IV.1)}$$

Similarmente, pode-se pensar que as MENSAGENS podem estar ou num 'pool' inicial (representado pelo predicado associado POOL/MENSAGEM (s, gr)), ou sendo numa FONTE, ou PRONTA, ou em TRÂNSITO, etc. Assim,

$$M_o(\text{POOL}(s, gr)) := N = \{ \bigcup N_s \mid N_s = \langle s, gr\rangle = s \times ER \frown \langle s,s\rangle\}, \qquad \text{(IV.2)}$$

o que indica que todas as mensagens, exceto aqueles "a si mesmo", estão neste POOL/MENSAGEM(s,gr) inicial.

Precisa-se também indicar as marcações iniciais em relação aos outros predicados como, por exemplo, para os caminhos vir

tuais, agrupados em CAMINHOS/VIRTUAIS (s, r, cv). A marcação inicial é, neste caso, simplesmente a indicação de todos os caminhos virtualmente possíveis:

$$M_o(\text{CAMINHO}(s, r, cv)) := \text{Card}(U \times U) - \text{Card}(U) = \sum_{s}^{W} \sum_{r}^{W} (<s, r, cv> | s \in EE, r \in ER, s \neq r). \quad \text{(IV.3)}$$

Todos os outros predicados p tem marcação inicial zero, indicado por:

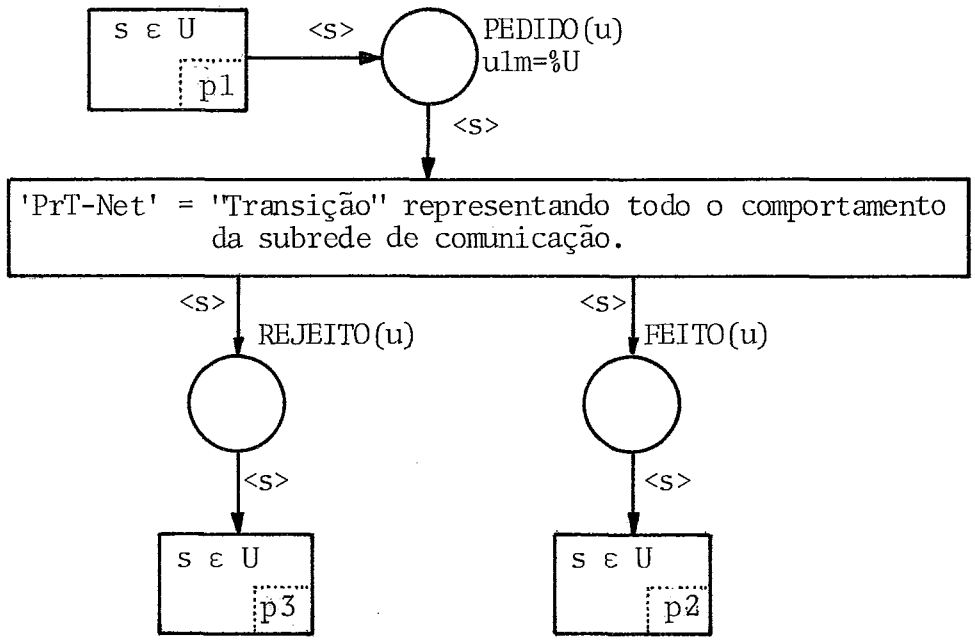
$$M_o(p) := 0, \quad \text{(IV.4)}$$

incluindo aí também as marcações para os predicados auxiliares que são, a nosso ver, dependentes diretos dos predicados básicos. Portanto, qualquer 2ª via (ou cópia) da mensagem, confirmação, etc., é criada por causa de uma mensagem sendo enviada (ou recebida corretamente, etc.) e deve ser, conseqüentemente, modelada neste sentido. Para, então, não interferir em demasiado com a finalidade explicativa deste exemplo, falaremos sobre estes predicados, e eventuais marcações, somente no momento quando realmente precisarmos deles na modelagem.

IV.2.d - Horizonte de observação.

Pode-se criar um horizonte de observação ('scope of concern'), a saber, decidir a partir de onde (ou de que situação, ou de quando, etc.), se deseja investigar. Formalmente, isto pode ser feito através de transições chamadas de "externas" (figura IV.3). Estas agem como "portas" de entrada ou saída, fornecendo os pedidos ou recebendo as respostas em relação aos predicados que estão colocados imediatamente antes ou depois do horizonte, respectivamente.

Assim, podemos dizer que um determinado processo (pertencente ao conjunto dos USUÁRIOS), s (s ε U), quer enviar uma, ou mais, mensagens; então, ele coloca este pedido, através de uma



onde: s ... determinado usuário, s ε U.

U ... conjunto de usuários conectados ao sistema.

PEDIDO(u), REJEITO(u), FEITO(u) ... predicados, indicando o desejo (ou a permissão) de um processo (ou usuário) de transmitir, assim como o "resultado" deste pedido.

ulm = %U ... indicando uma eventual restrição em relação ao número de usuários com permissão para transmitir (por exemplo, em caso de congestionamento pode-se reduzir drasticamente os acessos à subrede de comunicação.

pi ... portas ou transições externas.

Fig. IV.3: Colocação de transições "externas" (ou portas) para indicar o horizonte de observação ('scope of concern').

transição externa, p1, num predicado, digamos PEDIDO(u). A resposta, em relação a esta mensagem ou pedido, será recebida via outras transições externas, p2 e p3, a partir de determinados predicados como, por exemplo, FEITO(u) e REJEITO(u), respectivamente (figura IV.3).

Deste modo conseguimos delimitar as nossas preocupações, que estão voltadas para o funcionamento, e evitar perguntas polêmicas, tais como, porque mandar uma mensagem, qual é a vantagem, qual a implicação econômica ou política, etc. (compare, no capítulo III, com a figura III.1, que mostra os diversos conceitos relacionados à definição do horizonte de observação).

IV.3 - As fases para a criação de um modelo básico.

Tendo agora as considerações iniciais estabelecidas, podemos começar a modelar o problema em si. Faremos isto em etapas, seguindo, mais ou menos, a lógica de uma tentativa de transmissão de mensagens. Mas devemos alertar que nesta descrição das fases, numa maneira isolada, tem OMISSÕES no que diz respeito ao EQUILÍBRIO dos componentes.

No final deste capítulo, quando juntaremos as diferentes fases, poderíamos cuidar deste equilíbrio (fixando, por exemplo, algumas S-invariantes), seguindo uma velha sabedoria que diz que "energia não se cria, somente se transforma". Somente depois destes "ajustes" finais poderíamos dizer que temos um modelo final e representativo deste esquema simplificado, (figura II.2), descrito no capítulo II.

Finalmente, para facilitar a modelagem, podemos, antes de entrar nas fases de concepção do modelo, esboçar uma estrutura provável sem, contudo, entrar em detalhes. Isto servirá para melhor se situar e, também, para forçar um pouco o pensamento em termos "globais" na modelagem durante as diversas fases. Observe este modelo auxiliar na figura IV.4.

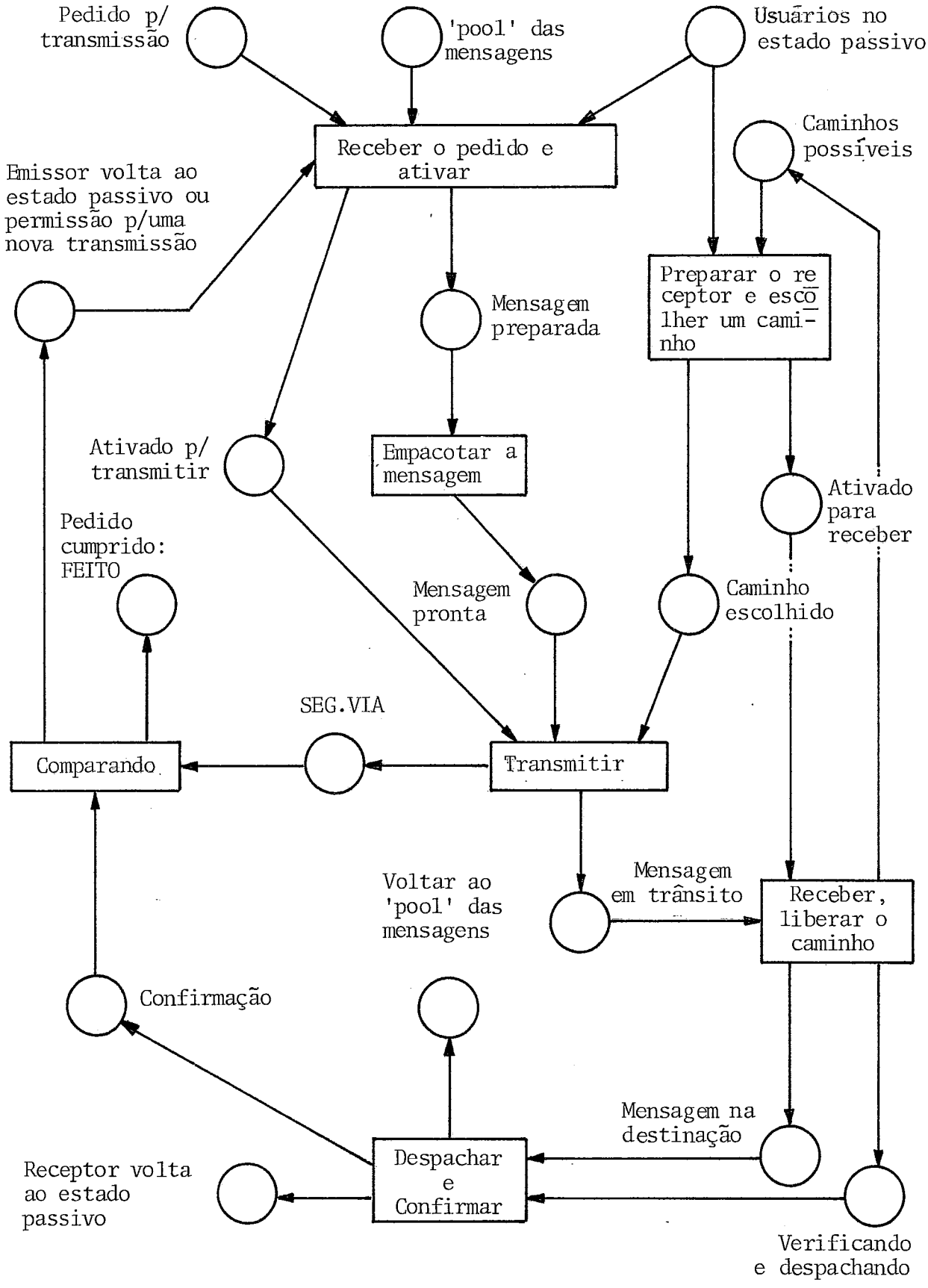


Fig. IV.4: Modelo auxiliar mostrando uma estrutura básica possível de um 'PrT-Net' para o esquema da figura II.2.

IV.3.a - Primeira fase: ativar um usuário para transmitir as mensagens a ele associadas.

Seja um processo s, s ε U, que deseja enviar alguma mensagem, representada pela tupla <s,r>. Observando a figura IV.5, vemos que, colocando o pedido do usuário s no predicado PEDIDO(u), obtemos já uma das pré-condições para disparar a transição 1.

As outras pré-condições (compare com as idéias iniciais mostradas na figura IV.4), devem estar relacionadas ao:

- Conjunto de usuários, U = $\{u_1, \ldots, u_W\}$, e ao
- Conjunto das mensagens, M = $\{m_1, \ldots, m_N\}$.

Dentro do conjunto dos usuários deve ser procurado, e ativado, aquele idêntico ao usuário s. E do conjunto das mensagens será isolado o grupo das mensagens associadas a s, isto é, aquelas representadas pela tupla <s, gr>. Para poder tratar as mensagens individualmente, isto é, as mensagens <s,r>, usamos uma listagem, LISTM(s,gr), que permite colocar primeiro uma mensagem, <s, r_1>, depois outra, <s, r_2>, etc., no predicado FONTE(s, r).

Ao mesmo tempo pode-se preparar uma listagem similar, LISTR(s, gr), que contém todos os receptores potenciais do usuário s; isto permite, mais tarde, obter a condição de que o usuário s enviou, uma por uma, todas as mensagens e que pode, depois, conseqüentemente, retornar ao estado passivo.

Resumindo, observando o modelo parcial na figura IV.5, pode-se dizer que o objetivo desta 1ª fase é o seguinte:

- Permitir que o usuário s passe do estado passivo ao ativo, representados pelos predicados PASSIVO(u), PREP/TRANSM(u) e ATIV/TRANSM(u);
- Separar as mensagens associadas ao usuário s, que estão agrupadas dentro de POOL/MENSAGEM (s, gr) na forma <s, gr>, e colocá-las, uma por uma, no predicado FONTE(s, r), permitindo,

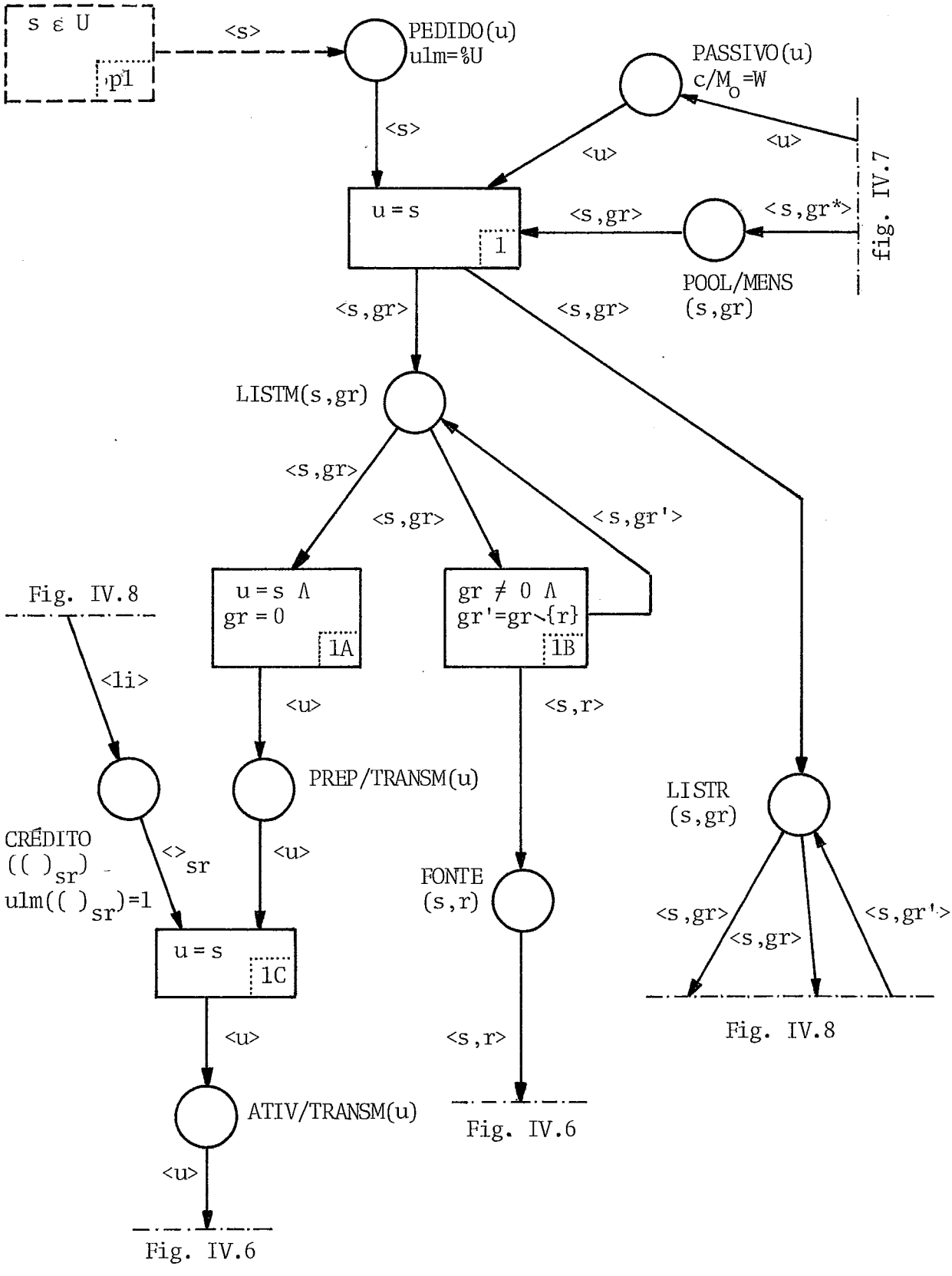


Fig. IV.5: Modelo parcial, representando a 1ª fase: associar ao usuário s as mensagens do grupo, <s,gr>, e listar os receptores, r ε gr, r ≠ s, destas mensagens.

assim, o tratamento de mensagens <s, r> de uma maneira individual.

- Preparar uma lista dos receptores em questão, LISTR(s, gr), que permitirá, mais tarde, a verificação se todas as mensagens do usuário s foram enviadas.

Como observação adicional, tendo em vista a continuação da modelagem, pode-se dizer que a mensagem <s, r> será submetida, por exemplo, "dentro do predicado FONTE(s,r)", a algum processamento como, por exemplo, a colocação de códigos adicionais.

Consideramos que isto é suficiente para dar alguma idéia sobre esta primeira fase da modelagem. Mas queremos enfatizar que o modelo parcial mostrado na figura IV.5, NÃO representará o estágio final, já que deve, mais tarde, ser incorporado no modelo global e, conseqüentemente, sofrer algum tipo de "reajuste".

IV.3.b - Segunda fase: associação entre usuários, mensagens e caminhos.

A passagem do estado ativo (representado pelos predicados PREP/TRANSM(u) e ATIV/TRANSM(u)), responsável pelo processamento mencionado (não modelado, por enquanto) em relação à FONTE (s, r) e PRONTA (s, r, p), para o estado de transmitir (predicado TRANSMITIR(u)), somente é possível se existe, ou melhor, se está à disposição, além das mensagens em questão, algum caminho virtual entre a fonte e a desejada destinação. Para fugir, neste momento, do problema complexo de como modelar a escolha deste caminho, criamos primeiramente um predicado chamado de CAMINHOS/VIRTUAIS (s, r, cv) que contém, como possível marcação inicial, o conjunto de todos os caminhos virtuais.

Agora, pensando que existem somente alguns destes caminhos que possibilitam uma comunicação eficiente entre algum par de vértices, podemos "juntar" aqueles caminhos num feixe de caminhos ('Wegbündel'), fc ε FC, que assim ficaria associado ao par de vértices em questão. Isto pode ser feito na própria

transição, usando alguma fórmula adequada de transição (p. ex., $fc_{ij} = f(cv_{ij},s,r)$), o que leva, assim, ao predicado em questão, ou seja, FEIXE/POSSÍVEL (s,r,fc). Numa transição subseqüente, pode-se escolher um, e somente um, dos caminhos deste feixe em questão, usando, por exemplo, a fórmula $c_{ij} = sel(fc_{ij})$.

Com respeito à mensagem pode-se dizer que ela deveria ser "embrulhada", isto é, tem que receber os endereços necessários e eventuais informações adicionais. Isto pode ser feito usando uma fórmula adequada de transição, do tipo $p_i = f(<s,r>)$ que assim forneceria uma mensagem do tipo <s,r,p>, quer dizer, pronta para transmissão, representada pelo predicado PRONTA (s,r,p).

Assim temos mais uma pré-condição para disparar a transição 2A (a outra foi a dos caminhos possíveis agrupados num feixe associado a s e r).

A mensagem, estando agora toda preparada, pode ser enviada e colocada em trânsito, predicado M-em-TRÂNSITO(s,r,p). Para se fazer isto, precisamos ainda do emissor s, passando do estado ATIV/TRANSM(u) para o de realmente TRANSMITIR(u), e do caminho escolhido dentro do feixe de caminhos possíveis. Esta escolha, feita como já mencionada, na própria transição $c_{ij} = sel(fc_{ij})$, permite a "fixação" deste caminho, CAMINHO/ESCOLHIDO(s,r,c), e da liberação dos demais.

Resumindo, podemos dizer que os objetivos desta 2ª fase são os seguintes (compare com a figura IV.6):

- Permitir ao usuário s passar da primeira fase ativa para a próxima, ou seja, para a verdadeira transmissão, TRANSMITIR(u);
- Preparar a mensagem individual, <s,r>, de uma maneira tal que contenha todas as informações necessárias (endereços, controles, etc.) e possa, assim, ser realmente colocada em trânsito, M-em-TRÂNSITO(s,r,p);

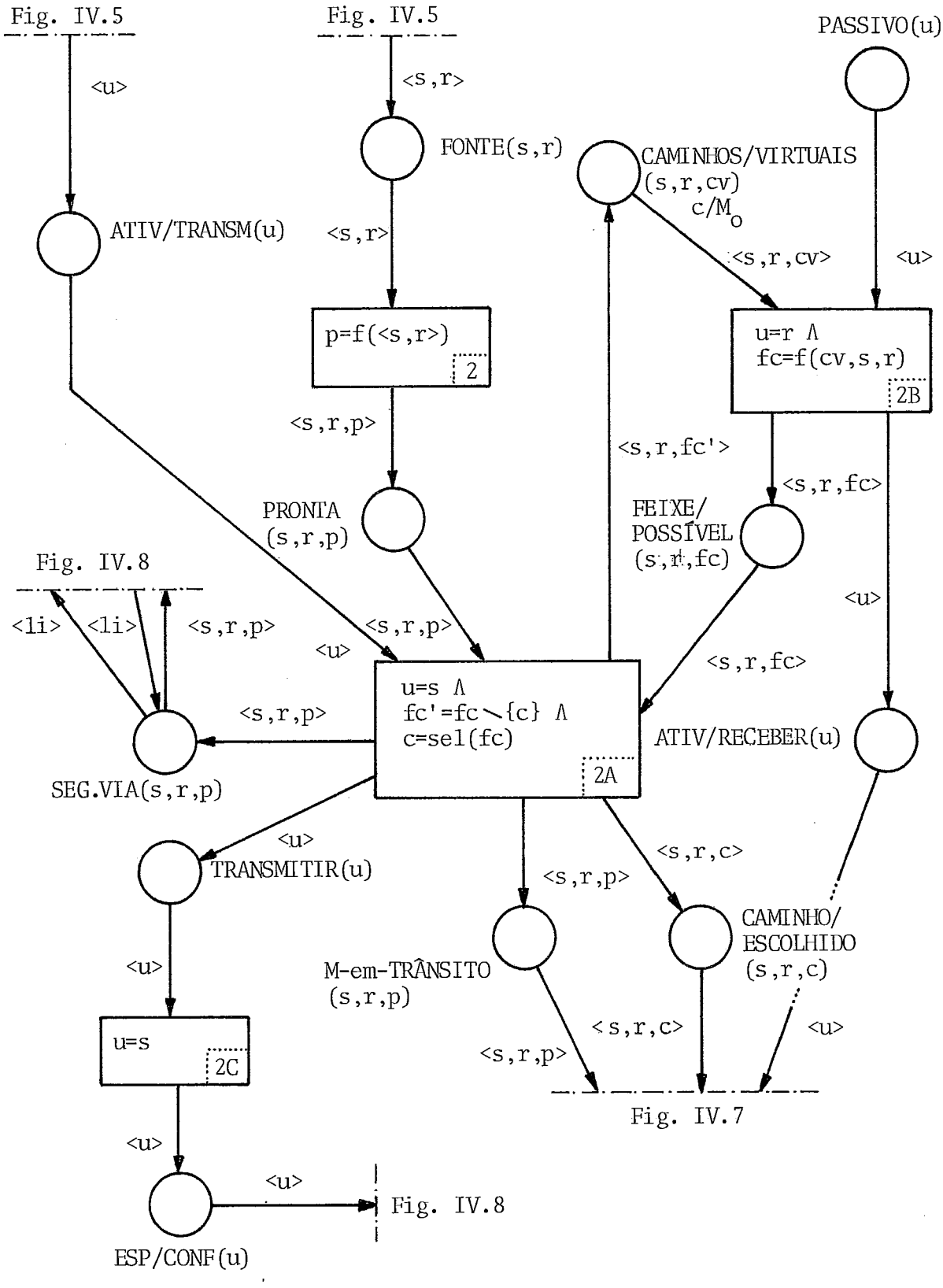


Fig. IV.6: Modelo parcial (2ª fase) em relação à passagem para o estado de transmitir, mostrando a dependência desta passagem da preparação da própria mensagem e da escolha de um determinado caminho.

- Para que estes dois objetivos (relacionados ao usuário e à mensagem) possam ser realizados, deve-se escolher, dentro de um feixe de caminhos possíveis, um único caminho para efetuar a transmissão, CAMINHO/ESCOLHIDO(s,r,c).

Como observação adicional podemos ainda mencionar o fato de que é necessário reter uma 2ª via da mensagem (uma cópia, como é chamada na literatura). Esta cópia serve como 'back-up' no caso em que a própria mensagem se perca na transmissão ou seja danificada, etc.

Assim, pode-se também dizer que o usuário, uma vez terminada a transmissão (que é independente da recepção), passa para um estado que podemos chamar de "esperando a confirmação" de que a mensagem chegou corretamente na destinação. Representamos isto com o predicado ESP/CONF(u).

Terminamos, assim, as observações relativas a esta fase, isto é, de preparação e transmissão. Agora, como o usuários fica no estado ESP/CONF(u) até que ele receba a confirmação em questão, os acontecimentos em torno do receptor r, descritos na própria fase, tornam-se interessante para nós.

IV.3.c - Terceira fase: a recepção de uma mensagem.

Da mesma maneira que não modelamos, em relação ao estado ATIV/TRANSM(u) e à situação da mensagem em FONTE(s,r), o processamento interno (de preparação dos pacotes, por exemplo), não vamos tentar, neste nível de investigação, modelar todo o processamento em relação à recepção das mensagens.

Então, ficamos somente com os pontos principais que podem ser representados, de um lado, por uma possível seqüência de estados que o usuário-receptor, r, possa assumir e, de outro lado, por uma seqüência que reflete a "situação" da mensagem. Assim temos predicados tais como ATIV/RECEBER(u), VERIFICANDO(u) e DESPACHANDO(u), relacionado ao usuário r, e M-em-TRÂNSITO(s, r,p), DESTINAÇÃO(s,r,p,b) e ACEITA(s,r,b), relacionados à mensagem, <s,r> (figura IV.7).

Para explicar o que acontece nesta fase, podemos começar com observações a respeito do caminho escolhido. Uma vez que a mensagem <s,r,p> chegou ao vértice/destinação, pode-se liberar o caminho e colocá-lo à disposição para outros usuários (observação: lembrar que estamos somente modelando a versão 1 do ARPANET que não previa, ainda, reserva de recursos, tais como caminhos, 'buffers', etc, antes do envio de uma mensagem. Esta liberação, no modelo, se faz simplesmente através de uma função na transição 3 (usando, por exemplo, cv' = cv U {c}) que incorpore este caminho selecionado no 'pool' de todos os caminhos possíveis, predicado CAMINHOS/VIRTUAIS(s,r,cv).

Já os acontecimentos em torno da mensagem são um pouco mais complexos. Vejamos: o disparo da transição 3 (veja figura IV.7) estava preparado devido às pré-condições satisfeitas, mas devemos observar que as pós-condições também deverão estar satisfeitas. Neste aspecto deve-se, por exemplo, incluir a necessidade de haver 'buffers' à disposição no vértice/destinação. Isto pode ser modelado em torno do predicado DESTINAÇÃO(s,r,p,b), ou seja, os 'buffers' são "fornecidos", via transição 3A, a partir de um 'pool' de 'buffers' (predicado POOL/BUFFER(b)). A fórmula b_{sr} = f(b,<s,r,p>), na transição 3A, simplesmente quer expressar que o fornecimento dos 'buffers' b_{sr} necessários para receber a mensagem <s,r> depende do espaço disponível b e também, da própria mensagem.

Aqui vemos ainda, como fator importante, porque nas versões 2 e 3 do ARPANET foi incluída a reserva prévia dos recursos; exemplificando podemos dizer que a transição 3A, responsável por colocar à disposição espaço na memória (os 'buffers'), não tem nenhuma relação com o emissor. Assim, pode acontecer que a mensagem tenha sido colocada em trânsito, mas que não haja espaço disponível para recebê-la. Veremos uma possível solução para este problema em trabalhos posteriores.

Admitindo, por enquanto, que haja espaço disponível, a mensagem pode chegar à destinação, predicado DESTINAÇÃO(s,r,p,b), onde será submetida a diversos testes (se houver erros na trans

missão, por exemplo) até que possa ser fornecida a condição "satisfeita" e ela, a mensagem, possa ser considerada como sendo aceita, modelado pelo predicado ACEITA(s,r,b). Se houver algum problema na transmissão, haveria a condição "não-satisfeita" e a transição 3C passaria a ser disparada, mostrando, assim, eventualmente, uma das razões porque um pedido de transmissão não pode ser realizado.

Mas, uma vez aceita, a mensagem pode ser despachada em direção ao HOST/DESTINAÇÃO, modelado aqui, neste exemplo, pelo "retorno" da mensagem <s,r> ao POOL/MENSAGEM(s,gr), fechando assim o "ciclo da vida" de uma mensagem.

Ainda, relacionado à mensagem temos que modelar o mecanismo de confirmação de uma mensagem com a finalidade de liberar o emissor. Assim, temos que gerar uma mensagem de controle, por exemplo, mc, que retorna, via algum caminho privilegiado, de r(rεEndR) para s(sεEndE), modelado pelo predicado MC/TRÂNSITO (mc,r,s), ao lugar da emissão da mensagem <s,r>. Usamos para este fim uma fórmula de transição do tipo mc = f(<s,r>) satisfazendo a inscrição no arco adjacente, <mc,r,s>, que indica a relação causal com o predicado em questão, MC/TRÂNSITO(mc,r,s).

Então, falta somente a modelagem em relação ao USUÁRIO/RECEPTOR. Como já indicado no início desta fase, haverá alguma "associação" entre a situação da mensagem e o USUÁRIO/RECEPTOR. Assim temos o estado VERIFICANDO(u) relacionado ao predicado DESTINAÇÃO(s,r,p,b), associação que verifica, por exemplo, se a mensagem foi recebida corretamente. O próximo estado, o de DESPACHANDO(u), será o último antes de retornar ao estado PASSIVO(u). Como já diz o nome, neste estado o receptor libera a mensagem recebida para que seja, se for declarada correta, enviada ao hospedeiro associado.

Resumindo, então, os objetivos desta 3ª fase, podemos dizer (observando a figura IV.7):

- O caminho escolhido para a transmissão da mensagem é, depois de ter conduzido a mensagem, imediatamente liberado;

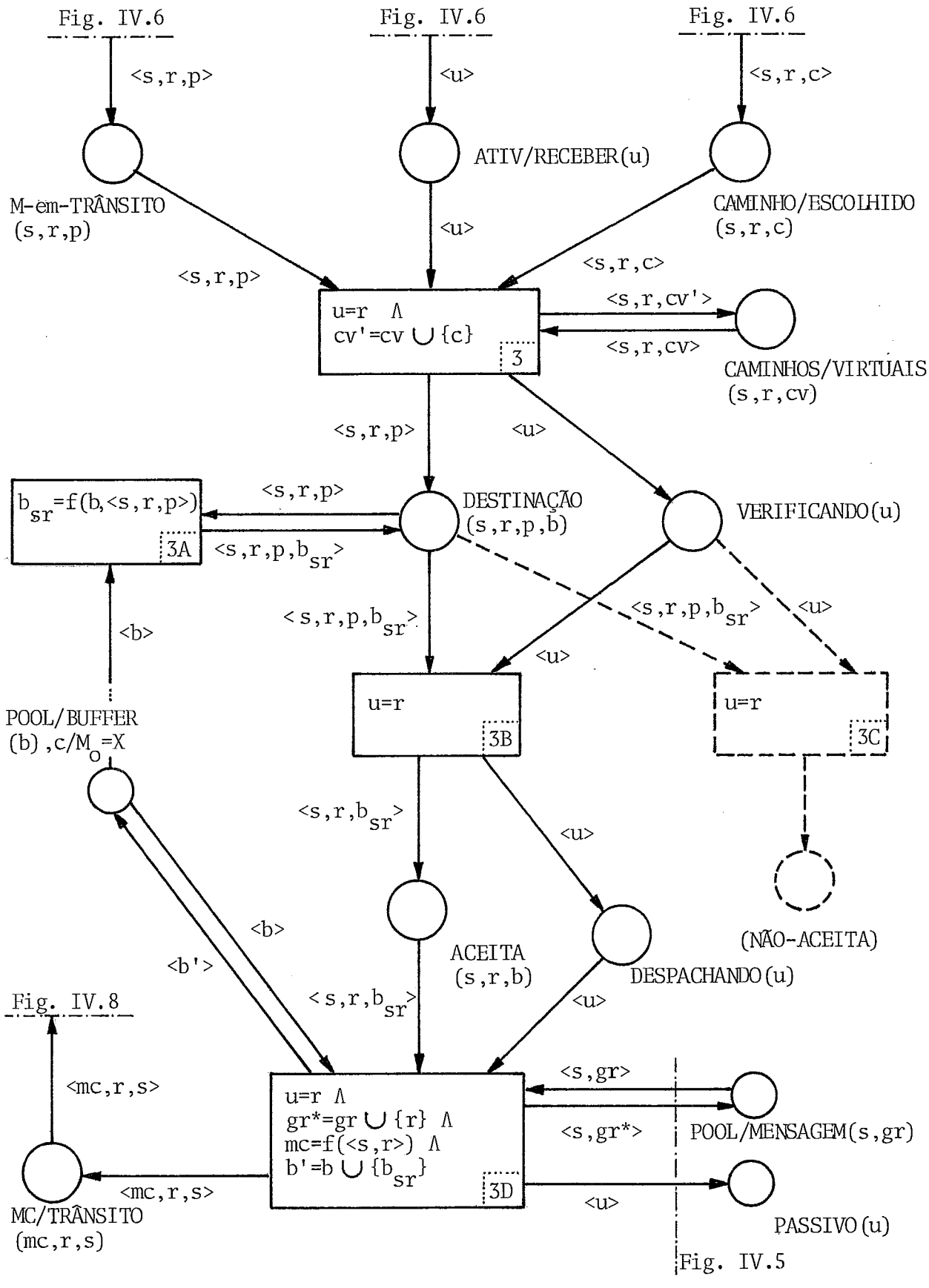


Fig. IV.7: Modelo parcial em relação à recepção de uma mensagem (3ª fase).

- A mensagem é recebida, processada e despachada. Gera-se ao mesmo tempo uma confirmação. Esta será tratada, como mensagem de controle, isto é, de maneira privilegiada, usando, para este fim, caminhos "especiais";

- O receptor simplesmente acompanha a mensagem e, depois do despacho final da mensagem, retorna ao estado passivo.

A próxima fase deve tratar a desativação do usuário/emissor, isto é, tem que ser verificado se todas as mensagens do usuário em questão foram enviadas e confirmadas. Mostraremos isto na seção IV.3.d.

IV.3.d - Quarta fase: a desativação do usuário/emissor.

Vimos que o receptor r, tão logo todas as atividades de recepção, verificação e confirmação terminaram, retornou ao estado passivo, modelado pelo predicado do mesmo nome, PASSIVO(u). Isto acontece depois de ter despachado a mensagem <s,r> em direção ao PROCESSO/RECEPTOR (considerado fora do horizonte de observação), representado pelo retorno da mensagem ao predicado POOL/MENSAGEM(s,gr) e se ter enviado a confirmação relativa a esta mensagem, modelado pela tupla <mc,r,s>.

Embora, para o emissor, a tarefa de transmitir a mensagem em questão também já tenha terminado (vide 2ª fase, figura IV.6), sabemos que ele está ainda esperando uma confirmação para saber se a mensagem chegou, corretamente, no receptor. Assim, somente no momento em que ele tem esta confirmação, ele está liberado para enviar uma eventual próxima mensagem ao mesmo receptor.

Vamos então representar esta restrição usando um predicado do tipo CRÉDITO($( )_{sr}$); este terá, neste caso, somente um 'token', para cada par s-r, à disposição, sendo a liberação deste 'token' relacionado ao "saldo" disponível deste crédito (figura IV.8). Observe, ainda, que no caso em que a confirmação é negativa, teríamos que modelar este fato com uma transição similar àquela de 4C; assim, em vez da função para liberar, li = f(mc,RFNM<s,r>),

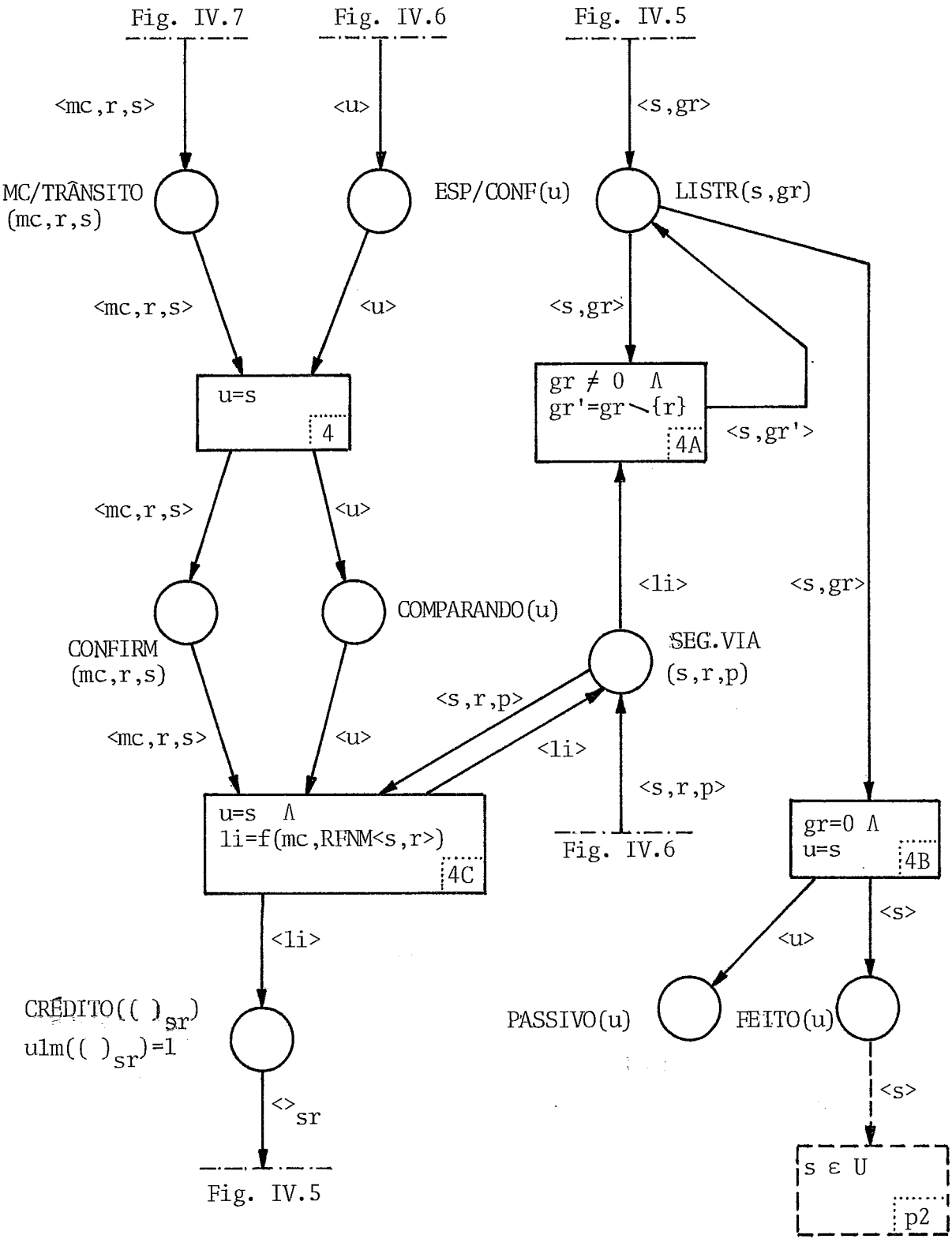


Fig. IV.8: Modelo parcial (4ª fase) sobre o envio de outras mensagens e o retorno do emissor, após ter enviado todas as mensagens, ao estado passivo.

teríamos uma outra, nli = f(mc,NEG<s,r>), que indicaria algum outro tipo de providência a ser tomada (por exemplo, investigar ou ativar uma retransmissão). Este fato não será modelado agora, mas será investigado em um próximo trabalho, onde modelaremos os acontecimentos indicados na figura II.5.

Agora, para retornar ao estado passivo, predicado PASSIVO(u), o usuário precisa verificar se todas as mensagens, isto é, não somente aquelas que têm um único receptor, mas como receptores todos os usuários/receptores contidos na lista LISTR(s,gr), foram confirmadas. Modelamos isto, usando o mencionado predicado LISTR(s, gr), como já indicado na figura IV.5, que somente no momento da confirmação de todas as mensagens do grupo <s,gr> fornece a condição para que a transição em questão, 4B, possa disparar. Ao mesmo tempo ele indica que o pedido do usuário s foi satisfeito, modelado pelo predicado FEITO(u); (vide figura IV.8).

Então, resumindo os objetivos desta 4ª fase, pode-se dizer:

- As cópias (2as vias) das mensagens são, depois que o emissor recebeu as confirmações, eliminadas. Ao mesmo tempo é ativado o mecanismo de crédito, permitindo, neste caso, que somente uma mensagem fique no sistema de transmissão sem ser confirmada;
- O emissor retorna, depois que enviou todas as mensagens, ao estado passivo, comunicando ao mesmo tempo que seu pedido foi feito.

Tendo modelado agora esta 4ª fase, podemos juntar todas as fases para obter o modelo global dos acontecimentos em torno da figura II.2. Faremos isto na seção IV.4 deste trabalho.

IV.4 - O modelo nº 1, na base de 'PrT-Net', representando o esquema simplificado de um controle 'end-to-end' em redes de computadores.

Juntando os diferentes modelos parciais (figura IV.5 até a figura IV.8), obtemos agora algum início para investigações

mais realistas e não somente neste nível, digamos, descritivo. Esta observação não quer diminuir a validade desta fase descritiva, já que a partir dela podemos obter uma base sólida e representativa para a compreensão de nosso problema de fluxo.

Mostramos, então, na figura IV.9, o 'PrT-Net' que representa o esquema simplificado (figura II.2) de controle de fluxo tipo 'end-to-end' e uma variante na representação da matriz de incidência (fig. IV.10) associada a este 'PrT-Net'. Como a matriz é muito esparsa (veja capítulo III), decidimos usar esta variante que mostra somente as informações úteis, completada por uma lista das transições (e suas fórmulas) envolvidas no controle de fluxo deste modelo nº 1 (fig. IV.11).

IV.4.a - Observações sobre eventuais testes em relação à consistência do modelo nº 1.

Em seguida indicamos como se poderia testar, através de marcações e de S-invariantes, se há "equilíbrio" no sistema. Contudo, uma investigação mais completa deixamos para uma oportunidade melhor, a saber, quando estivermos investigando um sistema mais representativo, e não somente um modelo-exemplo.

Então, tendo agora o modelo global (figura IV.9) da versão simplificada de controle de fluxo (figura II.2), e também uma variante da matriz de incidência associada (figura IV.10), poder-se-ia investigar se o modelo é representativo, consistente, etc. Pode-se, para este fim, usar alguns conceitos, tais como, por exemplo, a verificação das várias camadas desta matriz com a ajuda de marcações iniciais, a fórmula básica, $i' M = i' M_o$ e S-invariantes. Mas como já dissemos, vamos somente indicar o que poderia ser feito.

Por exemplo, a parte de "estados dos usuários" (modelados pelos predicados PASSIVO(u), ATIV/TRANSM(u), etc.) seria relativamente fácil. Usando a idéia que os usuários sempre devem estar em algum estado, isto é, num predicado, podemos dizer que a "soma" nas colunas e linhas da matriz de incidência deve sempre

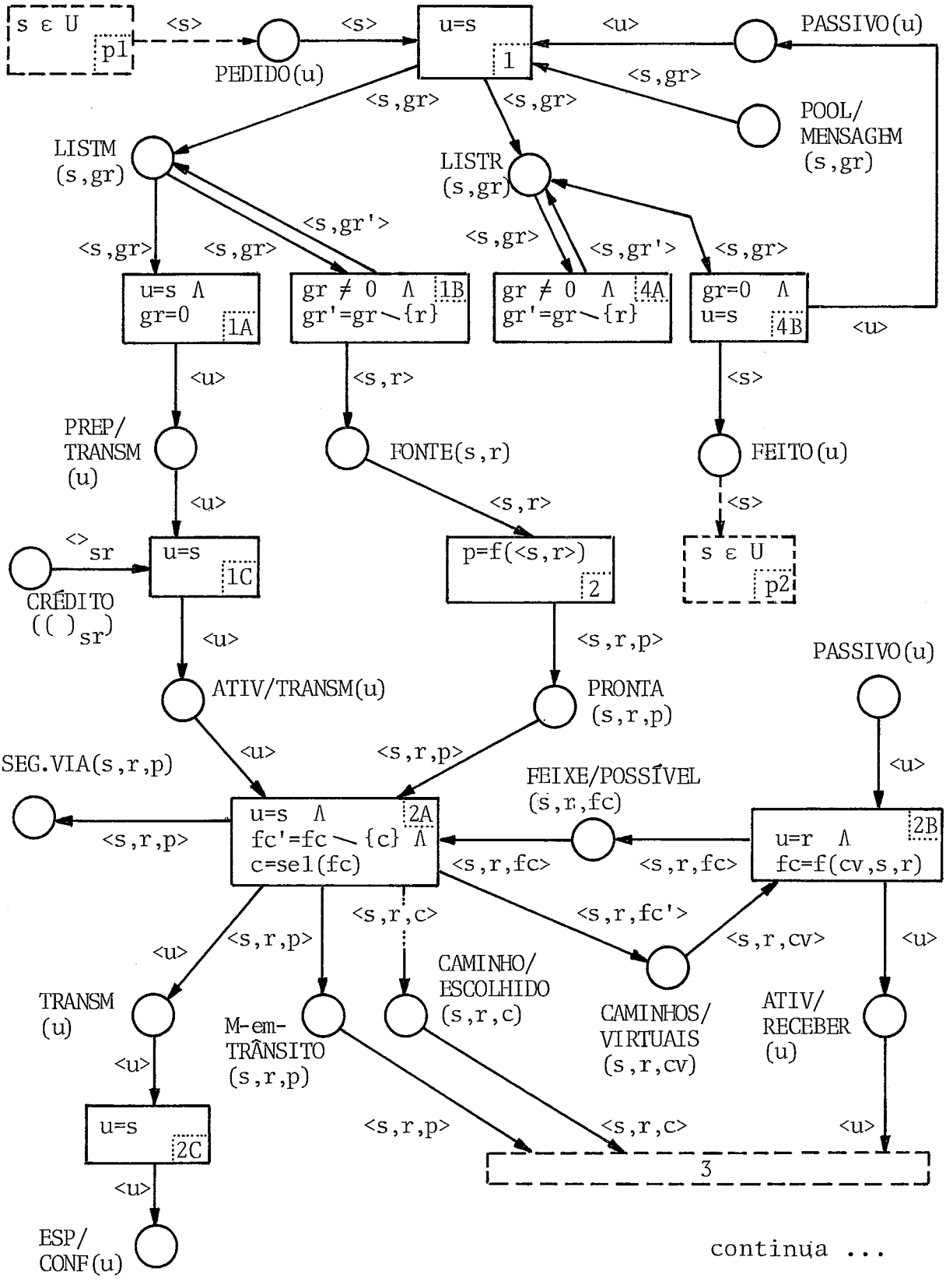


Fig. IV.9: Modelo nº 1, na base de 'PrT-Net', representando o esquema simplificado de um controle 'end-to-end' em redes de computadores (baseado nos modelos parciais fig. IV.5 até fig. IV.8).

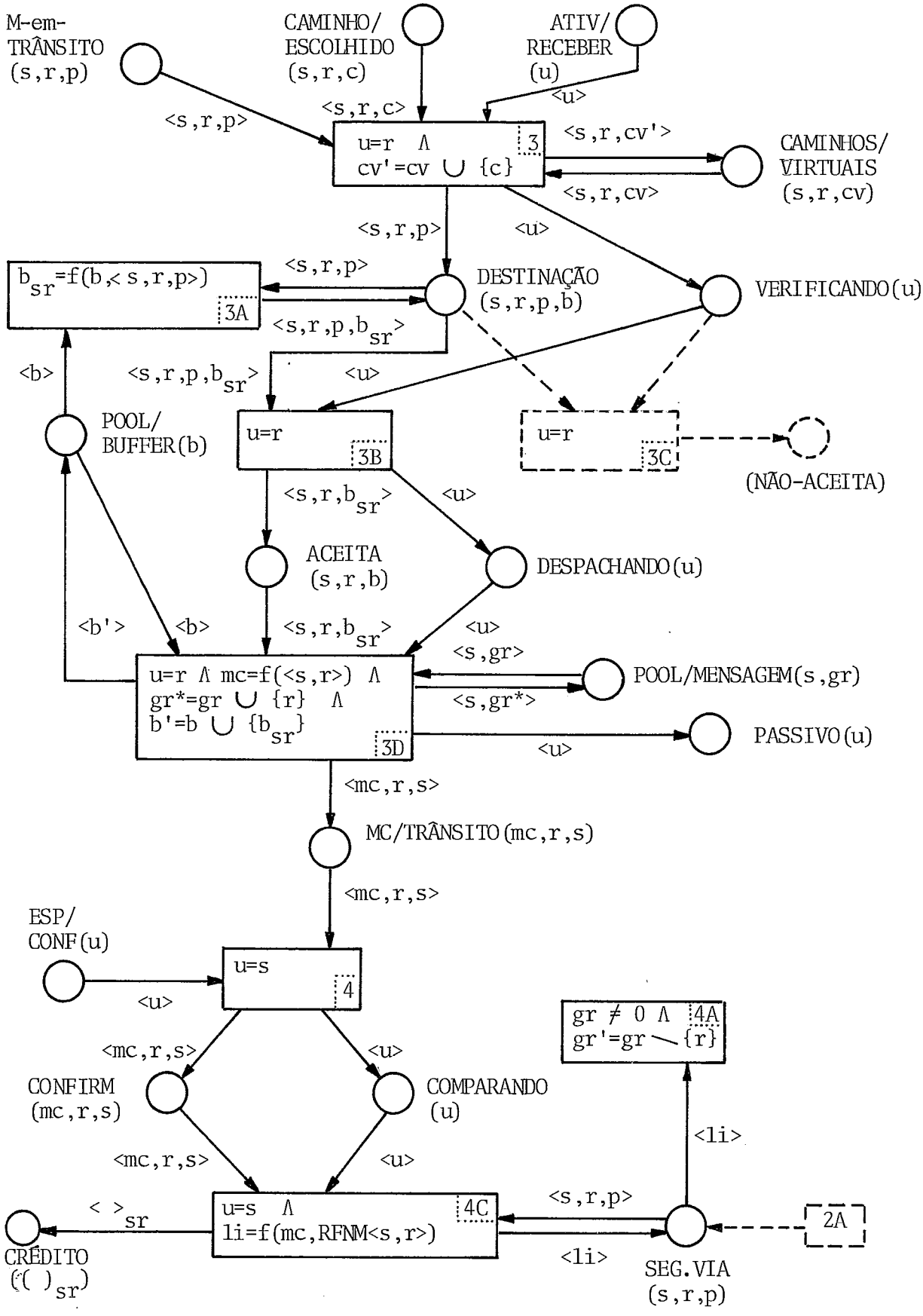


Fig. IV.9 (continuação - final): Modelo nº 1, ...

Predicados: MENSAGEM	relação causal com as transições: - ≙ precondição	trans	+ ≙ poscondição	trans
ACEITA(s,r,b)	$-\langle s,r,b_{sr}\rangle$	3D	$+\langle s,r,b_{sr}\rangle$	3B
DESTINAÇÃO (s,r,p,b)	$-\langle s,r,p\rangle$	3A	$+\langle s,r,p\rangle$	3
	$-\langle s,r,p,b_{sr}\rangle$	3B	$+\langle s,r,p,b_{sr}\rangle$	3A
	$-\langle s,r,p,b_{sr}\rangle$	3C		
FONTE(s,r)	$-\langle s,r\rangle$	2	$+\langle s,r\rangle$	1B
LISTM(s,gr)	$-\langle s,gr\rangle$	1A	$+\langle s,gr\rangle$	1
	$-\langle s,gr\rangle$	1B	$+\langle s,gr'\rangle$	1B
LISTR(s,gr)	$-\langle s,gr\rangle$	4A	$+\langle s,gr\rangle$	1
	$-\langle s,gr\rangle$	4B	$+\langle s,gr'\rangle$	4A
M-em-TRÂNSITO (s,r,p)	$-\langle s,r,p\rangle$	3	$+\langle s,r,p\rangle$	2A
NÃO-ACEITA (s,r,b)	indefinido		$+\langle s,r,b_{sr}\rangle$	3C
POOL/MENSAGEM (s,gr)	$-\langle s,gr\rangle$	1	$+\langle s,gr^*\rangle$	3D
	$-\langle s,gr\rangle$	3D		
PRONTA(s,r,p)	$-\langle s,r,p\rangle$	2A	$+\langle s,r,p\rangle$	2
SEG.VIA(s,r,p)	$-\langle li\rangle$	4A	$+\langle s,r,p\rangle$	2A
	$-\langle s,r,p\rangle$	4C	$+\langle li\rangle$	4C

Fig. IV.10: Lista dos principais predicados da fig. IV.9, agrupados segundo as suas diferentes "naturezas". Parte 1: MENSAGEM.

Predicados: USUÁRIOS	relação causal com as transições: - ≙ precondição	trans	+ ≙ poscondição	trans
ATIV/RECEBER(u)	-<u>	3	+<u>	2B
ATIV/TRANSM(u)	-<u>	2A	+<u>	1C
COMPARANDO(u)	-<u>	4C	+<u>	4
DESPACHANDO(u)	-<u>	3D	+<u>	3B
ESP/CONF(u)	-<u>	4	+<u>	2C
FEITO(u)	-<s>	p2	+<s>	4B
PASSIVO(u)	-<u>	1	+<u>	3D
	-<u>	2B	+<u>	4B
PEDIDO(u)	-<s>	1	+<s>	p1
PREP/TRANSM(u)	-<u>	1C	+<u>	1A
TRANSMITINDO(u)	-<u>	2C	+<u>	2A
VERIFICANDO(u)	-<u>	3B	+<u>	3
	-<u>	3C		
Predicados: CAMINHOS				
CAMINHO/ESCOLHIDO (s,r,c)	-<s,r,c>	3	+<s,r,c>	2A
CAMINHOS/VIRTUAIS (s,r,cv)	-<s,r,cv>	2B	+<s,r,fc'>	2A
	-<s,r,cv>	3	+<s,r,cv'>	3
FEIXE/POSSÍVEL (s,r,fc)	-<s,r,fc>	2A	+<s,r,fc>	2B
Predicados: AUXILIARES				
CONFIRM(mc,r,s)	-<mc,r,s>	4C	+<mc,r,s>	4
CRÉDITO($)_{sr}$	$-<>_{sr}$	1C	$+<>_{sr}$	4C
MC/TRÂNSITO (mc,r,s)	-<mc,r,s>	4	+<mc,r,s>	3D
POOL/BUFFER(b)	-<b>	3A	+<b'>	3D
	-<b>	3D		

Fig. IV.10 (cont.): Lista dos principais predicados .... Parte 2: USUÁRIOS, Parte 3: CAMINHOS, Parte 4: AUXILIARES.

T R A N S I Ç Õ E S :	
no.	fórmulas
p1	$s \in U$
p2	$s \in U$
1	$u=s$
1A	$u=s \wedge gr=0$
1B	$gr\neq 0 \wedge gr'=gr\setminus\{r\}$
1C	$u=s$
2	$p=f(<s,r>)$
2A	$u=s \wedge fc'=fc\setminus\{c\} \wedge c=sel(fc)$
2B	$u=r \wedge fc=f(cv,s,r)$
2C	$u=s$
3	$u=r \wedge cv'=cv\cup\{c\}$
3A	$b_{sr}=f(b,<s,r,p>)$
3B	$u=r$
3C	$u=r$
3D	$u=r \wedge gr^*=gr\cup\{r\} \wedge mc=f(<s,r>) \wedge b'=b\cup\{b_{sr}\}$
4	$u=s$
4A	$gr\neq 0 \wedge gr'=gr\setminus\{r\}$
4B	$gr=0 \wedge u=s$
4C	$u=s \wedge li=f(mc,RFNM<s,r>)$

Fig. IV.11: Tabela das transições, e suas principais fórmulas, envolvidas no controle de fluxo no modelo no. 1.

ser zero, já que um usuário não pode, e não deve, se "perder numa transição".

A variante usada por nós, onde "agrupamos" determinados predicados, permite este tipo de investigações, ao menos, neste modelo simplificado. Assim vemos, p. ex., no que diz respeito ao usuário s, que colocou o "pedido", a idéia que o usuário s que "entrou do 'PrT-Net' somente pode 'sair'" ou pelo predicado FEITO(u) ou pelo predicado REJEITO(u).

Já a parte das "mensagens" é um pouco mais complexa. Vejamos: esta parte (representada pelos predicados tais como, POOL/MENSAGEM(s,gr), FONTE(s,gr), etc), indica, individualmente, as mensagens associadas a um determinado usuário. Devemos então considerar que, para cada usuário s, haverá a necessidade de investigações isoladas, já que cada usuário pode enviar mensagens, às vezes mais que uma, para cada outro usuário.

Também, as tuplas, que representam as mensagens, não são tão mais simples como aquelas para os usuários, já que contêm informações adicionais tais como, em relação ao pacote formado, aos buffers necessários, etc. Assim, deve-se levar em consideração os elementos da matriz de incidência junto com as fórmulas de transição usadas. Mas, em princípio, vale também aqui que uma mensagem sempre deve estar em algum lugar, isto é, não pode se perder no modelo. Cabe a nós verificar no modelo se qualquer situação ficou realmente representada.

Temos ainda que verificar se os predicados auxiliares (por exemplo, SEG. VIA(s,r,p), CONFIRMAÇÃO(mc,r,s), etc.), satisfazem à idéia de que eles representam algo artificialmente criado em relação, por exemplo, à mensagem. Assim, depois de ter cumprido a sua função auxiliar, estes "artifícios" devem novamente "desaparecer" em determinados instantes dinâmicos no modelo.

Agora, a parte "caminho" é a verdadeira chave para um bom funcionamento em conjunto desta estrutura distribuída. Isto vem do fato, de que os caminhos são recursos fornecidos pela sub-

rede de comunicação e, portanto, alheio à vontade do usuário. Ao contrário da formação fácil do par "usuário-mensagem", (dependendo somente de informações locais), a associação "mensagem-caminho" depende não somente de informações locais mas, sobretudo, de informações presentes em algum lugar da rede mas raramente, na sua forma completa, no lugar da decisão. Como este processo da escolha de algum caminho é um procedimento bem complexo, deixamos, nesta fase de modelagem, toda esta complexidade reunida nas transições 2B e 2A, isto é, nas fórmulas de transição usadas, fc = f(cv,s,r) e c = sel(fc), respectivamente. Contentamo-nos, por enquanto, somente com a associação de um determinado caminho, representado pela tupla <s,r,c>, à mensagem, <s,r>, em questão.

No decorrer desta pesquisa dedicaremos um espaço maior a este problema, isto é, à "abertura" da escolha feita nas transições mencionadas.

Como já mencionamos inicialmente, este modelo-exemplo serve, principalmente, como primeiro passo no sentido de entender como unir a modelagem com as ferramentas à disposição. Como resultado principal podemos destacar que o modelo resultante facilitou o estudo do problema (somente pensando nas camadas e subcamadas) mas que, por outro lado, nos deixou uma boa quantidade de questões. Veja estes pontos na seção IV.4.b.

IV.4.b - Observações sobre pontos vulneráveis e questões não resolvidas.

Terminaremos com uma enumeração dos principais pontos vulneráveis, mal resolvidos, faltosos, etc., deste modelo simplificado sem, contudo, fazer nenhuma tentativa de revisão. Esta decisão se deve ao fato de que este modelo servirá, principalmente, para fins didáticos, por exemplo, mostrar COMO criar o modelo 'PrT-Net' e QUAIS os pontos deficientes.

Nos próximos capítulos, porém, tentaremos modelar um esquema bem mais complexo (p.ex., o esquema completo mostrado na figura II.5) com todas as suas melhorias em relação a este atual esque

ma simplificado.

Temos então, entre outras, as seguintes questões:

- Há equilíbrio em relação a todos os componentes? (cópias, mensagens, caminhos, etc.)?
- Como podemos modelar uma rejeição?
- Como podemos indicar que um usuário pode transmitir e receber ao mesmo tempo?
- Como se toma uma decisão se não houver um 'time-out'?
- O caminho não está reservado de antemão, assim como o receptor também não passa, explicitamente, para o estado de receber. Então, como fazer isto?
- etc...

Vimos que uma coisa ficou bem clara: mesmo sem conhecer as modificações e limitações feitas atualmente (na versão 2 e 3 do ARPANET), pode-se afirmar que este tipo de modelagem é válido porque mostrou, num nível de abstração razoável, claramente, as deficiências desta esquema simplificado.

IV.5 - Resumo do capítulo IV.

Mostramos, neste capítulo IV, o desenvolvimento de um modelo simplificado. O objetivo principal foi a familiarização com o 'PrT-Net', já aplicado a um exemplo real. Vimos que, uma vez definido o horizonte de observação, pode-se dividir a formação do modelo em fases que seguem, em nosso caso, o funcionamento seqüencial mostrado na figura II.2.

Juntando as fases, obtemos um primeiro resultado, o modelo nº 1 (figura IV.9), que permitiu descobrir claramente os pontos fracos deste esquema. Mas o modelo é válido como base e será utilizado nos próximos capítulos. Lá, introduziremos, passo a passo, grande parte das melhorias (tais como: ACK, NAK,

'time-out', multipacotes, etc.), conhecidas através do estudo do ARPANET.

C A P Í T U L O V

O uso de 'PrT-Nets' para modelar os acontecimentos nos vértices intermediários na transmissão de pacotes em redes de computadores.

V.1 - Introdução.

(Observação: uma versão preliminar deste capítulo foi publicada como relatório técnico, SCHWARZ (35)).

Antes de estudar os problemas de como considerar, na modelagem, os vértices intermediários, queremos mais uma vez, lembrar o que já fizemos como trabalhos preparatórios.

Entre estes trabalhos se encontra, por exemplo, um que trata do controle de fluxo em redes com linhas de transmissão terrestres, SCHWARZ (30). No trabalho mencionado, estudamos aspectos qualitativos de controle de fluxo em redes de computadores. Depois de uma introdução sobre termos gerais, necessária para fixar uma linguagem comum, descrevemos métodos para o controle de fluxo em canais, para depois ampliar estes conceitos para redes de computadores. Foram também incluídas, nesse trabalho, explicações quantitativas sobre algumas ferramentas principais, notadamente, sobre o crédito, a alocação dos 'buffers' e o roteamento adaptativo.

Um outro trabalho, SCHWARZ (33), também preparatório, tratou da apresentação de uma nova ferramenta para conseguir a modelagem de problemas dinâmicos em redes. Este método é parte da GNT ('General Net Theory'). A GNT foi apresentada em termos gerais, isto é, um pouco sobre a sua história e um pouco sobre os seus objetivos, preocupações e tópicos de destaque. Em seguida foi explicado como se chegou à técnica de 'PrT-Net', usando, para

esta finalidade, conceitos básicos de Petri-Nets ('place/ transition nets'), de sistemas tipo C/E ('condition/event systems') e exemplos associados. Assim, conseguimos apresentar a técnica de 'PrT-Net', usando uma explicação formal e uma outra, detalhada, sobre os componentes deste 'PrT-Net'. Este relatório serviu, entre outros, como base para o capítulo III deste trabalho.

Finalmente, já usando esses dois trabalhos, SCHWARZ (30) e SCHWARZ (33), começamos a modelagem de um sistema simplificado, isto é, a transmissão de um único pacote, a partir de uma fonte até uma destinação, considerando a subrede de comunicação como sendo um único recurso, SCHWARZ (34). Assim, esse último trabalho serviu para obter uma familiarização maior sobre a aplicação da técnica de 'PrT-Net' em problemas de nosso interesse. Usando-o como base para o capítulo IV, podemos afirmar que este primeiro resultado, o modelo no. 1, ajudou muito para compreender o problema em si, isto é, a base do controle de fluxo, e, também, para sentir maior segurança em usar esta técnica nova de 'PrT-Nets'.

Assim, podemos dizer que os preparativos, para conseguir uma modelagem mais realista, chegaram a tal ponto que podemos nos aventurar em modelar um problema um pouco mais avançado. Este objetivo consiste em ampliar o modelo básico, alcançado no trabalho de SCHWARZ (34) e mostrado no capítulo IV, no sentido de incluir os vértices intermediários e, obviamente, todos os problemas associados. Em outras palavras: temos que "ABRIR" a subrede de comunicação para ver onde e como podem, eventualmente, ocorrer conflitos.

V.2 - A transmissão de um pacote simples numa rede de computadores.

Do trabalho feito por SCHWARZ (30), onde descrevemos diferentes métodos de controle de fluxo em canais e em redes, escolhemos o método 'end-to-end', aplicado no ARPANET. Em particular, usaremos a versão 1 que, mesmo sendo substituída hoje pelas ver

sões 2 e 3, mostra muito bem a base para uma transmissão e, também, as complicações que possam surgir durante o funcionamento.

Como este capítulo V é uma continuação direta do capítulo IV, seria o mais lógico apontar as diferenças essenciais entre o esboço simplificado (figura II.2), que somente lida com PAC (pacote) e RFNM ('request for next message'), e aquele que mostra toda a anatomia de uma transmissão (figura II.5).

Então, quais são estas diferenças fundamentais? São elas as diversas filas de espera, os vértices intermediários, uma eventual retransmissão entre vértices vizinhos, baseados em NAK ('negative acknowledgement') ou num t_{out} ('time-out'), e o tratamento do RFNM ('request for next message') como mensagem de controle?

Falaremos, em seguida, neste capítulo V, um pouco sobre alguns destes pontos, a saber, os vértices intermediários, o tratamento dos ACK's ('acknowledgement') e NAK's ('negative ACK'), deixando, entretanto, o tratamento de 'time-out' para o capítulo VI (que investigará, entre outros, os tempos envolvidos em controle de fluxo) e o de RFNM (uma mensagem do tipo controle) para um eventual próximo trabalho onde serão investigadas as mensagens de controle em geral.

V.2.a - As filas de espera em relação a um vértice da subrede de comunicação.

Observando a figura II.5, vemos a indicação de um único tipo de fila; este tipo representa a espera para ter acesso a um vértice. Entretanto, devemos nos perguntar, imediatamente, se este tipo é o suficiente para modelar os acontecimentos reais numa transmissão. Então, cabe aqui a seguinte observação: sabemos que em sistema de computadores digitais há sempre o problema de filas devido ao comportamento predominante, quase exclusivamente seqüencial. Como já foi mencionado por HEART et alii (10), há filas

- relacionadas às tarefas associadas ao hospedeiro, ao sistema, ao modem, etc.;

- de saída para os hospedeiros, para os vértices vizinhos, para o modem das diversas linhas, etc.;
- para transmitir, para receber, para conferir, etc.

Ainda, cada uma destas filas pode ser dividida em relação à prioridade e a outros critérios.

Como não é o objetivo deste trabalho modelar minuciosamente cada uma destas filas, devemos nos concentrar aos seguintes tipos de filas que consideramos como sendo os mais importantes:

- Um tipo, relacionado à entrada num vértice, representa os aspectos de preprocessamento (e das decisões associadas) no que diz respeito à aceitação, à rejeição, à classificação, à fragmentação e formatação, etc.;
- Outro tipo, mais relacionado à saída de um vértice, é aquele que representa as mensagens já alocadas num buffer de saída. Neste caso, a espera na fila (onde a fila é representada pelo espaço alocado neste 'buffer') é relacionada à disponibilidade do servidor, isto é, do transmissor e da linha propriamente dita (incluindo todo o tratamento necessário para a emissão de uma mensagem);
- Finalmente, um tipo de filas que representará os acontecimentos internos num vértice e onde podemos mencionar todo o processamento que tem que ser feito, obrigatoriamente, na UCP. Como exemplo, devemos destacar os cálculos e as decisões para obter um roteamento eficiente que é, afinal, uma das partes mais importantes de controle de fluxo.

Prosseguiremos, então, da seguinte maneira: primeiramente, na seção V.2.a.1, mostraremos como se pode modelar o sistema geral de uma fila usando a técnica de 'PrT-Net'. Depois, tendo esta idéia bem assimilada, modelaremos, em outras seções, as mencionadas filas de entrada, de saída e da UCP (unidade central de processamento, 'CPU'), respectivamente.

V.2.a.1 - O modelo geral de uma fila de espera.

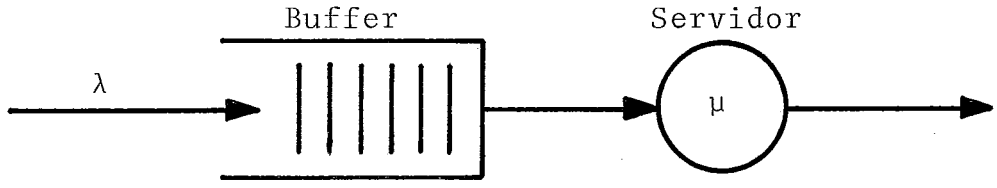
Como já mencionado por SCHWARZ (30), podemos representar uma fila de espera através de um conjunto de parâmetros. Pode-se ver a representação gráfica e o significado dos parâmetros na figura V.1.

Cabe aqui, de imediato, uma observação importante: não queremos substituir, de modo algum, a teoria de filas que se preocupa, principalmente, com o fluxo de alguma comodidade através de um sistema de filas.

Assim, para ela, esta teoria de filas, os parâmetros mais importantes, nos estudos básicos, são aqueles relacionados às distribuições dos tempos entre chegadas e tempos de serviço, A e B, respectivamente, completado pelo parâmetro m que representa o número de utilidades. Numa fase posterior dos estudos, esta teoria começa se preocupar, também, com o número da população N e a capacidade da fila, K.

Nós aqui, neste estudo, estamos preocupados em integrar algum sistema de filas num sistema grande, complexo e real. Assim, precisa-se tomar decisões nos momentos críticos como, por exemplo,no instante quando o 'buffer' de uma fila estiver cheio. Embora existam estudos em teorias de filas que tratam de chamado 'overflow' em filas, achamos que o mais importante é garantir que o espaço K numa fila seja corretamente administrado. Uma vez feito isto, podemos pensar em como atender os elementos armazenados na fila, isto é, nos preocupar com a política P de atendimento.

Assim justificado, chegamos ao momento de investigar como modelar estes dois fatores, isto é, a administração e a estratégia. Sem querer nos preocupar, nesta altura das investigações, com o "casamento" da teoria de filas com a de decisões, propomos a modelagem dos fatores mencionados, principalmente do fator administração, usando a técnica de 'PrT-Net'.



onde: λ ... taxa média de fluxo de entrada
μ ... taxa média de serviço

Parâmetros: A/B/m/K/N/P

A ... Distribuição de tempos entre chegadas de mensagens à rede, ao vértice, etc.;

B ... Distribuição de tempo de serviço no servidor, que pode ser um computador-vértice, a linha de transmissão, etc.;

m ... Número de utilidades, como, p. ex., o número de linhas de comunicação saindo de um vértice;

K ... Capacidade de buffer, como, p. ex., nos vértices intermediários ou no vértice/destinação;

N ... Tamanho da população;

P ... Política de atendimento, como, p. ex., FIFO ('first-in--first-out'), LCFS ('last-come-first-served'), RR ('round robin'), randômico, HOL ('head-of-line'), etc.

Fig. V.1: Representação gráfica de uma fila de espera e seus respectivos parâmetros.

Argumentos e predicados.

Usando os nossos conhecimentos adquiridos nos capítulos III e IV, precisamos, então, argumentos que representem os espaços no buffer da fila e números naturais para a contagem. Assim, temos:

$K = \{k_1, k_2, \ldots, k_n\}$; lugares ou espaços no buffer;

$N = \{1, 2, 3, \ldots\ldots\}$; números naturais.

Com estes argumentos, pode-se, assim, definir os predicados da seguinte maneira:

FILA(arg,k) ; representa a estratégia do atendimento de alguma mensagem no k-ésimo lugar do buffer, ($k \varepsilon K$);

BUFFER(k) ; representa a administração dos espaços k ($k \varepsilon K$) dentro do 'buffer';

$Z_i(n)$; representa contadores-sincronizadores, baseados em n ($n \varepsilon N$), ao que diz respeito à administração do espaço no 'buffer' da fila.

Tendo estas definições básicas, pode-se, agora, explicar o funcionamento propriamente dito.

Funcionamento (fig. V.2)

Para "entrar na fila, precisa-se, além da pós-condição satisfeita (p.ex., a situação na própria fila), que as pré-condições da transição f1 também estejam satisfeitas:

- a tupla <arg>, representando o fato de que o predicado precedente a esta transição fornece o elemento que quer entrar na fila para obter, p.ex., um determinado tratamento;

- a presença de um elemento identificador <z>, isto é, a condição que o z-ésimo espaço dentro de BUFFER(k) está disponível;

- o elemento contador <z>, que indica a posição de um contador-sincronizador $Z_1(n)$.

Então, cada tupla, que passa pela transição f1, "consome" um lugar no buffer; ao mesmo tempo, corrige-se o contador através da fórmula de transição, z' = z+1, mod. n, preparando, assim, a próxima entrada.

Agora, se a tupla <arg,z> terminou o seu tratamento na fila, ela deveria sair da fila e liberar o espaço no buffer. Observe que a identificação <arg,z> da entrada na fila não tem nada a ver com o tratamento dentro da fila. Portanto, o identificador <arg,z> de saída reflete somente que a tupla, que está saindo, é a z-ésima em relação à seqüência de saída da fila.

Assim, temos as seguintes pré-condições para disparar a transição f2:

- A própria situação na fila, ou seja, devido ao tratamento na fila, chegou-se à situação em que o elemento em questão pode ser despachado;
- A posição do segundo contador-sincronizador deverá permitir, através de <z>, que alguma z-ésima tupla, em relação à seqüência de saída da fila, seja liberada para sair.

Como pós-condição temos, por um lado, a disponibilidade do próprio sistema que necessita da tupla <arg> e, do outro lado, o conjunto BUFFER(k) e contador $Z_2(n)$. Assim, o BUFFER(k) incorpora, novamente, um lugar disponível e o contador se prepara, através da fórmula de transição, z' = z+1, mod. n, para dar a permissão necessária para que a próxima tupla, isto é, a (z+1)-ésima, possa sair da fila.

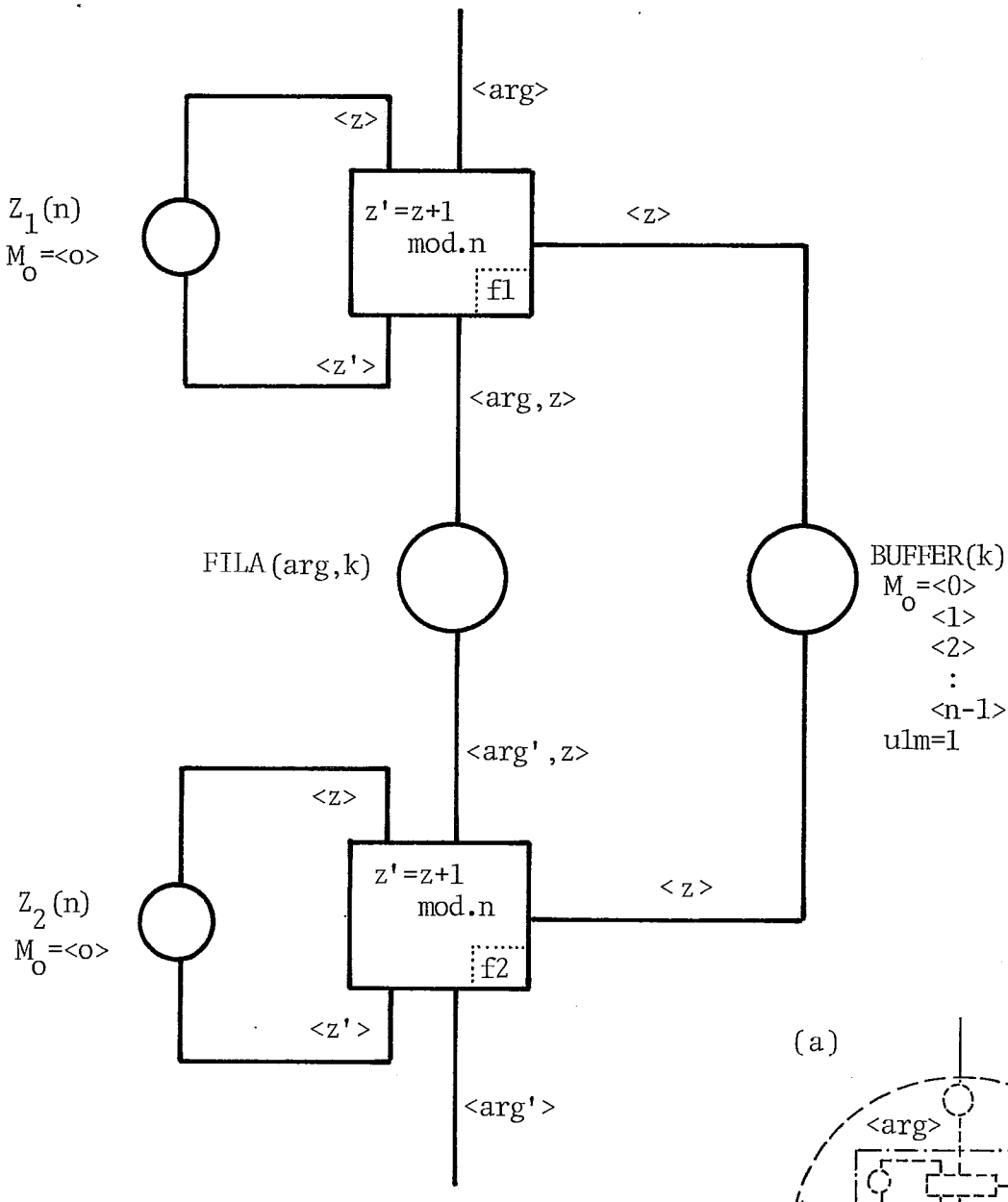
Vemos, então, que, com este modelo simples, garantimos uma administração correta do espaço no buffer sem, contudo, influenciar na política do atendimento na fila e sem restringir as "velocidades" de entrada e saída na fila (obviamente, no caso em que a fila estiver cheia, o sistema se auto-regula, isto é, a velocidade da entrada depende diretamente da liberação do espaço no buffer).

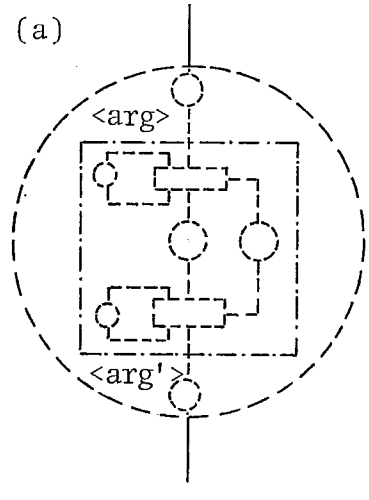
V.2.a.2 - As filas de "entrada" num vértice.

Poderíamos, para fins de modelagem, considerar como fila de entrada aquela mostrada na figura V.3. Este tipo de modelo foi utilizado, p. ex., por LAM (16); se considera, principalmente, a fila do processador como sendo a fila de entrada. Para nós, esta hipótese é somente em parte viável já que queremos tratar os canais de uma maneira independente, ou quase independente. da UCP. Entretanto, podemos usar este modelo para iniciar o nosso estudo e ver como chegar a um modelo mais de acordo com nossos objetivos.

Então, devemos considerar, entre outros, os seguintes pontos:

- Podemos tentar tratar as filas de entrada em paralelo. Isto se





onde:

$Z_i(n)$ ... contador-sincronizador, colocado inicialmente em zero;

FILA(arg,k) ... predicado que representa o atendimento (isto é, a política) dentro da fila;

BUFFER(k) ... capacidade da fila.

Fig. V.2: Modelo geral, na base de 'PrT-Net', em relação à capacidade de uma fila, indicando também (a) como se pode compactar a representação gráfica.

deve ao fato que, normalmente, os processadores de E/S funcionam em paralelo à UCP. Entretanto, num determinado momento, temos que modelar a parte da UCP que está envolvida com o processamento seqüencial dos problemas do fluxo das mensagens;

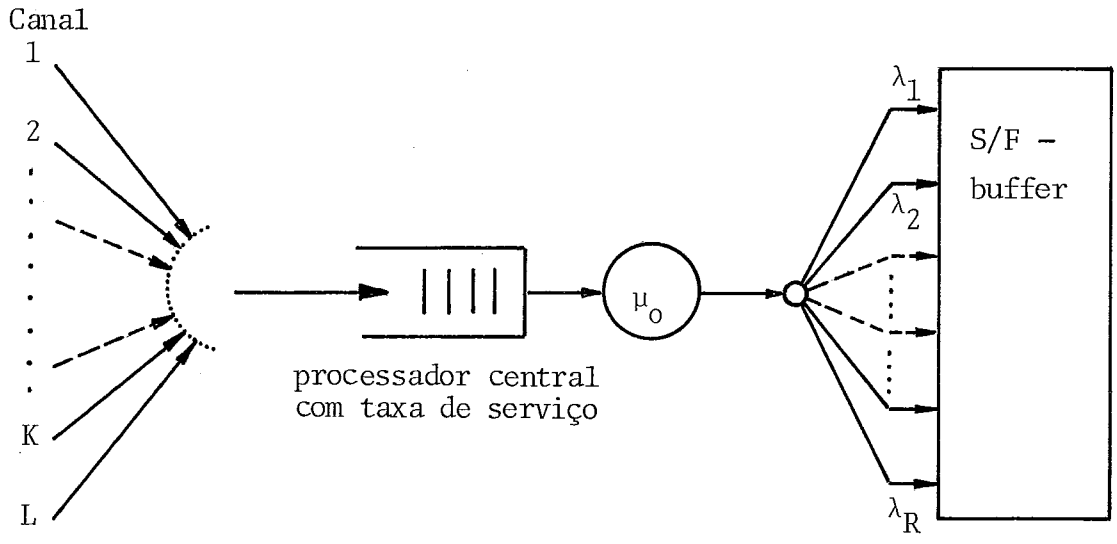
- As diversas filas tem capacidades finitas. Este fato deve ser levado em consideração no momento em que dividimos as filas, a saber, que elas, as filas principais, associadas aos canais, podem ser divididas em subfilas, em filas seqüenciais, em filas paralelas, etc., seguindo, por exemplo, idéias expostas por SCHWARZ (30) sobre a alocação do espaço disponível dentro de um 'buffer';
- Os predicados e transições, mostrados no modelo no. 1 (figura IV.9), são insuficientes para modelar os problemas de filas e do processamento associado. Devemos, então, criar "novos" predicados e transições ou, para ser mais exato, criar novos e, eventualmente, ABRIR aqueles já existentes.

Vamos, em seguida, detalhar o conjunto destes pontos seguindo, novamente, mais ou menos, o fluxo de uma mensagem desde a fonte até a destinação desejada.

V.2.a.2.α - Representação clássica das filas de entrada num vértice, indicando, também, as suas respectivas capacidades.

Para ser o mais realista possível precisamos associar capacidades finitas aos vértices em geral. Agora, para evitar ainda, em parte, os conflitos com outros vértices, devemos adaptar o esboço da figura V.3 de maneira a obtermos uma parte da capacidade associada aos canais e uma outra parte ligada à UCP, i.e., ao processador central propriamente dito. Veja esta idéia na figura V.4.

Este esboço mostra, então, que cada canal (com o 'buffer' de entrada associado) pode funcionar em paralelo com os outros e que um conflito ocorre somente no momento de "inscrição" e do "tratamento" do pacote (que chegou em algum destes canais) na fila do processador central.



onde:

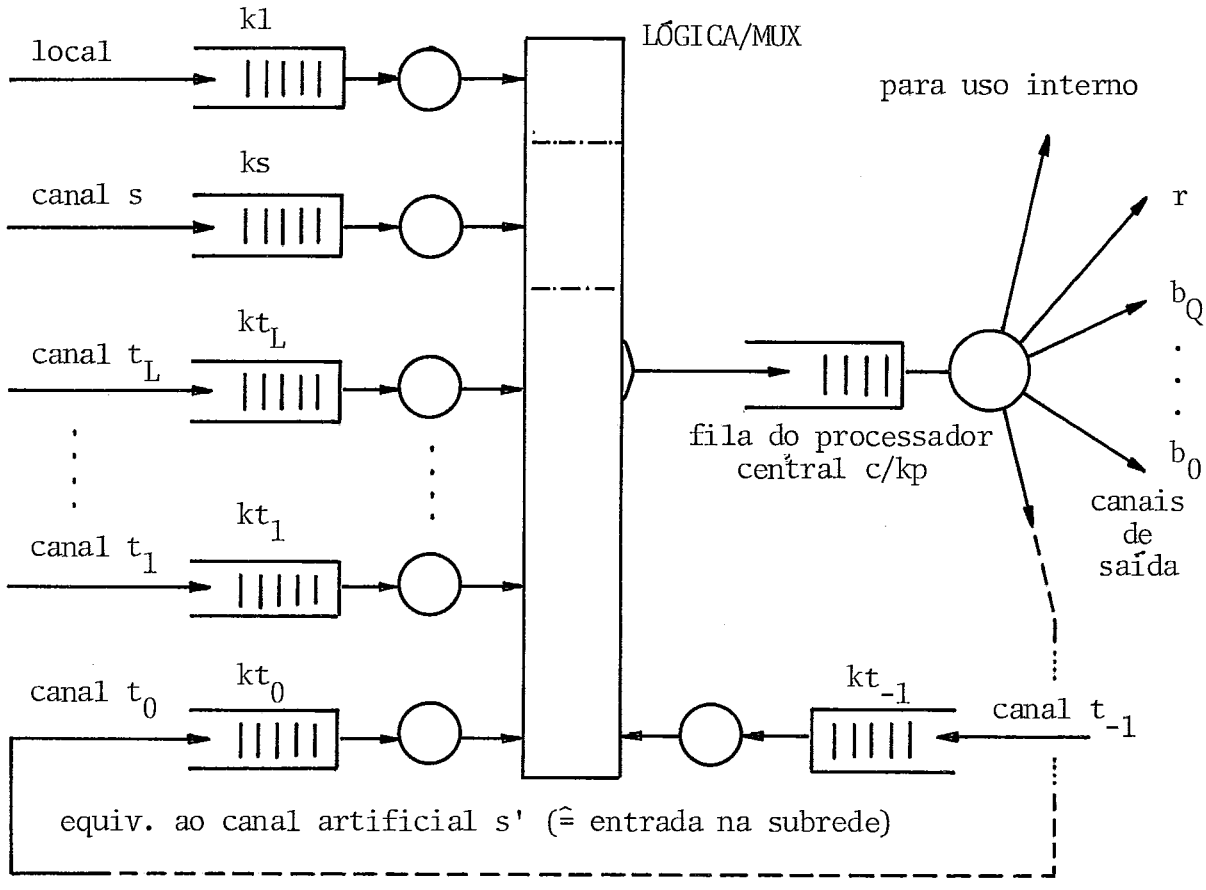
canal i .... representa o canal vindo de um outro vértice

λ_i ... representa a taxa de fluxo para um determinado S/F-buffer

R .... indica o número dos possíveis vértices vizinhos ligados à saída deste vértice S/F

μ_o ... taxa de serviço no processador central

Fig. V.3: Caminhos do fluxo das mensagens, passando pelo processador central de um vértice, LAM (16).



onde:

canal s	equivalente à ligação vindo do hospedeiro associado a este vértice/fonte
canal t_i, i=1, ..., L	ligação vindo de outros vértices (internos à subrede de comunicação)
canal t_0=s'	canal artificial que representa a "ligação" do canal de entrada s à subrede
canal t_{-1}	canal artificial indicando a "entrada para iniciar uma retransmissão"
canais r, b_0 até b_Q	canais de saída (veja fig. V.11)
kp, ks, kt_i, kl	representam as capacidades das filas associadas ao processador, canal s, canais t_i e ao uso interno, respectivamente
LÓGICA/MUX	responsável pela administração do espaço na fila do processador central ou UCP.

Fig. V.4: Esboço sobre a integração da ligação host-vértice, das ligações vindas de outros vértices e dos canais artificiais num vértice e a associação de capacidades às diversas filas.

O que vale destacar, observando este modelo, é que, com a separação do conjunto de filas em três subconjuntos (tarefas locais, canais dos vértices vizinhos e canal vindo do host), conseguimos uma distinção clara entre as funções de um vértice/fonte e as de um vértice/do tipo 'store-and-forward' (vértice S/F).

O primeiro, o V/F, se limita a receber e testar a mensagem vinda do hospedeiro, encaminhando-a, depois, à UCP para a empacotagem. Uma vez na forma de pacotes, a mensagem chegará ao canal artificial s', já que, agora, será submetida a funções da subrede de comunicação (como, p.ex., o roteamento) e, conseqüentemente, terá um tratamento igual aos pacotes vindo de vértices vizinhos. Estas funções da subrede são executadas pelos vértices S/F em conjunto, obviamente, com a mesma UCP do vértice em questão.

Assim, dizendo que temos agora uma única fila de entrada no vértice/fonte, podemos nos preocupar em como modelar, usando a técnica de 'PrT-Net', este fato.

V.2.a.2.β - Modelo da fila de entrada num vértice/fonte.

Tendo obtido, no esboço da figura V.4, uma separação das diversas filas de entrada, podemos agora nos preocupar em como modelar a fila de entrada ao vértice/fonte.

Então, podemos considerar que o predicado FONTE(s,r) do modelo no. 1 (figura IV.9 do capítulo IV) é agora, exclusivamente, representativo da situação das mensagens <s,r> na fila do canal s. Isto inclui, também, além do armazenamento, uma eventual verificação do pacote, feita na unidade E/S do canal s. Podemos, então, usando esta idéia e aquela do modelo geral (figura V.2), reconsiderar o predicado em questão, isto é, FONTE(s,r).

Como argumentos temos, além de s (s ε EndE) e r (r ε EndR), tirados do capítulo IV (figura IV.1), um que é novo:

$KS = \{ks_1, ks_2, \ldots, ks_W\}$, que representa as capacidades ks (ks ε KS) dos 'buffers' da fila associada ao canal s.

Este argumento aparece, então, num predicado de grau 2

FONTE/BUF(s,ks), onde s ε EndE e ks ε KS,

que representa a administração do espaço na fila do canal s e, também, num predicado de grau 3

FONTE/FILA(s,r,ks), onde s ε EndE, r ε EndR e ks ε KS,

que representa o atendimento dos elementos da fila, onde vale salientar que este atendimento é, por convenção, restrito à recepção e verificação. A preparação do pacote, isto é, a empacotagem da mensagem, ou parte dela, deve ser feita, obrigatoriamente, na UCP do vértice em questão.

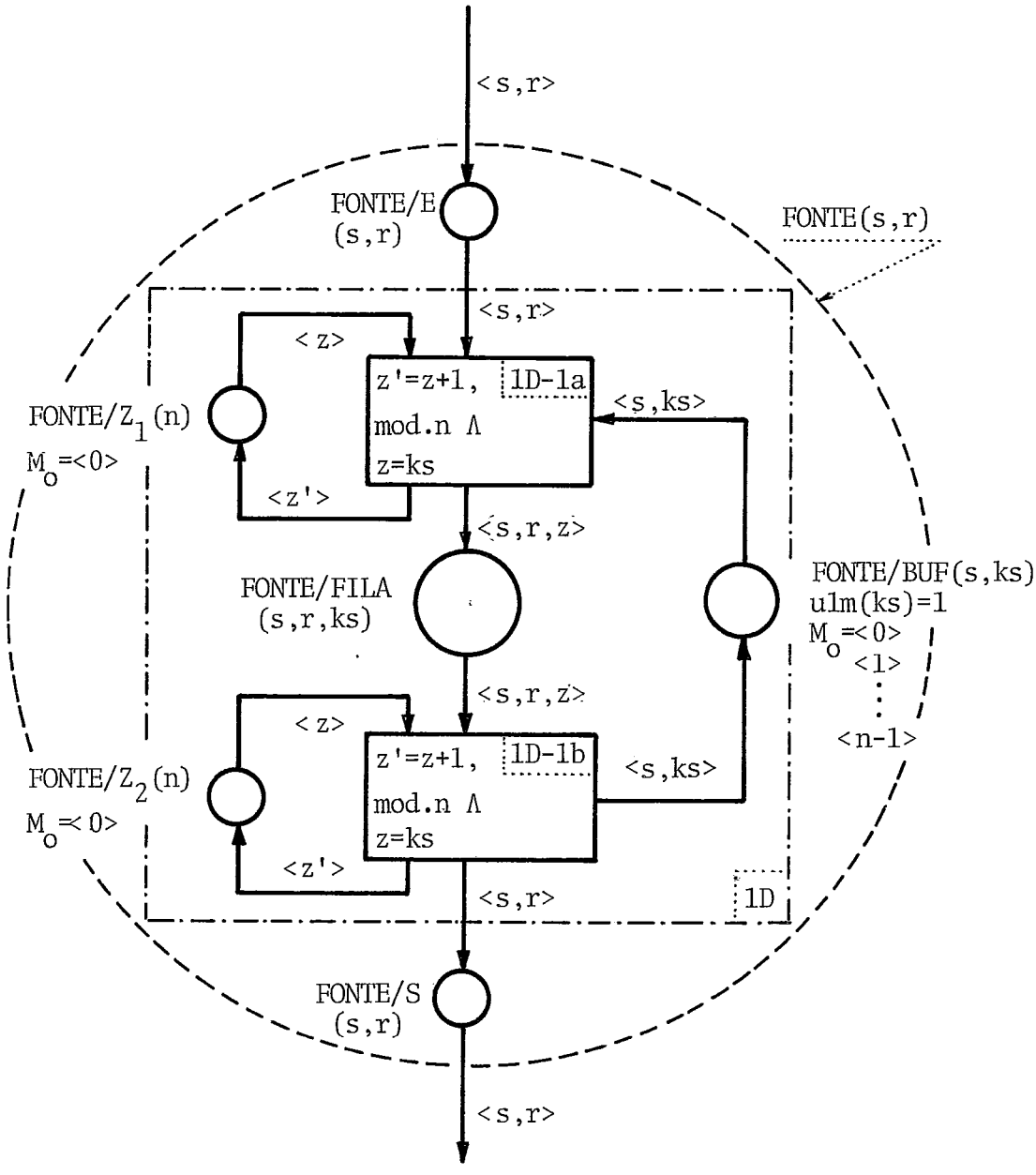
Veja estas idéias em relação à modelagem do predicado FONTE(s,r) na figura V.5.

V.2.a.2.γ - A passagem dos elementos da fila de entrada no V/F para a fila do processador central.

Observando a figura V.4, vemos que temos, em princípio, dois pontos a considerar:

- A passagem (de um elemento de uma das filas de entrada para a fila do processador central) depende da lógica do multiplexador e, obviamente, se há espaço na fila deste processador;
- O atendimento pelo processador dependeria da política do atendimento regente nesta fila da UCP e, também, eventualmente, da capacidade interna de multiprocessamento.

Começando a pensar na estrutura deste conjunto de filas, temos que considerar dois sistemas de filas em cascata. Cada uma destas filas segue alguma das políticas comumente usadas em tratamento de filas (p.ex., FIFO, LCFS, RR, randômico, HOL, etc.). Já o conjunto das filas, que competem pela fila da UCP, deveria



onde: ——————— modelo simplificado do predicado FONTE

-·-·-·-·- modelo simplificado da transição 1D

Fig. V.5: Modelo parcial do predicado FONTE(s,r), considerando este predicado como sendo uma fila

ser atendido por técnicas em multifilas (veja, p.ex., investigações feitas por VASCONCELLOS (38)) observando, ainda, que este conjunto consiste de três subconjuntos, a saber, um em relação à ligação ao 'host', outro associado aos canais vindo dos vértices vizinhos e, finalmente, um que representa as tarefas internas da UCP.

Mas, voltando à preocupação principal desta seção, isto é, à passagem dos elementos da fila de entrada (V/F) para a fila do processador, podemos observar o seguinte: uma vez que a precondição principal para esta passagem, a saber, a presença no predicado FONTE/S(s,r), que é um subpredicado do principal, FONTE(s,r), é satisfeita, podemos nos restringir a investigar como modelar as outras condições para garantir esta passagem.

As outras precondições devem ser associadas, como já dissemos, à lógica do MUX e, posteriormente, à situação dentro da fila da UCP. Considerando, então, o esquema básico de uma fila em geral (figura V.2), auxiliado por predicados intermediários em torno da fila da UCP, chegamos, usando esta técnica de 'predicate/transition-net', ao esquema da figura V.6.

No esquema da figura V.6, pode ser observada, ainda, a necessidade de uma identificação dos vértices e das funções associadas a eles. Isto se deve ao fato que não podemos mais evitar "ABRIR" a subrede de comunicação; conseqüentemente, não estamos mais lidando com um único recurso para todos os usuários, mas, isto sim, com um conjunto de partes da subrede usadas por um determinado grupo de usuários.

Agora, como não queremos mostrar, no esquema gráfico, cada vértice da subrede de comunicação, optamos, dentro do esquema esboçado, para a representação de um único vértice, mas, ainda, com alguma inscrição relativa à identificação. Deste modo, precisamos colocar identificações nos predicados do tipo LÓGICA/MUX, UCP/BUF, etc.; fazemos isto através do seguinte argumento:

$V = \{v_1, v_2, \ldots, v_W\}$, usando, no texto, $v \in V$,

que representa os vértices que formam a subrede de comunicação. Este argumento, junto com

$KP = \{kp_1, kp_2, \ldots, kp_W\}$, representado por $kp \in KP$, e

$P = \{p_1, p_2, p_3, \ldots\ldots\}$, usado como $p \in P$,

aparece, então, nos predicados envolvidos na modelagem de algum "acontecimento específico" num vértice.

Assim, temos, como predicados principais,

UCP/BUF(v,kp), que representa a administração do espaço na fila da UCP, onde $v \in V$ e $kp \in KP$;

INTERNO/FILA(v,arg) e INTERNO/USO(v,arg'),que representam, por exemplo, um possível funcionamento interno de um vértice, com $v \in V$ e arg, arg' $\in$ ARG;

LÓGICA/MUX$(()_V)$, que representa a permissão para inscrição de um elemento das filas de entrada (veja figura V.4) na fila do processador central,

auxiliados pelos predicados

CF/PREP/INI(s,r) e CF/PREP/FIM(s,r,p), onde $s \in$ EndE, $r \in$ EndR e $p \in P$.

Estes últimos já podem ser associados ao processamento na UCP, que acontece numa parte da fila da UCP, representado pelo predicado

CF/PREPARAÇÃO(s,r,p,z) .

Como observação, em relação ao predicado LÓGICA/MUX$(()_V)$, podemos dizer que usamos o mesmo raciocínio (para conseguir uma identificação da tupla zero $<>_V$) como foi usado para a identificação da tupla zero associada ao predicado CRÉDITO$((()_{sr})$ no capítulo IV. Ainda, em relação à este predicado LÓGICA/MUX, devemos admitir, que os critérios usados, para decidir quais das informações tratar, fogem, neste momento, de nosso interesse imediato.

Na figura V.6 esboçamos ainda, só para indicar, a existência de alguma informação interna (em relação ao próprio vértice) que, ela também, quer entrar nesta fila da UCP. Assim, por exemplo, um argumento.

$$ARG = \{arg_1, arg_2, \ldots, arg_W ; arg_1', arg_2', \ldots, arg_W'\}$$

aparece em predicados do tipo INTERNO/PROBL(v,arg), CF/INT.PROC/INI(v,arg), CF/INT.PROC(v,arg',z), CF/INT.PROC/FIM(v,arg') e INTERNO/USO(v,arg'), que representam estas "necessidades internas".

Deste modo, conseguimos modelar a permissão para iniciar a colocação na fila da UCP. Na figura V.7 mostraremos mais detalhes sobre a parte de transição 2 envolvida no processamento (empacotagem) da mensagem ($p=f_p(s,r)$, onde $f_p:(U \times U \rightarrow P)$) para depois, na seção V.2.a.2.δ, falar sobre o processamento propriamente dito.

V.2.a.2.δ - O processamento na fila da UCP.

O que se tentou mostrar, nas figuras V.6 e V.7, foi a inscrição dos elementos <s,r>, vindo de FONTE(s,r), na fila da UCP. Entretanto, como o modelo de uma fila consiste de entrada, processamento e saída, surge automaticamente a indicação da saída desta fila do processador central. Em outras palavras, os elementos processados, <s,r,p>, deixam a fila, permitindo assim a liberação de um espaço no buffer da UCP, e aparecem num predicado PRONTO(m,em,re), que representa um estado similar aquele que os elementos vindos dos vértices vizinhos assumem. Trataremos deste ponto mais detalhadamente na seção V.2.a.3.

O que falta indicar, nesta seção, é algo em relação ao processamento. Assim, uma vez que algum elemento das filas de entrada está inscrito na fila da UCP, pode-se, dependendo das regras existentes na mesma, iniciar o processamento, p.ex., representado por um predicado do tipo CF/PREPARAÇÃO(s,r,p,z). Entretanto, como isto foge um pouco de nossas preocupações imediatas, podemos modelar o funcionamento de uma maneira bem simpli-

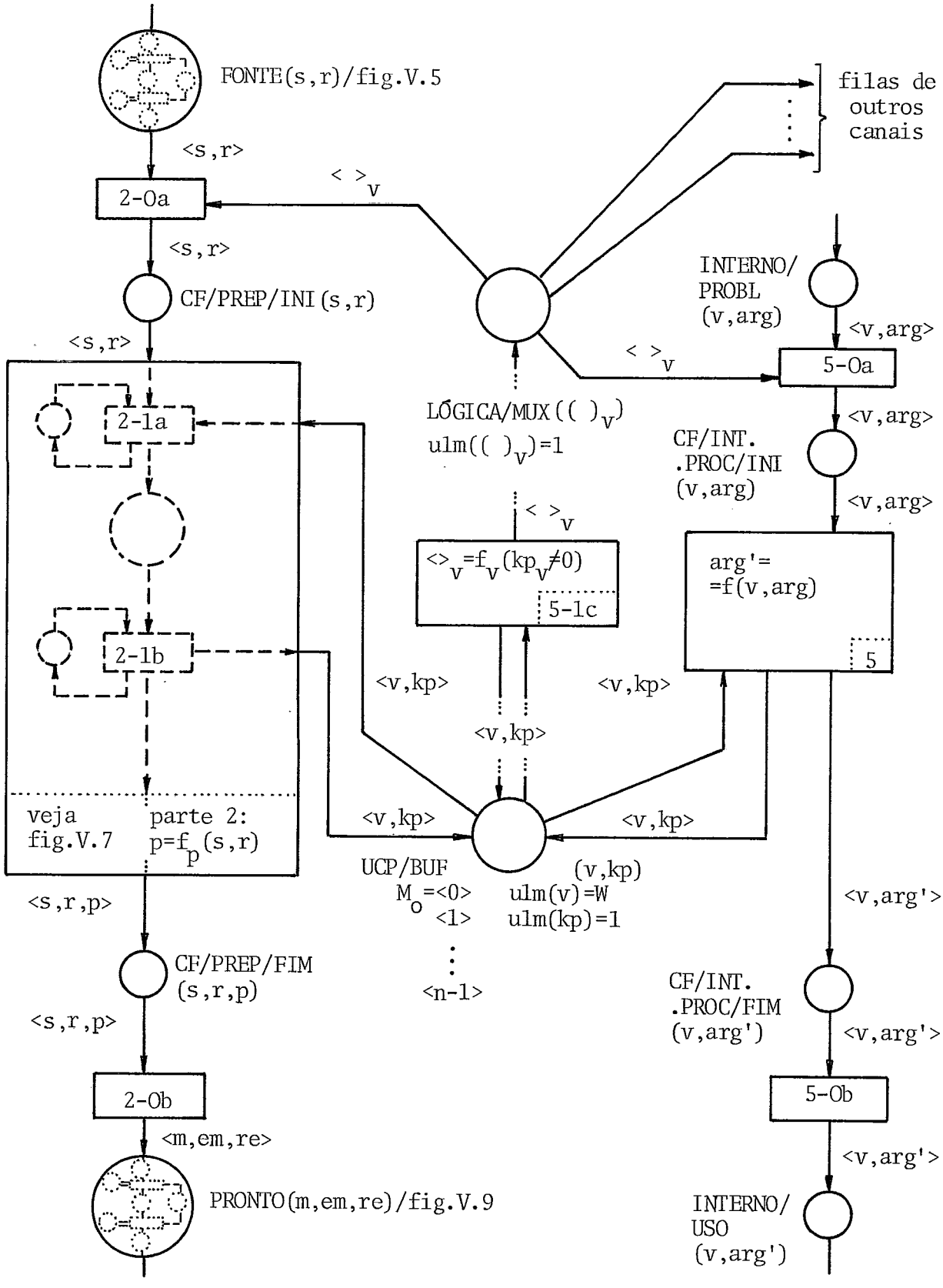
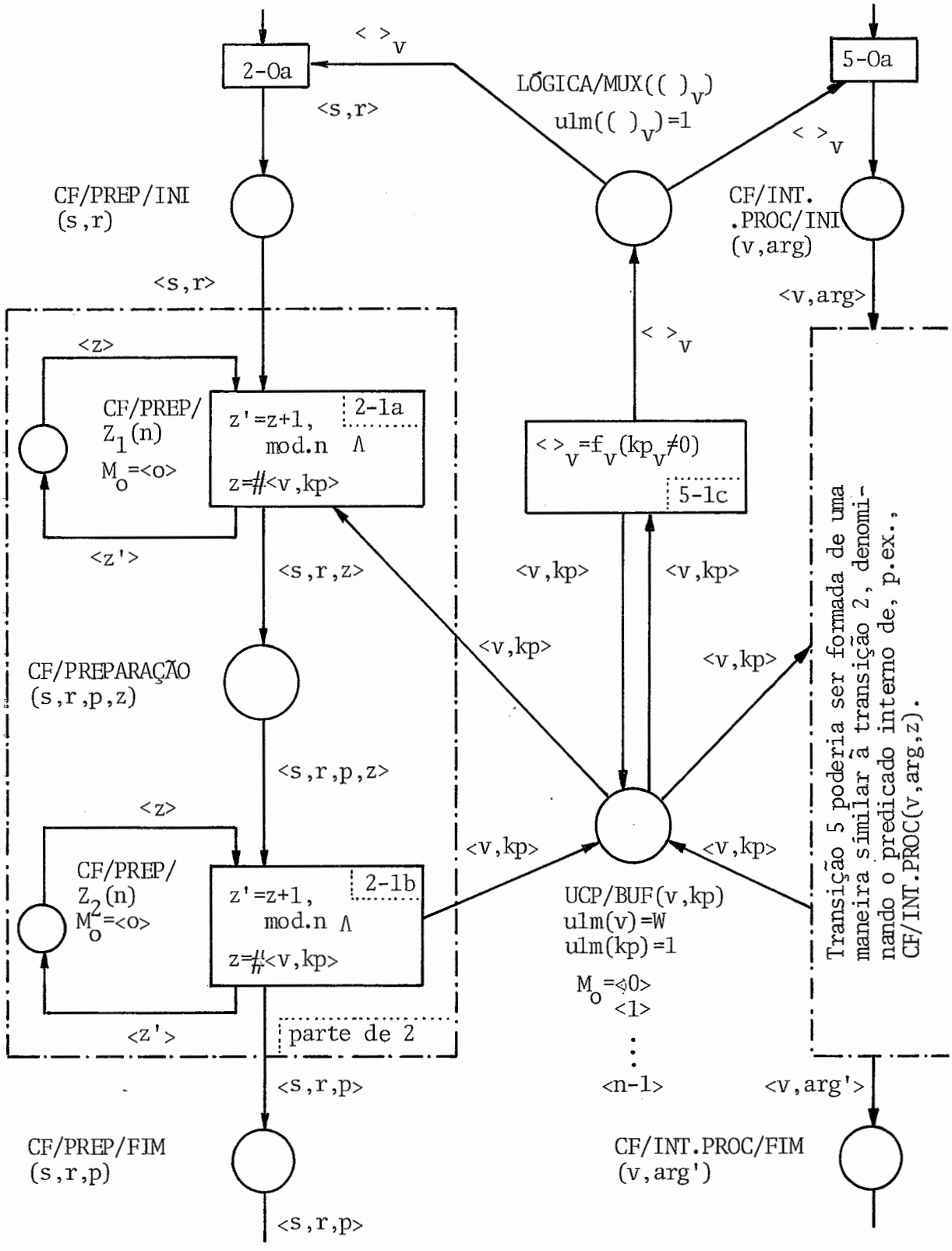


Fig. V.6: Transferência, baseada na lógica do MUX, dos elementos das diversas filas (fonte, interna e vértices vizinhos) para a fila do processador central. (Veja detalhes da transição 2 na fig. V.7).



Obs.: #<v,kp> indica a n-ésima tupla em relação à fila; detalhes do predicado CF/PREPARAÇÃO se encontram na fig. V.8.

Fig. V.7: Detalhe da parte da transição 2: $p=f_D(s,r)$, da figura V.6, observando o espaço comum, representado pelo predicado UCP/BUF, e a administração (contador/sincronizador) individual.

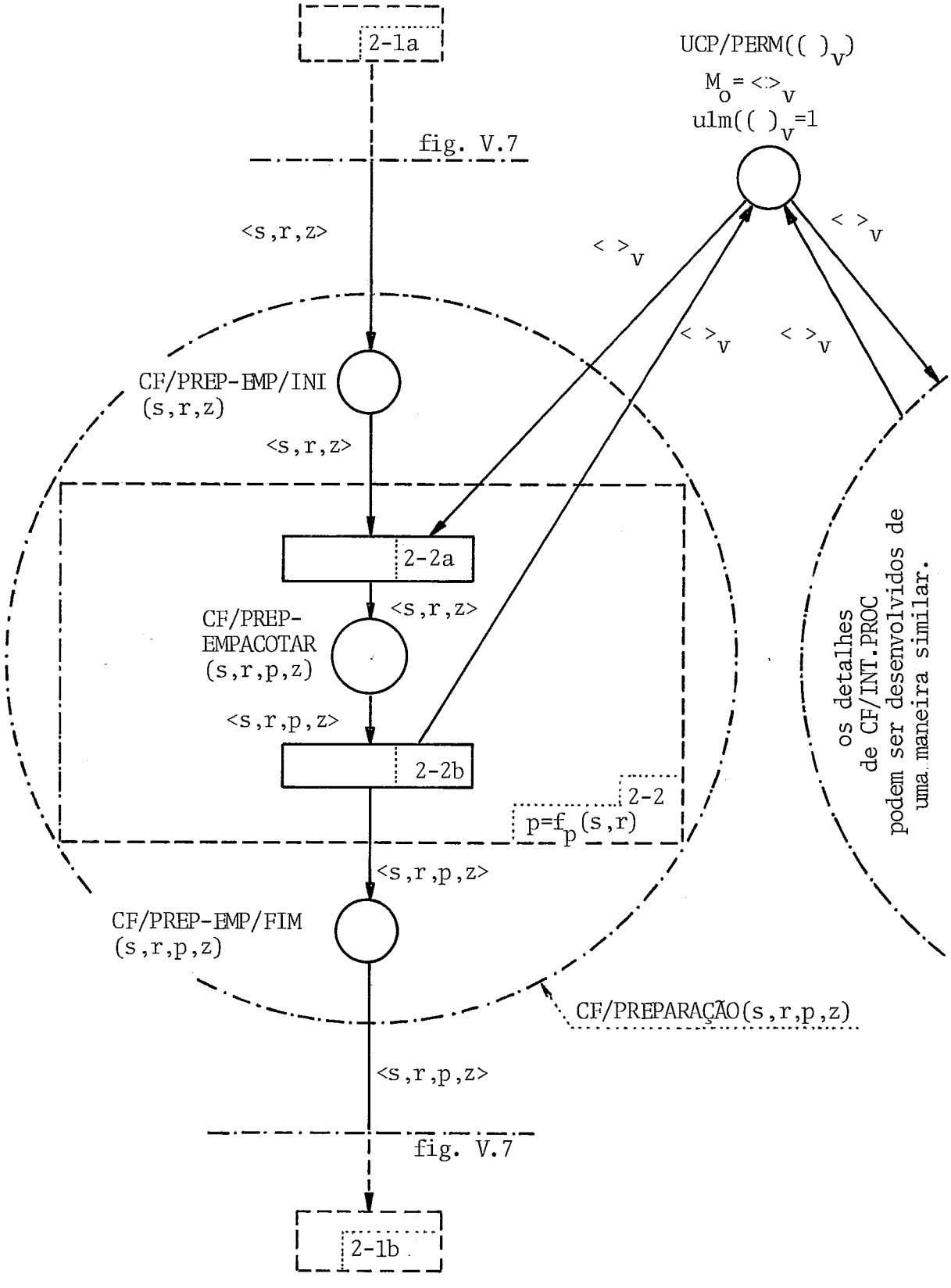


Fig. V.8: Competição dos elementos da fila do processador pelo atendimento na UCP, detalhando o predicado CF/PREPARAÇÃO da figura V.7.

ficada. Indicamos, então, somente a "permissão" $<>_v$, vindo de um UCP/PERM($()_v$), para realizar alguma tarefa (por exemplo, "empacotar" a mensagem, representado pelos predicados CF/PREP-EMP/INI(s,r,z), CF/PREP-EMPACOTAR(s,r,p,z) e CF/PREP-EMP/FIM(s,r,p,z)), como tentamos mostrar na fig. V.8.

O que deve ficar claro é, que esta indicação é bem superficial. Em realidade, tanto a inscrição na fila, como, também, o processamento, é feito pela UCP do vértice. Portanto, deveríamos indicar como o vértice seria "ativado". Poder-se-ia pensar, p.ex., num 'pool' (POOL/SF(v)), que representaria os vértices da subrede num estado passivo, e um predicado do tipo NODO(v), que representaria, justamente, esta ativação. Mas, como já apontamos, não entraremos nestes detalhes, ao menos, não neste trabalho.

V.2.a.3 - As filas de "saída" de um vértice .

V.2.a.3.α - Idéias gerais.

Como idéia inicial podemos dizer o seguinte: para que uma mensagem empacotada possa ser realmente transmitida, precisa-se ter, ainda, escolhido um determinado caminho. Baseado nesta escolha, pode-se saber qual das filas de saída será responsável pelo envio. Vale ainda observar, que "caminho", neste contexto, é definido como uma seqüência de canais que, por sua vez, são especificados como o conjunto de todos os componentes envolvidos na transmissão, a saber, a fila de saída, seu servidor associado (emissor, modem, linha, etc.) o que, em termos gerais, é considerado como a linha de transmissão, e a fila de entrada noutro "lado", isto é, no próximo vértice.

Concernente à escolha, pode-se admitir que a operação da formação do feixe dos caminhos possíveis (entre um determinado par emissor-receptor) foi feita de uma maneira independente da UCP-vértice (por exemplo, num processador dedicado, num centro de roteamento, etc.) ou, ao menos, de um modo antecipado ao momento da necessidade real para escolher um determinado caminho.

Assim teremos, para encaminhar a mensagem empacotada <s,r,p>, somente que escolher o melhor caminho possível a partir das informações à disposição (p.ex., uma tabela de roteamento), verificar se a fila de saída associada a este melhor caminho tem espaço disponível e, caso afirmativo, colocar a mensagem nesta fila de saída.

Agora, observando bem este último parágrafo, podemos constatar que a função de roteamento é uma função que cabe à subrede de comunicação, isto é, uma vez que a mensagem a ser transmitida está na forma <s,r,p>, a função da escolha independe do usuário (emissor e receptor). Assim, este roteamento deverá ser modelado numa seção sobre os vértices da subrede, isto é, os vértices intermediários, e, obviamente, o tratamento de filas de saída, em geral, será associado a esta escolha das rotas.

Fica, então, a pergunta se seria possível o "enquadrar" uma mensagem vindo do hospedeiro, isto é, não estando, ainda, na forma <s,r,p>, nestas idéias gerais sobre as filas de saída. Trataremos disto na seção V.2.a.3.β.

V.2.a.3.β - Modelo da fila de saída de um vértice/fonte.

Podemos agora, neste item, justificar o porque da existência de um canal artificial s', como já foi mostrado na figura V.4.

Querendo transformar uma mensagem vinda do hospedeiro numa forma igual aquela dos pacotes vindos dos vértices vizinhos, precisamos um tratamento preliminar da mensagem. Mas isto já foi feito na "passagem" pela UCP, a saber, durante o empacotamento. Olhando as figs. V.6, V.7 e V.8, vimos que a mensagem <s,r> passou pela transição 2 (com a fórmula $p=f_p(s,r)$) onde a mensagem foi transformada na forma <s,r,p>.

Pode-se, então, constatar que o predicado PRONTO(m,em,re) representa, por um lado, a saída do processador central mas, ao mesmo tempo, também a recepção numa fila similar ao tipo 'store-and-forward'. Considerando a estrutura básica de uma fi-

la (figura V.2), podemos usar conceitos similares aos usados para modelar FONTE(s,r):

- a recepção será, no caso deste canal artificial s', modelada pelo predicado PRONTO(m,em,re) que já contém tuplas que foram preparadas, na transição 2-Ob, para receber as informações em relação a emissor e receptor, respectivamente;

- o processamento, isto é, testes e verificação em relação à consistência e estrutura do pacote, será modelado por um predicado a criar, digamos PRONTO/FILA(m,em,re,ks'), onde o argumento ks' (ks' ε KS') representa a capacidade da fila do canal $s'=t_o$;

- o fim destes testes será modelado por um predicado do tipo PRONTO/S(m,em,re).

Assim, usando, globalmente, o predicado PRONTO(m,em,re), que representa o conjunto total, conseguimos um modelo da saída de um vértice/fonte e, também, da entrada em relação ao canal s'. Veja este modelo do predicado PRONTO(m,em,re) na figura V.9.

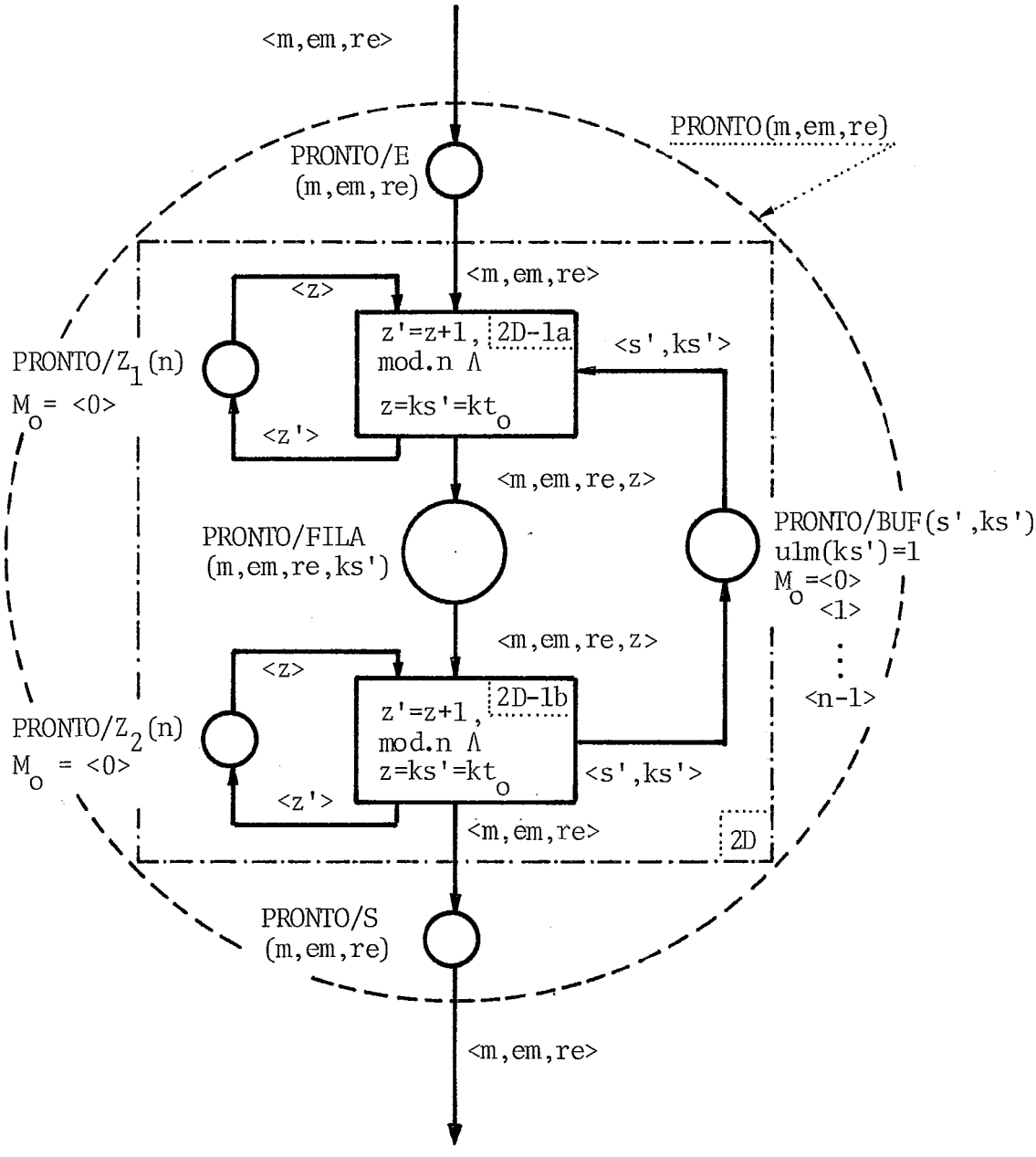
Esta saída do vértice/fonte significa, ao mesmo tempo, a delegação da responsabilidade pela transmissão ao vértice, tipo S/F, ou, em outras palavras, à subrede de comunicação. Assim, já antecipando algumas observações em relação aos vértices intermediários, precisamos, além de uma simplificação no manejo da tupla <s,r,p>, ainda explicar a definição dos "emissores e receptores" (feita na transição 2-Ob) dentro da subrede; então, o argumento

$$m = f_m(s,r,p), \text{ onde } f_m : (U \times U \times P \rightarrow M),\ m \in M,$$

representa uma simplificação de ponto de vista do manejo, isto é, o conjunto M é constituído de todas as mensagens virtualmente possíveis, já na forma de pacotes:

$$M = \{m_1, m_2, \ldots\ldots\}$$

$$= \{s_1r_1p, s_1r_2p, \ldots\ldots, s_Wr_Wp\} \quad .$$



onde: ——————— -·-·-·-·-

modelo simplificado do predicado PRONTO

modelo simplificado da transição 2D

Fig. V.9: Modelo detalhado do predicado PRONTO(m,em,re), considerando este predicado como sendo uma fila.

O argumento "em" representa o emissor local na subrede de comunicação, papel que, em termos mais específicos, é desempenhado pelo processador de saída de uma determinada fila de saída b_k, associado a um vértice v_i. Assim,

$$em = v_i b_k = f(\text{emissor local}),\ em \in EM,\ i=1,\ldots,W;\ j=1,\ldots,Q,$$

onde

$$EM = \{em_1, em_2, \ldots, em_W\}$$

$$= \{v_1b_o, v_1b_1, \ldots, v_1b_k; v_2b_o, v_2b_1, \ldots; v_wb_o, \ldots, v_wb_k\}.$$

O argumento "re" representará o receptor local na subrede; ele é formado similarmente:

$$re = v_j t_\ell = f(\text{receptor local}),\ re \in RE,\ j=1,\ldots W,\ \ell=1,\ldots,L,$$

onde

$$RE = \{re_1, re_2, \ldots, re_W\}$$

$$= \{v_1t_o, v_1t_1, \ldots., v_Wt_o, \ldots., v_Wt_L\},$$

o que contém a indicação do vértice v_j com um processador de entrada para uma fila t_1.

Tendo estas indicações, podemos agora tranqüilamente iniciar a modelagem dos problemas na subrede, começando, na seção V.2.a.3.γ, com a delagação da responsabilidade à subrede.

V.2.a.3.γ - Transferência da responsabilidade em relação à transmissão (parte emissão).

O que conseguimos na seção V.2.a.3.β foi uma desassociação do roteamento (e, assim, do encaminhamento a uma determinada fila de saída em relação ao vértice global) da função do vértice/fonte. O que falta mostrar é como esta passagem da fila de saída do V/F (predicado FONTE(s,r)) para a fila de entrada no canal s' (predicado PRONTO(m,em,re)) aparece no esquema geral do

modelo no. 1 (fig. IV.9 no capítulo IV). Também devemos nos preocupar com a mencionada transferência da responsabilidade pela transmissão.

Então, considerando estes novos aspectos, temos que modificar as antigas transições 2 e 2A (como utilizadas na fig. IV.9). Veja estas idéias expostas em seguida e compare com as indicações provisórias feitas na figura V.10:

- Pensando em futuras versões (por exemplo, aquelas que já incluem multipacotes), podemos considerar que a segunda via (cópia) foi gerada no momento em que a mensagem empacotada <s,r,p> entrou na fila do canal s', i.e.,quanto entrou no domínio das funções do S/F (modelado pelo predicado PRONTO(m,em,re)). Assim, o predicado FONTE(s,r) representa o último estágio da mensagem em relação à fonte e, conseqüentemente, o predicado SEG.VIA(s,r,p) será substituído pelo predicado SEG.VIA.M(s,r,p) que representará a cópia da mensagem empacotada, sendo liberado pelo RFNM associado;

- Nesta mesma transição acontece que a função de transmitir (predicado TRANSMITIR(u)) do usuário s terminou e ele passa para o estado ESP/CONF(u) que, agora, poderíamos chamar melhor de ESP/RFNM(u) já que se trata, futuramente, da mensagem inteira. Conseqüentemente, chamaremos o predicado associado à confirmação, em vez de CONFIRM(mc,r,s), agora de RFNM(mca,sc,rc) (veja a explicação na seção V.2.c);

- Porém, como não pode haver um pacote na rede sem um "responsável" pela transmissão, precisamos de um "novo" emissor que será responsável pela transmissão do pacote em questão até o próximo vértice. Antecipando resultados em relação à modelagem dos problemas nos vértices intermediários, podemos dizer que existe, para cada canal de entrada num vértice, um processador de entrada, ou receptor, cuja identificação (física) "re" consiste de uma dupla formada pelo vértice v_j e do canal de entrada t_1. Agora, cada um destes processadores pode tratar pacotes (de q diferentes classes (usuários)) cujos caminhos se "cruzam" no enlace associado a este processador. Por-

tanto, o responsável por um pacote p deve ter um argumento que indica, digamos, a existência física, e um outro que indica por quem ele é "responsável". Assim, um predicado do tipo 'pool', que representa todos estes responsáveis, poderia ser um predicado POOL/PE(re,p) que, deste modo, fornece uma das precondições para disparar a transição 2; a poscondição relacionada a esta idéia seria, então, um predicado PE/VERIFICAR(re,p), o que significaria a passagem do estado passivo para um ativo;

- Falta ainda a identificação física exata para esta fase, o que pode ser feito na transição com as indicações $v_j t_1 = v_s t_o$, já que se trata aqui do "primeiro" vértice na cadeia de um caminho e, conseqüentemente, do canal artificial $s' = t_o$.

Por enquanto, podemos considerar resolvida a modelagem dos problemas básicos em relação às filas de entrada na subrede. Tendo conseguido a separação das funções V/F e S/F, podemos, agora, nos dedicar às seguintes perguntas básicas, todas elas relacionadas aos vértices S/F:

- Onde e como pode-se colocar a idéia do roteamento (escolha do "melhor" caminho) na passagem do predicado PRONTO(m,em,re) até aquele que indicou que a mensagem está em trânsito (predicado M-em-TRÂNSITO, na figura IV.8)?

- Ou, em outras palavras: O que acontece realmente na "nova" transição 6, que será a substituta da transição 2A da figura IV.9?

- Como garantir e indicar, na representação gráfica, a identidade dos vértices intermediários S/F nesta cadeia que, provavelmente, será usada repetitivamente?

Tentaremos responder a estas perguntas na seção V.2.b, que tratará dos problemas relacionados aos vértices intermediários do tipo S/F.

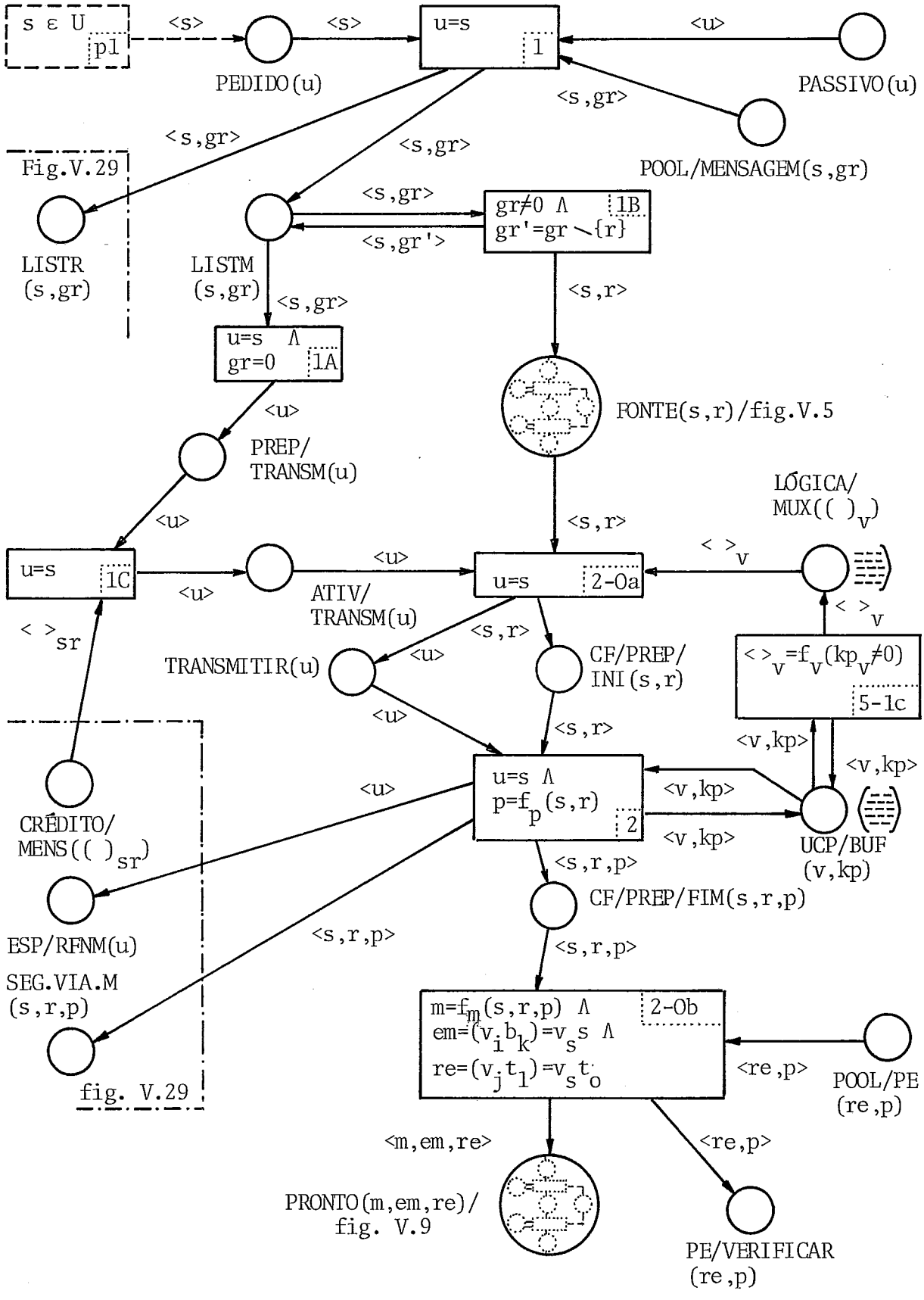


Fig. V.10: Modelo parcial provisório, mostrando a idéia da separação das funções V/F e S/F e as modificações causadas em relação ao modelo número 1 (figura IV.9).

V.2.b - Os vértices intermediários do tipo S/F.

Observando as figuras II.2 e II.5, podemos constatar que a idéia do esquema simplificado se repete, aparentemente, tantas vezes até que o próximo vértice S/F coincida com o vértice do tipo V/D. Assim, para ser mais específico, podemos dizer que temos, entre vértices vizinhos, um esquema de controle em canais (fig. II.1), entre os quais poderíamos, por exemplo, escolher algum esquema de crédito. Mas como, neste exemplo inicial de modelagem, estamos somente modelando a transmissão de um único pacote, podemos usar o método simples com ACK, NAK e 'time-out' para mostrar o que acontece entre vértices vizinhos.

Adicionalmente queremos alertar que este fato vale, em princípio, somente para as mensagens <s,r> de um determinado usuário. Entretanto, na realidade acontece que todas as mensagens (de todos os usuários), que passam pelo mesmo trecho de um caminho, estão sendo tratadas como pertencentes, ao menos temporariamente, a um "mesmo grupo". Assim, todas elas seguem a política do atendimento regente na fila em questão e do método de transmissão na linha de saída associada. Então, não se deve esquecer, no momento de substituir o método simples mencionado (válido para as mensagens de um único usuário) pelo sistema de crédito, de determinar exatamente se o crédito é válido para uma determinada classe de mensagens ou para o 'pool' inteiro presente numa determinada fila de saída de um vértice S/F.

Outro ponto a observar está relacionado à escolha do caminho. Observando o esquema da figura IV.9, desenvolvido no capítulo IV, vemos que este esquema somente será válido se tivermos escolhido, e reservado, já na fonte, o caminho virtual completo, isto é, toda a seqüência de canais entre os vértices adjacentes para formar o caminho entre a fonte e a destinação. Este tipo de procedimento somente é usado em caso de roteamento fixo, SCHWARZ (30), o que não daria muita flexibilidade do ponto de vista de adaptação aos acontecimentos reais dentro de uma rede de computadores.

Agora, sabendo que numa rede existe uma certa dinâmica, que deve impor uma adaptação durante o funcionamento, podemos usar, por exemplo, um método distribuído, SCHWARZ (30), para resolver o encaminhamento das mensagens. Precisa-se, então, determinar onde e como podemos escolher o caminho (ou, melhor, o próximo vértice para compor este caminho), levando em consideração as informações disponíveis (global e localmente) e a capacidade dos vértices em tomar certas decisões sozinhos.

Propomos, então, a seguinte metodologia para incluir estes pontos mencionados no modelo número 1 (figura IV.9), já considerando alguns dos aspectos levantados na seção sobre as filas.

- Primeiramente, ver como podemos "embutir" a seqüência dos vértices intermediários do tipo S/F no modelo básico, introduzindo, também, idéias um pouco mais concretas sobre a maneira de como escolher o caminho;

- Segundo, estudar como evitar, na visualização gráfica, a representação repetitiva dos vértices intermediários usando, para esta finalidade, inscrições mais complexas.

O estudo será feito seguindo, novamente, o fluxo de uma mensagem, ou pacote, através da subrede de comunicação.

V.2.b.1 - A seqüência dos vértices S/F num caminho entre fonte e destinação.

Observando a figura V.10, vimos que em torno da transição 2 começam as preocupações em relação à subrede de comunicação. Em particular, podemos enumerar os seguintes pontos:

a) Como incluir o argumento "vértice S/F" no esquema?

- Primeiramente, podemos afirmar o que já foi mencionado na seção V.2.a.2.γ, isto é, que $v \in V$, onde V representa o conjunto dos vértices disponíveis na subrede (lembrar que cada vértice pode ser do tipo S/F);

- Segundo, como já mencionada na seção V.2.a.2.δ, não vamos modelar, explicitamente, a ativação de um vértice. Consideramos que a rede como um todo, isto é, incluindo aí as linhas, vértices, etc., funciona e que não precisamos nos preocupar com este nível de modelagem;

- Finalmente, da mesma maneira como temos um 'pool' para os usuários (que, uma vez ativados, são "responsáveis" pela transmissão de uma mensagem), deveríamos trabalhar com 'pools' que representarão os processadores de entrada e saída em relação a um vértice. Estes processadores são responsáveis pela transmissão, dentro da subrede, de um vértice para outro e serão ativados a partir de predicados

  POOL/PE(re,p), para entradas, (re ε RE, p ε P), e

  POOL/PS(em,p), para saídas, (em ε EM, p ε P), respectivamente.

  Nestes predicados aparecem os argumentos re (re ε RE) e em (em ε EM) que indicam, como foi explicado na seção V.2.a.3.β, o conjunto de processadores. Assim, a identificação física é sempre através re = $v_j t_l$, para entradas, e em = $v_i b_k$, para saídas. Entretanto, a verdadeira identificação se faz através das tuplas <re,p> e <em,p>, o que permite, assim, obter um "responsável" por um pacote p no caminho entre saída e entrada de dois vértices vizinhos.

b) Como escolher o próximo vértice S/F num caminho?

- Considerando um caminho entre V/F e V/D, podemos dizer, em geral, que ele é formado por uma seqüência de vértices S/F, digamos,

  $$v_s, v_{s+1}, \ldots, v_i, v_j, \ldots, v_{r-1}, v_r .$$

  Como já tentamos indicar através dos índices usados, existem, em princípio, três tipos de vértices S/F neste caminho: o primeiro "coincide" com o vértice/fonte, o segundo é associado exclusivamente ao "interior" da subrede e o

terceiro "coincide" com o vértice/destinação;

- Deste modo, no momento em que a mensagem saiu do "domínio" do usuário s ($s \in U$), isto é, quando é entregue à subrede, o vértice em questão (o primeiro tipo) assume o papel de "emissor", denominado v_s, até que receba, do vértice visinho, o ACK em relação ao pacote enviado. Assim, podemos dizer, que a escolha do vértice inicial do caminho entre V/F e V/D depende de alguma função

$$v_s = f(s)$$

que está relacionada com o usuário. Como observação queremos repetir, mais uma vez, que o "emissor" não é simplesmente o vértice, mas, sim, um processador de saída de uma determinada fila de saída neste vértice em questão;

- O segundo tipo dos vértices intermediários depende, exclusivamente, da escolha do caminho para alcançar o ponto final V/D. Agora, como o caminho completo não escolhido totalmente no vértice/fonte, mas, sim, de uma maneira progressiva nos vértices S/F, precisamos uma outra função para escolher o S/F. Então, assinalando idéias do modelo número 1 (figura IV.9), em particular, a escolha do caminho na transição 2A desta figura IV.9, podemos considerar o seguinte: existe, em cada vértice S/F, uma tabela que indica, para cada destinação, um feixe de caminhos possíveis para alcançar uma determinada destinação. Assim, a escolha, digamos, c = sel(fc,...), baseada em diversos critérios, indica um determinado caminho via um vértice vizinho específico. Então, considerando o vértice v_i sempre como "início" do caminho restante, a escolha dos vértices v_j poderia ser determinada, de um modo geral, através de uma função do tipo

$$v_j = f(c_{v_i \to v_r}) \text{ , onde } c_{v_i \to v_r} = sel(fc_{v_i \to v_r} , \ldots) .$$

Veremos mais detalhes sobre esta escolha na seção V.2.b.3.β;

- O terceiro tipo dos vértices S/F, o "receptor final v_r", num caminho entre V/F e V/D, é similar ao segundo tipo, acrescentando somente a condição que $v_j = v_r$.
  Esta condição será também necessária para determinar, na representação gráfica, o momento quando "sairmos" da subrede de comunicação para entrar no "domínio" do receptor r ($r \varepsilon U$).

c) Como simplificar as inscrições?

- Para fazê-las de uma maneira simplificada, na representação gráfica, sem, contudo, modificar o conteúdo das mesmas, podemos usar as idéias lançadas na seção V.2.a.3-β, isto é,

$$m = f_m(s,r,p) \quad ,$$
$$em = v_i b_k \quad ,$$
$$re = v_j t_1 \quad .$$

  Assim, temos inscrições simples, fácil de manejar e que, no caso em que isto se faz necessário, também podem ser substituídas pelas inscrições originais e mais completas.

V.2.b.2 - Modelagem da primeira fase: a entrada na subrede de comunicação.

Podemos considerar a modelagem em torno do canal s' como sendo a "entrada na subrede" de comunicação no nível dos vértices S/F. Já discutimos bastante a este respeito na seção V.2.a.3.β (sobre a saída de um vértice/fonte) e mostramos estas idéias na figura V.10. Nesta figura, se vê, entre outras, as transições 2-Oa, 2 e 2-Ob, que refletem a passagem de uma mensagem <s,r> do canal s (predicado FONTE(s,r)) para o canal s' (predicado PRONTO(m,em,re)), sendo ela ainda submetida, na UCP, a algum tratamento (p. ex., empacotar).

Assim, para conseguir "entrar" na subrede, precisamos passar por três etapas:

- sair do canal s
- tratamento na UCP
- entrar no canal s' .

Recapitulamos, primeiramente, o que já sabemos com respeito à saída do canal s, que acontece em torno da transição 2-Oa (figura V.10):

Temos, como uma das precondições, a posição da mensagem no canal s, representado pelo predicado FONTE(s,r). Em particular, quando a mensagem já foi tratada na FONTE/FILA(s,r), ela está presente no subpredicado FONTE/S(s,r) que representa a disposição de uma continuação na progressão, entre V/F e V/D, desta mensagem.

A passagem até o predicado CF/PREP/INI(s,r), que representaria a disposição para entrar na fila da UCP, se faz com a permissão $<>_V$, dada pela LÓGICA/MUX$(()_V)$, que, afinal, é o responsável pela coordenação da competição em torno da UCP.

Indicamos, também, a presença do responsável pela mensagem, representado pelos predicados ATIV/TRANSM(u) e TRANSMITIR(u), que, eventualmente, poderia ser identificado como sendo, nesta fase da modelagem, o processador de entrada na fila s do vértice v_s. Mas, trataremos, por convenção, tudo o que está "fora da subrede" como sendo "usuário".

O tratamento na UCP se faz entre os predicados CF/PREP/INI (s,r), precondição da transição 2, e CF/PREP/FIM(s,r,p), sua pos condição:

Já indicamos, nas figuras V.6, V.7 e V.8, como a mensagem entraria na fila da UCP, o que se pode fazer neste processador central (p.ex., executar a função $p=f_p(s,r)$) e como se libera novamente o espaço ocupado durante o processamento.

Assim, a mensagem presente em CF/PREP/FIM(s,r,p) está pronta para realmente entrar no canal artificial s'.

O que vale destacar, neste momento, é que o responsável pela mensagem, o usuário s, se coloca num estado ESP/RFNM(u) que indica que, para ele, a transmissão, no nível V/F-V/D, já começou; ou melhor, que a mensagem saiu do seu domínio para entrar no domínio de meio de transporte.

Para ter, na chegada do RFNM associada à mensagem, condições de verificações, foi guardada também uma cópia da mensagem, representada pelo predicado SEG.VIA.M(s,r,p).

Para realmente entrar no canal artificial s', representado pelo predicado PRONTO(m,em,re), precisa-se preparar os pacotes no sentido de determinar quais os "novos responsáveis" por eles:

Faremos isto na transição 2-Ob que, usando as suas funções, transformou o pacote <s,r,p>, vindo de CF/PREP/FIM(s,r,p), para que contenha indicações sobre emissor e receptor dentro da sub-rede.

Então, do ponto de vista do canal artificial s', o emissor responsável pelo pacote foi o canal s, indicado pela fórmula $em = (v_i b_k) = v_s s$; e o receptor é o canal s', indicado por $re = (v_j t_1) = v_s t_o$.

O que falta é somente ativar este receptor. Faremos isto a partir de um 'pool', predicado POOL/PE(re,p), que representa o estado passivo, para chegar a PE/VERIFICAR(re,p), que representaria um dos estados ativos deste responsável. Lembre que o primeiro argumento, re, representa a identificação física, e o segundo, p, o pacote por quem ele é responsável.

V.2.b.3 - Modelagem da segunda fase: o encaminhamento inicial da mensagem à fila de saída associada ao caminho para o próximo vértice.

Antes de considerar os problemas da modelagem em torno da transmissão 6, precisamos pensar um pouco mais sobre a estrutura de um vértice S/F e, também, sobre a maneira de como seleciо

nar o próximo vértice no caminho entre V/F e V/D.

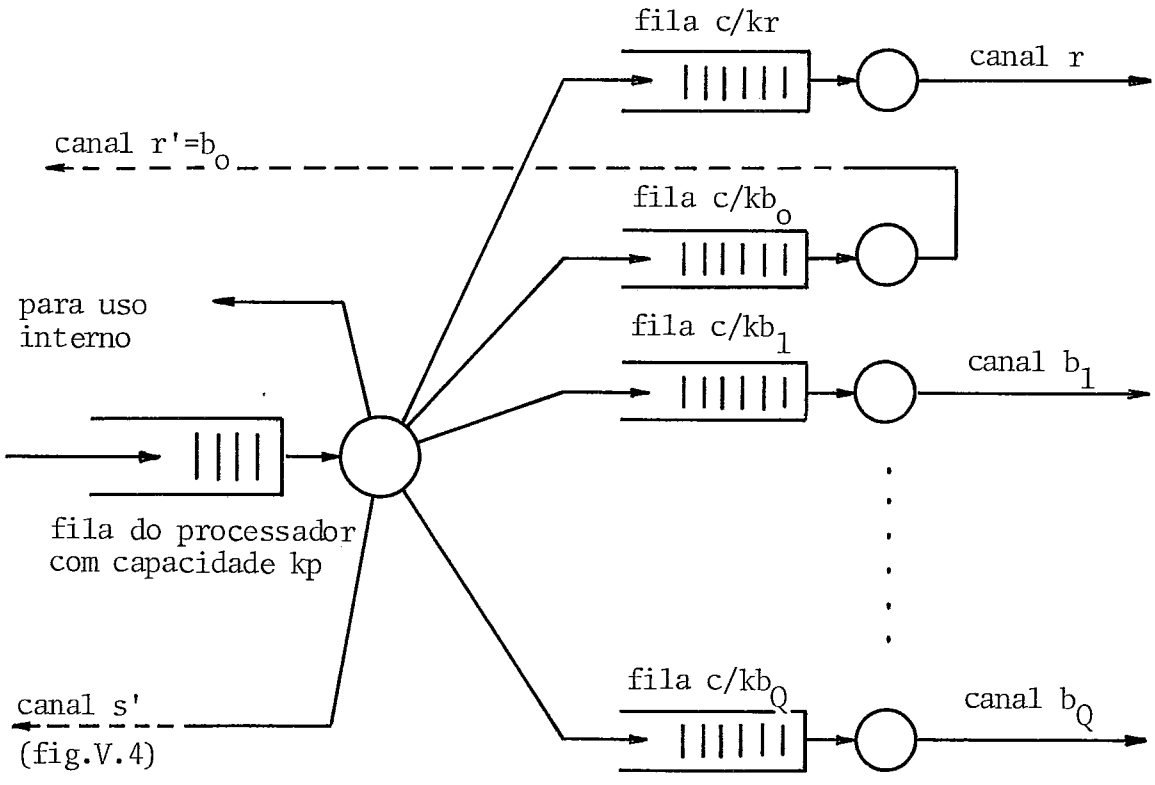
V.2.b.3.α - Estrutura de um vértice.

Observando a estrutura básica de um vértice (figura V.3), vemos que, além de especificar mais detalhadamente a "entrada" (figura V.4), precisamos também pormenorizar mais a "saída". Então, sem tentar resolver, por enquanto, questões relativas ao método de distribuição do espaço disponível, SCHWARZ (30), podemos dizer que cada canal para um vértice vizinho pode ser considerado como sendo o servidor da fila associada. Atribuindo ainda capacidades às filas, chegamos à representação gráfica como mostrada na figura V.11.

Assim, de um modo geral, um predicado similar ao M-em-TRÂNSITO da figura IV.9 poderia, eventualmente, representar a situação da mensagem na fila de saída, isto é, como sendo em "trânsito". Vejamos: o predicado M-em-TRÂNSITO(s,r,p) da fig. IV.9 precisa ser adaptado levando em consideração de que se trata, agora, de um pacote em trânsito entre vértices vizinhos. Então, conforme já explicamos na seção V.2.b.2, seria evidente usar identificadores para o emissor e receptor imediato. O identificador para o emissor do pacote p será a tupla $<em,p>$ ($\hat{=} <v_i b_k,p>$), o que indica o k-ésimo processador de saída no i-ésimo vértice S/F, e, para o receptor deste pacote p, a tupla consiste de $<re,p>$ ($\hat{=} v_j t_1,p$).

Deste modo, utilizando ainda a simplificação $m=f_m(s,r,p)$, chegaríamos a um predicado do tipo P-em-TRÂNSITO(m,em,re). Entretanto, a pergunta que surge, de imediato, é: Como chegar, dentro do esquema da modelagem, até este predicado se queremos, ou melhor, se devemos considerar, entre outros, o pacote com suas informações inerentes, os espaços disponíveis na fila de saída, os processadores de saída (responsável pela emissão) e de entrada (responsável pela recepção no próximo vértice)?

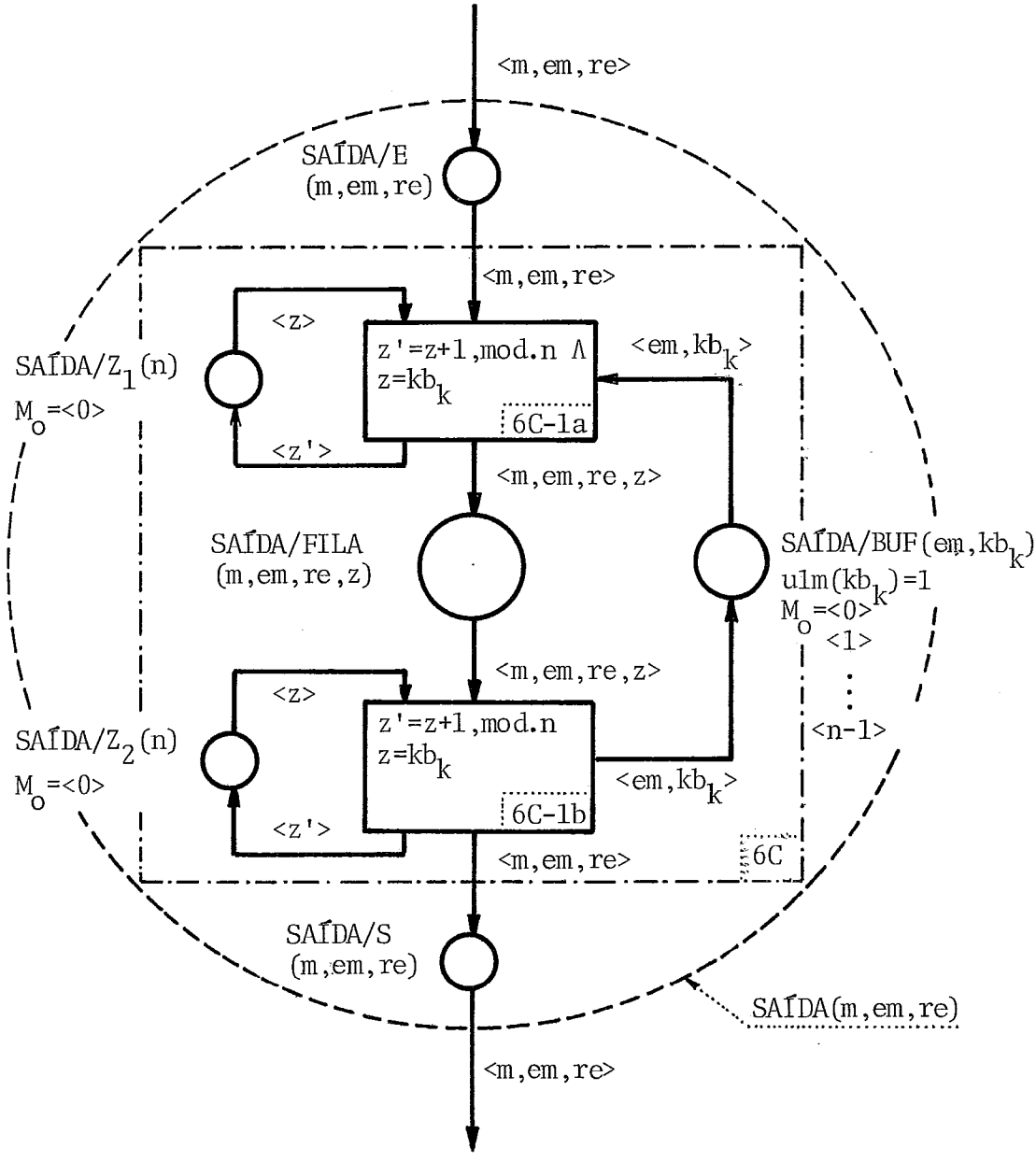
A primeira providência a tomar é a de intercalar mais um estado entre PRONTO e P-em-TRÂNSITO. Esta decisão se deve, princi



onde:

canal r	equivalente à ligação ao hospedeiro associado.
canal r' ou b_o	equivalente à ligação abstrata deste vértice (≙ saída da subrede) à fila r
canal b_i	canais para outros vértices (internos à subrede de comunicação)
kb_i, kr, kp	capacidades das diversas filas

Fig. V.11: Esboço sobre as filas de saída de um vértice S/F e a associação de diversas capacidades.



onde: ——————— modelo simplificado do predicado SAÍDA

-·-·-·-·- modelo simplificado da transição 6C

Fig. V.12: Detalhamento do predicado SAÍDA(m,em,re), modelado como sendo uma fila.

palmente, ao fato que queremos, nesta modelagem bem mais complexa que aquela da figura IV.9, resolver problemas ligados à ativação real do receptor antes de colocar o pacote em trânsito. Assim, deixamos a explicação do predicado P-em-TRÂNSITO para uma fase posterior (seção V.2.b.4) e contentamo-nos, por enquanto, com um predicado do tipo SAÍDA(m,em,re), como foi mostrado na figura V.12.

Este predicado foi modelado conforme o modelo geral de uma fila (figura IV.2), a saber, que consta de uma seqüência de predicados SAÍDA/E, SAÍDA/FILA e SAÍDA/S, auxiliados pelos contadores/sincronizadores SAÍDA/Z_i,i=1,2, para administrar corretamente o espaço em SAÍDA/BUF.

Em seguida, conforme já indicamos no modelo número 1 no capítulo IV, precisamos nos preocupar com a principal precondição para obter uma transmissão eficiente, ou seja, precisamos escolher o melhor caminho possível.

V.2.b.3.β - Escolha do caminho

Começamos, então, a escolha do caminho (figura V.13), com o predicado CAMINHO(s,r,cv) da figura IV.9, que representou todos os caminhos virtuais imagináveis cv (cv ε CV) entre todos os pares possíveis fonte-receptor. Pode-se admitir, que estes caminhos dependem da configuração inicial (topologia, capacidades, fluxos estimados, etc.) e que, assim, representam um conjunto de caminhos que somente será alterado por modificações extremas (p. ex., ruptura de uma linha, decisão arbitrária da gerência, etc.); portanto, não será necessário uma modelagem no contexto deste trabalho. Assim, temos um predicado

CAMINHOS/VIRTUAIS(s,r,cv) , s ε EE, r ε ER, cv ε CV ,

que representa estes caminhos cv no nível dos usuários.

O que precisamos agora é colocar estes caminhos no nível da subrede de comunicação já que a escolha efetiva de um determinado caminho, no contexto do funcionamento da rede, se fará entre

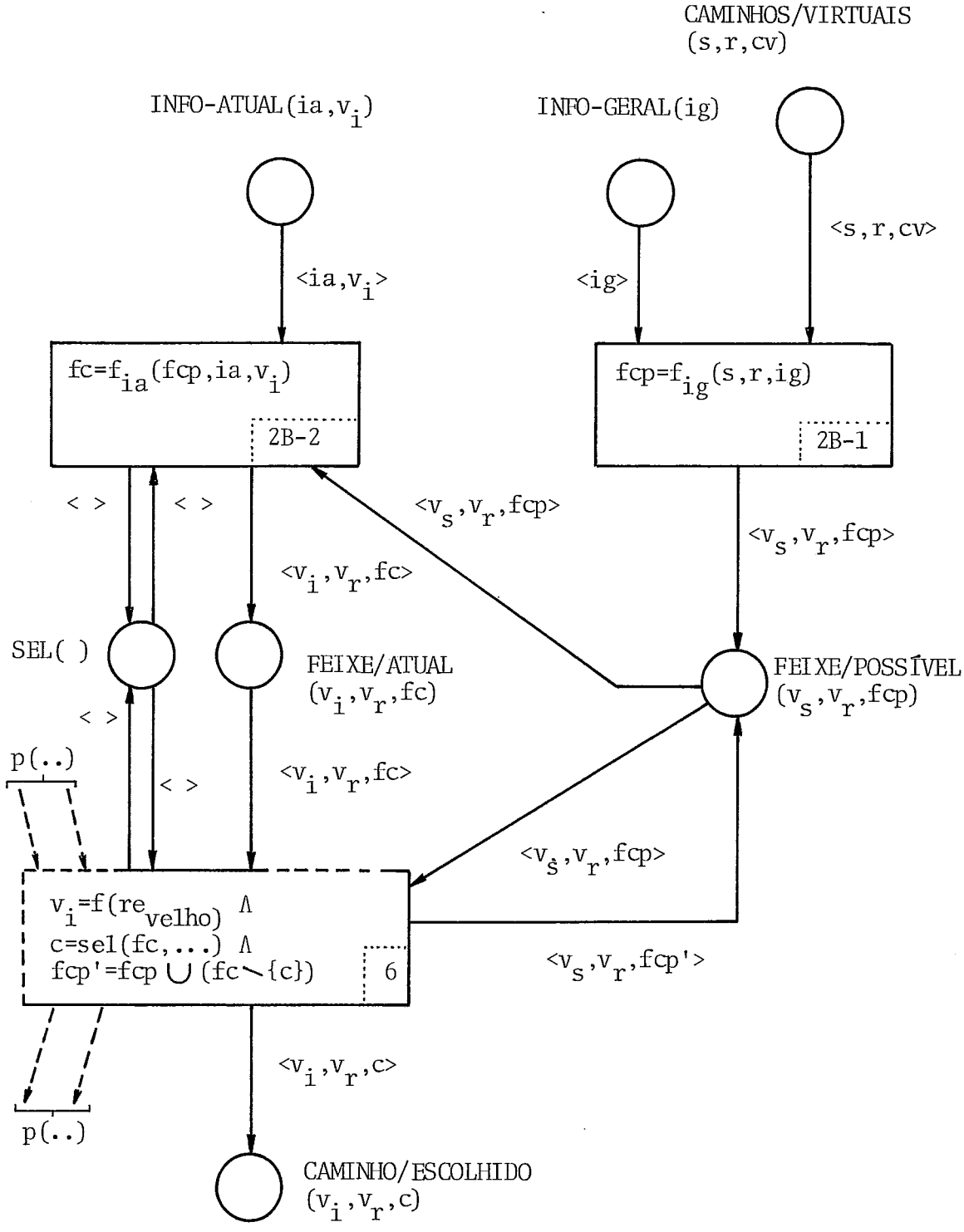


Fig. V.13: Esboço sobre as idéias de como preparar a escolha final (c=sel(fc,m,emv,rev,v_i)) de um caminho.(p(..) indica os outros predicados em torno da transição 6; veja figura V.14).

vértices desta subrede. Assim, chegamos ao predicado FEIXE/POSSÍVEL(s,r,fc) da figura IV.9, que, conforme acabamos de explicar, se tornará

$$\text{FEIXE/POSSÍVEL}(v_s, v_r, fcp) \quad , \quad v_s, v_r \in V \quad , \quad fcp \in FCP \ .$$

Este predicado representa um feixe de caminhos possíveis fcp ($fcp \in FCP \subset CV$), escolhido dentro do conjunto de caminhos virtuais, cuja escolha é dependente de regras preestabelecidas e de informações gerais. Estas informações, válidas no sentido amplo, estão relacionadas aos próximos vértices, aos vértices atuais, ao desempenho da rede, etc. Representamos esta fonte de informações pelo predicado INFO/GERAL(ig) onde o argumento ig ($ig \in IG$) representará os elementos desta fonte.

Deve ficar claro que a formação destes feixes e, mais tarde, também a sua atualização são acontecimentos dinâmicos. Mas fica agora a pergunta: Quão dinâmicas devem ser esta formação e atualização? Bom, como esta pergunta é um pouco polêmica e como a resposta envolve investigações sobre a quantidade e o conteúdo das informações necessárias para uma atualização, assim como sobre a freqüência de um intercâmbio destas informações entre os vértices, vamos tentar, novamente, separar os acontecimentos no sentido de necessidades imediatas e aquelas dentro de uma certa margem de tempo.

Então, voltando ao problema da formação do feixe de caminhos possíveis (ou de vários feixes, dependendo, eventualmente, de classes e prioridades diferentes das mensagens), podemos dizer que ela é considerada como sendo um acontecimento raro que ocorre, p.ex., somente na inicialização do funcionamento da rede ou no caso da confirmação de modificações significativas e permanentes no comportamento da rede. Podemos modelar isto pela fórmula na transição 2B-1, $fcp = f_{ig}(s,r,ig)$, evitando, assim, nesta altura da modelagem, as investigações relativas à maneira de como adquirir e usar estas informações gerais. Observe que se justifica, nesta fórmula, o uso de s e r, porque o critério da escolha de um caminho, ou de feixe de caminhos, depende dos usuários envolvidos, considerando, eventualmente, questões relativas a

classes, prioridades, etc.

O próximo ponto, a atualização deste feixe, para obter um feixe atual fc (fc ε FC⊂ FCP), depende do funcionamento efetivo da rede, isto é, das informações atuais (no sentido tempo real), disponíveis em qualquer vértice v_i, onde v_i é sempre considerado como sendo o início de um caminho. Modelamos este tipo de informação pelo predicado similar ao INFO/GERAL(ig), i.e., INFO/ATUAL(ia,v_i). O uso destas informações deve ocorrer "bastante freqüentemente", especificando, p.ex., que o intervalo das atualizações, entretanto, deve ser bem maior que, digamos, o tempo de envio de uma mensagem. Modelamos este fato de atualização pela transição 2B-2 e sua fórmula, $fc=f_{ia}(fcp,ia,v_i)$. O predicado subseqüente,

$$\text{FEIXE/ATUAL}(v_i,v_r,fc)\ ,\ v_i,v_r \in V,\ fc \in FC\ ,$$

representa os feixes de caminhos, a partir de um determinado vértice v_i, para qualquer destinação v_r e contém, assim, as precondições para a seleção final.

Como observação adicional podemos ainda avisar que precisa-se ter o cuidado de "congelar" o feixe fc neste predicado no momento da seleção efetiva de um caminho, ou seja, as transições 6 e 2B-2 não devem disparar ao mesmo tempo (a modelagem através do predicado SEL() é supersimplificada, mas não a querendo detalhar mais neste momento, ela serve, pelo menos, para alertar sobre este fato).

Feita esta atualização, podemos afirmar que a escolha final, isto é, as decisões e ações em relação ao que caminho realmente escolher, é restrita a poucas funções que podem ser executadas num tempo mínimo (observação: o tempo de processamento envolvido diretamente no controle de fluxo deve ter, obrigatoriamente, a característica de operações em tempo real!):

- Dependendo da classe, tipo, prioridade, etc., da mensagem empacotada, olhar na tabela em questão e escolher o caminho para aquele vértice vizinho que indica o "melhor" caminho para

uma determinada destinação;

- Verificar se há espaço disponível na fila de saída que leva a este vértice escolhido. Em caso afirmativo, colocar a mensagem nesta fila de saída. Em caso negativo, temos que tomar outras providências como, p.ex., escolher um outro caminho, rejeitar o pacote, esperar um certo tempo, etc. (não modelaremos este caso negativo neste trabalho).

Então, tendo refletido um pouco sobre a estrutura do vértice em relação à fila de saída e linha de transmissão (trânsito) e, também, sobre a maneira como preparar a escolha final de um caminho, podemos agora nos dedicar à modelagem em torno da transição 6.

V.2.b.3.γ - Observações preliminares em relação à modelagem para colocar o pacote na fila de saída associado ao caminho escolhido.

Vejamos, primeiramente, o que temos à disposição para iniciar a modelagem:

- Da seção V.2.b.2 sabemos que o pacote está representado pelo predicado PRONTO(m,em,re). Mais exatamente, podemos dizer que é o subpredicado PRONTO/S(m,em,re), que indica que o pacote está realmente pronto para ser submetido ao roteamento;

- Também temos o responsável por este pacote, a saber, o processador de entrada (no canal artificial s'), representado pelo predicado PE/VERIFICAR(re,p);

- Temos, ainda, um predicado FEIXE/ATUAL(v_i,v_r,fc), que permitirá a escolha do melhor caminho a partir deste vértice atual.

Tendo estas condições, poderíamos, já, pensar em escolher o caminho. Entretanto, já antecipando o caso em que o pacote recebido chegou de um verdadeiro vértice vizinho, a saber, que chegou num canal t_i, observamos que o predicado PRONTO/S(m,em,re)

não será, certamente, o lugar adequado para "juntar os dois fluxos".

Criamos, então, em termos de modelagem, um novo predicado, digamos, PAC/ACEITO(m,em,re), que refletiria que há um pacote pronto para ser submetido ao processo de encaminhar, não importando se ele vem da fila de entrada artificial s' ou dos canais t_i .

Este predicado poderia também ser chamado de CF/ENCAM/INI(.) se pensássemos em termos da fila da UCP. Observe, na figura V.15, a parte da fila da UCP engajada neste processamento, representado pelo predicado CF/ENCAMINHAR(m,em,re,z). Então, para atender a estas duas idéias, daremos a este predicado o nome PAC/ /ACEITO(m,em,re), sabendo que é um pseudônimo de CF/ENCAM/INI (m,em,re).

Deste modo, agora sim, temos as precondições para iniciar o processamento na UCP em relação ao encaminhamento:

- PAC/ACEITO(m,em,re), m ε M, em ε EM, re ε RE;
- PE/APRONTAR(re,p), o que significa um estado subseqüente ao de verificar, assegurando, assim, que o pacote somente sairá de sua responsabilidade no momento exato em que a UCP assume, ou seja, quando o elemento realmente entrou na fila da UCP, valendo re ε RE, p ε P;
- FEIXE/ATUAL(v_i,v_r,fc), com v_i,v_r ε V, fc ε FC .

Precisamos, ainda, enumerar o que queremos obter depois da transição, representado pelas poscondições:

- um determinado caminho escolhido (p.ex., representado pelo predicado CAMINHO/ESCOLHIDO(v_i,v_r,c), v_i,v_r ε V, c ε FC);
- o pacote em questão presente na fila de saída em relação a este caminho (p.ex., predicado SAÍDA(m,em,re), com m ε M, em ε EM, re ε RE);
- alguém responsável pelo processamento na fila de saída e pela

emissão a fazer (p.ex., predicado PROC/SAÍDA(em,p), onde em ε EM, p ε P).

Tendo estas condições, podemos, já, esboçar um modelo preliminar. Veja isto na figura V.14. Mas como, obviamente, queremos saber algo mais a respeito dos acontecimentos na transição 6, vamos, em seguida, "abrir" esta transição e pormenorizá-la mais.

V.2.b.3.δ - Detalhes sobre a "passagem" do pacote e do processador pela transição 6.

Tendo determinado o conjunto das pre e poscondições em torno da transição 6, podemos nos preocupar, agora, com o que acontece realmente dentro desta transição 6 já que, no esboço da figura V.14, isto foi indicado de uma maneira meio informal.

Admitindo que a parte da fila da UCP, encarregada da função de encaminhar, seja modelada de uma maneira bem similar àquela que tratou da preparação (figura V.6), podemos "seguir" o fluxo de um pacote desde PRONTO(m,em,re) até SAÍDA(m,em,re) e estabelecer, assim, as funções necessárias para esta passagem.

Observamos, então, na figura V.15, as fases de entrada pelas quais o pacote e o processador PE passam:

- Sabemos que o fato de que o pacote está pronto para se submeter ao processamento de encaminhar (predicado PAC/ACEITO) é uma das precondições principais para disparar a subtransição 6-1a; isto foi modelado pelo predicado mencionado que, também, repetimos isto, poderia ser chamado de outra maneira, ou seja, CF/ENCAM/INI(m,em,re). Agora, tendo em mente a figura V.6, sabemos que os elementos deste predicado competem pelo acesso à fila da UCP da mesma maneira como o fizeram aqueles da fila FONTE ou INTERNO/PROBL. Então, o que, na figura V.6, foi modelado pelas transições 2 e 5, respectivamente, será agora modelado pela transição 6;

- Devemos ainda falar da outra precondição principal que representa a responsabilidade pelo pacote. Então, passando o pacote da fila de entrada (canal s' ou t_i) para a UCP, a responsabilidade pelo pacote p passará do processador de entrada, modelado pelos predicados PE/VERIFICAR(re,p) e PE/APRONTAR(re,p), para a própria UCP. Assim, o processador de entrada pode "voltar" ao 'pool' (poscondição, representada pelo predicado POOL/PE(re,p)), enquanto a ação da UCP poderia ser representada como indicado na figura V.8 (não modelaremos este fato aqui);

Evidentemente, poderíamos discutir se esta transição 6-1a é realmente responsável pela passagem da responsabilidade, ou se esta transferência deveria ser feita em etapas (p.ex., o PE deixa ser responsável a partir da transição 2A e a UCP começa assumir a partir de 6-1a). Como este problema já envolve bastante a arquitetura de um computador/vértice, deixamos somente este alerta registrado aqui.

- Todas as outras condições em torno da subtransição 6-1a estão associadas à UCP; assim, a "entrada" na fila da UCP é representada, neste contexto, pela poscondição: predicado CF/ /ENCAMINHAR(m,em,re,z), auxiliado pelos predicados UCP/BUF(v,kp) e CF/ENCAM/Z_1(n);

- Para completar as descrições em torno desta subtransição 6-1a, precisa-se ainda colocar as suas fórmulas. Devemos, então, além das inscrições habituais para entrar numa fila

$$z' = z+1,\text{mod.}n \;\Lambda\; z \;\#\; \langle v,kp\rangle \;,$$

indicar a "neutralização ou negação" das informações sobre emissor e receptor. Podemos fazer isto, usando, p.ex., a expressão

$$em_{velho} = emv = (v_i b_k) = em \;\Lambda\; re_{velho} = rev = (v_j t_1) = re.$$

Estas informações "velhas" servirão, do ponto de vista da seleção de um caminho, somente para evitar que se "retorne" no caminho em vez de avançar.

Supondo que já tenhamos o resultado em relação ao caminho, podemos descrever o que acontece, na saída da UCP (figura V.15), com o pacote e o processador PS (subtransição 6-1b):

- O pacote, já na forma final para a próxima emissão, <m,em,re>($\hat{=}$ <m,$v_i b_k$,$v_j t_l$>), ou seja, já contendo as indicações novas sobre emissor e receptor, está inscrito num predicado CF/ENCAM/INI(m,em,re) que, por sua vez, pode ser condiderado como sendo associado ao predicado principal CF/ENCAMINHAR(m,em,re,z). Este predicado, que é uma das pós-condições para sair da UCP, indica que há, pelo menos, um espaço disponível, podendo, assim, "liberar" a UCP para outras tarefas. No caso em que este espaço mínimo não está disponível, tem que se tomar providências urgentes para não bloquear a UCP. Estas providências podem ser a escolha de um outro caminho, a rejeição do pacote, esperar mais um tempo, etc.; como estamos tentando criar um modelo "positivo", isto é, sobre acontecimentos normais e não-conflitantes, não modelaremos este fato agora. Entretanto, em estudos posteriores, devemos, obrigatoriamente, estudar todos estes fatores perturbadores;

- Em relação ao responsável pelo pacote podemos, a partir das informações contidas no pacote sobre emissor e receptor, ativar o processador de saída necessário. Então, a partir de um 'pool', predicado POOL/PS(em,p), chegamos ao processador de saída, predicado PROC/SAÍDA(em,p);

- As fórmulas da subtransição 6-1b consistem, além das inscrições já conhecidas para sair de uma fila

  $$z' = z+1,\mathrm{mod}.n \wedge z = \# \langle v,kp \rangle ,$$

  da identificação dos "novos" emissor e receptor, ou seja, a expressão

  $$em = emn = em_{novo} \wedge re = ren = re_{novo} ,$$

  permitindo, assim, ativar o processador de saída em questão.

Tendo discutido os problemas em relação ao pacote e aos processadores, falta ainda alguma explicação em relação à escolha

definitiva do caminho.

V.2.b.3.ε - Detalhes sobre a escolha definitiva do caminho.

Já falamos bastante sobre a preparação do feixe de caminhos na seção V.2.b.3.β. O que falta agora é a escolha final, isto é, olhar na tabela e verificar se há espaço na fila de saída. Todas estas ações são feitas pela UCP e se pode imaginar um esquema similar aquele da figura V.16:

- A partir de um subpredicado de CF/ENCAMINHAR(m,em,re,z), digamos, o predicado CF/ENC-SEL/INI (m,emv,rev,z), pode se fazer a escolha do caminho. Observe que todas as informações necessárias para esta escolha estão contidas no pacote <m,emv,rev,z> . Assim, usando como outra precondição, para a transição 6-2, o predicado FEIXE/ATUAL(v_i,v_r,fc), pode-se, p.ex., na subtransição 6-2a, fazer a escolha e liberar os outros caminhos do feixe:

  v_i=f(rev) Λ c=sel(fc,m,emv,rev,v_i) Λ fcp'=fcp ⋃ (fc∖{c}) ;

- Ignoramos, por enquanto, como esta seleção se realiza dentro do processador central. Por isso, colocamos todo o conjunto de operações dentro da transição 6-2. Assim, uma vez escolhido o caminho, pode-se, p.ex., na subtransição 6-2b, usar funções apropriadas para preencher os campos de endereço com as novas indicações. Então, teremos como "novo" emissor e receptor, usando identificações gerais, a seguinte expressão:

  $em_{novo} = f_{em}(c) = (v_i b_k) = emn$ Λ

  $re_{novo} = f_{re}(c) = (v_j t_1) = ren.$

  Desta maneira, a precondição CF/ENC-SEL/FIM(m,emn,ren,z), para disparar a transição 6-2, contém todas as informações necessárias para colocar o pacote na fila de saída apropriada.

Como observação complementar, em relação às idéias mencionadas no final da seção V.2.b.3.β, podemos dizer que o conjunto em torno do predicado CF/ENCAMINHAR(m,em,re,z) representa a escolha

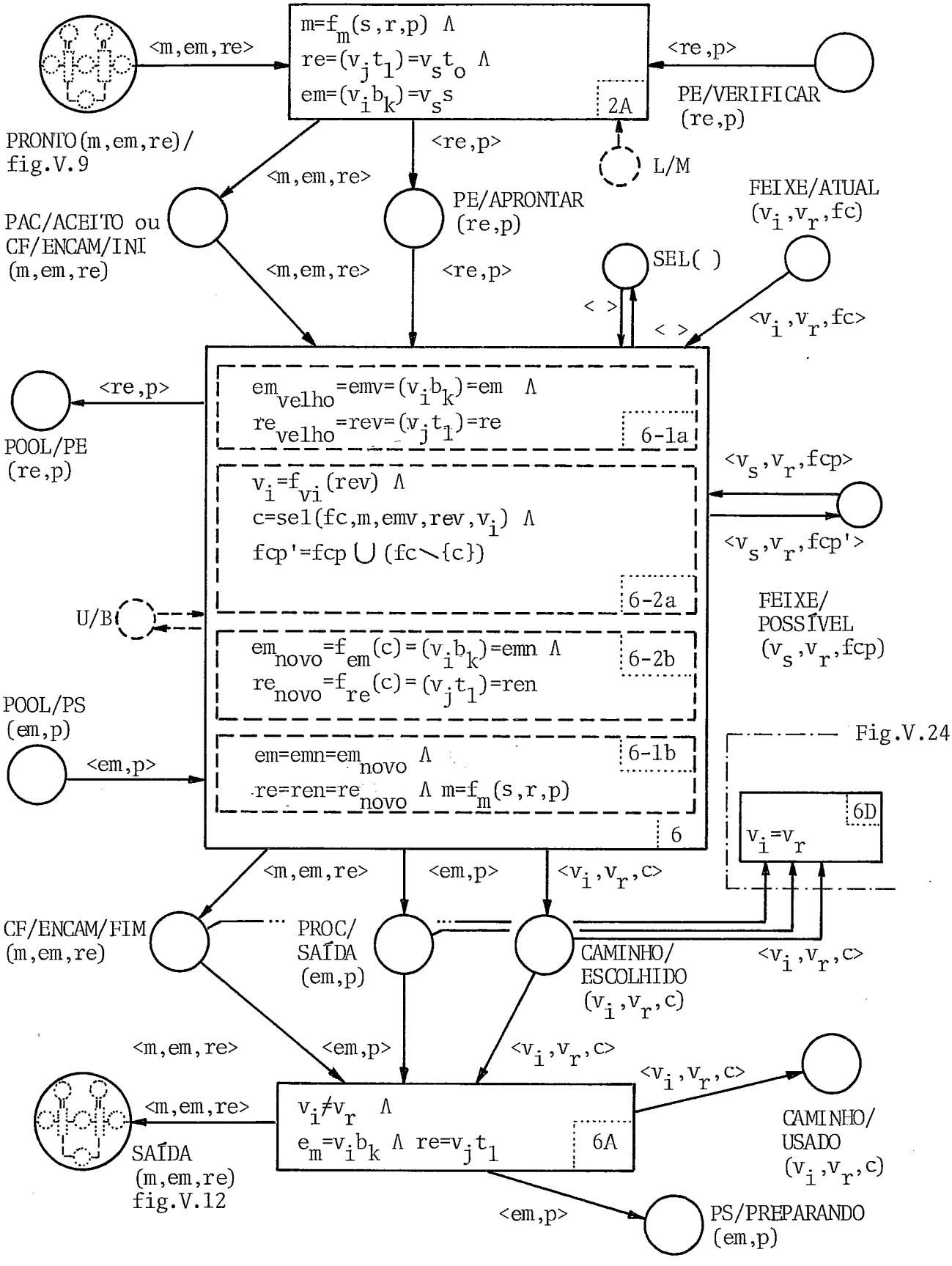


Fig. V.14: Esboço dos acontecimentos em torno e dentro da transição 6 (veja os detalhes da seqüência do pacote e do processador na fig. V.15, e os detalhes em relação ao caminho na fig. V.16).

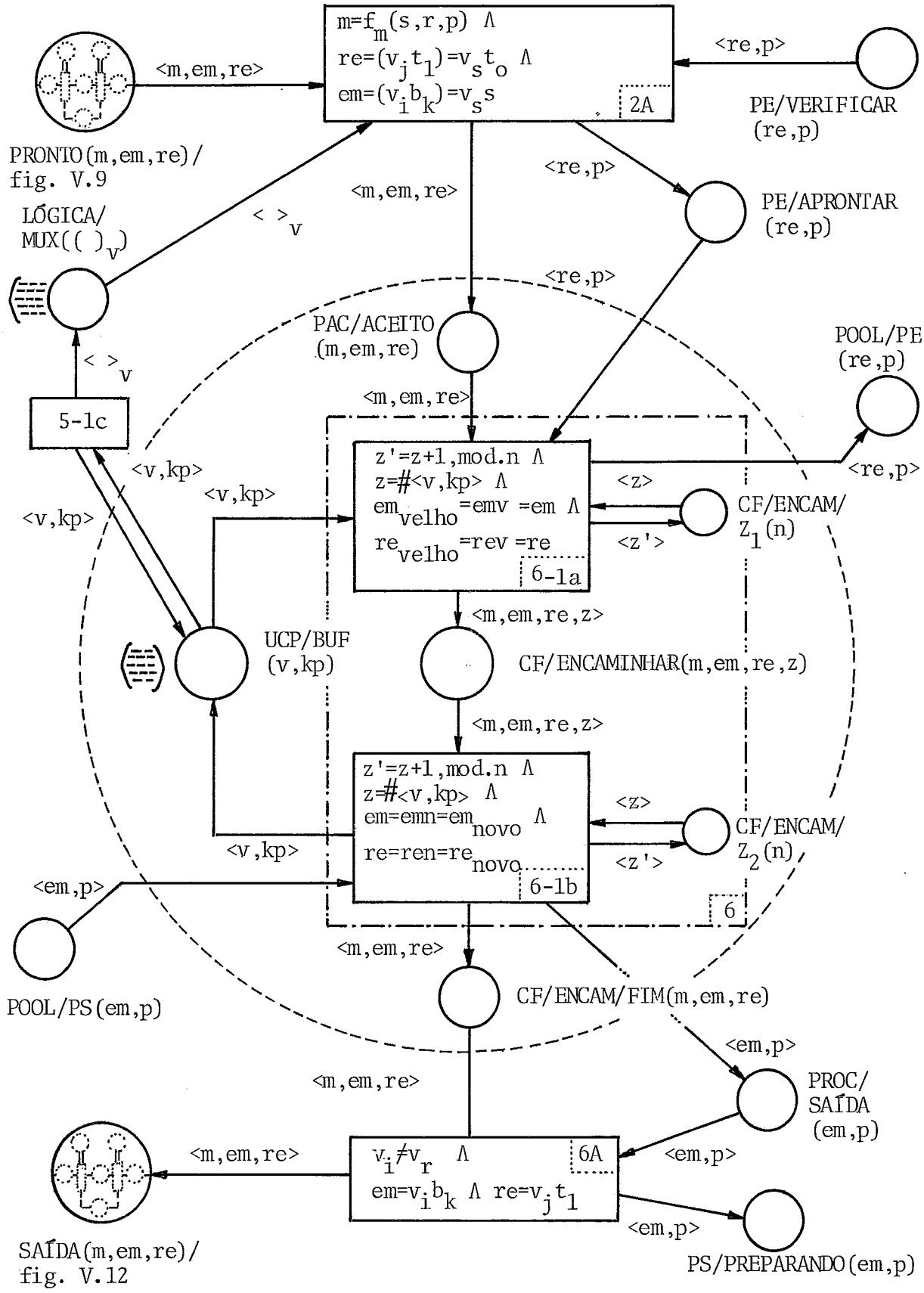
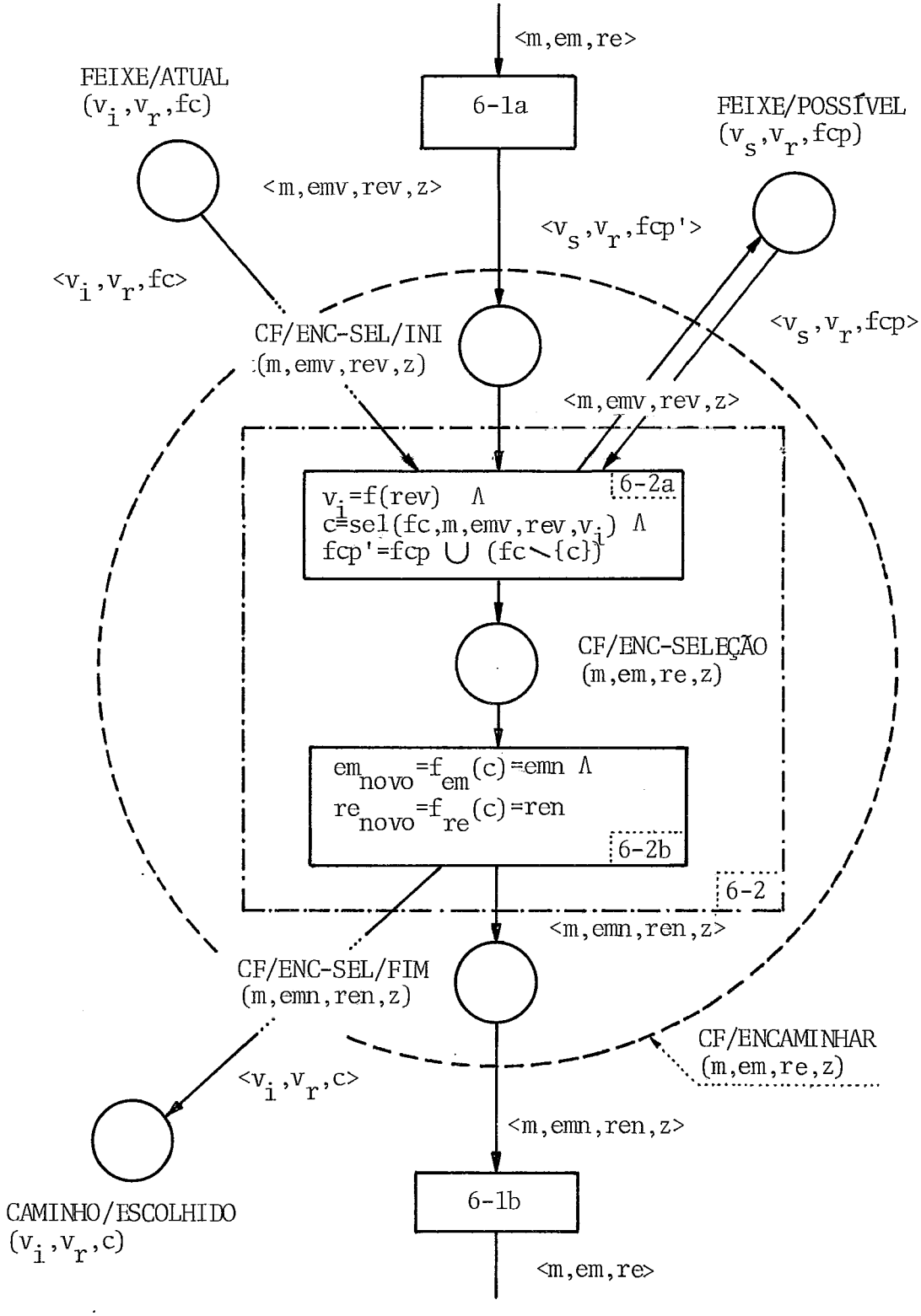


Fig. V.15: Detalhe da seqüência do pacote e do processador entre o canal de entrada (s' ou t_1) e o da saída (canal b_k).



Obs.: A competição pelo atendimento na UCP poderia ser similar aquela do esquema na figura V.8.

Fig. V.16: Esquema simplificado da seleção de um caminho (detalhe do predicado CF/ENCAMINHAR da fig. V.15).

do caminho (p.ex., o mencionado "olhar na tabela), destacando o predicado CF/ENCAM/FIM(m,em,re) como indicação de que a escolha já foi feita, e que uma das poscondições da transição 6A, o predicado SAÍDA(m,em,re), notadamente, o subpredicado SAÍDA/ /E(m,em,re), reflete a verificação de se há espaço na fila de saída escolhida.

Tendo colocado o pacote na fila de saída, podemos nos preocupar, finalmente, como colocá-lo em trânsito. Veja isto na seção V.2.b.4.

V.2.b.4 - Modelagem da terceira fase: ativação do vértice vizinho e colocação do pacote em trânsito.

Continuando a usar, como referência, o modelo número 1 (fig. IV.9), desta vez em relação à recepção de um pacote, poderíamos adaptar as idéias da terceira fase na criação deste modelo (cap. IV.3.c) ao nosso objetivo atual. Mas, ao contrário daquela modelagem, onde a ativação do receptor foi modelada de uma maneira bem superficial, devemos nos preocupar, desta vez, com alguma interdependência entre receptor e emissor. Foi esta a razão pela qual não colocamos, na seção V.2.b.3, o pacote em trânsito, mas somente numa fila de saída.

Assim, podemos dizer que já temos algumas das precondições básicas para iniciar uma transmissão, a saber, o caminho a ser usado (CAMINHO/USADO(v_i,v_r,c)), o pacote colocado na fila de saída associado a este canal (SAÍDA(m,em,re)) e o processador daquela saída num estado ativado (PS/PREPARANDO(em,p)), que permitem efetuar uma ativação do receptor o que, por sua vez, será uma das precondições para receber, posteriormente, este mesmo pacote no vértice vizinho.

O que falta, então, principalmente, para realizar o objetivo mencionado, é a modelagem desta ativação do receptor. No modelo número 1 (figura IV.9), esta recepção se referiu ao usuário o que, em nosso caso atual, somente acontecerá quando o pacote "sair" da subrede de comunicação. Mas, também, não modelamos ex-

plicitamente esta ativação.

Agora, para ativar o responsável pela entrada no próximo vértice, podemos usar a idéia da fig. V.16. Lá foi mostrado que o próximo vértice seria uma função do caminho escolhido, ou seja, $re_{novo}=f_{re}(c)$. Assim, já pensando um pouco na realidade física, podemos ativar, primeiramente, o enlace entre estes dois vértices, modelado pelo predicado ENLACE/ATIVADO(enl), e depois, através do enlace ativado, colocar o receptor em prontidão. Observe, na figura V.17, este tipo da modelagem onde, a partir de um 'pool' para o receptor, predicado POOL/PE(re,p), chegamos ao processador de entrada, no vértice vizinho, modelado pelo predicado PROC/ENTRADA(re,p).

Em relação ao enlace e ao caminho (que contém este enlace) temos ainda o seguinte a dizer:

Observe, que usamos as informações "caminhos virtuais, feixes possíveis, feixe atual de caminhos, etc." somente para determinar que rota propor para o pacote. Agora, em torno da transição 6B-1, aparece um predicado chamado POOL/ENLACE(enl) que representaria, em termos abstratos, um recurso real da subrede: a ligação física/lógica entre vértices vizinhos. Assim, podemos "devolver" a indicação sobre o caminho escolhido para o conjunto de informações sobre caminhos em geral (FEIXE/POSSÍVEL (v_s,v_r,fcp), onde são feitas as atualizações), e nos preocupar em modelar conscientemente este recurso mencionado e disputado pelos pacotes de todas as classes de usuários. A modelagem se faz similarmente àquela de outros recursos, ou seja, a partir de um 'pool' de enlaces, predicado POOL/ENLACE(enl), através da fórmula da transição 6B-1,

$$em=v_i b_k \wedge re=f(c)=v_j t_l \wedge enl=c_{v_i \to v_r} \cap c_{v_i \to re} \wedge$$

$$fcp'=fcp \cup \{c\} ,$$

obtemos o enlace desejado, representado pelo predicado ENLACE/ATIVADO(enl).

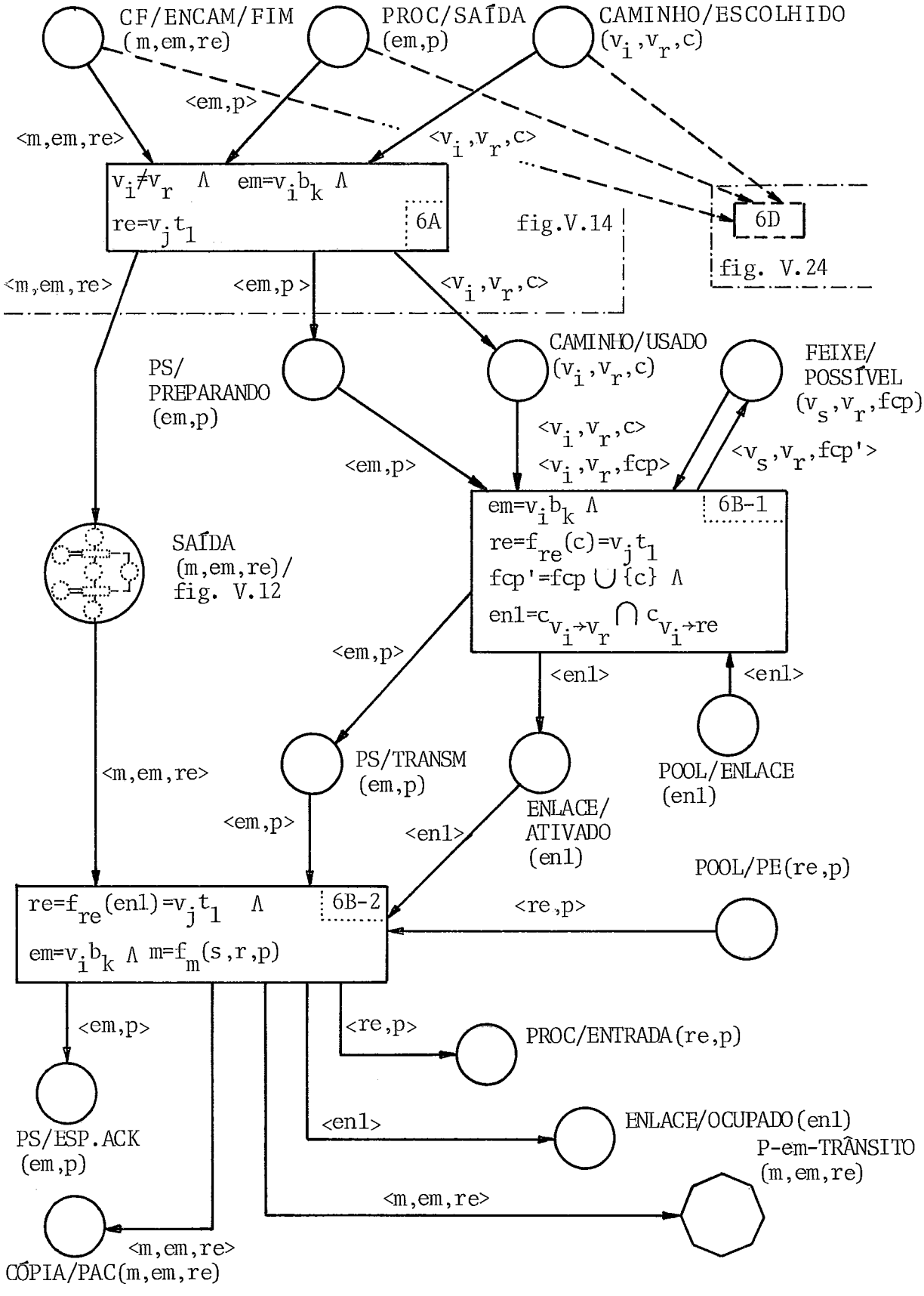


Fig. V.17: Modelagem da ativação do enlace e do receptor (processador de entrada no vértice vizinho) e colocação do pacote em trânsito.

Desta maneira, mostramos não somente que o "caminho" entre dois vértices consiste de dois recursos bastante distintos: os processadores de entrada e saída, respectivamente, pertencentes ao "mundo da computação", e o enlace com toda sua complexidade (modem, meio de comunicação, etc.), que pertence ao "mundo das comunicações", mas, também, que é possível, na modelagem, concentrar todos os possíveis problemas da comunicação (transmissão, isto é, emissão e recepção, dos sinais que representam, no mundo físico real, as mensagens abstratas) nestes predicados do tipo ENLACE/..(..).

Voltando à preocupação principal desta seção, ou seja, à colocação dopacote em trânsito, podemos dizer que as precondições já foram devidamente esclarecidas; podemos acrescentar, eventualmente, ainda a informação que o pacote deve, para ser colocado em trânsito, ter passado o processamento na fila da saída, isto é, deve estar presente no predicado associado, SAÍDA/S(m, em,re).

Agora, para modelar as precondições, podemos usar a nossa experiência (figuras IV.9 e V.11) para chegar aos seguintes predicados (figura V.17), usando m ε M, em ε EM, re ε RE, enl ε ENL e p ε P:

- P-em-TRÂNSITO(m,em,re), representando o pacote; modelamos este predicado usando um octógono para indicar que está, na realidade, associado ao "mundo das comunicações";

- ENLACE/OCUPADO(enl) que assume a transmissão física nos meios de comunicação;

- PS/ESP.ACK(em,p), que indica, por um lado, que o processador de saída transferiu a responsabilidade ao meio de transmissão, mas, por outro lado, que ele ainda espera uma confirmação em relação à transmissão bem sucedida do pacote;

- CÓPIA/PAC(m,em,re), que permite ao processador de saída conduzir verificações e, eventualmente, retransmissões;

- PROC/ENTRADA(re,p), que assumirá a responsabilidade sobre o pacote quando está saindo do meio de comunicação.

Não tendo muitas novidades em relação a estas precondições, podemos concluir que a modelagem da colocação do pacote em trânsito está, ao menos por enquanto, satisfeita.

A seção V.2.b.5, a seguir, tratará da recepção deste pacote que, agora sim, vem realmente de um vértice vizinho, ou seja, o canal t_i não é mais abstrato, como no caso de $s' = t_o$, mais real. Veremos, então, como modelar isto e, também, se há possibilidades de aproveitar conceitos usados na modelagem deste canal abstrato (p.ex., modelar similarmente ao predicado PRONTO(m,em,re)).

Mais uma coisa deve ser dita: observando as transições 6A e 6D, vemos que existe uma parte da fórmula que já se preocupou em saber se o vértice é intermediário ou final. Assim, a transição 6A indica, através de $v_i \neq v_r$, que o vértice é intermediário e a transição 6D, com a fórmula $v_i = v_r$, avisa que o pacote deveria ser passado ao canal artificial r', que representaria a saída da subrede.

V.2.b.5 - Modelagem da quarta fase: recepção do pacote no vértice vizinho.

Como já avisamos na seção V.2.b.4 e mostramos na figura V.17, precisamos várias precondições para realizar uma recepção. Compreendemos, sob a palavra recepção, primeiramente, a colocação na fila de entrada associada ao canal de transmissão em questão para poder, posteriormente, testar o pacote que chegou. Nesta segunda ação (descrita na seção V.2.b.6) será decidido se o pacote está correto e, conseqüentemente, aceito, gerando uma mensagem de controle do tipo ACK, ou, no caso contrário, gerando um NAK.

Observando, então, nesta seção V.2.b.5, esta primeira parte, podemos dizer, e mostrar na figura V.18, que já temos três precondições para disparar a transição 7 (o que será, mais ou menos, equivalente à recepção física no vértice vizinho):

- P-em-TRÂNSITO(m,em,re), que representa o pacote sendo transmitido e o que é indispensável para que possa haver uma recepção; m ε M, em ε EM, re ε RE;

- ENLACE/OCUPADO(enl), que foi o responsável pela transmissão de ponto de vista meio de comunicação; enl ε ENL;

- PROC/ENTRADA(re,p), que reflete a disponibilidade do processador de entrada (neste vértice receptor) para assumir a responsabilidade sobre o pacote; re ε RE, p ε P.

Considerando a transição 7 representativa da recepção, sabemos que ela somente pode disparar se as poscondições, associadas a ela, também estão satisfeitas. Temos, então:

- ENTRADA(m,em,re), que representa a fila de entrada associada a este canal ou enlace (figura V.19). Mais precisamente, podemos nos restringir, por enquanto, ao subpredicado ENTRADA/E (m,em,re) que pode ser considerado como sendo o modelo de um buffer de entrada; ele garante, além da recepção física de um pacote, que nenhum pacote simples se perde uma vez enviado (conseqüentemente, um enlace somente deve ser considerado "à disposição", se este buffer está realmente livre); m ε M, em ε EM, re ε RE;

- PE/RECEBER(re,p), que representa a disposição do processador de entrada para realmente receber o pacote e, depois, para assumir um estado de processamento, predicado PE/TESTAR(re,p), que, neste caso, seria o teste da consistência do pacote. Como este teste somente pode começar quando o pacote passou do buffer de entrada para a fila propriamente dita, podemos usar esta condição como sendo a responsável pela liberação efetiva do enlace; re ε RE, p ε P;

- ENLACE/DESATIVADO(enl), que representa o fato de que a linha de transmissão poderia ser liberada mas que falta ainda a liberação do 'buffer' de entrada (condição necessária para declarar o enlace "à disposição"). Assim, o predicado POOL/ /ENLACE(enl) representaria o fato que, além da linha de transmissão, também o buffer de entrada foi liberado, isto é, o e-

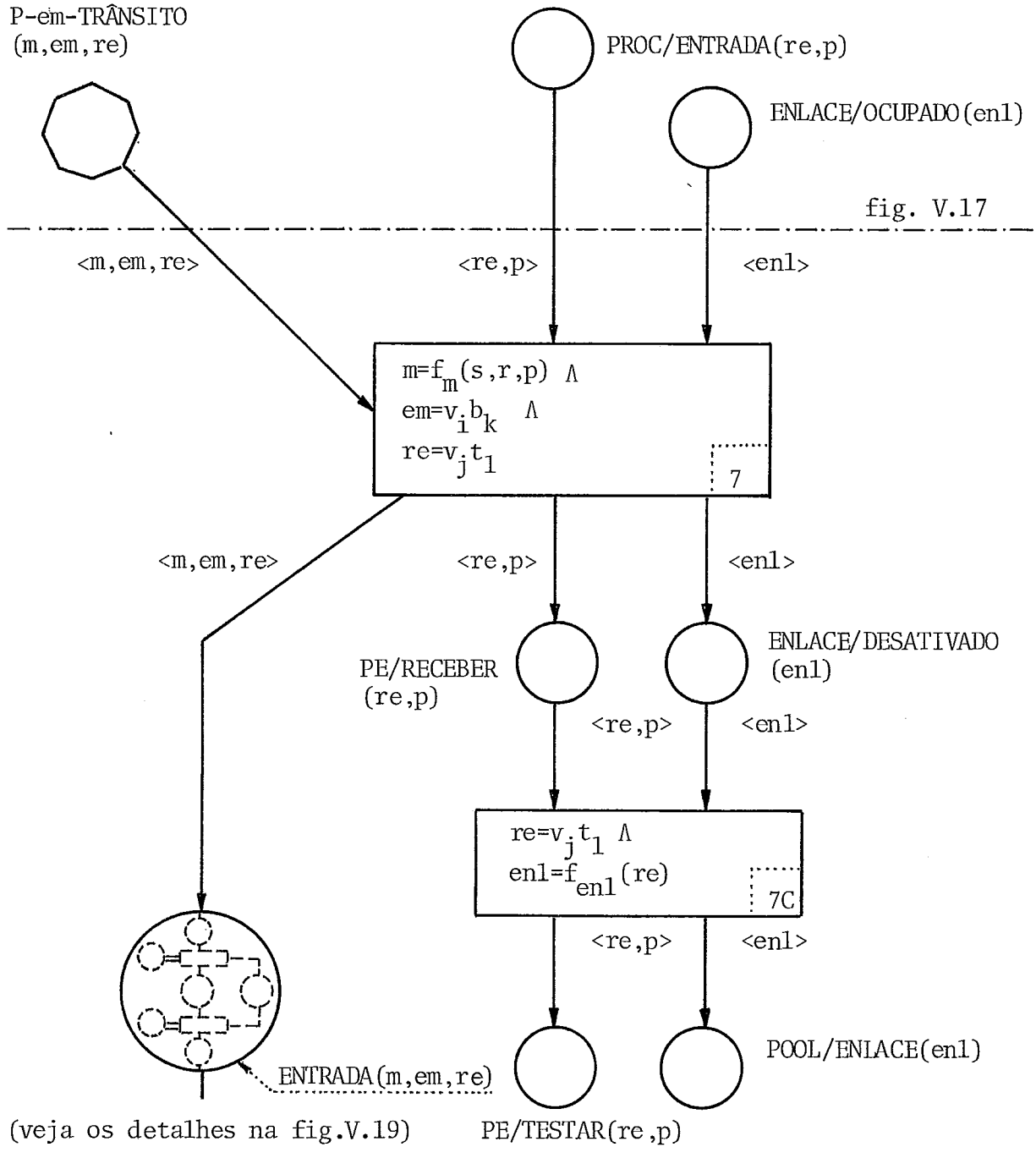
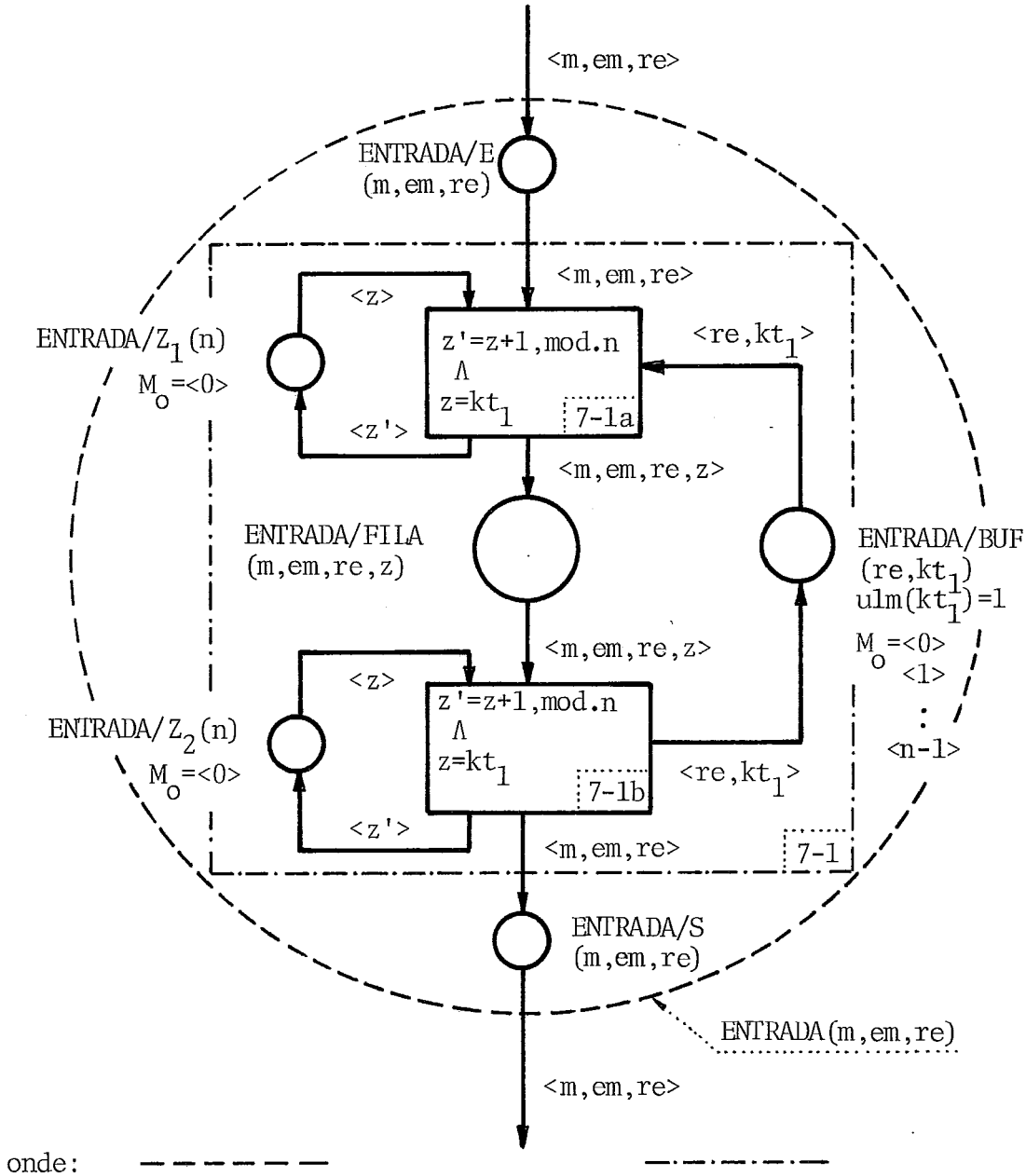


Fig. V.18: Pre e poscondições em torno da transição 7, responsáveis pela recepção física de um pacote.



Obs.: O resultado do teste da consistência pode, p.ex., ser colocado dentro do 'frame' do pacote p. Assim, o argumento $m=f_m(s,r,p)$ contém este resultado.

Fig. V.19: Detalhamento do predicado ENTRADA(m,em,re), modelando-o como sendo uma fila.

lemento em questão conseguiu ser inscrito na fila associada a esta entrada; enl ε ENL.

Vemos, então, que a recepção física não representa nenhuma dificuldade de ponto de vista da modelagem neste nível. Entretanto, achamos que chegou o momento de dar um pouco mais de atenção ao predicado ENTRADA(m,em,re).

Como visto na seção V.2.a.3, onde modelamos a saída de um vértice tipo V/F como sendo, abstratamente, a entrada num vértice S/F (a fila do canal s' foi modelada pelo predicado PRONTO(m,em,re)), devemos, agora, nos preocupar com uma comparação entre os canais s'($\hat{=}$ t_o) e t_1 o que, em termos de modelagem, seriam os predicados PRONTO e ENTRADA.

Como já propomos no início deste trabalho, queremos estruturas iguais para estes dois tipos de canais. Isto já conseguimos através do esquema de filas. Entretanto, há uma diferença baseada no fato de que o pacote que chegou ao canal s' já está "pronto" no sentido de não haver necessidade de conduzir testes em relação à integridade do pacote. Mas agora, que o pacote chegou de um vértice vizinho, ou seja, que transitou realmente por um meio físico, sendo, assim, sujeito a erros, devemos nos preocupar com testes neste sentido. Faremos isto na seção V.2.b.6.

V.2.b.6 - Modelagem da quinta fase: testar o pacote recebido e gerar uma mensagem de controle.

O que, na seção V.2.b.5, somente foi sugerido (mensagens de controle do tipo ACK ou NAK), tem que, agora, ser realmente modelado de um modo coerente.

Assim, a partir dos predicados ENTRADA(m,em,re) e PE/TESTAR (re,p), devemos verificar de que natureza é o resultado do teste. Se for um pacote confirmado com ACK ('acknowledgement'), podemos encaminhar o pacote para o próximo vértice ou para a destinação final. Mas, no caso em que for considerado falho ('negative acknowledgement'), devemos iniciar um procedimento para uma eventual retransmissão do pacote em questão.

Tendo então duas possibilidades, com conseqüências bastante distintas, precisamos continuar a modelagem em dois ramos distintos: um associado a ACK, e outro a NAK.

V.2.b.6.α - Modelagem a partir de um pacote classificado com NAK ('negative acknowledgement').

Rejeição do pacote.

Para poder disparar a transição 7A, que representará o início do ramo da modelagem relacionada a NAK, precisamos de duas precondições:

- ENTRADA(m,em,re), que representa a presença do pacote nesta fila de entrada; m ε M, em ε EM, re ε RE;

- PE/TESTAR(re,p), que representa o responsável pelo pacote p neste canal $v_j t_1$ de entrada (obs.: a interdependência entre estes dois predicados não foi explicitamente modelado, pois se situa num nível diferente daquele que interessa no momento); re ε RE, p ε P.

Agora, um dos argumentos das precondições, modelado, na fig. V.20, por ENTRADA(m,em,re), ou melhor, pelo subpredicado ENTRADA/S(m,em,re), tem que ser relacionado com a fórmula da transição incidente a este predicado. Assim, com uma fórmula do tipo $m=f_m(s,r,p)=m_{NAK}$, que indica a classificação que o pacote recebeu na fila de entrada, podemos iniciar o mencionado "ramo NAK" no modelo. Este ramo deve conter, lembrando a descrição qualitativa feita por SCHWARZ (30), alguma indicação de que o pacote foi rejeitado e, também, algo a respeito de uma mensagem de controle que transmitirá esta informação ao vértice responsável pela emissão.

Como uma das poscondições desta transição 7A, precisamos modelar a rejeição do pacote, o que será representado pelo predicado PAC/REJEITADO(m,em,re). Como observação adicional vale dizer que a rejeição de um pacote, por causa de uma falha na con-

sistência do mesmo, resulta em "volatilizá-lo".

A outra poscondição básica, desta transição 7A, será representativa da mensagem de controle mencionada. Sabemos que o pacote foi testado e recebeu a classificação NAK. Então, a partir desta classificação podemos, com a ajuda de uma função do tipo $pcn=f_{pcn}(m_{NAK})$, criar esta mensagem de controle e colocá-la em trânsito. Como já mencionado em relação à mensagem de controle mc no capítulo IV, não queremos, por enquanto, nos preocupar se esta mensagem de controle será submetida ao mesmo esquema de controle de fluxo que as mensagens principais. Portanto, vamos considerar que ela vai diretamente para a transmissão. Modelamos este fato com o predicado PCN/TRÂNSITO(pcn,emc,rec), na forma octagonal, onde este "pacote de controle do tipo NAK" aparece como um dos argumentos.

Falta ainda decidir o que acontecerá com o processador de entrada responsável pelo pacote p. Do mesmo modo que o emissor (processador de saída no vértice precedente) foi liberado no momento em que o pacote foi colocado em trânsito, podemos proceder agora. O responsável pela entrada, modelado pelo predicado PE/TESTAR(re,p), terminou não somente sua função em relação à passagem do pacote pela transição 7A (p.ex., testar o pacote e gerar um pacote de controle) mas, também, não tem mais nada a fazer, já que o pacote foi rejeitado. Então, no mesmo momento em que o pacote de controle, representado pela tupla <pcm,emc,rec>, foi colocado em trânsito, ele pode ser liberado. Podemos modelar este fato pelo predicado POOL/PE(re,p).

Observe, ainda, que ele, assim, pode ser considerado (lembre que não estamos modelando, por enquanto, o controle de fluxo de mensagens de controle!) como um "emissor" deste pacote de controle. Então, para seguir o mesmo esquema como foi usado em relação a mensagens comuns, isto induz à indicação pela letra c adicional nos lugares dos argumentos sobre emissor e receptor.

Retransmissão.

O pacote de controle, <pcn,emc,rec>, que contém todas as informações necessárias com respeito ao pacote rejeitado, será, então, responsável pelo início de uma retransmissão do pacote em questão. Em termos de modelagem, temos, assim, as seguintes precondições, usando os argumentos pcn ε PCN, emc ε EMC, re ε REC, em ε EM, p ε P, m ε M, re ε RE, para uma retransmissão:

- predicado PCN/TRÂNSITO(pcn,emc,rec), representativo do pacote de controle mencionado;

- PS/ESP.ACK(em,p), que representa o responsável pela emissão original do pacote e que está esperando uma confirmação associada a este pacote original;

- CÓPIA/PAC(m,em,re), que representa a cópia do pacote original enviado e que se tornará a "fonte" para uma retransmissão deste pacote.

O que se precisa decidir é a partir de que ponto no modelo começará esta retransmissão. Vejamos:

- Primeiramente, sabe-se que o enlace enl entre o emissor $em=v_i b_k$ e o receptor $re=v_i t_1$ foi liberado no momento em que o pacote passou do buffer do fim do canal, predicado ENTRADA/E(m,em,re), para a própria fila de entrada, modelado pelo predicado ENTRADA/FILA(m,em,re,z), mostrado nas figuras V.18 e V.19;

- Segundo, o responsável pela recepção, modelado pelo predicado PE/TESTAR(re,p), também retornou ao 'pool';

- Assim, o único associado a este enlace, que ainda está disponível, seria o emissor, que está esperando a confirmação, modelado pelo predicado PS/ESP.ACK(em,p).

Conseqüentemente, deveríamos novamente escolher o enlace. Agora, "escolher" não é exatamente o termo certo, já que a informação, sobre o canal usado, está contido no predicado CÓPIA/PAC

(m,em,re) e, ainda, que o responsável pela emissão não está no estado 'pool', isto é, desativado, mas, sim, num estado PS/ /ESP.ACK(em,p) que pode ser considerado "semiativo". Então, seria melhor dizer que tentaremos "reativar" o mesmo enlace como na primeira transmissão. Esta idéia está de acordo com modelos clássicos da teoria de filas, usados na literatura, como, p.ex., em MARUYAMA (17), que usou este modelo.

Então, tendo como precondição principal da transição 8A o predicado PCN/TRÂNSITO(pcn,emc,rec), podemos, usando as informações mencionadas, tentar modelar a retransmissão similar à transmissão original. Assim, em primeiro lugar, devemos escolher, na transição 8A, o "melhor" caminho dentro do feixe fc, para, depois, nas transições 9A e 9B, fazer a verificação se ele, o caminho, ainda é o melhor. A seleção deve ser feita a partir do predicado FEIXE/ATUAL(v_i,v_r,fc), já que é nele que se materializam as informações atuais (veja figura V.13), e, ainda, podemos usar uma fórmula do tipo similar aquela da transição 6, isto é, c_{verif}=sel(fc,pcn,emc,rec,v_i=f_{vi}(pcn,em)). Tendo este c_{verif}, podemos, nas transições subseqüentes, ou seja, em 9A e 9B, através das fórmulas $c_{verif}=c_{velho}$ e $c_{verif}\neq c_{velho}$, testar o caminho e tomar as providências necessárias.

Esta ação de escolha poderia ser considerada como problema interno do vértice, induzindo, assim, uma modelagem similar àquela da figura V.6. Já que a verificação está sob a responsabilidade do processador de saída, que passou, via transição 8A, do estado PS/ESP.ACK(em,p) para o estado PS/REATIVANDO(em,p).

Dependendo do resultado, isto é, se o caminho ainda é o mesmo ou não, temos duas opções:

1) A primeira, digamos, quando o canal novo corresponde ao anteriormente escolhido, permite o uso das informações existentes em CÓPIA/PAC(m,em,re) e PS/REATIVANDO(em,p), já que elas foram confirmadas. Assim, temos a possibilidade de chegar quase diretamente às precondições para disparar a transição 6B-2 (figura V.17) e conseguir uma retransmissão a par-

tir da mesma fila de saída, o que, em termos computacionais, significa um acontecimento "local" sem envolvimento direto da UCP. Observe, na figura V.20, que, através da transição 9A, conseguimos o aproveitamento direto das condições contidas em CÓPIA/PAC(m,em,re) e PS/REATIVANDO(em,p), para chegar às poscondições da transição 9A, predicados SAÍDA(m,em,re) e PS/TRANSM(em,p), que são, como já foi mencionado, as precondições para disparar a transição 6B-2.

As outras condições, relacionadas ao caminho e ao enlace, seguem o mesmo esquema como na figura V.17, já que, por definição, nenhuma linha de transmissão deverá ficar ocupada ou reservada além do necessário, e, assim, deverão ser novamente ativadas.

2) A segunda opção, quando houve alguma mudança no sistema e, assim, o novo caminho é diferente do anterior, exige um tratamento igual aos pacotes que chegaram normalmente ao vértice. Mas, para conseguir isto, precisamos de um artifício. Tendo como exemplo o predicado PRONTO(m,em,re), que também representou algo artificial (lembre que a canal s' é artificial no sentido de simular a entrada na subrede), podemos pensar em criar mais um canal artificial, digamos, t_{-1}. Este representaria a "reentrada" dos pacotes que foram confirmados com NAK e que não podem ser retransmitidos "localmente", isto é, no mesmo canal de saída que na transmissão original. Assim, eles deveriam ser submetidos à LÓGICA/MUX$(\ )_V$ para que pudessemos iniciar a retransmissão a partir dos predicados CF/ENCAM/INI(m,em,re) e PE/APRONTAR(re,p).

Temos, então, como precondições principais da transição 9B (veja na figura V.21) aquelas similar à transição 9A (figura V.20), ou seja:

- CÓPIA/PAC(m,em,re), que representa a nova "fonte" para a retransmissão; m ε M, em ε EM, re ε RE;
- PS/REATIVANDO(em,p), que, ainda, é responsável pelo pacote; em ε EM, p ε P;

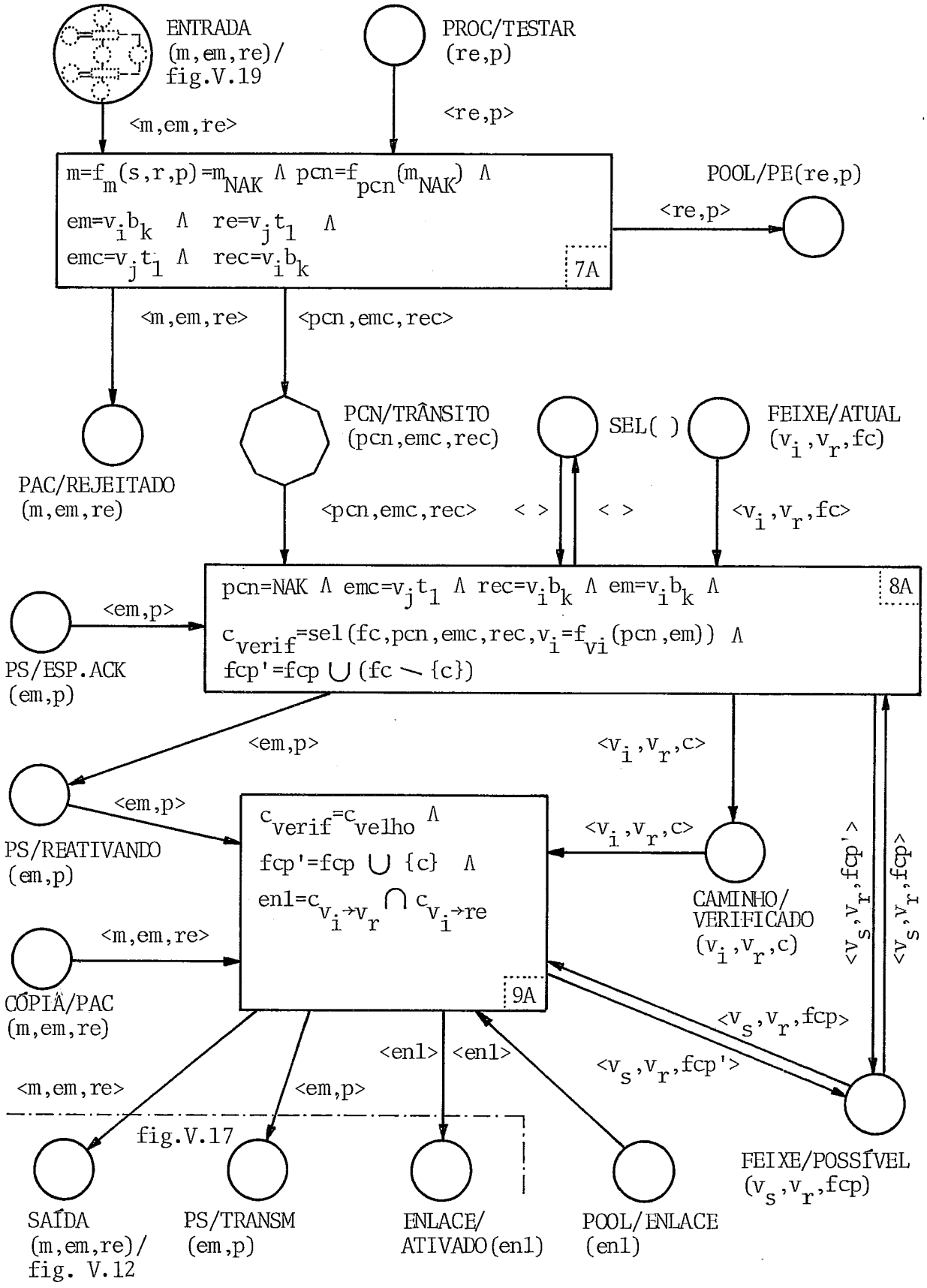


Fig. V.20: Modelo da ação de iniciar uma retransmissão quando se pode utilizar o mesmo caminho.

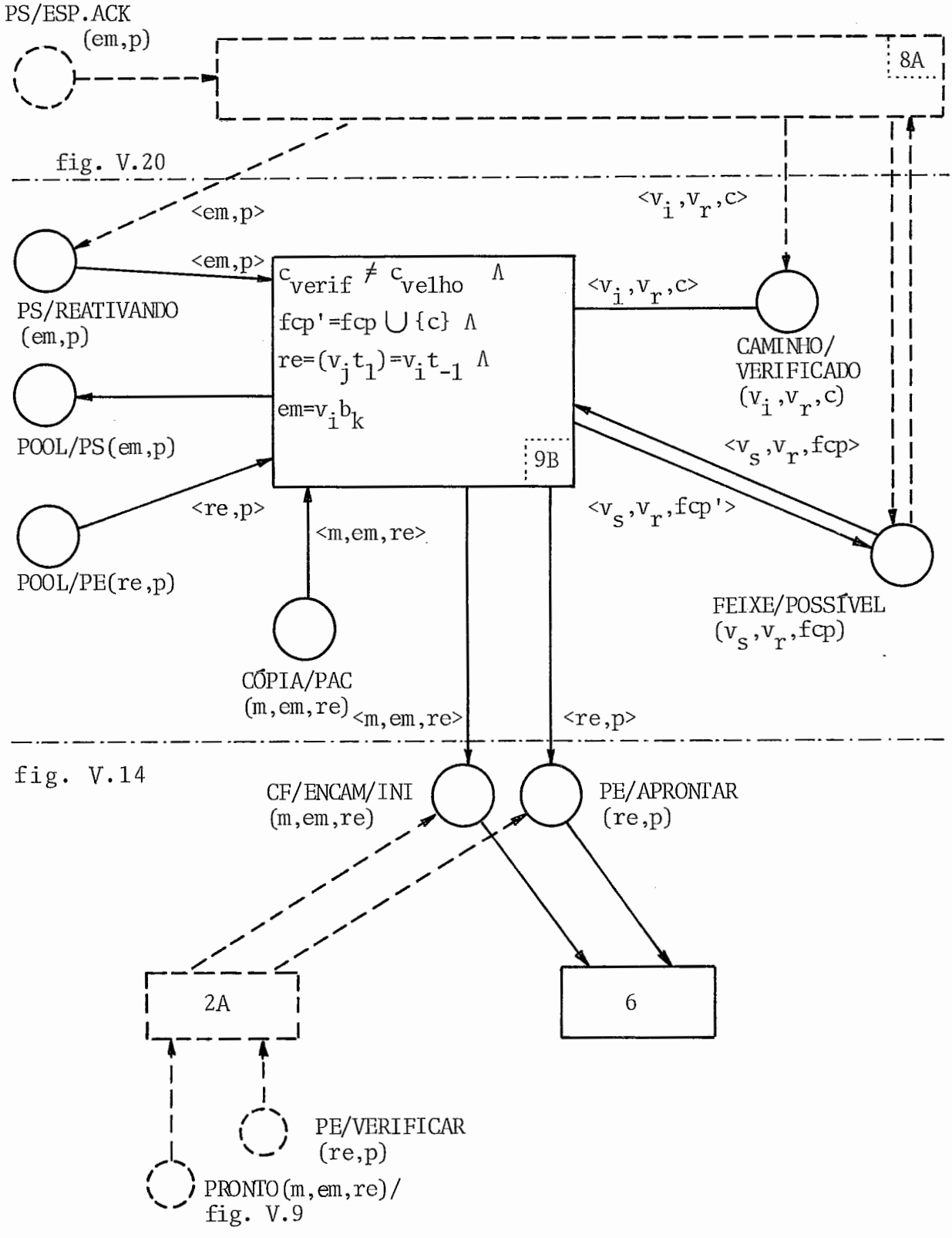


Fig. V.21: Variante da figura V.20, representando o caso quando não se pode usar o mesmo caminho para retransmissão (indicado também o "encontro" com as tuplas vindo da transição 2A).

- CAMINHO/ESCOLHIDO(v_i,v_r,c), que representaria a base para uma comparação; v_i,v_r ε V, c ε FC.

A fórmula da transição 9B indica que $v_{verif} \neq c_{velho}$; assim, podemos devolver o caminho e ignorar o enlace associado. Optamos por este tipo de moldagem porque, em primeiro lugar,não se sabe quanto tempo passará até que a retransmissão seja realmente iniciada e se até este momento o caminho verificado ainda é o "melhor". Segundo, achamos mais conveniente unificar ao máximo possível os esquemas utilizados.

Associado à mesma fórmula, vemos que o responsável, que estava esperando um ACK e que tentou reativar o enlace inicial, deve voltar ao POOL/PS(em,p) porque não será mais o responsável pela transmissão.

Já a outra parte da fórmula deve indicar o receptor artificial e podemos realizar isto através de $re=(v_j t_1)=v_i t_{-1}$.

Deste modo, chegamos às poscondições da transmissão 9B, isto é, aos predicados mencionados, e vemos que eles realmente são, ao mesmo tempo, as precondições da transmissão 6, garantindo assim o uso do mesmo esquema usado na figura V.14.

Como observação final desta seção queremos avisar que não nos preocupamos, neste trabalho, com o problema de como modelar o ato de evitar retransmissões consecutivas de um mesmo pacote. Entretanto, podemos pensar em "marcar" o pacote retransmitido e assim obter condições de controle adequadas para esta finalidade.

V.2.b.6.β - Modelagem a partir de um pacote classificado com ACK ('acknowledgement').

As primeiras observações são similares àquelas da seção V.2.b.6.α. Assim temos agora (veja figura V.22), em vez do predicado PAC/REJEITADO(m,em,re), mostrado nas figuras V.20 e V.21, um outro que representa a aceitação, ou seja, um predicado do

tipo PAC/ACEITO(m,em,re) que é pseudônimo do predicado CF/ENCAM/INI(m,em,re) na figura V.14.

Também, temos a passagem do responsável pela recepção do estado de testar para um outro que podemos chamar de aprontar e que, assim, coincide com PE/APRONTAR(re,p) da figura V.14. Ele ainda é responsável pelo pacote até que seja realmente aceito pela UCP para outros tratamentos (obs.: "em paralelo", ele poderia também ter a responsabilidade do envio do pacote de controle mas, por enquanto, não modelamos este fato). Entretanto, não se deve esquecer que ele, na passagem pela transição 7B, gerou um pacote de controle. Esta mensagem servirá para destruir a cópia guardada no vértice/emissor, ao contrário do caso NAK, onde o pacote de controle enviado para o emissor serviu para iniciar uma retransmissão. Novamente, colocamos o pacote <pca,emc,rec> diretamente em trânsito sem preocupações, por enquanto, como será o controle de fluxo para estas mensagens de controle.

A mencionada destruição da cópia pode ser modelada da seguinte maneira: temos como precondições da transição 8B o pacote de controle, <pca,emc,rec>, contido no predicado associado PCA/ /TRÂNSITO(pca,emc,rec), a cópia do pacote original, modelada por CÓPIA/PAC(m,em,re), e o processador de saída que está esperando o ACK, representado pelo predicado PS/ESP.ACK(em,p). Assim, uma vez satisfeitas estas precondições, podemos disparar a transição 8B considerando, obviamente, também as poscondições. Estas são o 'pool' para o responsável da emissão, predicado POOL/PS(em,p), e um outro predicado, do tipo CÓPIA/DESTRUÍDA (m,em,re), que simplesmente representa a eliminação da cópia, o que, se considerássemos mais detalhadamente a arquitetura de um computador, resultaria na liberação de algum espaço na memória.

Já indicamos vagamente, na figura V.22, como continuar a partir de um pacote aceito. Na seção V.2.b.7 podemos nos preocupar com estes acontecimentos, ou seja, com a modelagem em relação ao encaminhamento para um próximo vértice ou, no caso em que $v_j = v_r$ = =vértice final (no caminho entre v_s e v_r), em relação à saída da

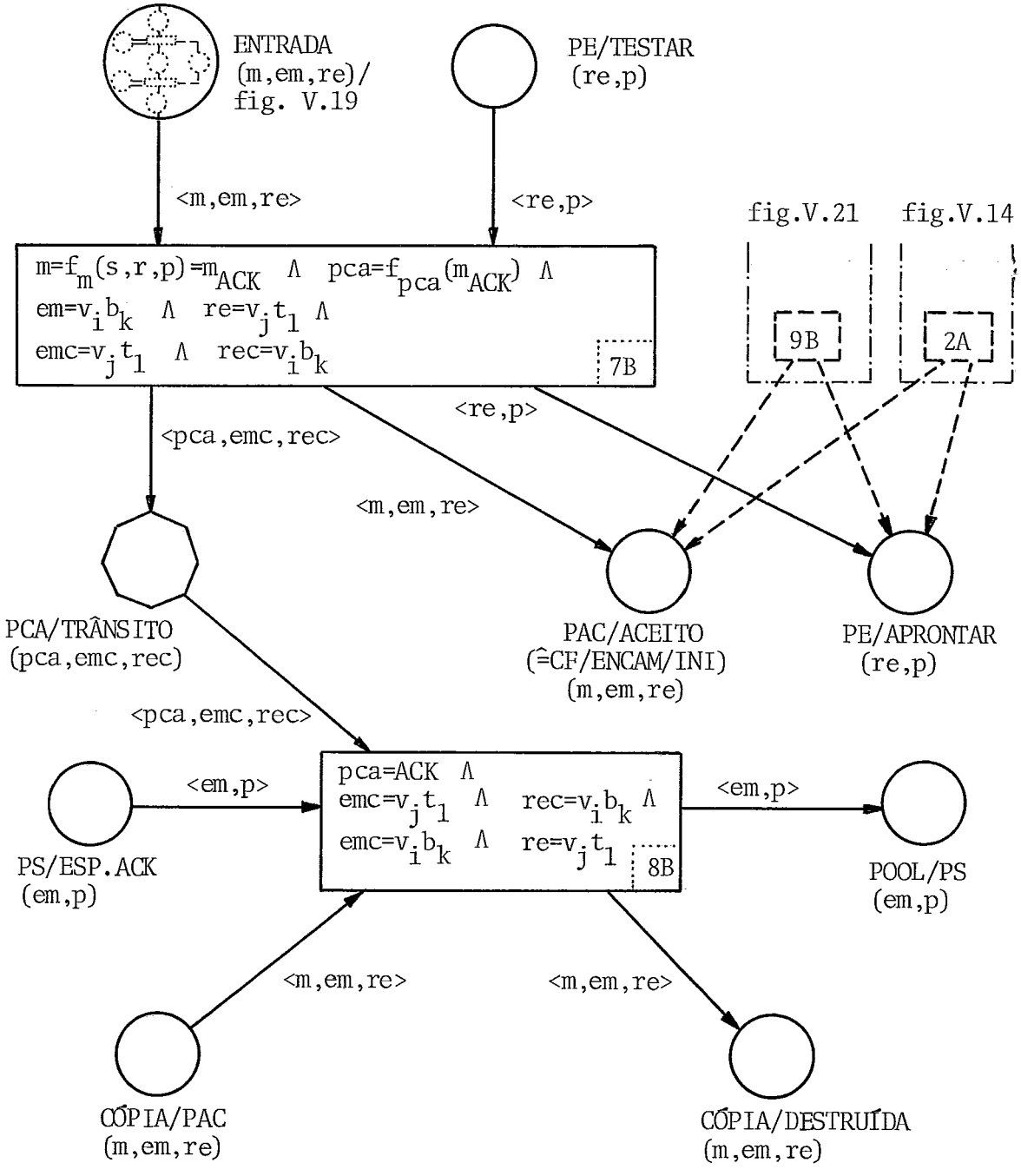


Fig. V.22: Modelagem parcial em relação ao ramo resultante de um ACK ('acknowledgement').

subrede de comunicação.

V.2.b.7 - Modelagem da sexta fase: seqüência para um próximo vértice S/F, utilizando o mesmo esquema gráfico.

Para poder investigar este ponto da seqüência, gostaríamos de, antes, mostrar, na figura V.23, o "encontro de pacotes" que vêm de diferentes canais. Assim, temos de um lado os pacotes que vêm da fonte, ou seja, do "exterior" da subrede, outros, que vêm dos vértices vizinhos, ou seja, de "dentro" da subrede, e, ainda, aqueles que vêm do canal artificial para serem retransmitidos.

Esta investigação é importante pois queremos, deste modo, justificar a nossa proposta inicial que dizia que todos os pacotes, uma vez na forma <s,r,p>, deveriam ser submetidos ao mesmo processo como, p.ex., o seu encaminhamento. Então, observando as diferentes fases da modelagem, vemos que a seqüência, do ponto de vista da representação gráfica, já existe. O que se deve fazer é investigar se as inscrições estão coerentes e, em particular, se a escolha do caminho será realmente feita a partir do vértice correto. Lembre que, na transição 6, a função da escolha é dada como

$$c = sel(fc,m,emv,rec,v_i=f_{vi}(rev)) \ ,$$

o que indica que o caminho será escolhido dentro do feixe de caminhos atualizado (fc), a partir de vértice (v_i), usando informações adicionais (m,emv,rev) para evitar que se retorne no caminho.

Pacotes vindo da fonte.

As identificações são em=($v_i b_k$)=$v_s s$, para o emissor (canal s), e re=($v_j t_1$)=$v_s t_o$, para o receptor (canal artificial s'), ambas colocadas dentro da transição 2A, incidente aos predicados CF/ENCAM/INI(m,em,re) e PE/APRONTAR(re,p), que darão início ao roteamento.

Portanto, como, dentro da função c=sel(.), o vértice inicial do feixe de caminhos é calculado na base do "velho" receptor, isto é, $v_i=f_{vi}(rev)$ temos como próximo emissor algum processador de saída do vértice v_s. Assim, a seleção apontará realmente um caminho que parte de v_s em direção v_r, via v_j.

Pacotes vindo de vértices S/F vizinhos.

Como todo o procedimento da modelagem é baseado na seqüência de vértices S/F, não há dificuldades nesta verificação. Então, temos $em=v_ib_k$ e $re=v_jt_1$ como sendo a base para o cálculo, de onde se tira $v_i=f_{vi}(rev)$. Isto indica que o caminho restante começará lá onde terminou o caminho já percorrido, ou seja, o "antigo" v_j vira o "novo" v_i.

Pacotes precisando ser retransmitidos.

Este ponto é, de fato, o único que poderia realmente causar problemas, já que, em vez de continuar a seqüência progressiva, esta foi quebrada no sentido de repetir um trecho do caminho. Então, como, na fórmula c=sel(.) da transição 6, o $v_i=f_{vi}(rev)$, precisamos manipular a informação contida em CÓPIA/PAC(m,em,re). Faremos isto através de um procedimento similar àquele usado na passagem de canal s para s', ou seja, consideramos como emissor da cópia o processador do canal b_k do vértice v_i e como receptor o processador de um canal artificial t_{-1}. Assim, como $re=(v_jt_1)=v_it_{-1}$, o resultado de $v_i=f_{vi}(rev)$ indica que o caminho deve começar no mesmo vértice que aquele escolhido anteriormente.

Entre parêntesis, podemos colocar, como complemento, uma observação em relação à retransmissão no mesmo canal. Usaremos, na transição 8A, uma fórmula similar àquela usada na transição 6, ou seja, $c_{verif}=sel(fc,pcn,emc,rec,vi=f_{vi}(pcn,em))$. Ela indica, baseado no pacote de controle, que o novo caminho deverá começar no mesmo vértice que anteriormente. Agora, no caso em que o novo caminho coincide com o velho (transição 9B), se coloca o pacote (ou melhor, a cópia do pacote) diretamente na fi-

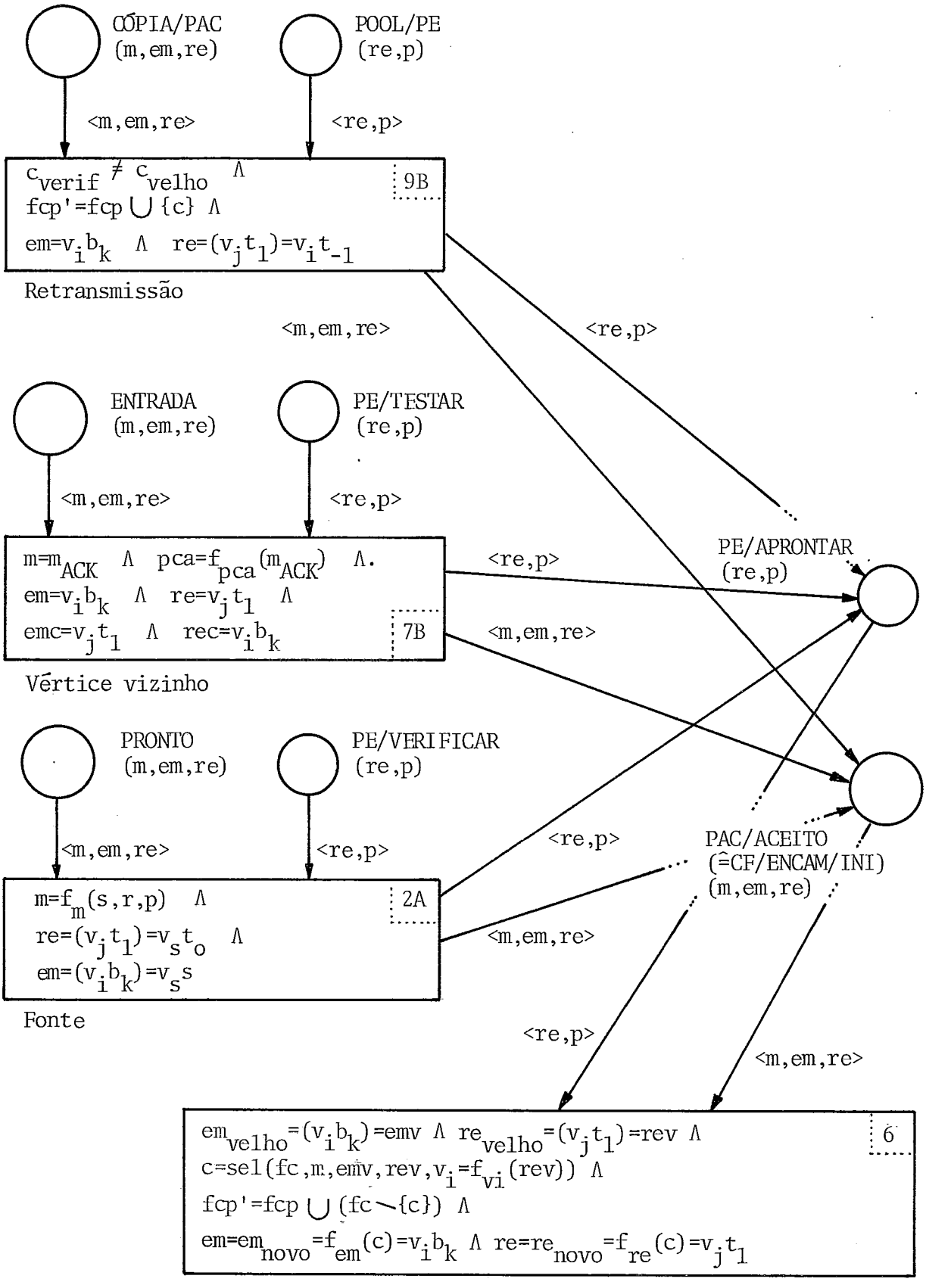


Fig. V.23: Esboço sobre o "encontro", na representação gráfica, dos pacotes vindo da fonte, de vértices vizinhos e do canal artificial para retransmissões.

la de saída associada a este caminho, evitando, assim, qualquer problema de identificação como descrito anteriormente.

V.2.b.8 - Modelagem da sétima fase: a saída da subrede de comunicação.

Lembrando a seção V.2.b.2, onde modelamos a entrada na subrede, achamos conveniente representar sua saída baseada em idéias similares. Assim, como já indicado na figura V.11, usaremos um canal artificial, digamos b_o ou r', que servirá, junto com a fila associada, como lugar da remontagem da mensagem. Lembre, que remontagem, no sentido amplo, significa recompor a mensagem que, na entrada da subrede, foi dividida em pacotes que representam, do ponto de vista da subrede, "mensagens" individuais. Aqui, neste capítulo, que trata de mensagens de um pacote só, esta remontagem se restringe, por enquanto, a uma simples "desempacotagem", ou seja, tirar da tupla <s,r,p> todo o adicional para obter, novamente, a tupla original <s,r>.

Posteriormente, na seção V.2.c, podemos modelar a "entrada" no vértice/destinação que se fará, a partir desta fila de remontagem, via UCP, por uma fila que poderíamos chamar de destinação. Esta fila, ou melhor, o canal r associado a esta fila. levará, depois, para o usuário r, destinatário previsto da mensagem <s,r>.

Então, a primeira providência a tomar é a de se decidir sobre o fim do caminho. Poderíamos ter verificado isto já na fila de entrada neste vértice, p.ex., baseado no predicado ENTRADA(m,em,re), entretanto, achamos mais conveniente colocar esta decisão para depois da transição 6, baseando-a, principalmente, no predicado CAMINHO/ESCOLHIDO(v_i,v_r,c).

Assim, observando a figura V.24, temos como precondições:

- CF/ENCAM/FIM(m,em,re), que representa o fim do encaminhamento realizado pela UCP; m ε M, em ε EM, re ε RE;

- PROC/SAÍDA(em,p), que representa o responsável para a próxima emissão; em ε EM, p ε P;

- CAMINHO/ESCOLHIDO(v_i,v_r,c), que representa o caminho a tomar para o vértice final do caminho, v_r, o que, em nosso caso do "final" do caminho, se restringe à fila de saída associada ao canal artificial r'; v_i,v_r ε V, c ε FC.

A transição 6D tem, como fórmula principal, a indicação que $v_i=v_j=v_r$ (ao contrário da transição 6A, que tinha $v_i \neq v_r$, como mostrado nas figuras V.14 e V.17) e, assim, é possível passar os elementos para os predicados que contêm as seguintes poscondições para disparar esta transição 6D:

- MONTAGEM(m,em,re), fig. V.25, que representa, em princípio, uma fila de saída, tal como o predicado SAÍDA(m,em,re), mas que, neste caso do canal artificial b_o=r', indicará também a necessidade de manipular os pacotes no sentido de remontá-los para obter a mensagem original; m ε M, em ε EM, re ε RE;

- PS/MONTAR(em,p), que indica o responsável pela remontagem e o encaminhamento ("emissão"), via UCP, ao canal de saída da subrede de comunicação; em ε EM, p ε P.

Então, vimos que o primeiro passo para sair da subrede pode ser considerado quase trivial. As dificuldades estão agora concentradas nesta fila de saída, MONTAGEM(m,em,re), e na passagem, posteriormente, pela UCP, que concluirá a transmissão pela parte da subrede. Mostraremos isto na seção V.2.c que representará o complemento da seção V.2.a onde descrevemos as filas em geral e as filas em torno de um vértice/fonte.

V.2.c - As filas de espera em relação à saída da subrede de comunicação.

Da mesma maneira como classificamos as filas em torno de um vértice/fonte, ou seja, uma fila de entrada (canal s e predica-

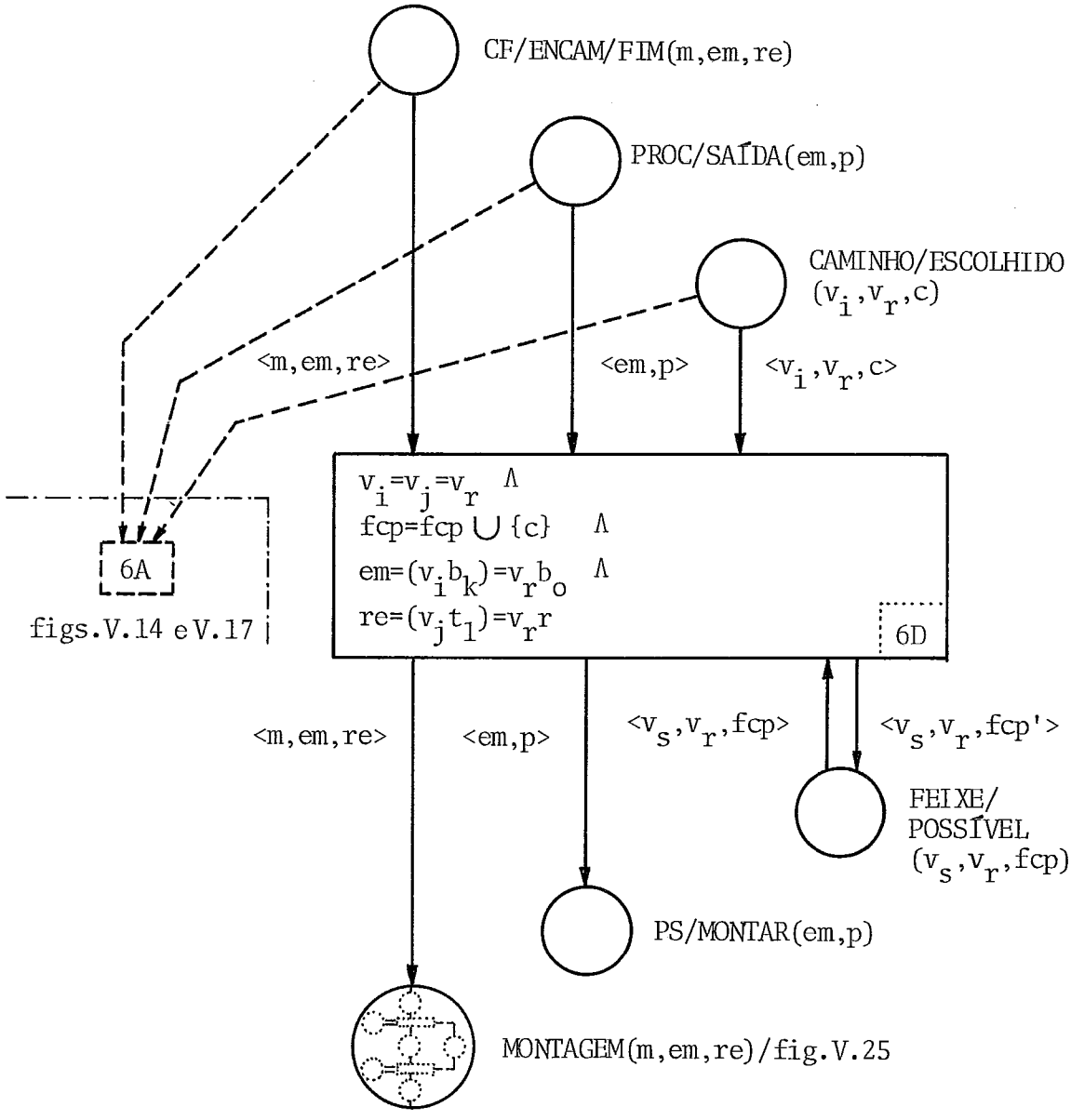
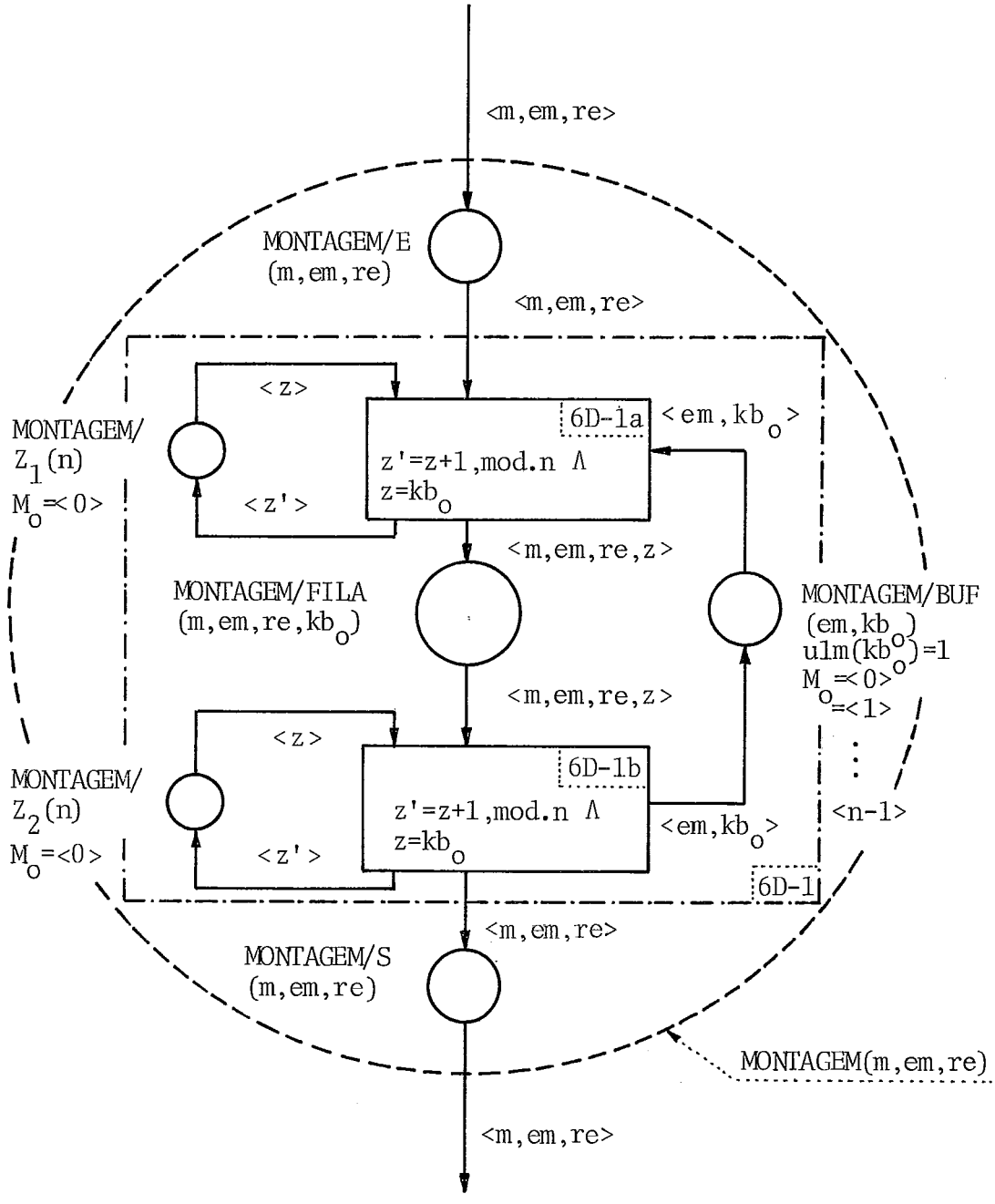


Fig. V.24: Modelo parcial em torno da transição 6D que indica o ramo no modelo que iniciará a remontagem da mensagem na saída da subrede.



onde: ——— modelo simplificado do predicado MONTAGEM

—·—·—·— modelo simplificado da transição 6D-1

Fig. V.25: Detalhamento do predicado MONTAGEM(m,em,re), modelando-o como sendo uma fila.

do FONTE(s,r)) e uma fila de saída (canal artificial s' e predicado PRONTO(m,em,re), que coincidiu com a entrada na subrede), podemos verificar e modelar as filas em torno de um vértice/destinação.

Assim, temos uma fila de entrada no V/D, que coincide com a fila de saída da subrede, canal b_o=r' e predicado associado MONTAGEM(m,em,re), e uma outra de saída de V/D, que poderíamos representar pelo canal r e o predicado DESTINAÇÃO(s,r). Falta, então, somente a passagem, via UCP, de uma fila para outra. Isto se faz similarmente à passagem de FONTE(s,r) para PRONTO (m,em,re), ou seja, precisamos um tratamento na UCP, que podemos chamar, digamos, de CF/CONCLUIR(s,r,p,z).

Já conhecendo muitos dos detalhes em torno de filas da UCP, precisamos somente observar atentamente a figura V.26 para saber o que acontecerá na passagem via UCP. Claro, a fórmula de transição 3-2, $(s,r) = f_p^{-1}(p)$, é muito simplificada, mas indica o essencial desta passagem, ou seja, a recuperação da mensagem original. Parte desta recuperação já foi preparada em MONTAGEM(m,em,re) sob responsabilidade de predicado PS/MONTAR (em,p), mas cabe à UCP de dar a palavra final, isto é, confirmar que a mensagem está na forma original <s,r>.

A incorporação destas idéias, mostradas na figura V.26, no esquema global pode ser vista na figura V.28, onde pode ser observada a separação das funções V/D e S/F (compare com a figura V.10), ou seja, a distinção entre a subrede de comunicação e o nível do usuário. Do ponto de vista do nível do usuário vemos que a figura V.28 já contém a parte da figura IV.9 envolvida com a recepção (cap. IV, fig. IV.7).

Em relação ao modelo número 1 (figura IV.9) cabe dizer que algumas das indicações estão um pouco modificadas como, p.ex., a parte relacionada ao "buffer". Como, neste trabalho atual, já estamos modelando o predicado DESTINAÇÃO(s,r) como sendo uma fila, não precisamos mais especificar esta parte. Isto pode ser visto na figura V.27 e, em especial, o predicado

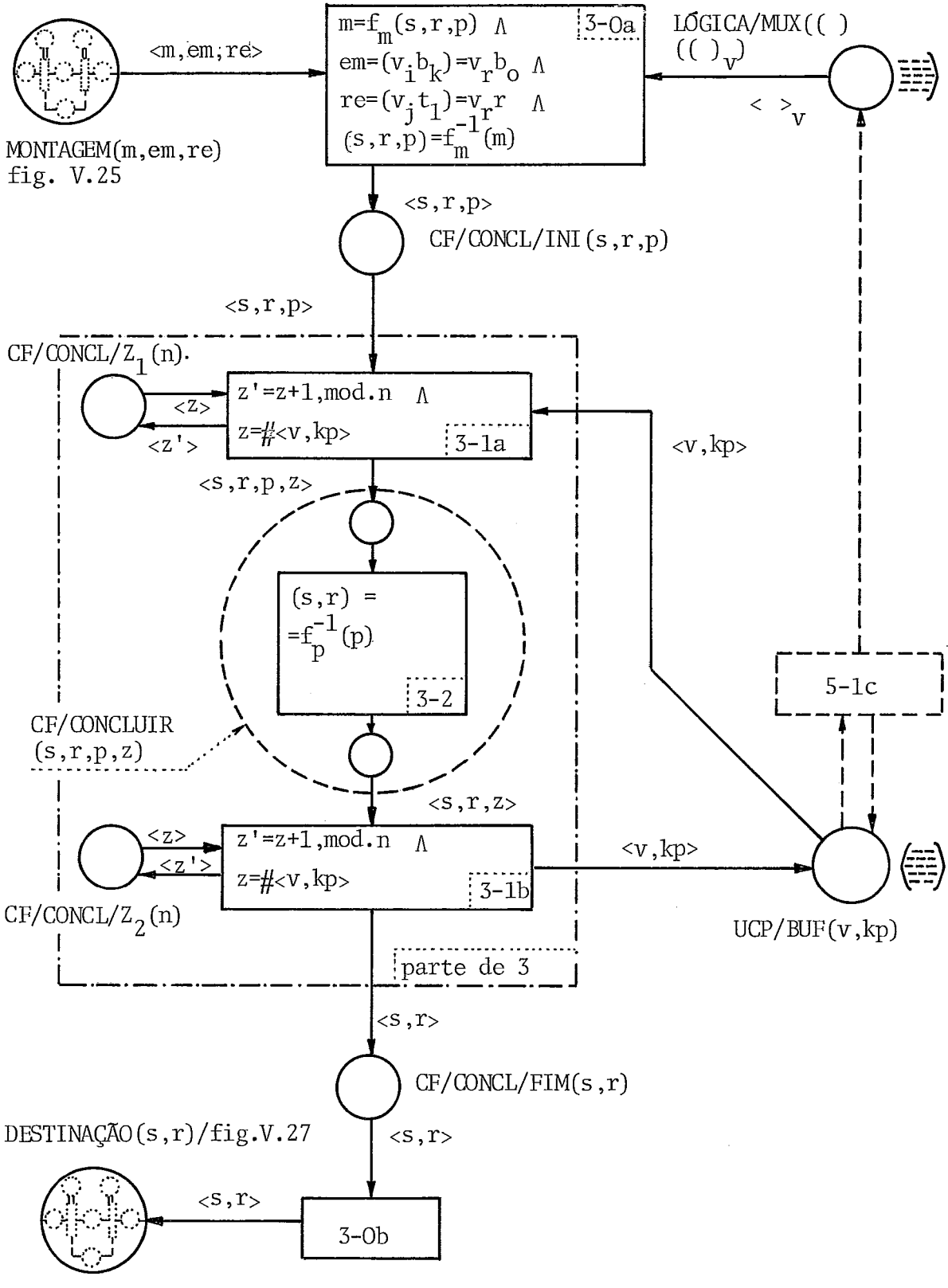
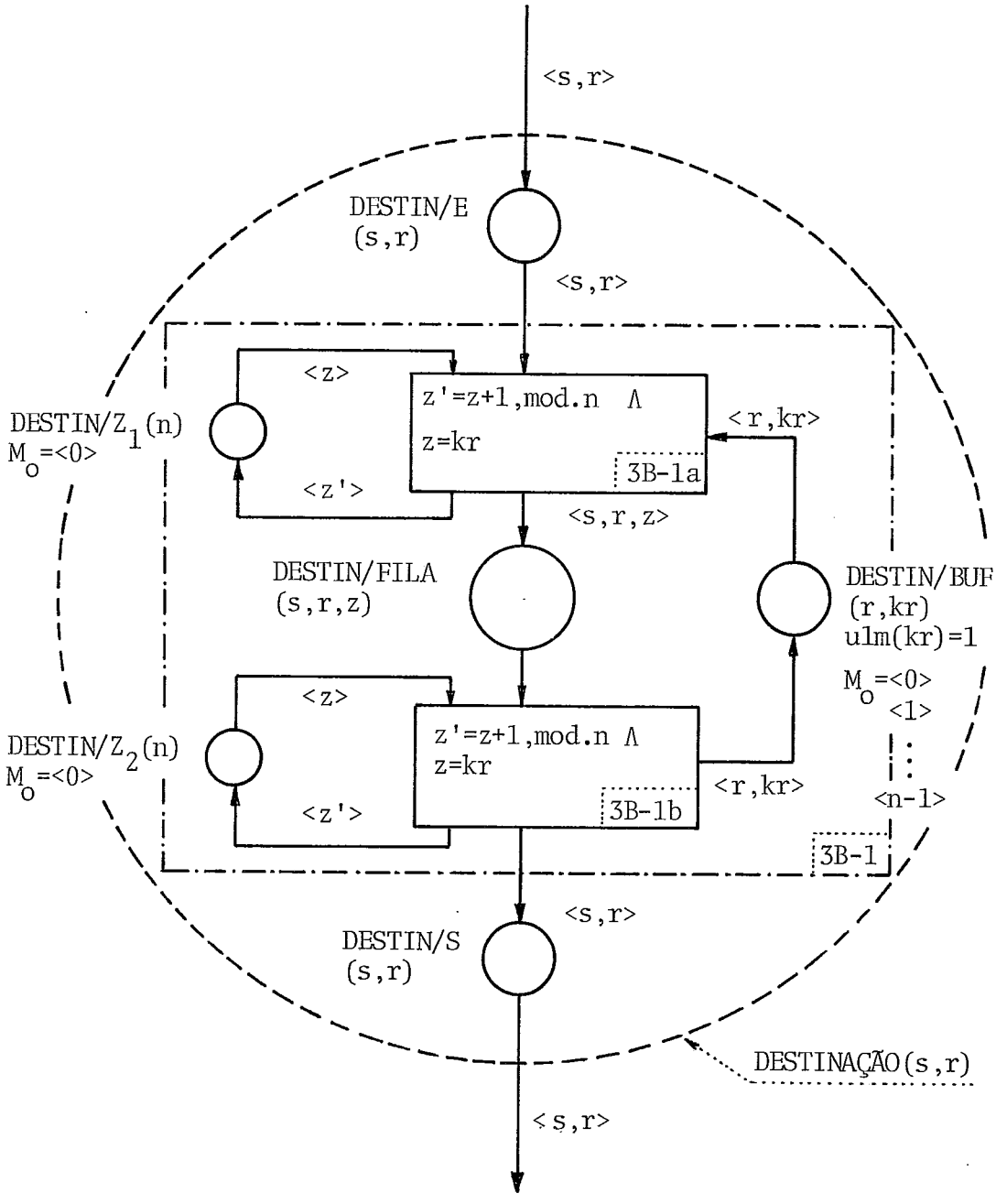


Fig. V.26: Passagem do pacote de predicado MONTAGEM(m,em,re) para a fila do receptor, DESTINAÇÃO(s,r), mostrando a parte da transição 3 envolvida.



onde:

———	—·—·—·—
modelo simplificado do predicado DESTINAÇÃO	modelo simplificado da transição 3B-1

Fig. V.27: Modelo parcial detalhado do predicado DESTINAÇÃO (s,r), considerando-o como sendo uma fila.

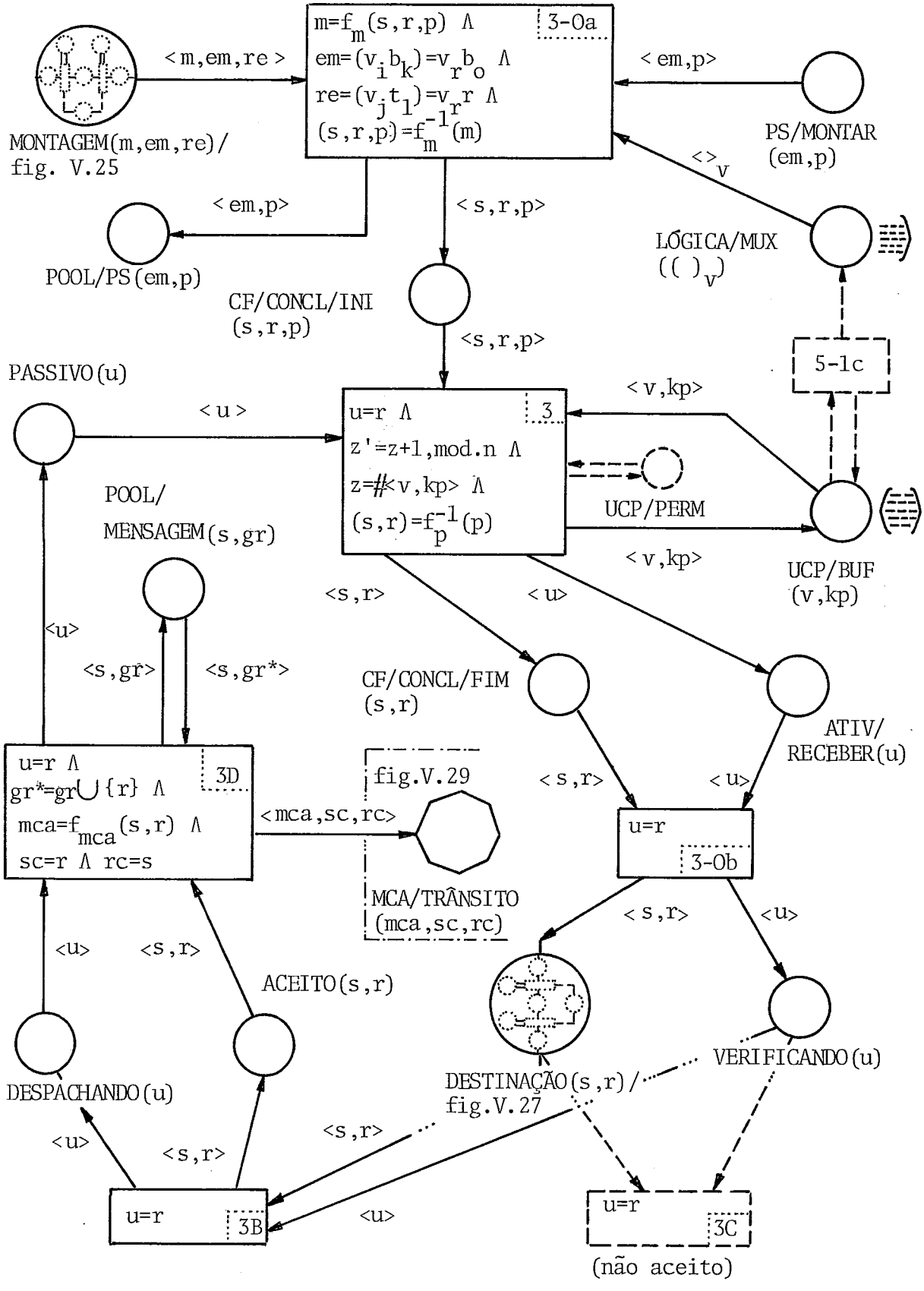


Fig. V.28: Modelo parcial mostrando a incorporação dos predicados MONTAGEM(m,em,re) e DESTINAÇÃO(s,r) no esquema da recepção no vértice/destinação.

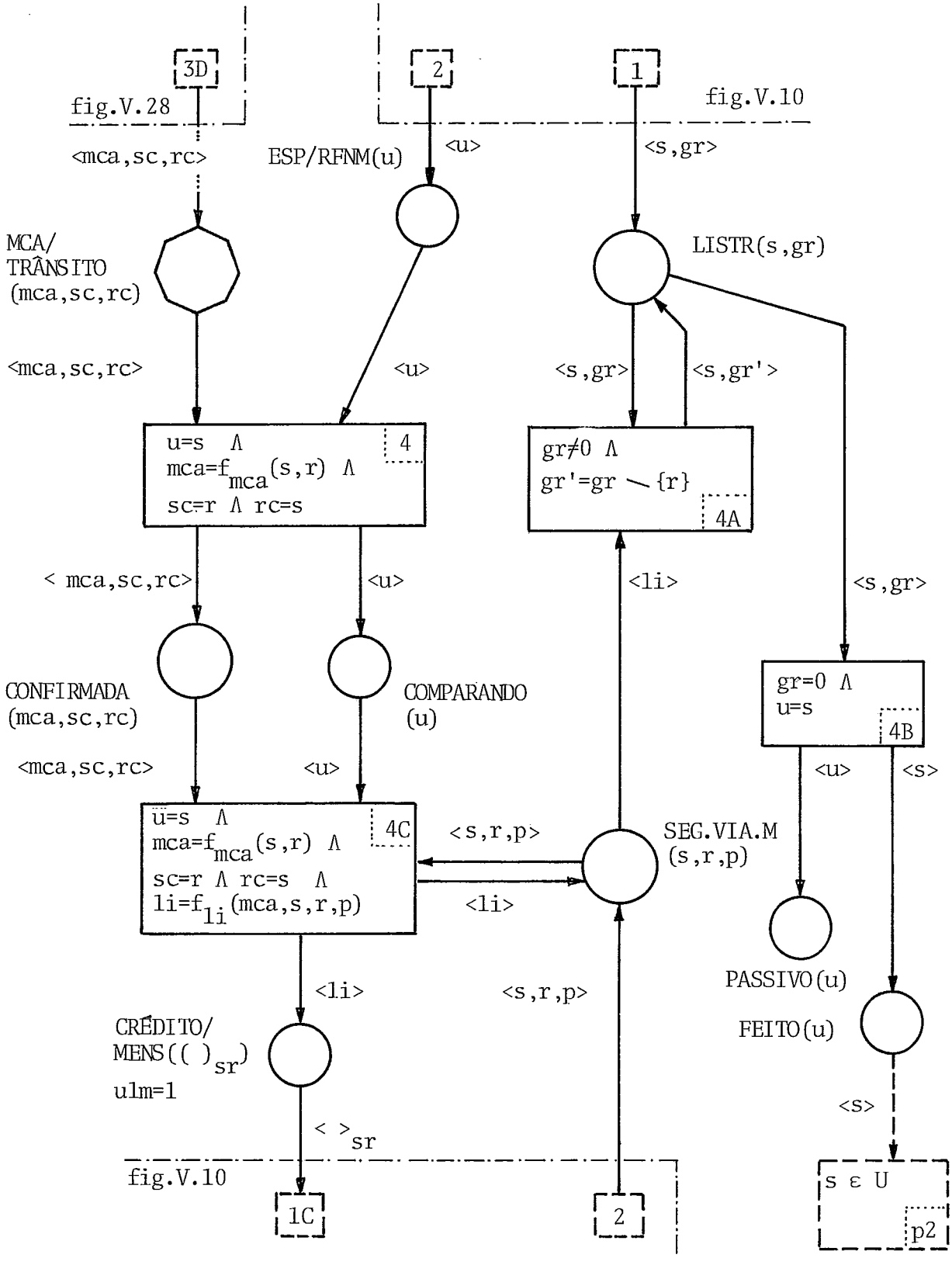


Fig. V.29: Modelo parcial sobre o envio de outras mensagens e o retorno do emissor s, após ter enviado todas as mensagens, ao estado passivo.

DESTIN/BUF(r,kr), que fornece "automaticamente" as condições necessárias para receber uma mensagem.

Outra parte que é diferente do modelo número 1 (figura IV.9) é a parte da confirmação. Esta parte é chamada, agora, de mca para mostrar que é uma mensagem de controle afirmativa. Este mca será, na realidade, um RFNM ('request for next message') e que poderá ser tratada como sendo uma mensagem comum, uma mensagem de controle prioritária ou uma mensagem de controle embutida numa outra mensagem. Trataremos disto em um próximo trabalho.

Entretanto, devemos mencionar, ainda, como se usa esta mensagem de controle na "fonte" e, também, como ela chegou lá. Para conseguir isto, podemos aproveitar, quase integralmente, a parte do modelo número 1 (figura IV.9) envolvida nisto (figura IV.8 no capítulo IV). Precisa-se somente integrar as poucas modificações mencionadas no decorrer deste trabalho para chegar ao esquema da figura V.29; o significado deste esquema pode ser tirado do capítulo IV que foi, afinal, o ponto de partida para o modelo atual.

V.3 - O modelo número 2 da transmissão de um pacote simples numa rede de computadores.

O modelo número 2, representando a anatomia da transmissão de um pacote simples numa rede de computadores (mostrada na figura II.5), se faz simplesmente juntando os diversos modelos parciais.

Assim, temos, sem considerar os diversos detalhamentos, o seguinte conjunto de modelos parciais que representam o fluxo principal dos pacotes:

- Fig. V.10, que mostra, principalmente, a colocação do pedido do usuário s ($s \in U$), as condições para sair do nível usuário (FONTE(s,r) e ATIV/TRANSM(u)), a transferência do nível do usuário ao nível da subrede, via UCP, pela transição 2, e

a entrada na subrede (PRONTO(m,em,re));

- Fig. V.13, que indica algumas maneiras de escolher o "melhor" caminho. Esta figura é considerada auxiliar mas, como a escolha do caminho influenciará diretamente no controle de fluxo, é importante ter ao menos alguma idéia sobre este procedimento;

- Fig. V.14 é a chave de como se pode modelar a passagem da entrada num vértice (PRONTO(m,em,re)), via UCP (onde é feita a escolha final do caminho), para a saída de um vértice (SAÍDA(m,em,re));

- Fig. V.17, que mostra como se pode, baseado no caminho escolhido, ativar o enlace (físico/lógico) e o receptor no vértice vizinho. Feito isto, pode se colocar o pacote em trânsito;

- Fig. V.18 mostra a recepção neste vértice vizinho, modelado pelo predicado ENTRADA(m,em,re);

- Figs. V.20/V.21 indicam como se poderia fazer uma retransmissão: usando o mesmo caminho (fig. V.20), fechando o laço via a figura V.17, ou, usando um outro caminho (fig. V.21), fechando a malha via a figura V.14;

- Fig. V.22 mostra a recepção positiva (ACK) que permite o encaminhamento ao próximo vértice;

- Fig. V.23 mostra, na representação gráfica, como se pode proceder para continuar a modelagem do fluxo dos pacotes, ou seja, como fechar o laço para iniciar um encaminhamento para um próximo vértice (figura V.14);

- Fig. V.24 mostra, se o próximo vértice foi aquele associado ao receptor, o início da remontagem da mensagem (MONTAGEM) para garantir as condições para sair da subrede;

- Fig. V.27 completa esta saída, via UCP, da subrede para que

possamos colocar os pacotes, na seqüência certa para formar a mensagem, no vértice/destinação (DESTINAÇÃO(s,r));

- Fig. V.28 fecha o ciclo da transmissão de uma mensagem;

- Fig. V.29 mostra como responder ao usuário que o seu pedido (figura V.10) foi satisfeito.

Associado a este fluxo dos pacotes está, nestas mesmas figuras, a transferência contínua da responsabilidade pelo pacote, que passa do usuário/emissor s, via os processadores de entrada, UCP's e processadores de saída, para o usuário/receptor r.

Apresentamos, na fig. V.30, o esquema gráfico completo (para isto, basta juntar as figuras mencionadas) e, também, mostraremos, na figura V.31, uma tabela completa dos principais predicados usados, agrupados convenientemente, e, na figura V.32, uma tabela com todas as principais transições. Estas tabelas servirão, mais tarde, em outros trabalhos, para realizar uma representação do esquema gráfico dentro de um computador. Isto se faz necessário porque a complexidade da estrutura e, também, das inscrições aumentou consideravelmente em relação ao esquema inicial, o modelo número 1 (figura IV.9). Assim, estas tabelas representam o início para criar esta ajuda automatizada.

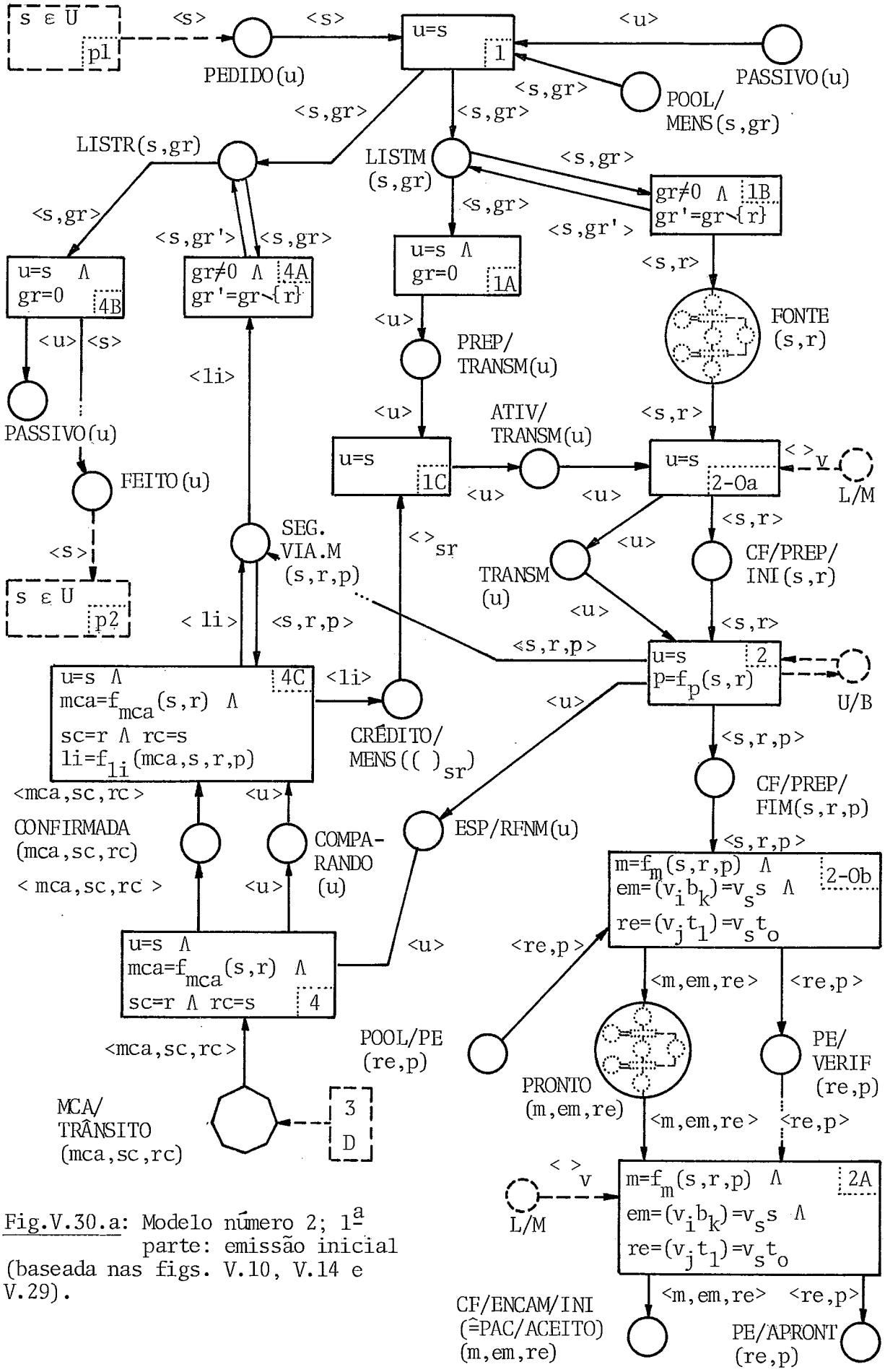


Fig.V.30.a: Modelo número 2; 1ª parte: emissão inicial (baseada nas figs. V.10, V.14 e V.29).

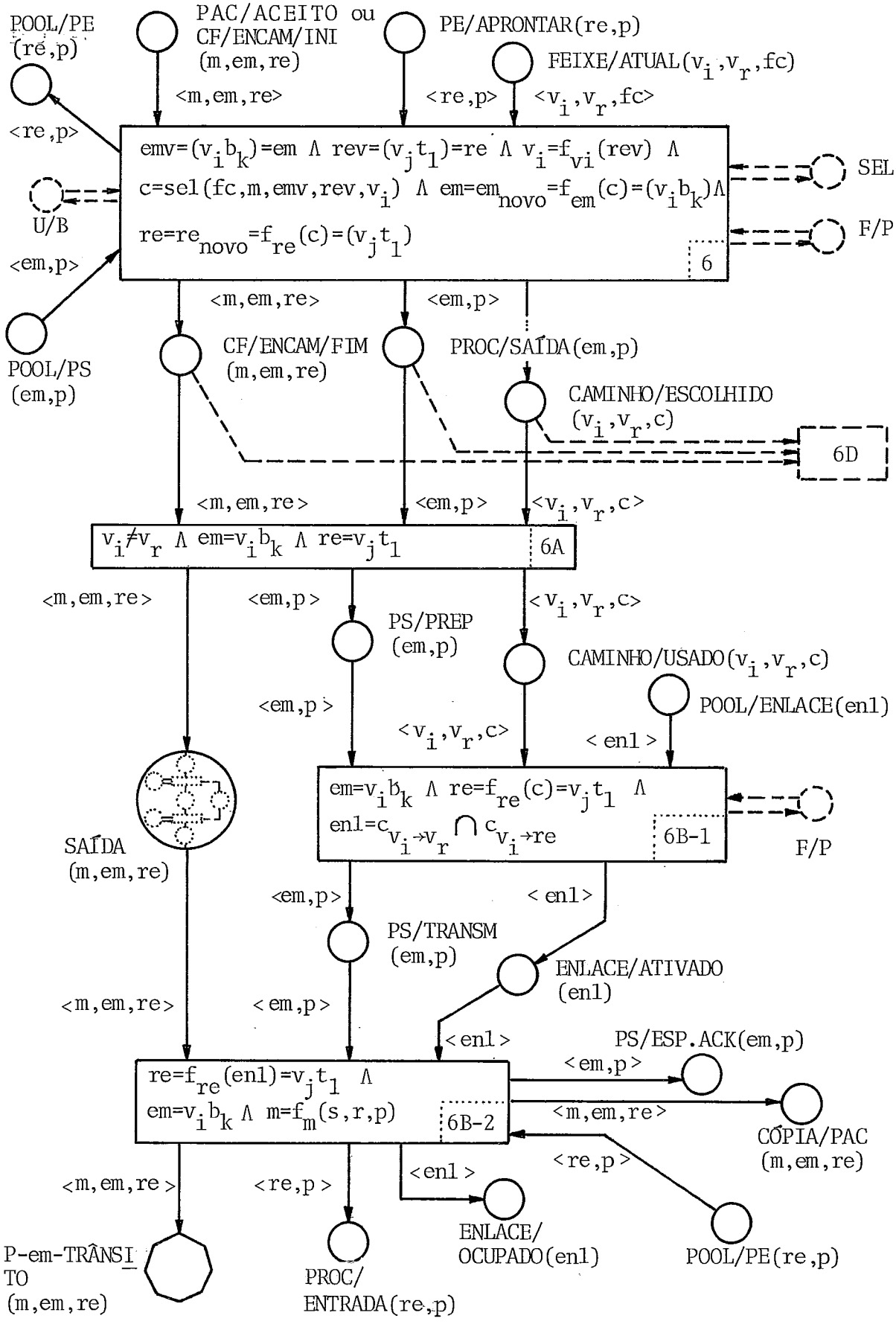


Fig.V.30.b: Modelo número 2; 2ª parte: emissão intermediária (baseada nas figuras V.14 e V.17).

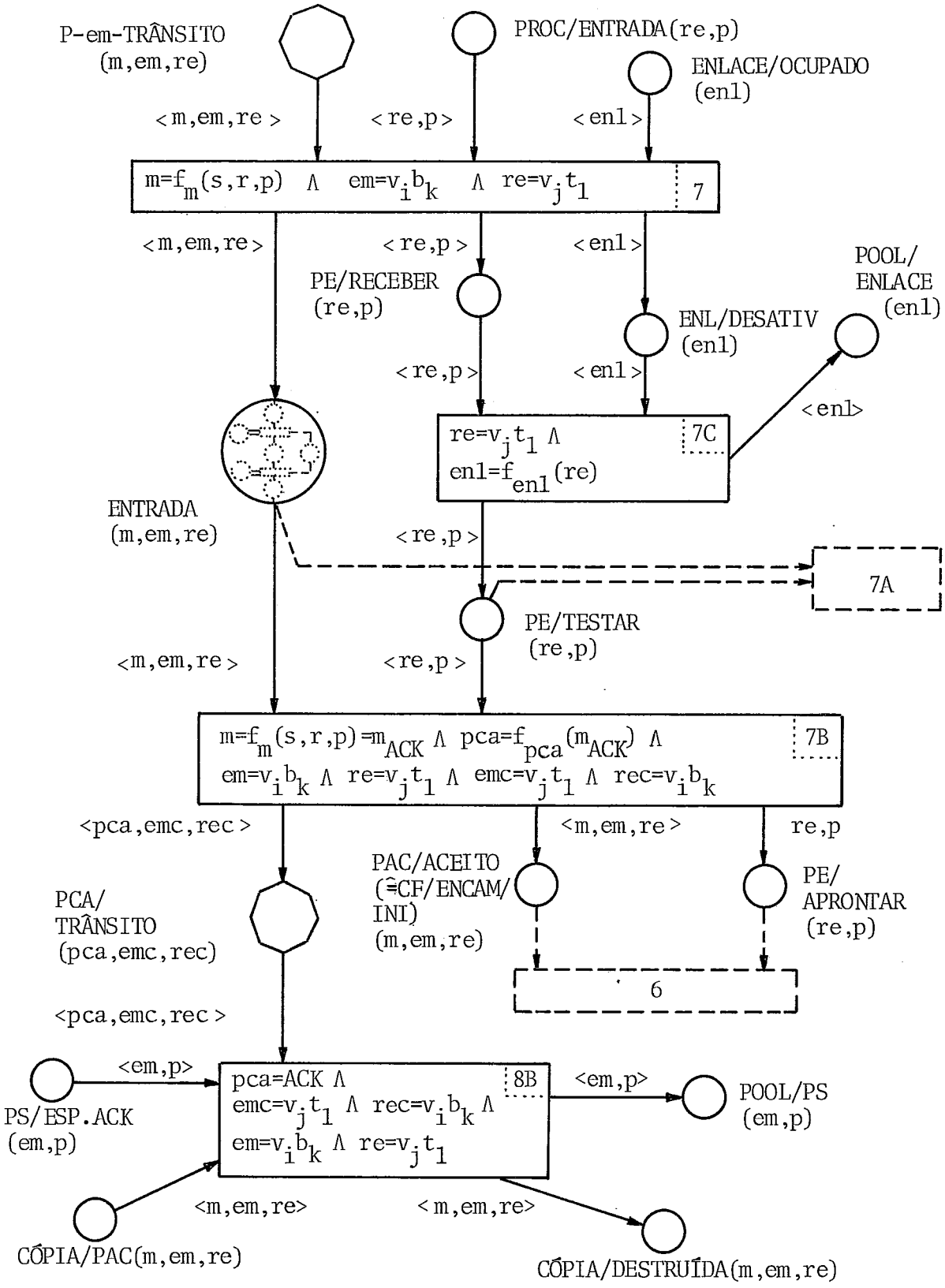


Fig. V.30.c: Modelo número 2; 3ª parte: recepção intermediária (baseada nas figuras V.18 e V.22).

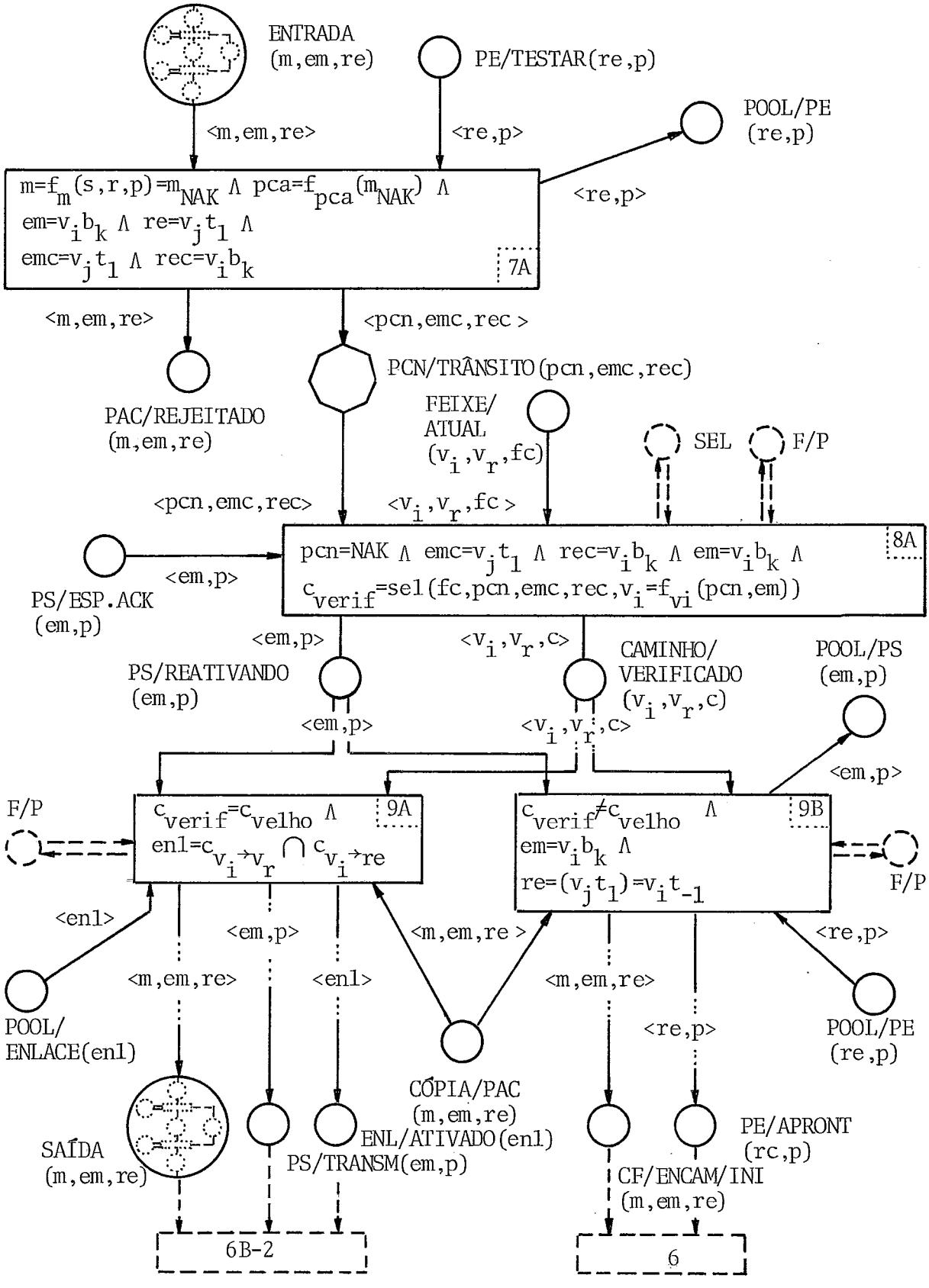


Fig. V.30.d: Modelo número 2; 4ª parte: retransmissão intermediária (baseada nas figuras V.20 e V.21).

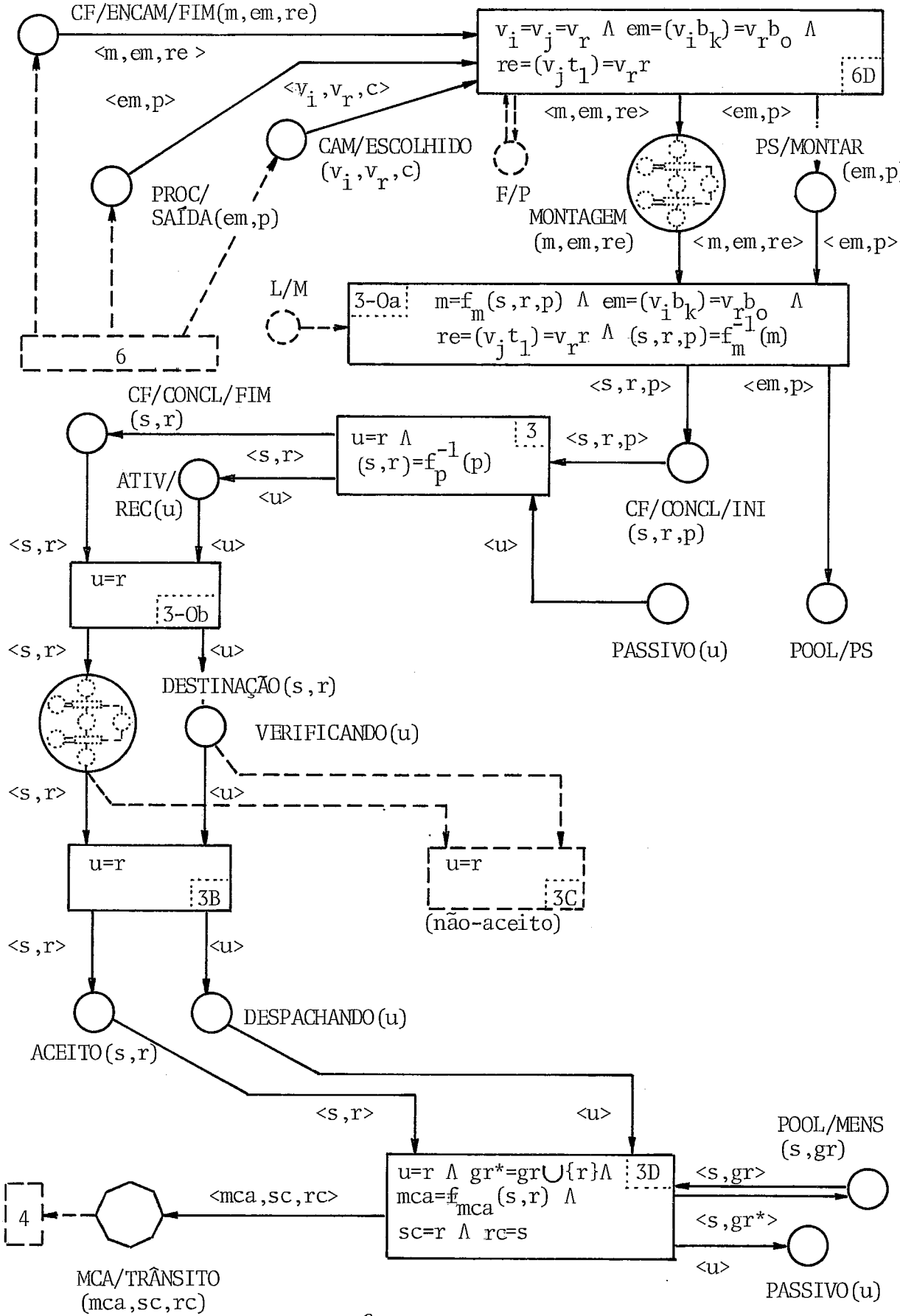


Fig.V.30.e: Modelo número 2; 5ª parte: recepção final (baseada nas figuras V.24 e V.28).

Predicados: MENSAGEM	relação causal com as transições:			
	- ≙ precondição	trans	+ ≙ poscondição	trans
ACEITO(s,r)	-<s,r>	3D	+<s,r>	3B
CÓPIA/DESTRUÍDA (m,em,re)	indefinido		+<m,em,re>	8B
CÓPIA/PAC (m,em,re)	-<m,em,re> -<m,em,re> -<m,em,re>	8B 9A 9B	+<m,em,re>	6B-2
DESTINAÇÃO (s,r) fila: fig. V.27	-<s,r> -<s,r>	3B 3C	+<s,r>	3-Ob
ENTRADA (m,em,re) fila: fig. V.10	-<m,em,re> -<m,em,re>	7A 7B	+<m,em,re>	7
FONTE(s,r) fila: fig. V.5	-<s,r>	2-Oa	+<s,r>	1B
LISTM(s,gr)	-<s,gr> -<s,gr>	1A 1B	+<s,gr> +<s,gr'>	1 1B
LISTR(s,gr)	-<s,gr> -<s,gr>	4A 4B	+<s,gr> +<s,gr'>	1 4A
MONTAGEM (m,em,re) fila: fig. V.25	-<m,em,re>	3-Oa	+<m,em,re>	6D
PAC/ACEITO (m,em,re)	veja: CF/ENCAM/INI		veja: CF/ENCAM/INI	
PAC/REJEITADO (m,em,re)	indefinido		+<m,em,re>	7A
P-em-TRÂNSITO (m,em,re)	-<m,em,re>	7	+<m,em,re>	6B-2
POOL/MENSAGEM (s,gr)	-<s,gr> -<s,gr>	1 3D	+<s,gr*>	3D

continua .....

Observações: São mostrados somente os predicados envolvidos diretamente no controle de fluxo;
O processador central é, em princípio, composto dos mesmos elementos que uma fila em geral, só que a UCP/FILA é dividida em diferentes partes, p.ex., UCP/FILA/PREPARAÇÃO, UCP/FILA/ENCAMINHAMENTO, etc. Mostramos somente aqueles predicados que apareçam dentro da progressão do fluxo.

Fig. V.31: Tabela dos principais predicados do modelo número 2 (fig. V.30.a até V.30.e), agrupados em relação a mensagens em si, mensagens processadas pelo processador central, usuários, processadores de entrada e saída, caminhos e enlaces, controle e auxiliares. São também mostradas as relações (pre e poscondições) com as transições.
Parte 1: MENSAGEM

Predicados: MENSAGEM (cont.)	relação causal com as transições: - ≙ precondição	trans	+ ≙ poscondição	trans
PRONTO(m,em,re) fila: fig. V.9	-<m,em,re>	2A	+<m,em,re>	2-Ob
SAÍDA(m,em,re) fila: fig. V.12	-<m,em,re>	6B-2	+<m,em,re> +<m,em,re>	6A 9A
SEG.VIA.M(s,r,p)	- -<s,r,p>	4A 4C	+<s,r,p> +	2 4C
Predicados: MENSAGEM/UCP				
CF/CONCL/INI(s,r,p)	-<s,r,p>	3	+<s,r,p>	3-Oa
CF/CONCL/FIM(s,r)	-<s,r>	3-Ob	+<s,r>	3
CF/ENCAM/INI (≙PAC/ACEITO) (m,em,re)	-<m,em,re>	6	+<m,em,re> +<m,em,re> +<m,em,re>	2A 7B 9B
CF/ENCAM/FIM	-<m,em,re> -<m,em,re>	6A 6D	+<m,em,re>	6
CF/PREP/INI(s,r	-<s,r>	2	+<s,r>	2-Oa
CF/PREP/FIM(s,r,p)	-<s,r,p>	2-Ob	+<s,r,p>	2
Predicados: USUÁRIOS				
ATIV/RECEBER(u)	-<u>	3-Ob	+<u>	3
ATIV/TRANSM(u)	-<u>	2-Oa	+<u>	1C
COMPARANDO(u)	-<u>	4C	+<u>	4
DESPACHANDO(u)	-<u>	3D	+<u>	3B
ESP/RFNM(u)	-<u>	4	+<u>	2
FEITO(u)	-<s>	p2	+<s>	4B
PASSIVO(u)	-<u> -<u>	1 3	+<u> +<u>	3D 4B
PEDIDO(u)	-<s>	1	+<s>	p1
PREP/TRANSM(u)	-<u>	1C	+<u>	1A
TRANSMITIR(u)	-<u>	2	+<u>	2-Oa
VERIFICANDO(u)	-<u> -<u>	3B 3C	+<u>	3-Ob

Fig. V.31: (continuação: Tabela dos principais predicados do modelo número 2 (fig. V.30.a até V.30.e).

Parte 1 (cont.): MENSAGEM; Parte 2: MENSAGEM/UCP; Parte 3: USUÁRIOS.

Predicados: PROCESSADOR/ENTR.	relação causal com as transições: - ≙ precondição	trans	+ ≙ poscondição	trans
PE/APRONTAR (re,p)	- <re,p>	6	+ <re,p> + <re,p> <re,p>	2A 7B 9B
PE/RECEBER (re,p)	- <re,p>	7C	+ <re,p>	7
PE/TESTAR (re,p)	- <re,p> - <re,p>	7A 7B	+ <re,p>	7C
PE/VERIFICAR (re,p)	- <re,p>	2A	+ <re,p>	2-Ob
POOL/PE (re,p)	- <re,p> - <re,p> - <re,p>	2-Ob 9B 6B-2	+ <re,p> + <re,p>	6 7A
PROC/ENTRADA (re,p)	- <re,p>	7	+ <re,p>	6B-2
Predicados: PROCESSADOR/SAÍDA				
POOL/PS (em,p)	- <em,p>	6	+ <em,p> + <em,p> + <em,p>	3-Oa 8B 9B
PROC/SAÍDA (em,p)	- <em,p> - <em,p>	6A 6D	+ <em,p>	6
PS/ESP.ACK (em,p)	- <em,p> - <em,p>	8A 8B	+ <em,p>	6B-2
PS/MONTAR (em,p)	- <em,p>	3-Oa	+ <em,p>	6D
PS/PREPARANDO (em,p)	- <em,p>	6B-1	+ <em,p>	6A
PS/REATIVANDO (em,p)	- <em,p> - <em,p>	9A 9B	+ <em,p>	8A
PS/TRANSMITINDO (em,p)	- <em,p>	6B-2	+ <em,p> + <em,p>	6B-1 9A
Predicados: CAMINHOS/ENLACES				
CAMINHO/ESCOLHIDO (v_i,v_r,c)	- <v_i,v_r,c> - <v_i,v_r,c>	6A 6D	+ <v_i,v_r,c>	6
CAMINHO/USADO (v_i,v_r,c)	- <v_i,v_r,c>	6B-1	+ <v_i,v_r,c>	6A

continua ...

Fig. V.31: (continuação): Tabela dos principais predicados do modelo número 2 (fig. V.30.a até V.30.e). Parte 4: PROCESSADOR DE ENTRADA; Parte 5: PROCESSADOR DE SAÍDA; Parte 6: CAMINHOS E ENLACES.

Predicados: CAMINHOS (cont.)	relação causal com as transições: - ≙ precondição	trans	+ ≙ poscondição	trans
CAMINHO/VERIFICADO (v_i,v_r,c)	- <v_i,v_r,c> - <v_i,v_r,c>	9A 9B	+<v_i,v_r,c>	8A
ENLACE/ATIVADO (enl)	- ⩽enl>	6B-2	+<enl> +<enl>	6B-1 9A
ENLACE/DESATIVADO (enl)	- <enl>	7C	+<enl>	7
ENLACE/OCUPADO (enl)	- <enl>	7	+⩽enl>	6B-2
FEIXE/ATUAL (v_i,v_r,fc)	- <v_i,v_r,fc> - <v_i,v_r,fc>	6 8A	indefinido neste nível da modelagem	
FEIXE/POSSÍVEL (v_s,v_r,fcp)	auxiliar; veja nos modelos parciais		idem	
POOL/ENLACE (enl)	- <enl> - <enl>	6B-1 9A	+<enl>	7C
SEL(( )$_v$)	auxiliar; veja nos modelos parciais		idem	
Predicados: CONTROLE				
CONFIRMADO (mca,sc,rc)	- <mca,sc,rc>	4C	+<mca,sc,rc>	4
CRÉDITO/MENSAGEM (( )$_{sr}$)	- < >$_{sr}$	1C	+	4C
LÓGICA/MUX(( )$_v$)	auxiliar; veja nos modelos parciais		idem	
MCA/TRÂNSITO (mca,sc,rc)	- <mca,sc,rc>	4	+<mca,sc,rc>	3D
PCA/TRÂNSITO (pca,emc,rec)	- <pca,emc,rec>	8B	+<pca,emc,rec>	7B
PCN/TRÂNSITO (pcn,emc,rec)	- <pcn,emc,rec>	8A	+<pcn,emc,rec>	7A
UCP/BUFFER (v,kp)	auxiliar; veja nos modelos parciais		idem	

Fig. V.31: (cont.-final): Tabela dos principais predicados do modelo número 2 (fig. V.30.a até V.30.e)

Parte 6: (cont.): CAMINHOS E ENLACES; Parte 7: CONTROLE.

T R A N S I Ç Õ E S	
nº	fórmulas
p1	$s \in U$
p2	$s \in U$
1	$u=s$
1A	$u=s \wedge gr=0$
1B	$gr\neq 0 \wedge gr'=gr \setminus \{r\}$
1C	$u=s$
2	$u=s \wedge p=f_p(s,r) \wedge$ (2-1a) $\wedge$ (2-1b)
2-0a	$u=s$
2-0b	$m=f_m(s,r,p) \wedge em=(v_i b_k)=v_s s \wedge re=(v_j t_1)=v_s t_o$
2-1a	$z'=z+1,\text{mod.}n \wedge z=\#\langle v,kp\rangle$
2-1b	$z'=z+1,\text{mod.}n \wedge z=\#\langle v,kp\rangle$
2A	$m=f_m(s,r,p) \wedge em=(v_i b_k)=v_s s \wedge re=(v_j t_1)=v_s t_o$
3	$u=r \wedge (s,r)=f_p^{-1}(p) \wedge$ (3-1a) $\wedge$ (3-1b)
3-0a	$m=f_m(s,r,p) \wedge em=(v_i b_k)=v_r b_o \wedge re=(v_j t_1)=v_r r \wedge (s,r,p)=f_m^{-1}(m)$
3-0b	$u=r$
3-1a	$z'=z+1,\text{mod.}n \wedge z=\#\langle v,kp\rangle$
3-1b	$z'=z+1,\text{mod.}n \wedge z=\#\langle v,kp\rangle$
3B	$u=r$
3C	$u=r$
3D	$u=r \wedge gr^*=gr \cup \{r\} \wedge mca=f_{mca}(s,r) \wedge sc=r \wedge rc=s$
4	$u=s \wedge mca=f_{mca}(s,r) \wedge sc=r \wedge rc=s$
4A	$gr\neq 0 \wedge gr'=gr \setminus \{r\}$
4B	$gr=0 \wedge u=s$
4C	$u=s \wedge mca=f_{mca}(s,r) \wedge sc=r \wedge rc=s \wedge li=f_{li}(mca,s,r,p)$
6	(6-1a) $\wedge$ (6-1b) $\wedge$ (6-2)
6-1a	$em_{velho}=(v_i b_k)=emv \wedge re_{velho}=(v_j t_1)=rev \wedge z'=z+1,\text{mod.}n \wedge z=\#\langle v,kp\rangle$
6-1b	$em=em_{novo} \wedge re=re_{novo} \wedge m=f_m(s,r,p) \wedge z'=z+1,\text{mod.}n \wedge z=\#\langle v,kp\rangle$
6-2	(6-2a) $\wedge$ (6-2b)

Fig. V.32: Tabelas das principais transições (incluindo algumas secundárias) da fig. V.30 e suas fórmulas envolvidas diretamente com o controle de fluxo.

T R A N S I Ç Õ E S	
nº	fórmulas
6-2a	$v_i=f_{vi}(rev) \wedge c=sel(fc,m,emv,rev,v_i) \wedge [fcp'=fcp \cup (fc \setminus \{c\})]$
6-2b	$em_{novo}=f_{em}(c)=v_i b_k \wedge re_{novo}=f_{re}(c)=v_j t_1$
6A	$v_i \neq v_r \wedge em=v_i b_k \wedge re=v_j t_1$
6B-1	$em=v_i b_k \wedge re=f_{re}(c)=v_j t_1 \wedge [fcp'=fcp \cup \{c\}] \wedge enl=c_{v_i \to v_r} \cap c_{v_i \to re}$
6B-2	$re=f_{re}(enl)=v_j t_1 \wedge em=v_i b_k \wedge m=f_m(s,r,p)$
6D	$v_i=v_j=v_r \wedge [fcp'=fcp \cup \{c\}] \wedge em=(v_i b_k)=v_r b_o \wedge re=(v_j t_1)=v_r r$
7	$m=f_m(s,r,p) \wedge em=v_i b_k \wedge re=v_j t_1$
7A	$m=f_m(s,r,p)=m_{NAK} \wedge pcn=f_{pcn}(m_{NAK}) \wedge em=v_i b_k \wedge re=v_j t_1 \wedge$ $emc=v_j t_1 \wedge rec=v_i b_k$
7B	$m=f_m(s,r,p)=m_{ACK} \wedge pca=f_{pca}(m_{ACK}) \wedge em=v_i b_k \wedge re=v_j t_1 \wedge$ $emc=v_j t_1 \wedge rec=v_i b_k$
7C	$re=v_j t_1 \wedge enl=f_{enl}(re)$
8A	$pcn=NAK \wedge emc=v_j t_1 \wedge rec=v_i b_k \wedge em=v_i b_k \wedge$ $c_{verif}=sel(fc,pcn,emc,rec,v_i=f_{vi}(pcn,em)) \wedge [fcp'=fcp \cup (fc \setminus \{c\})]$
8B	$pca=ACK \wedge emc=v_j t_1 \wedge rec=v_i b_k \wedge em=v_i b_k \wedge re=v_j t_1$
9A	$c_{verif}=c_{velho} \wedge [fcp'=fcp \cup \{c\}] \wedge enl=c_{v_i \to v_r} \cap c_{v_i \to re}$
9B	$c_{verif} \# c_{velho} \wedge [fcp'=fcp \cup \{c\}] \wedge em=v_i b_k \wedge re=(v_j t_1)=v_i t_{-1}$

Obs.: As funções são, na maioria dos casos, definidas de uma maneira similar a este exemplo: $p=f_p(s,r)$, onde $f_p:(U \times U \to P)$.

Fig. V.32: (cont.-final): Tabela das principais transições ...

V.4 - Resumo do capítulo V.

Tendo terminado a modelagem do esquema da figura II.5, isto é, a transmissão de pacotes simples, via vértices intermediários, numa rede de computadores, podemos concluir, basicamente, duas coisas.

Primeiramente, este modelo número 2 obtido é bem mais instrutivo que aquela da figura IV.9. Isto se deve ao fato que incluímos, além dos vértices intermediários, também os caminhos que representam recursos muito disputados. Vimos que a técnica de 'PrT-Net' ajudou bastante para modelar e, através da modelagem, compreender o sistema em questão.

Entretanto, por outro lado, tentando "entrar" mais na estrutura do sistema, vimos que precisamos definir exatamente até onde penetrar. Por exemplo, no modelo preliminar (isto é, no modelo número 1) ficamos no nível do usuário; conseqüentemente, a subrede e todas as implicações envolvidas ficaram transparentes para nós. Já agora, "abrindo" esta subrede, vimos que começa a verdadeira competição pelos recursos. Tentamos modelar estes problemas, baseados em nossos conhecimentos limitados e, também, na intuição e no bom senso, mas sentimos que se precisa que isso.

Isso nos leva, então, às sugestões para a continuação deste trabalho. Deve ter ficado claro que não podemos, tampouco, chamar este modelo número 2 de "final" mas somente de "intermediário". Mas acreditamos que ele já representa uma boa base e que os melhoramentos podem ser feitos "localmente". Compreendemos como "local" a possibilidade de explorar mais, do ponto de vista da modelagem, os acontecimentos reais dentro, por exemplo, das filas de espera (atendimento e processamento), a distribuição ótima dos espaços dentro dos 'buffers', as estratégias usadas pela UCP, etc.. Achamos que podemos, em princípio, "abrir" qualquer predicado, ou conjunto de predicados, para investigações mais aprofundadas.

A título de exemplo podemos sugerir explorações sobre:

- LÓGICA/MUX, UCP/PERM e UCP/BUFFER;
- INFO/GERAL e INFO/ATUAL;
- P-em-TRÂNSITO, PCA/TRÂNSITO, etc;
- MONTAGEM,

só para mencionar algumas possibilidades.

Mas queremos alertar que não se deve perder de vista o funcionamento global do sistema.

Então, as nossas intenções imediatas são as seguintes: melhorar este modelo número 2, já nos próximos capítulos, no sentido de incluir

- a realização de um 'time-out' e 'check-time' como ferramentas de vigilância;
- o atendimento de mensagens do tipo múltipacotes.

Completar, num outro trabalho, o modelo obtido (já com os melhoramentos mencionados) com investigações em relação a mensagens de controle, classes dos usuários e certas prioridades cabíveis.

Em paralelo a isto, devemos nos preocupar com a representação da estrutura e das inscrições dentro de um computador. Este ponto, por um lado, visa ser uma ajuda valiosa na modelagem já que as estruturas e as inscrições ficam bastante complexas e, por outro lado, representa uma ferramenta para investigações sobre a dinâmica de um sistema (p.ex., investigações sobre as invariantes do sistema, transições mortas, concorrências, etc.).

Somente quando estes pontos forem resolvidos podemos, em nossa opinião, pensar em investigações sobre a dinâmica de sistemas complexos cuja base de interconexão será uma rede de computadores.

C A P Í T U L O VI

Melhoramentos do modelo número 2 sobre a transmissão de pacotes simples em redes de computadores no sentido de incluir a verificação de tempos.

VI.1 - Introdução.

(Obs.: uma versão preliminar deste capítulo foi publicada como relatório técnico, SCHWARZ(36)).

Este capítulo pode ser considerado como sendo complemento do capítulo V no sentido de adicionar aspectos de modelagem com respeito aos tempos envolvidos na transmissão de pacotes. No capítulo mencionado, que tratou dos problemas da transmissão de pacotes simples através de uma rede de computadores, mostramos os problemas nos vértices intermediários incluindo, também, as conseqüências em relação a mensagens de controle do tipo ACK ('acknowledgement') e NAK ('negative ACK').

Mostraremos agora, neste capítulo, como incluir o controle, ou, melhor dizer, a verificação dos tempos envolvidos, notadamente o chamado 'time-out', que é associado ao tempo de transmissão entre vértices vizinhos, e o 'check-time', que funciona no nível 'end-to-end'.

VI.2 - A indicação dos tempos envolvidos na transmissão de um pacote simples numa rede de computadores.

Observando, em geral, transmissões de sinais, pacotes, mensagens, etc., vemos que temos sempre duas dimensões a considerar. Uma é representada pelo espaço físico e envolve, principalmente, a topologia da estrutura que suporta as transmissões.

A outra dimensão trata dos tempos envolvidos para conseguir, por exemplo, um certo deslocamento físico nesta estrutura de suporte.

Agora, é graças à dimensão "tempo" que podemos utilizar e reutilizar espaços físicos, rotas, etc. por vários usuários diferentes, obviamente, em momentos diferentes. Mas igualmente, é devido a esta dimensão que se precisa ter controles mais sofisticados para evitar conflitos quando dois ou mais elementos lutam pelo mesmo recurso no mesmo instante. As filas, presença constante em qualquer estrutura que envolve o fluxo de comodidades, são exemplos clássicos de como controlar este tipo de competição.

Mas, existe ainda uma outra preocupação, ou seja, queremos também saber por quanto tempo um determinado recurso é, ou será, ocupado para poder, eventualmente, melhor articular as nossas ferramentas de controle. As verificações do 'time-out' e do 'check-time' se enquadram nesta preocupação e é através dos resultados obtidos, em relação a estes tempos, que podemos tomar outras ações como, por exemplo, liberar um espaço na memória, iniciar uma retransmissão ou atualizar uma estatística, só para mencionar algumas das conseqüências possíveis.

Nas seções VI.2.a e VI.2.b explicaremos, brevemente, os dois tempos mencionados e, na seção VI.3, tentaremos modelar estas idéias usando, novamente, a técnica de 'PrT-Net'.

VI.2.a - Controle do tempo de transmissão entre vértices vizinhos ('time-out').

Tomando, novamente, como base as descrições qualitativas, apresentadas por SCHWARZ (30), podemos usar, como ponto de partida, o esquema que mostrou a anatomia de uma transmissão de um único pacote, fig. II.5, e que inclui o método simples de transmissões em canais, fig. II.1. Nesta figura II.5 podemos destacar o 'time-out' (indicado como sendo 125 ms) que verifica o tempo necessário para conseguir a transmissão de um pacote e re

ceber a subseqüente confirmação (ACK ou NAK) deste pacote em questão.

No caso em que não chegou nenhuma confirmação dentro do tempo previsto, devemos tomar uma atitude que, na maioria dos casos, é a retransmissão do pacote em questão. Deve ser observado, porém, que esta ação de retransmitir é baseada numa hipótese porque não sabemos se o pacote foi transmitido erroneamente ou lento demais, ou se houve algum problema com as confirmações; conseqüentemente, a nossa decisão é sujeita a erros o que, por sua vez, leva a outros tipos de controles adicionais como, p. ex., "marcar" os pacotes retransmitidos.

VI.2.b - Controle do tempo de transmissão entre vértice/fonte e vértice/destinação ('check-time').

Do mesmo modo como se pode verificar o tempo em relação a transmissões intervértice, podemos verificar o tempo envolvido no controle 'end-to-end'.

Como foi mencionado por KLEINROCK (14), o vértice/fonte espera aproximadamente 30 segundos antes de investigar o porque da não-chegada do RFNM esperado. Esta observação indica, já, uma diferença fundamental em relação ao 'time-out', ou seja, não se retransmite automaticamente depois de ter constatado um excesso de tempo; ao contrário, através do envio de uma mensagem de controle tenta-se descobrir a causa desta falha. Deve ficar claro, que este comportamento é bem mais racional já que a transmissão 'end-to-end' e a associada transmissão do RFNM envolvem toda a estrutura da rede (através dos caminhos utilizados, ou seja, enlaces e vértices intermediários) e, também, a ligação para o H/D que, afinal, gera o RFNM em questão.

Assim, como não modelamos ainda, neste trabalho, as mensagens de controle de uma maneira explícita, contentamos-nos com a modelagem de um RFNM bem recebido, isto é, dentro de tempo previsto ('check-time').

Na figura II.3 podem-se observar as possíveis ações, ignorando, contudo, o conteúdo das mensagens de controle envolvidas e, também, as ações a serem tomadas, num caso falho, ao que diz respeito à liberação do próximo pacote para transmissão. Só para mencionar, as ações extremas podem ser a liberação do próximo pacote (correndo, assim, o risco de novos problemas) ou bloquear este par até que chegue a resposta à mensagens de controle (provocando, eventualmente, congestão na entrada a este V/F).

Seja como for, as conseqüências, num caso falho, fogem de nosso interesse imediato e ficam, assim, para um próximo trabalho.

VI.3 - Modelagem, na base de 'PrT-Net', dos TEMPOS envolvidos na transmissão de pacotes simples em redes de computadores.

VI.3.a - Introdução.

Já falamos sobre os tempos em geral e a impressão que ficou é que a ferramenta "controle e/ou observação dos tempos envolvidos" tem uma importância bastante singular. Então, o que será mais lógico que tentar modelar este envolvimento?

Contudo, devemos alertar que modelaremos somente o 'time-out' e o 'check-time', deixando eventuais outros tempos envolvidos para estudos posteriores.

Também deve ser salientado que ficaremos, na modelagem, num nível lógico e que não nos preocuparemos com detalhes que envolvem o compartilhamento e funcionamento dos relógios de computadores, sejam eles os fornecedores dos pulsos básicos ou derivações na base de ns, μs, ms ou segundos.

Tampouco trataremos, neste trabalho, o problema de como sincronizar estes relógios numa estrutura distribuída, mesmo sabendo que a sincronização significa um fator vital em sistemas que

envolvem rádio-difusão ou acesso aos enlaces via satélite como alertado por SCHWARZ (31,32).

Assim, o conteúdo desta seção se restringe a explicações sobre como modelar tempos em geral (seção VI.3.b), seguido pelas tentativas de modelar o 'time-out' (seção VI.3.c) e o 'check--time' (seção VI.3.d). Estas tentativas consistem, inicialmente, na definição sobre o "início e o fim do trabalho", sendo assim, depois, a base para modelar as "conseqüências" provocadas.

VI.3.b - A maneira como modelar TEMPOS, em geral, usando a técnica de 'PrT-Net'.

Temos algum TRABALHO que deve ser executado num prazo máximo de, digamos, t segundos (obs.: usamos, de propósito, o nome trabalho para não confundir com os conceitos usados em nosso modelo que são tarefas, subtarefas, seqüências, etc.). Assim, se este trabalho for executado dentro deste prazo, isto é, em $t \leq t_{max}$, o resultado pode ser considerado positivo. Por outro lado, se ele não terminou a execução neste prazo, ou seja, se $t > t_{max}$, o resultado deve ser considerado como negativo. Um primeiro esboço desta idéia pode ser observado na fig. VI.1.

Queremos agora saber o que realmente acontece DENTRO da transição esboçada. Numa primeira aproximação podemos dizer que são feitos

- a execução do trabalho;
- a contagem regressiva do tempo restante para execução deste trabalho;
- uma comparação do tempo percorrido com o prazo estabelecido.

Tendo isto, podemos estabelecer os predicados cujos argumentos fornecem, através das relações causais (interligação via os arcos), as condições para disparar as transições (lembrar das explicações sobre o 'PrT-Net' no capítulo III deste trabalho.

Predicados.

TRABALHO(arg,td); representa algum trabalho, descrito pelo argumento arg, a ser executado num tempo ou prazo determinado, td (td ε N);

EXECUÇÃO/INÍCIO(arg); representa o início da execução do trabalho em questão, onde arg ε ARG;

EXECUÇÃO/FIM(res); representa o fim (o resultado) desta execução, onde res=f_{res}(arg), res ε RES;

TEMPO/RESTANTE(arg,td); representa um contador regressivo que será iniciado com o tempo determinado td associado ao trabalho a ser executado, onde arg ε ARG, td ε N;

TRABALHO/nãoEXECnoPRAZO(neg); representa o fato que, dentro do tempo determinado, o trabalho não conseguiu ser terminado, onde neg=f_{neg}(arg Λ tr ≤ 0), tr ε N, com tr sendo denominado de tempo restante;

TRABALHO/EXECUTADO(pos); representa que a execução conseguiu ser terminada dentro do tempo determinado (prazo estabelecido), onde pos=f_{pos}(res Λ tr > 0), pos ε POS.

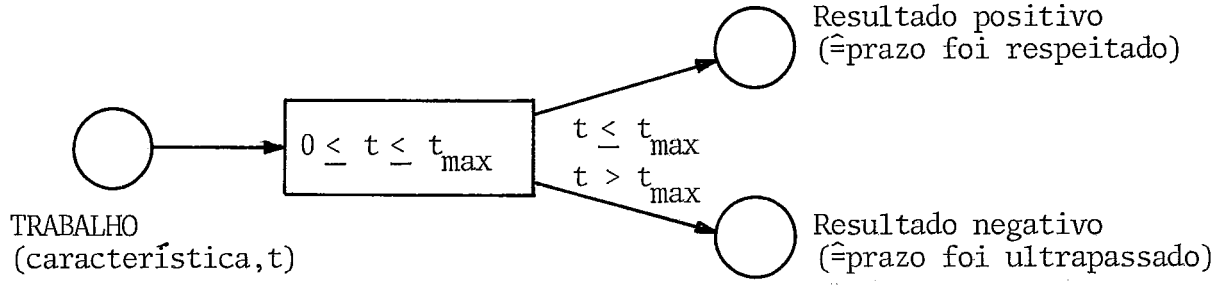


Fig. VI.1: Esboço sobre o resultado da execução de um trabalho em relação a um dado prazo.

Argumentos.

ARG = $\{arg_1, arg_2, \ldots, arg_n\}$; significa alguma descrição das características do trabalho;

N = {1, 2, 3, ...}; números naturais que servem para expressar o tempo determinado (o prazo) td (td ε N) e o tempo restante tr (tr ε N), sendo estes, assim, baseados em números discretos em vez de usar um t contínuo;

NEG = $\{neg_1, neg_2, \ldots, neg_n\}$; resultado negativo, onde a expressão $neg=f_{neg}(arg \wedge tr \leq 0)$;

RES = $\{res_1, res_2, \ldots, res_n\}$; resultado, em geral, da execução do trabalho, $res=f_{res}(arg)$;

POS = $\{pos_1, pos_2, \ldots, pos_n\}$; resultado positivo desta execução porque foi obtido dentro do tempo predeterminado, onde $pos=f_{pos}(res \wedge tr \leq 0)$.

Modelo inicial.

Com estes predicados e argumentos podemos, já, esboçar uma tentativa de um modelo na base de 'PrT-Net'. Veja isto na figura VI.2 e observe as ligações causais entre os predicados e transições com suas respectivas inscrições.

Problemas.

Observando atentamente este modelo, percebemos problemas inerentes do seguinte tipo:

- A transição 2 "funciona" num ritmo independente do trabalho em si (execução, comparação dos resultados, outras verificações, etc.). Portanto, queremos colocar aqui o alerta, já visando o uso prático deste modelo, de que a unidade de tempo escolhida, para ser usada no contador regressivo, deve ser

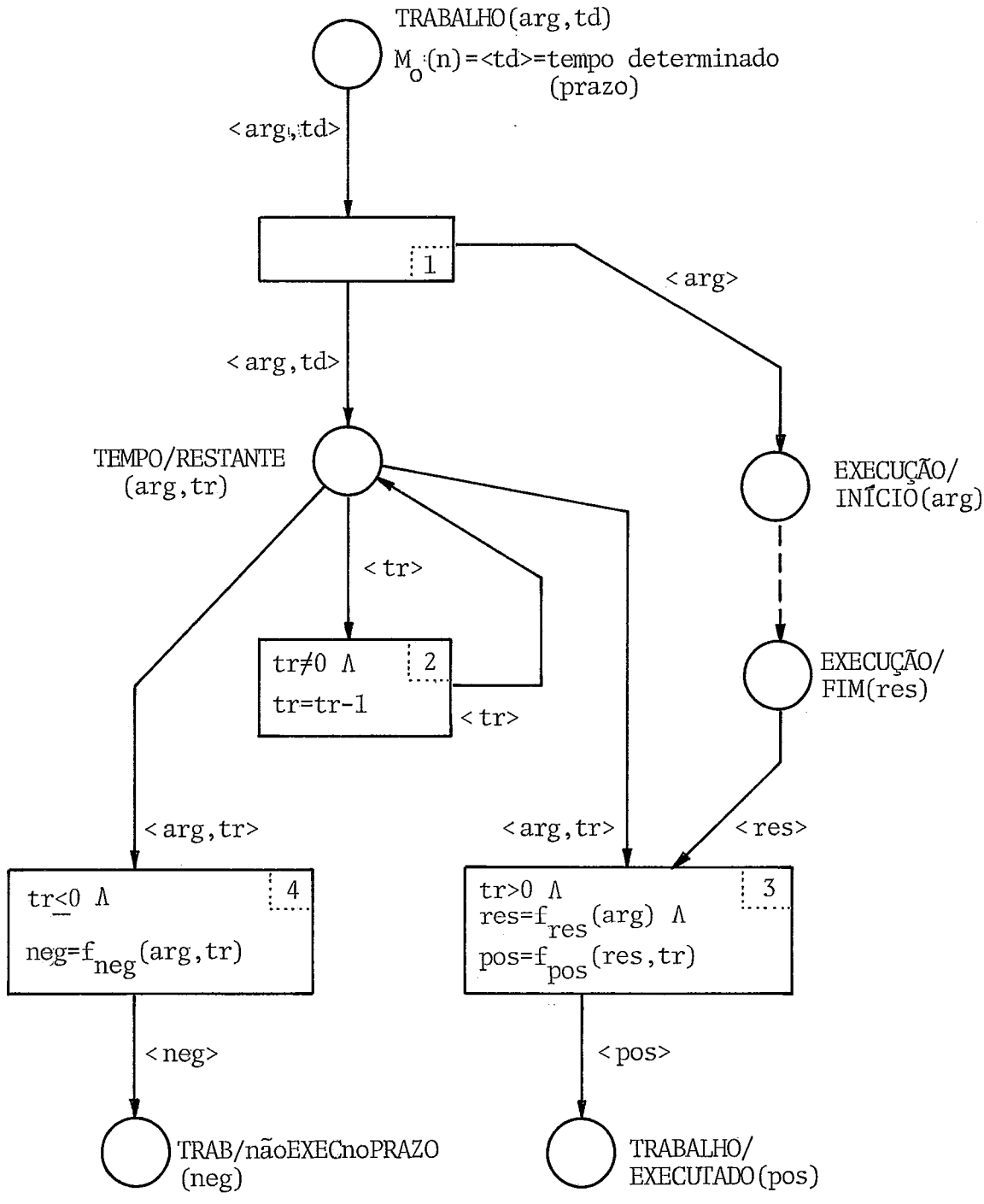


Fig. VI.2: Primeira tentativa de modelar a execução de um trabalho em relação a um determinado prazo ou tempo.

menor que o tempo de execução do trabalho mas, ao mesmo tempo, precisa ser estritamente maior que o tempo necessário para fazer as operações pelo computador (p.ex., comparação, decisão, etc.). Se não o fizermos, devemos refazer este modelo para incorporar os problemas relacionados a este fato;

- As transições 3 e 4 não devem disparar ao "mesmo" tempo. Portanto, como NÃO podemos PARAR a transição 2, precisamos evitar que o "novo" resultado da contagem regressiva seja colocado em TEMPO/RESTANTE(arg,n), acusando, eventualmente, tr=0 e, assim, fornecendo a condição necessária para que a transição 4 disparasse; isto pode ser considerado falho no caso em que a transição 3 já disparou com o "velho" resultado da contagem regressiva.

Solução destes problemas.

Para resolver esta imprecisão no modelo, podemos colocar, junto ao tempo restante, tr, um elemento travador ou bloqueador, bl, que:

- por um lado liberará o resultado da execução, representado pelo predicado EXECUÇÃO/FIM(res,bl), com ulm(bl)=1;

- por outro lado, se não for "usado" para liberar o resultado (que, p.ex., não estiver ainda pronto), serviria para liberar o resultado da contagem regressiva.

Deste modo, chegamos ao modelo final que, usando a técnica de 'PrT-Net', mostra como relacionar a execução de um trabalho a um prazo preestabelecido. Veja isto na figura VI.3 e observe, também, as modificações, no grau dos predicados e nas tuplas, por causa deste elemento bloqueador.

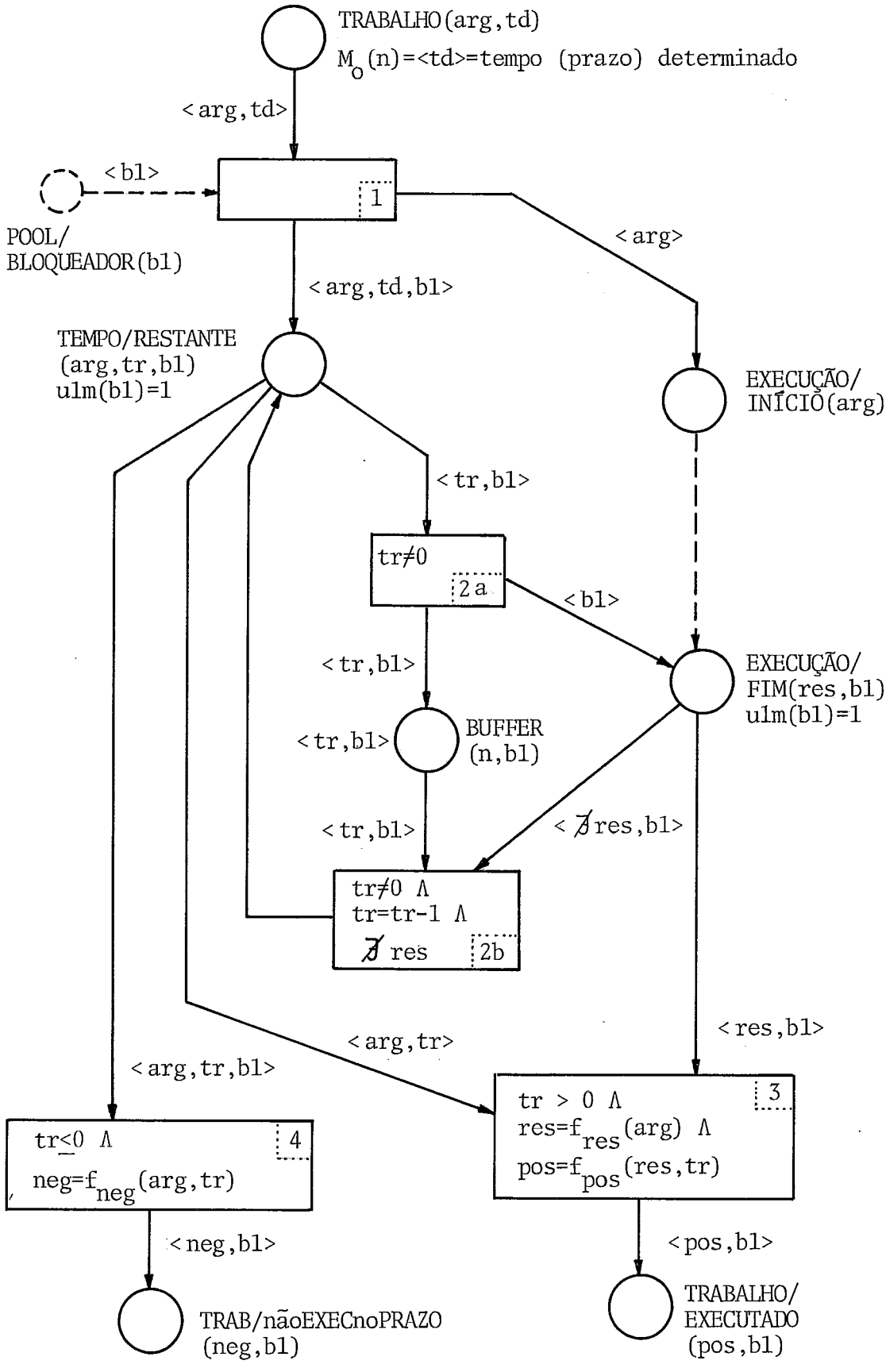


Fig. VI.3: Modelo final para mostrar a relação entre a execução de um trabalho com um tempo ou prazo predeterminado.

VI.3.c - Tempos envolvidos no controle intervértice ('time-out' no nível S/F).

VI.3.c.1 - Idéias gerais.

Primeiramente, devemos declarar que usaremos, em vez do nome TRABALHO (veja fig. VI.3), expressões ligadas a redes de computadores. Assim, temos TAREFA que se refere à transmissão 'end--to-end' e SUBTAREFA que representa a transmissão entre vértices vizinhos.

Começando, nesta seção, com o controle intervértice, devemos definir a expressão SUBTAREFA. Ela consiste, em princípio, da seguinte seqüência:

- transmissão (emissão e recepção) de um pacote entre vértices vizinhos;
- teste no receptor;
- preparação de alguma confirmação (ACK e NAK);
- Transmissão (emissão e recepção) desta confirmação positiva ou negativa.

Portanto, do ponto de vista da modelagem, podemos considerar que o argumento $\hat{a}rg \in \hat{A}RG = \{\hat{a}rg_1, \hat{a}rg_2, \ldots\})$, que aparece, por exemplo, num predicado do tipo SUBTAREFA(ârg,tds), representaria toda a seqüência dos eventos mencionados. Como observação podemos já adiantar que o circunflexo foi usado para indicar que o argumento não é completamente previsível, mas que tem partes estimadas ou esperadas. Entretanto, não precisamos, felizmente, pormenorizar esta seqüência, porque, para modelar a relação com algum prazo estabelecido, é somente necessária a definição exata do início e do fim desta subtarefa.

Mesmo tendo dito que não precisamos pormenorizar o argumento ârg, achamos conveniente explicar como se poderia chegar a ele, já que aparece não somente nos diversos predicados como, p.ex.,

em SUBTAREFA (âr̂g,t̂ds), mas também na fórmula que calcula o tempo t̂ds. Assim, teoricamente, devemos levar em consideração que, para definí-lo, temos uma parte das condições "presente", como, p.ex., m, em ou re, e uma outra parte "previsível", representada, p.ex., por pca, pcn, emc ou rec. Será, então, necessário criar uma multitude de funções para especificar âr̂g. Seriam elas, entre outras, as seguintes expressões básicas:

$\hat{p}cn = \hat{p}ca \vee \hat{p}cn = \hat{f}_p(m)$; representa um pacote de controle que está sendo esperado como, por exemplo, neste caso, uma confirmação da recepção contendo uma afirmação ou negação em relação à consistência do pacote recebido. Observe, que usamos o símbolo "$\hat{}$" porque no momento em que precisamos este definição para calcular o âr̂g, ela ainda não está conhecida neste lugar do cálculo, mas somente no receptor (p.ex., na fórmula da transição 7B : $m=f_m(s,r,p)=m_{ACK} \wedge pca=f_{pca}(m_{ACK})$);

$\hat{e}mc = \hat{f}_e(re)$; representa o emissor do pacote de controle esperado, ou seja, provavelmente se usa uma função especial do receptor para emissão da confirmação (p. ex., parte da faixa de freqüência reservada para este fim);

$\hat{r}ec = \hat{f}_r(em)$; explicação similar àquela usada para esclarecer êmc.

A partir destas expressões básicas (p̂cc, êmc, r̂ec) e das informações contidas na emissão do pacote original (m,em,re) podemos "criar" as seguintes derivações:

$m^* = f^*_m(m,em,re)$; é necessária para ajudar na identificação da primeira parte na seqüência da execução da subtarefa, ou seja, que a mensagem m passa de em ($\hat{=}v_i b_k$) para re ($\hat{=}v_j t_l$) ;

$\hat{m} = \hat{f}_m(\hat{p}cc,\hat{e}mc,\hat{r}ec)$; idem a m*, só que serve para identificar a seqüência na transmissão do pacote de controle.

Deste modo, podemos dizer que

$$\hat{a}rg = f_{arg}(m^* \wedge \hat{m}) \ ;$$

Agora, para evitar todo este conjunto de equações, usaremos, na representação gráfica, uma "função especial" do tipo

$$\hat{a}rg = \hat{f}^*_{arg}(m,em,re)$$

que engloba todos estes passos intermediários, sejam eles previsíveis ou não.

Assim, falando em termos do modelo número 2, obtido no capítulo V, a transição 6B-2 deveria conter, além das inscrições usadas, também esta fórmula mencionada.

Então, chegamos à segunda parte do predicado mencionado, ou seja, devemos definir o tempo determinado $\hat{t}ds$ (o prazo estabelecido) para esta subtarefa.

Este tempo deve ser, inicialmente, estimado de uma maneira tal que incluia, no mínimo, os tempos da transmissão do pacote, da sua verificação e da transmissão da confirmação. Este tempo básico, mais alguma margem de segurança, daria um primeiro valor, digamos $\hat{t}ds$. Agora, para obter um desempenho da rede perto do ótimo, precisa-se verificar este $\hat{t}ds$ inicial durante a operação da rede para poder, eventualmente, adaptá-lo às necessidades reais. Assim, devemos assegurar que $\hat{t}ds$ não é pequeno demais (o que pode induzir retransmissões desnecessárias), nem muito grande (o que poderia causar perda no desempenho no caso em que haja necessidade de iniciar uma retransmissão).

Com respeito à modelagem podemos dizer que o cálculo do $\hat{t}ds$ depende, por um lado, do pacote em questão e, por outro lado, da situação na rede, (em especial do enlace envolvido na transmissão). Supondo, então, que $\hat{a}rg$ (através de m,em,re) contenha todas as informações necessárias em relação ao "tipo da mensagem" (p.ex., pacote pequeno, arquivo, urgente, prioritária, etc.) e

que existe algum predicado, p.ex., denominado de INFO/TEMPO-LOCAL(itl), que contenha todas as informações necessárias sobre a situação na rede, em especial, sobre os tempos envolvidos em relação aos diversos enlaces.

Assim, o cálculo de $\hat{t}ds$ se faria, p.ex., usando uma fórmula do tipo

$$\hat{t}ds = \hat{f}_{tds}(\hat{a}rg,itl) \ ,$$

que apareceria, em termos do modelo número 2 da figura V.30, na transição 6B-2.

VI.3.c.2 - O modelo parcial em torno do "início" da subtarefa.

Tendo em vista o modelo geral da figura VI.3, podemos dizer que a transição 6B-2 da figura V.30 pode ser considerada como sendo o início da subtarefa. É nela, onde se deve determinar o que será o argumento $\hat{a}rg$ e o tempo determinado $\hat{t}ds$.

Já tendo as precondições básicas, contidas nos predicados SAÍDA(m,em,re), PS/TRANSM(em,p) e ENLACE/ATIVADO(enl), precisamos somente adicionar as precondições em relação ao cálculo do tempo determinado $\hat{t}ds$ e a respeito do elemento para bloquear o resultado da contagem regressiva. Faremos isto através dos predicados INFO/TEMPO-LOCAL(itl) e POOL/BLOQUEADOR-LOCAL(bl).

Falta somente um pequeno detalhe, ou seja, precisamos ainda de uma identificação para saber a que vértice intermediário estes elementos serão associados. Assim, usaremos o argumento v_i ($v_i \in V$) para identificar em que vértice intermediário precisamos a informação sobre o TEMPO (p.ex., tempo de transmissão, tempo de processamento, etc.), resultando num predicado

$$\text{INFO/TEMPO-LOCAL}(itl,v_i) \ .$$

O bloqueador, cuja função é similar às "permissões" (p.ex., o crédito, UCP, etc.) será identificado através de uma indexação do tipo bl_{vi}. O predicado em questão ficaria, então, como

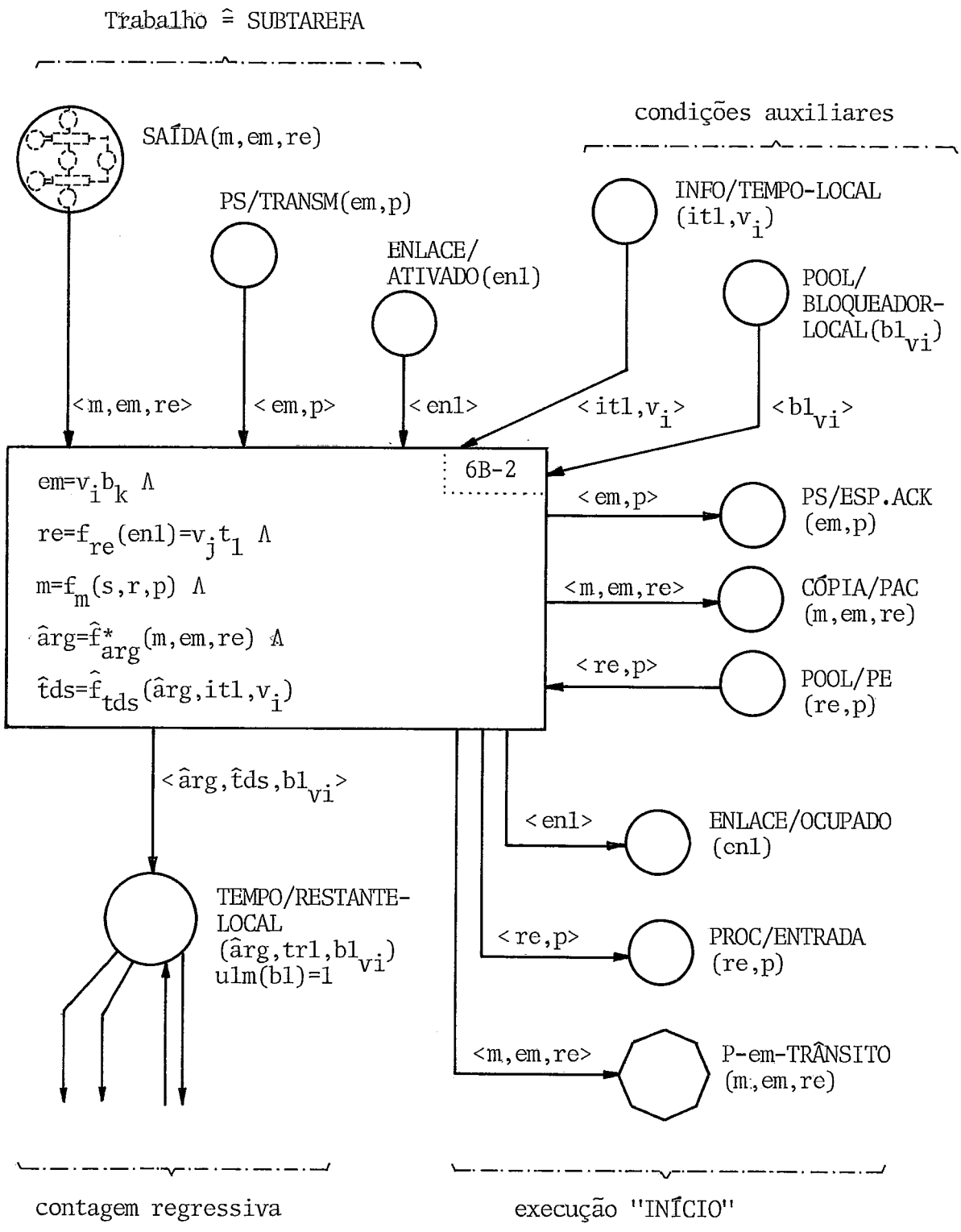


Fig. VI.4: Modelo parcial em torno do "início" da subtarefa, ou seja, no início da transmissão intervértice de um pacote simples.

POOL/BLOQUEADOR-LOCAL(bl_{vi}) .

As poscondições básicas ficam como estão; falta somente acrescentar a poscondição em relação ao contador regressivo do tempo restante, sendo ele iniciado, obviamente, com o tempo predeterminado. Assim, temos mais um predicado em torno da transição 6B-2, a saber,

TEMPO/RESTANTE-LOCAL(ârg,trl,bl_{vi}) .

O resultado destas idéias pode ser visto na figura VI.4.

VI.3.c.3 - O modelo parcial em torno do "fim" da subtarefa.

Pensando sobre o que obtivemos como modelo número 2 no capítulo V, podemos adaptar este resultado para "criar" um predicado que indica que a execução da subtarefa terminou (isto é, o equivalente ao predicado EXECUÇÃO/FIM(res,bl) da figura VI.3).

Então, para facilitar esta modelagem, achamos melhor trabalhar, ao invés de com dois predicados (PCA/TRÂNSITO e PCN/TRÂNSITO), com um único predicado que representaria o fim da execução. Este, assim, representa a transmissão de algum "pacote de controle do tipo confirmação", seja ele afirmativo ou negativo:

PCC/TRÂNSITO(pcc,emc,rec) .

Assim, as transições 7A e 7B da figura V.30 deverão conter, adicionalmente, as fórmulas pcc=pcn e pcc=pca, respectivamente.

Agora, na recepção usaremos, além de uma "nova" transição 8, também "novos" predicados com a finalidade de, por um lado, preservar a estrutura do modelo número 2 (figura V.30) e, por outro lado, obter um predicado que represente as condições necessárias a serem relacionadas com o contador regressivo, ou seja, com o 'time-out' envolvido. Então, tendo o predicado mencionado, PCC/TRÂNSITO(pcc,emc,rec), como sendo representativo de transmissão do pacote de controle, podemos, através da fórmula

da transição 8, pcc=pca $\vee$ pcn, chegar a estes novos predicados mencionados:

PCA/RECEBIDO(pca,emc,rec)

em substituição do "antigo" PCA/TRÂNSITO(pca,emc,rec), e

PCN/RECEBIDO(pcn,emc,rec)

em substituição ao PCN/TRÂNSITO(pcn,emc,rec).

O predicado relacionado ao 'time-out' podemos chamar de

SUBTAREFA/FIM(esf,bl_{vi}) ,

onde o argumento, que indica que a "execução da subtarefa foi finalizada", esf=f_{esf}(pcc,emc,rec), foi determinado na transição 8.

Veja estas primeiras idéias para determinar o fim da execução na figura VI.5 e observe, também, nas transições 8A e 8B, a substituição das expressões pcn=NAK e pca=ACK pelas fórmulas

$$m_{NAK} = f^{-1}_{pcn}(pcn) \quad e \quad m_{ACK} = f^{-1}_{pca}(pca) \ ,$$

respectivamente.

Uma vez definido o fim da execução da subtarefa, ou seja, sabendo que a seqüência indicada na seção VI.3.c.1 foi concluída, podemos nos preocupar com a modelagem que representa a relação com o 'time-out'. Tendo, por um lado, o predicado relacionado ao tempo, ou seja, TEMPO/RESTANTE-LOCAL(ârg,n,bl_{vi}), da figura VI.4 e, por outro lado, o predicado SUBTAREFA/FIM(esf,bl_{vi}), precisamos somente adaptar a figura VI.3 neste sentido.

Confira esta adaptação na figura VI.6 e observe que temos, no lado "tarefa executada", dois casos a considerar. Trataremos estas implicações na seção VI.3.c.4.

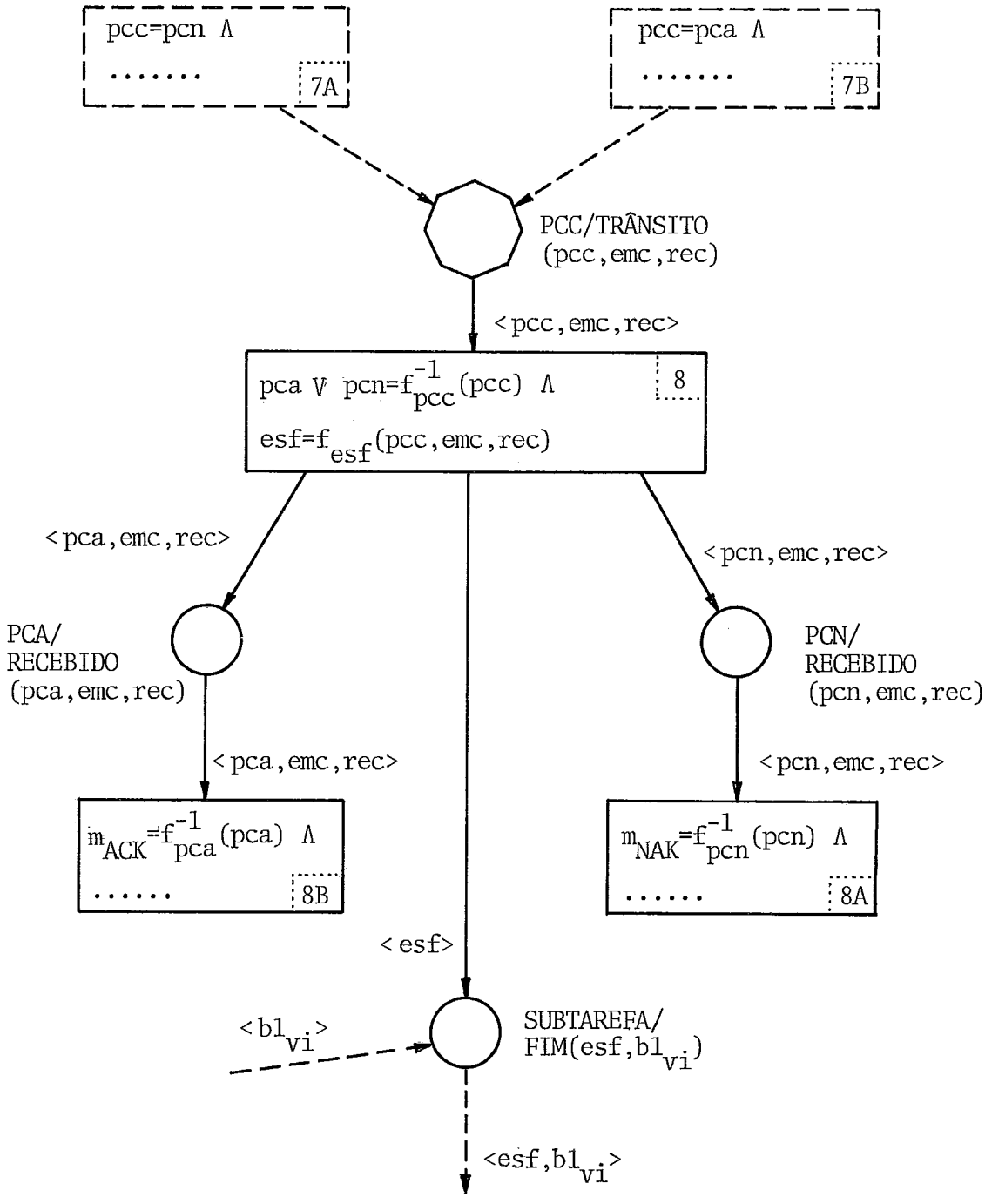


Fig. VI.5: Modelo parcial para determinar o fim da execução da subtarefa.

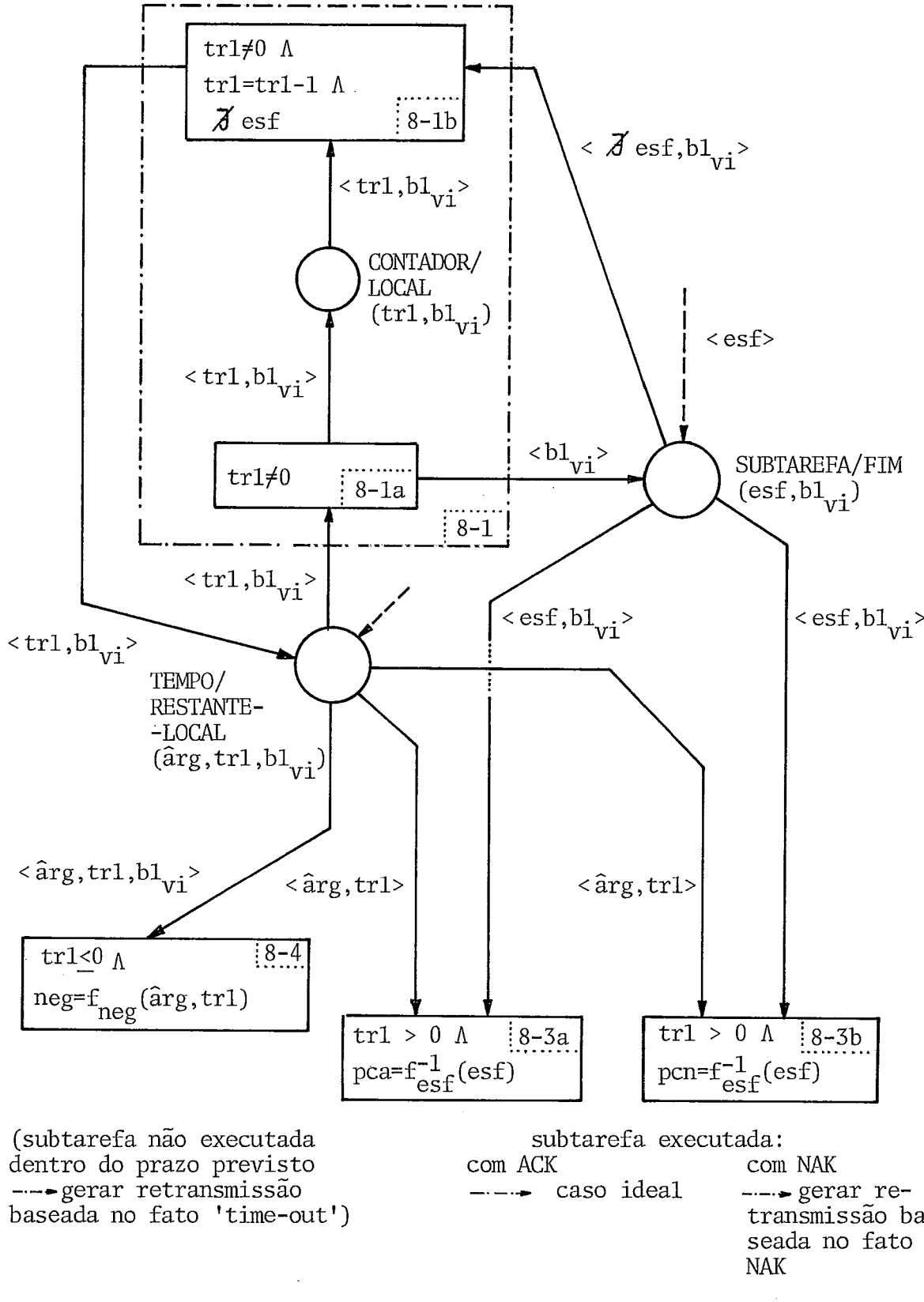


Fig. VI.6: Modelo parcial em torno do "fim da execução" mostrando a relação com o 'time-out' e indicando possíveis implicações.

VI.3.c.4 - Conseqüências em relação ao 'time-out'.

Como já indicado na figura VI.6, temos agora que avaliar as conseqüências do resultado concernente ao 'time-out'. Podemos, então, enumerar três casos distintos:

Caso 1: recepção de um ACK dentro do prazo previsto.

Podemos considerar este caso como sendo ideal, e assim desejado, porque recebemos, dentro do prazo previsto, uma confirmação que é afirmativa (ACK). Assim, via transição 8-3a, criamos a precondição para que possamos destruir a cópia do pacote (guardada no vértice/emissor) e liberar este espaço no buffer para outros pacotes. Ao mesmo tempo, permitimos que o processador de saída, $v_i b_k$, responsável pela emissão do pacote p em questão, retorne ao 'pool' e libere, assim, o "crédito" para o envio de outros pacotes (observe que, neste caso, o crédito entre dois vértices vizinhos é considerado "UM" ao que toca o processador de saída em relação a uma determinada classe de mensagens).

Veja este caso 1 na figura VI.7 e observe, também, a função $res=f_{res}(m,\hat{a}rg,m_{ACK})$, dentro da transição 8B, que tem como objetivo testar se o pca recebido realmente confirma o m que "passou" pela seqüência âarg.

Caso 2: recepção de um NAK dentro do prazo previsto.

Este caso, mesmo não sendo mais o ideal, ainda pode ser considerado como sendo aceitável. Isto se deve ao fato que o sistema indicou que houve algum problema na transmissão do pacote (na maioria dos casos este problema se resume, provavelmente, em ruídos aleatórios na linha de transmissão).

Assim, se pode tomar uma atitude corretiva com base num fato real e concreto, ou seja, iniciar uma retransmissão com a consciência que ela realmente é necessária. Veja, na figura VI.8, como se pode integrar os resultados em relação ao 'time-out', no caso de um NAK recebido, no esquema básico (modelo número 2) mos

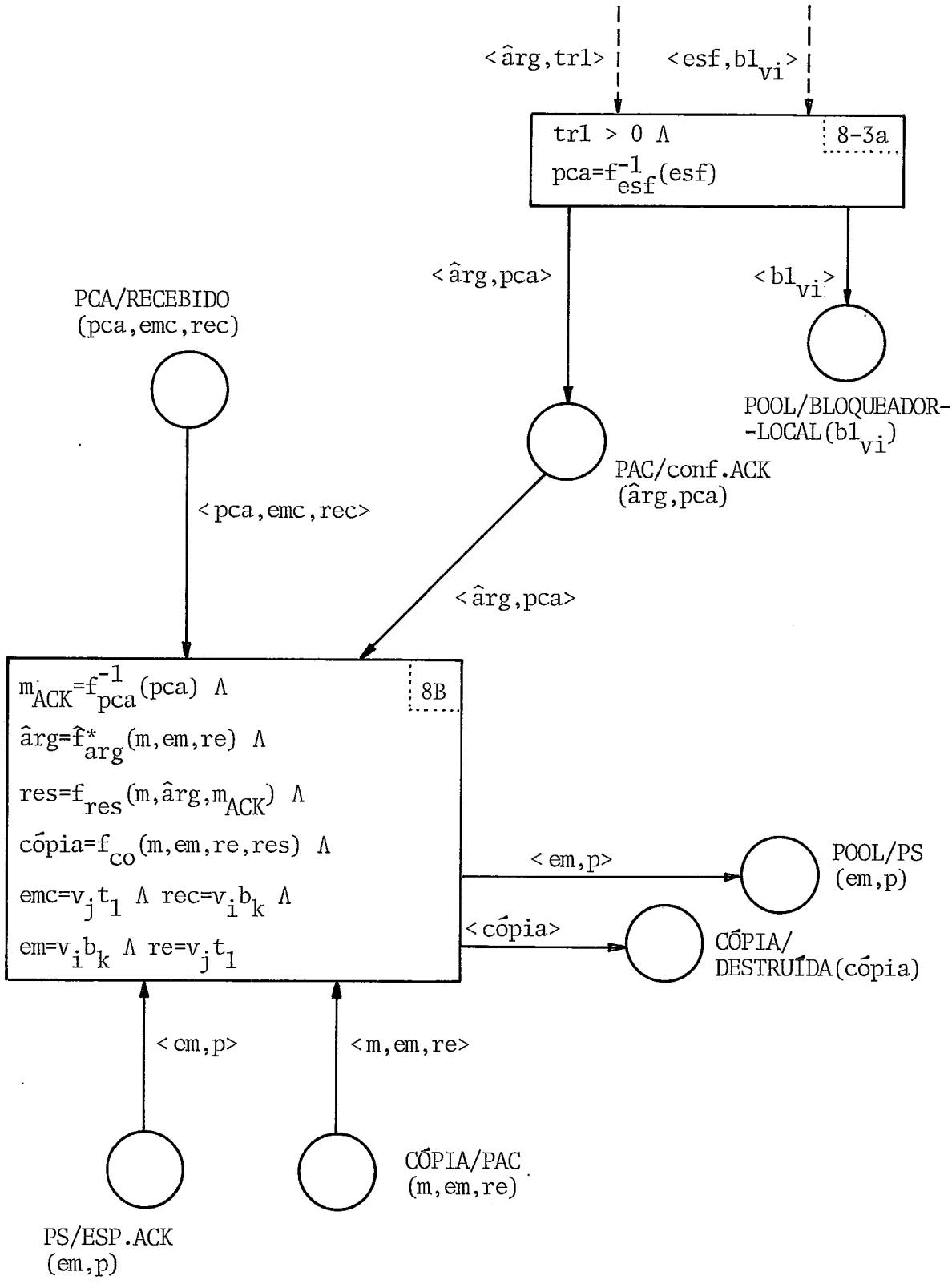


Fig. VI.7: Caso 1: conseqüências em relação à recepção de um ACK dentro do prazo previsto.

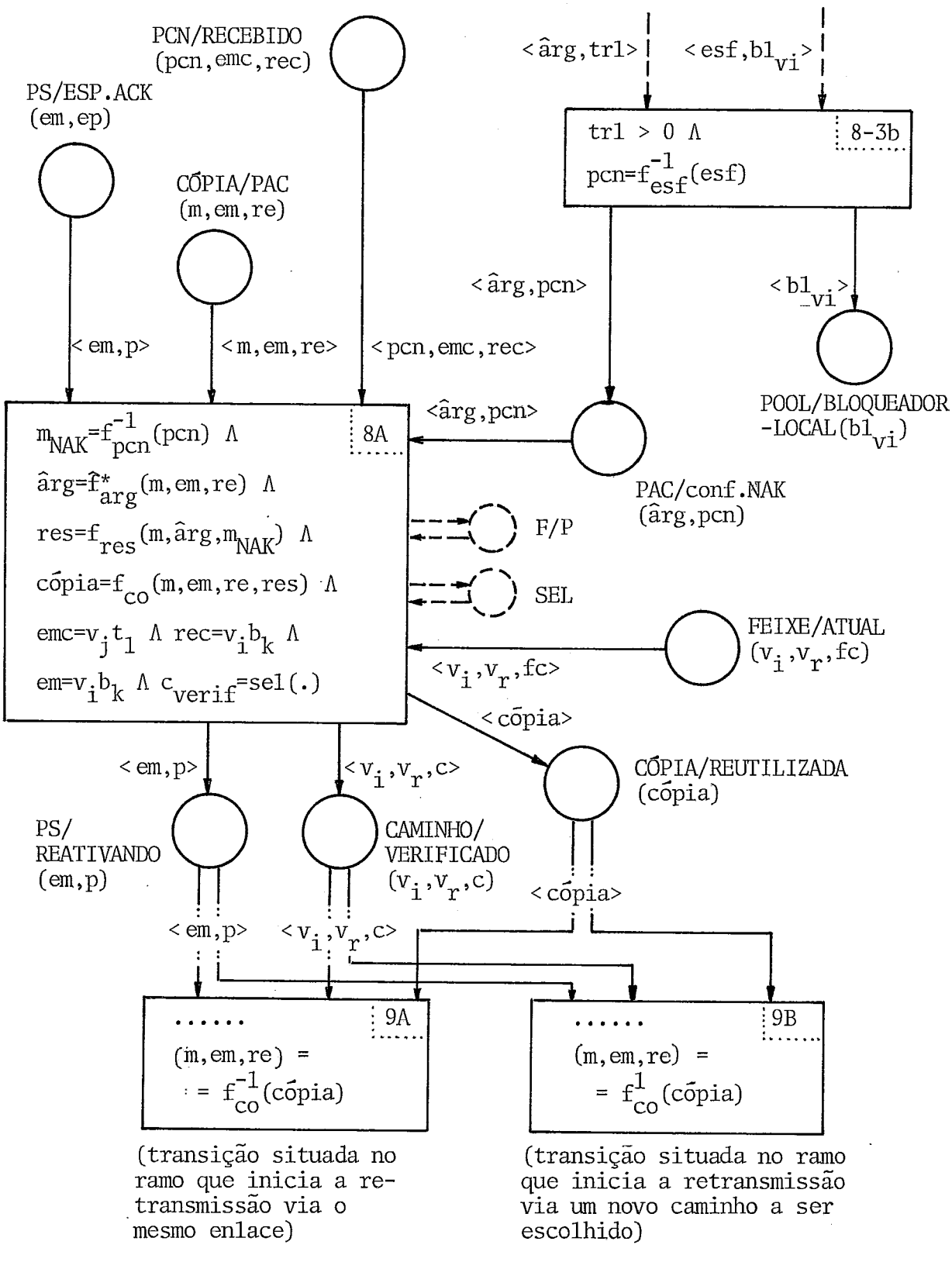


Fig. VI.8: Caso 2: conseqüências em relação à recepção de um NAK dentro do prazo previsto.

trado na figura V.30.

Caso 3: não foi recebida, dentro do prazo previsto, nenhuma confirmação (isto é, nem ACK, nem NAK).

Este caso é o pior de todos porque se deve tomar uma atitude sem saber a verdadeira razão pela qual aconteceu o 'time-out'. Assim, para evitar problemas, escolhemos o início de uma retransmissão, causada pelo 'time-out', a partir da ESCOLHA de CAMINHO (transição 6). Desta maneira podemos manter o esquema na esperança que INFO/GERAL e/ou INFO/ATUAL contenham, já, eventuais informações a respeito das situações nos enlaces usados.

Confira, na figura VI.9, a integração destas idéias no esquema do modelo número 2 (figura V.30) e, observe, que temos que "criar" novas transições (8C e 9C) para permitir o uso do esquema da escolha de um caminho.

VI.3.c.5 - Problemas, em relação ao 'time-out', ainda não incluídos no modelo.

Iniciar uma retransmissão por causa de um NAK é uma ação consciente porque está se baseando numa informação "real". Já a retransmissão a partir de um 'time-out' é uma ação que está baseada numa suposição, embora o fato de que o tempo predeterminado foi ultrapassado, seja algo real.

Então, o que acontece se o tempo já se esgotou e recebemos a confirmação, digamos, um ACK, em relação ao pacote em questão, fora deste prazo? A resposta é simples: vamos retransmitir um pacote, já transmitido com sucesso, mais uma vez, ou seja, que existem agora dois pacotes, em vez de um único, que competem pelos recursos da subrede. Isto diminuirá certamente o desempenho da rede porque somente o usuário/receptor perceberá esta duplicação, já que os vértices intermediários do tipo S/F, que compõem a subrede da comunicação, não têm "memória neste sentido". Pode-se, eventualmente, testar os pacotes no vértice/destinação, mas isto traz um outro inconveniente, a saber, quanto tempo se deve

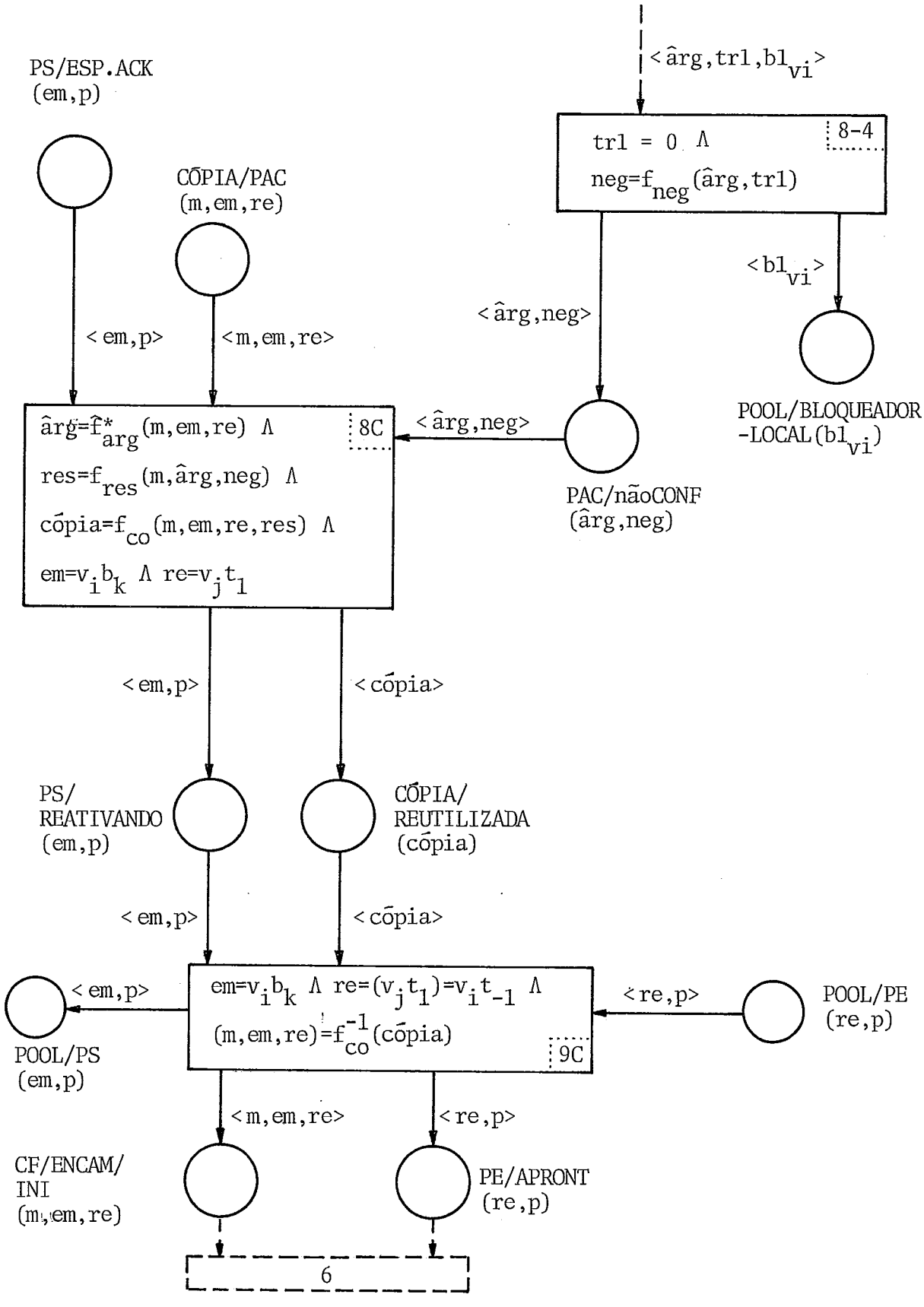


Fig. VI.9: Conseqüências em relação ao não-recebimento de nenhuma confirmação dentro do prazo previsto.

esperar para ter certeza que um pacote não será duplicado.

Vimos, então, que o nosso modelo está ainda longe de ser definitivo só pensando que, além do problema mencionado, o modelo não considera, por exemplo, um pacote de controle de confirmação que chegou atrasado. Devemos, assim, numa etapa posterior a este trabalho, investigar todos estes problemas sempre se baseando em suposições, hipóteses, simulações e experiências reais. Será este o próximo desafio, ou seja, testar e provar o modelo até que seja consistente em relação aos acontecimentos físicos e que não hajam nenhum 'deadlock' ou situações indesejadas nele.

VI.3.d - Tempo envolvido no controle 'end-to-end' ('check-time' no nível V/F-V/D).

VI.3.d.1 - Idéias gerais.

Da mesma maneira que se pode medir ou estimar o tempo necessário para enviar um pacote de um vértice S/F para o vizinho, e receber uma confirmação (ACK ou NAK) associada, podemos tratar o tempo envolvido na passagem, e confirmação, de uma mensagem entre um vértice/fonte (V/F) e um vértice/destinação (V/D). Lembrando do esquema simplificado (modelo número 1 no capítulo IV, figura IV.9), poderíamos pensar em usar, eventualmente, até o mesmo esquema que foi usado para modelar o 'time-out' para o controle intervértice. Isto é, em princípio, válido, só que estimar o tempo correto necessário (isto é, o tempo equivalente ao 'time-out' entre vértices vizinhos) é um empreendimento bastante difícil porque não sabemos, neste caso, nem o que aconteceria depois da "entrega" da mensagem empacotada à subrede.

Seguindo um raciocício similar aquele usado na seção VI.3.c, devemos, primeiramente, definir que a descrição da TAREFA deve constar da seguinte seqüência básica:

A) transmissão (emissão e recepção) da mensagem empacotada entre o vértice/fonte e o vértice/destinação;

B) pré-teste no vértice/destinação e entrega ao host/destinação (usuário/receptor);

C) teste final no H/D, seguida (se tudo for correto) da preparação de uma mensagem de controle RFNM ('request for next message') e colocação do RFNM no V/D;

D) transmissão (emissão e recepção) deste RFNM entre V/D e V/F.

Se existe interesse em saber o que acontece numa transmissão (pontos A e D), pode-se observar a seguinte seqüência:

a) processamento necessário para calcular o caminho;

b) passar a mensagem empacotada para o canal de saída associado a este caminho;

c) transmissão entre vértices vizinhos;

d) receber, testar e, eventualmente, retransmissão local;

e) repetir de a) até d) até chegar ao vértice/destinação.

Assim, pensando na modelagem, temos um predicado inicial, TAREFA(d̂es, t̂dt), cujo argumento "descrição" d̂es (d̂es ε D̂ES), onde D̂ES = {$\hat{des}_1$, $\hat{des}_2$, ...} , especificaria todos estes acontecimentos. Felizmente, pelo menos em relação a este primeiro argumento, precisamos somente definir o início da tarefa (isto é, quando a subrede "assume" a responsabilidade sobre a mensagem empacotada) e fim desta seqüência (ou seja, quando recebemos o RFNM em questão no vértice/fonte).

Já com o outro argumento, t̂dt (tempo determinado para tarefa), não temos tanta sorte. Isto se deve ao fato que, a priori, não sabemos, além da destinação, nada sobre o trajeto do pacote, nem, obviamente, sobre os tempos envolvidos nestas transmissões (e, eventualmente, retransmissões) entre vértices vizinhos. Também não sabemos nada específico sobre os tempos do processamento envolvidos (p.ex., escolher as rotas, programas de teste, etc.) ou se aconteceu algum atraso imprevisto no atendimento em relação às filas de entrada e saída. Então, o jeito é usar, como no

caso de 'time-out' para o controle intervértice, algum predicado, digamos, o predicado INFO/TEMPO-GLOBAL(itg), para poder estimar, da melhor maneira possível, o tempo determinado para executar a tarefa. Uma fórmula do tipo $\hat{t}dt=\hat{f}_{tdt}(\hat{d}es,itg)$ ajudará, assim, a determinar este segundo argumento no predicado TAREFA ($\hat{d}es,\hat{t}dt$).

VI.3.d.2 - A inclusão destas idéias no modelo número 2, obtido no Capítulo V.

Usando como guia o raciocínio usado para especificar o argumento $\hat{a}rg$ no predicado SUBTAREFA($\hat{a}rg$,tds), podemos esperar que exista uma função especial que consiga estabelecer a descrição mais provável da tarefa a ser executada: $\hat{d}es=\hat{f}^{*}_{des}(s,r,p)$.

Associado a esta função pode se prever o tempo predeterminado para esta tarefa, considerando basicamente a descrição provável $\hat{d}es$ e as informações sobre os tempos em geral, itg. A função usada, $\hat{t}dt=\hat{f}_{tdt}(\hat{d}es,itg)$, acrescenta ainda as margens de erros em relação à estimativa inicial e, também, os erros com respeito às alternativas no caminho. Os tempos necessários para eventuais retransmissões devem ser incluídos, até um certo limite, neste acréscimo.

Observando o modelo número 2, figura V.30, vemos que o início da tarefa acontece na transmissão 2 onde se começou a espera em relação ao RFNM. Incluindo nela as duas fórmulas mencionadas, obtemos o modelo parcial em relação ao início da tarefa. Veja este resultado na figura VI.10 e observe, ainda, o acréscimo dos predicados INFO/TEMPO-GLOBAL(itg), POOL/BLOQUEADOR-GLOBAL(bg_s) e TEMPO/RESTANTE-GLOBAL($\hat{d}es$,trg,bg_s).

O final da tarefa é alcançado quando recebemos o RFNM. Mesmo que a geração deste RFNM seja somente modelada de uma maneira simplificada, podemos usar o predicado MC/TRÂNSITO (o equivalente ao predicado MCA/TRÂNSITO na figura V.30) como sendo representativo do final de tarefa. A observação sobre a modelagem aproximada do RFNM se refere ao fato que não explicamos, no capítulo V,

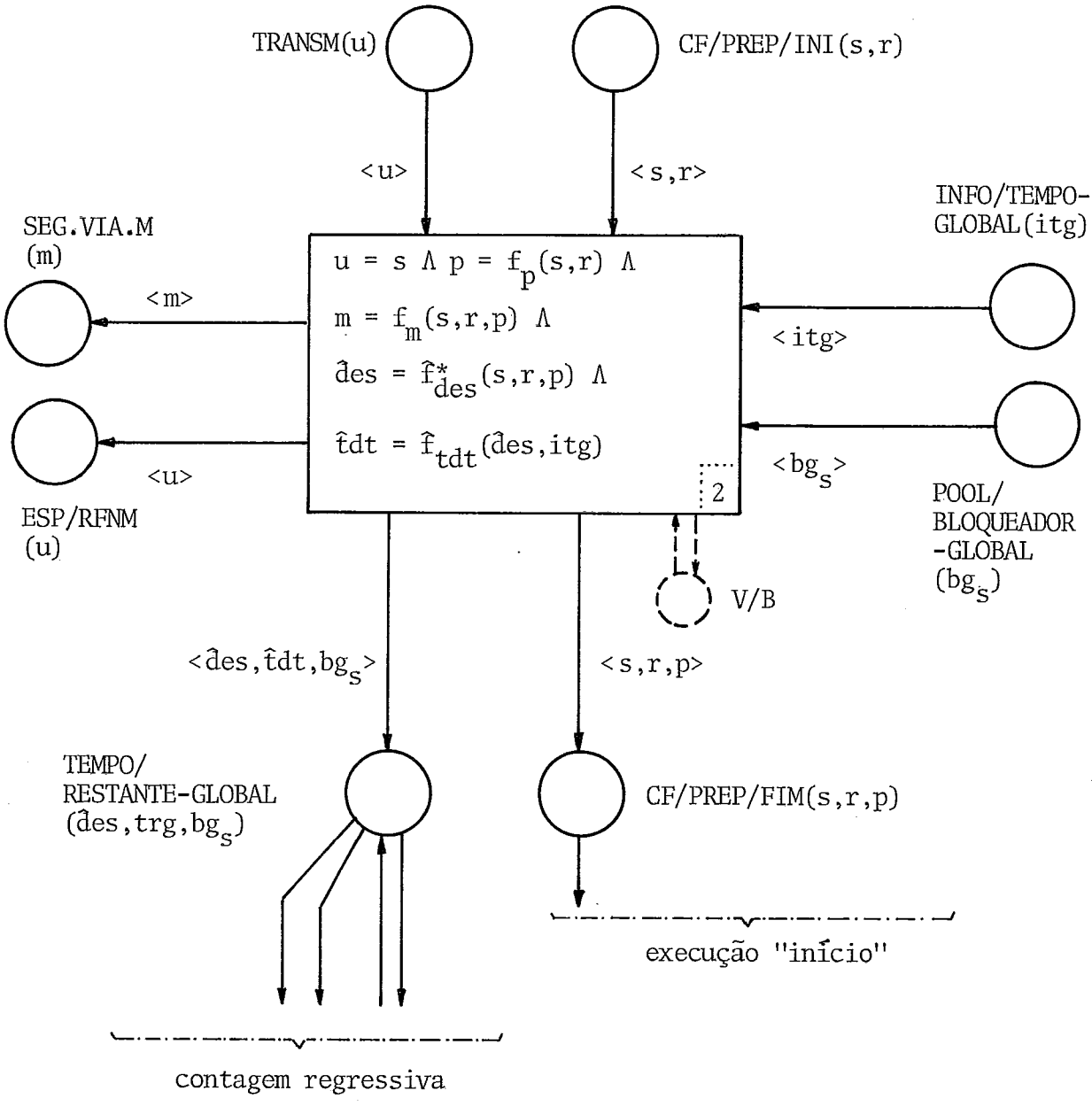


Fig. VI.10: Modelo parcial em torno do início da tarefa, ou seja, do início da transmissão entre V/F e V/D de uma mensagem empacotada.

que a função $mca=f_{mca}(s,r)$, na transição 3D, deveria ser representativa de entrega da mensagem ao H/D e pela sua resposta em relação à mensagem, ou seja, que a mensagem original saiu da subrede e que o RFNM é uma "nova" mensagem que vem de "fora".

Mostraremos a modelagem do fim da tarefa na figura VI.11 e aproveitamos a oportunidade para modificar o modelo número 2 (figura V.30) no que diz respeito às observações sobre o RFNM.

Uma vez determinado o fim da tarefa, pode-se modelar o resultado da comparação com a contagem regressiva. Este resultado pode ser negativo, se a tarefa não foi executada no prazo previsto, ou positivo, se foi recebido um RFNM (*m*ensagem de controle *a*firmativa, mca) ou, ao menos, uma *m*ensagem de controle *q*ualquer, mcq.

Estas indicações podem ser vistas na figura VI.12.

VI.3.d.3 - Conseqüências em relação ao 'check-time' no nível V/F-V/D.

Na figura VI.12 indicamos, em princípio, três possíveis conseqüências em relação ao 'check-time' neste nível entre vértices fonte e destinação. Porém, o tratamento destes três casos não é mais tão simples como no caso de 'time-out' no nível intervértice onde a questão era simplesmente se retransmitimos ou não. Mesmo tendo omitido os problemas em relação a retransmissões sucessivas, pode-se dizer que a decisão a ser tomada era bastante simples. Isto se deveu ao fato que somente se testou se a transmissão de um único pacote era correta ou não, e, também, que se tratou somente de um único enlace envolvido.

Agora, neste caso do 'check-time' no nível V/F-V/D, temos um número de problemas envolvidos que é consideravelmente maior que no caso anterior; podemos mencionar, por exemplo, que a mensagem empacotada pode consistir de um ou vários pacotes, que cada um destes pacotes pode tomar um caminho diferente, que temos mais que um enlace envolvido, que se trata não somente de testar se a

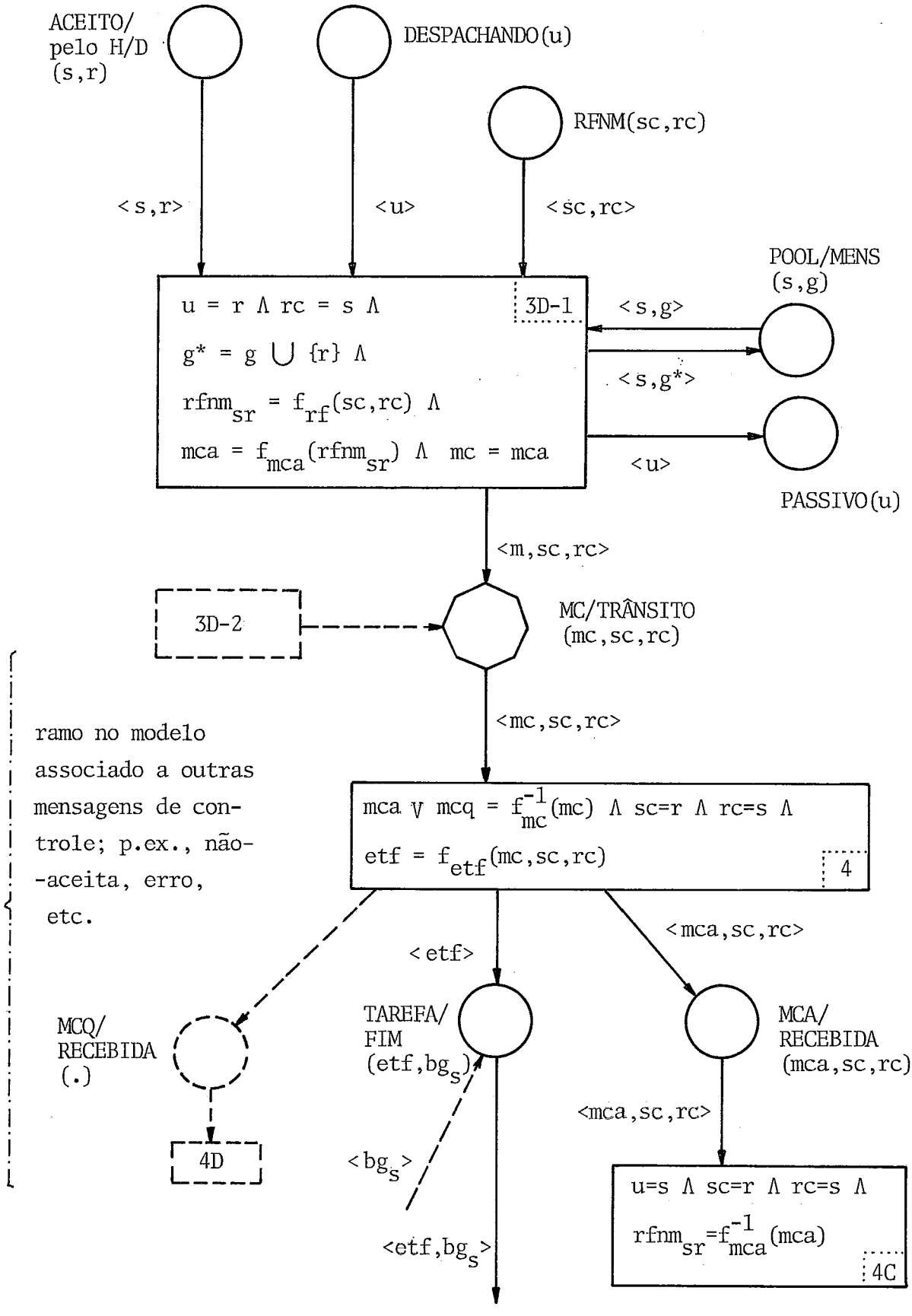


Fig. VI.11: Modelo parcial em torno do fim da execução da tarefa para determiná-lo.

transmissão foi correta ou não, mas, também, de verificar a consistência da mensagem, etc. Para evitar, neste momento, um desvio tão grande de nosso objetivo imediato, que é de construir um modelo básico de problemas associados principalmente à subrede de comunicação, deixamos estas questões, que envolvem já em parte o nível do usuário (como, por exemplo, no caso de testar a consistência da mensagem), para uma outra ocasião.

Então, depois deste registro de nossas preocupações, podemos dirigir a nossa concentração às conseqüências em relação ao caso em que tivermos recebido um RFNM, o que representa, aliás, o caso ideal (transmissão correta e dentro do prazo).

Assim, podemos observar, na figura VI.13, que a recepção desta mensagem de controle afirmativa significa não somente a liberação do espaço na memória ocupada pela cópia da mensagem, mas, também, a atualização do crédito. Para fugir ainda, neste momento, do problema de como modelar o crédito (o que envolve, para garantir o bom desempenho da rede, estimativas iniciais e atualizações contínuas delas), ficamos com um crédito igual a "UM", o que significa simplesmente a abertura da janela a partir do momento do recebimento do RFNM e posterior teste a respeito desta mensagem de controle para que o responsável pela transmissão das mensagens do usuário possa enviar a próxima mensagem.

VI.4 - O modelo número 3, incluindo o controle de tempo, da transmissão de um pacote simples numa rede de computadores.

VI.4.a - O modelo número 3 na sua forma gráfica.

Mostraremos, na figura VI.14, o modelo número 3, resultante do estudo neste capítulo VI, que inclui, agora, também as verificações em relação ao 'time-out' e o 'check-time'. Como base usamos o modelo número 2, obtido por SCHWARZ (35) e repetido no capítulo V, figura V.30, e tentamos conservar a separação em partes distintas. Assim, temos:

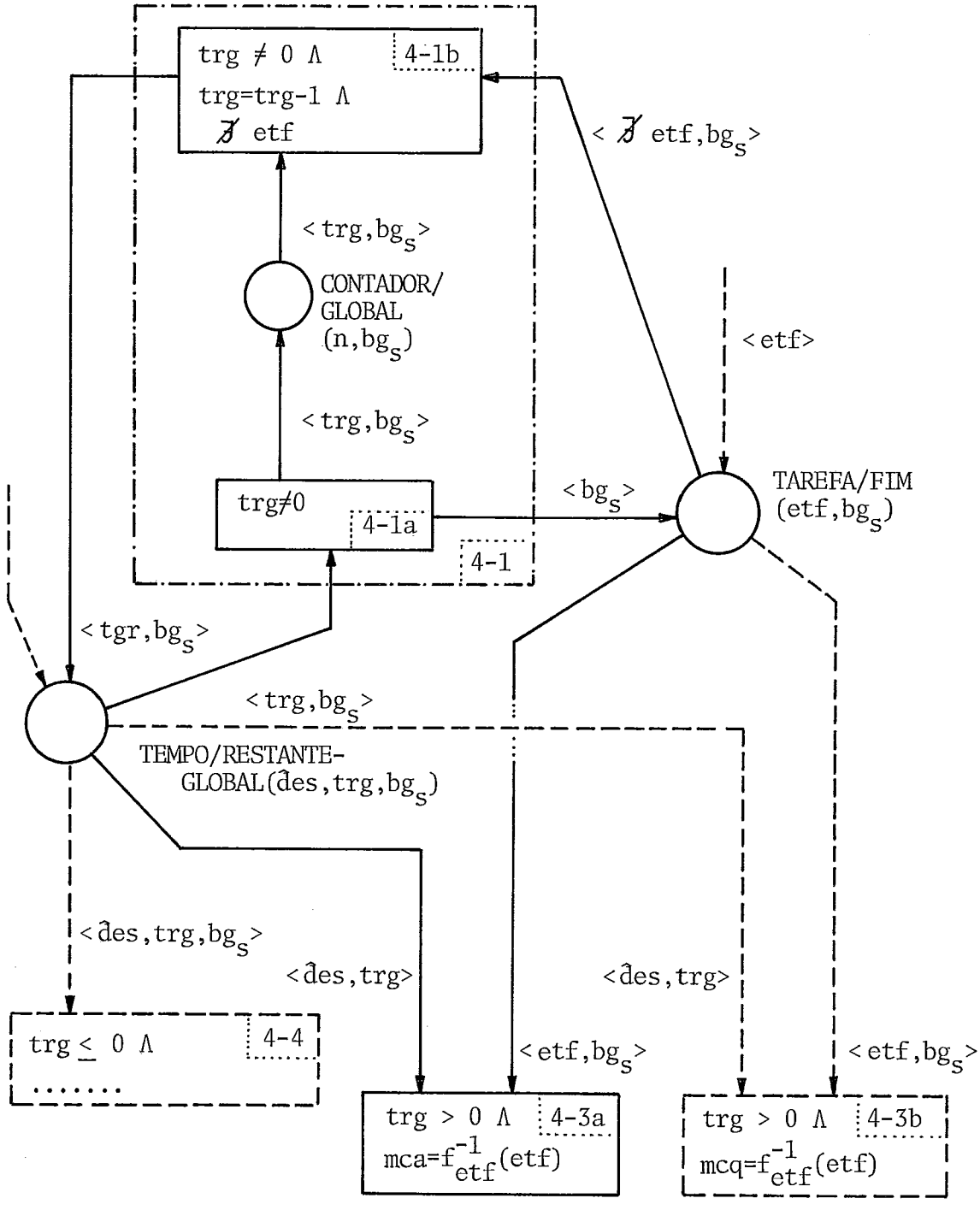


Fig. VI.12: Modelo parcial em torno do fim da execução, mostrando a relação com o 'check-time' no nível V/F-V/D.

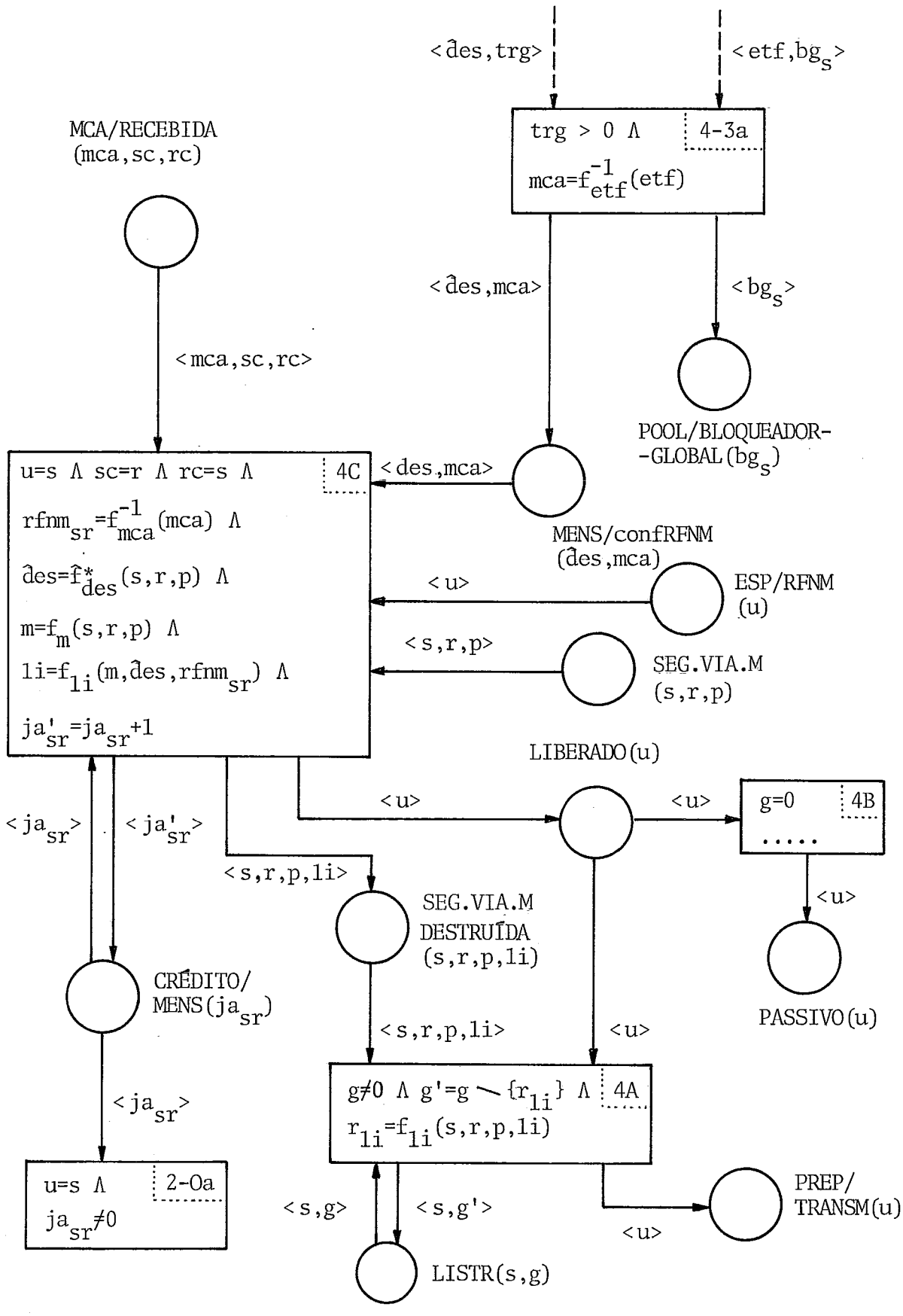


Fig. VI.13: Conseqüências em relação ao recebimento de um RFNM dentro do 'check-time' previsto no nível vértice/fonte-destinação.

- Fig. VI.14.a: Emissão inicial em torno do vértice/fonte;
- Fig. VI.14.b: Integração do 'check-time';
- Fig. VI.14.c: Emissão nos vértices intermediários;
- Fig. VI.14.d: Recepção bem sucedida nos vértices intermediários;
- Fig. VI.14.e: Retransmissão a partir dos vértices intermediários;
- Fig. VI.14.f: Recepção no vértice/destinação.

Observando o modelo obtido, vemos que o número de detalhes aumentou bastante, mesmo já tendo suprimido alguns detalhes menos importantes. Isto resultou, primeiramente, numa representação gráfica muito sobrecarregada (o que poderia ser evitado usando um outro formato ou uma outra divisão em partes) e, segundo, numa "fonte de erros" se insistirmos em continuar a manipulação manual deste modelo.

Então, se queremos evitar este segundo problema, devemos pensar em introduzir o modelo num computador para facilitar as modificações (p.ex., melhoramentos, etc.) do modelo, mas, sobretudo, para poder investigar a consistência e a dinâmica do modelo. Seria isto um dos próximos trabalhos a fazer.

VI.4.b - O modelo em forma de lista.

Pensando na apresentação num computador, precisamos colocar o modelo gráfico numa forma mais facilmente compreensível por um sistema automatizado. Assim, usando os conceitos introduzidos no capítulo III, sobretudo a matriz de incidência mencionada, resolvemos continuar usando (como fizemos também em relação aos modelos número 1 e número 2) um tipo de lista baseada nesta matriz de incidência.

Observe, então, na figura VI.15, esta lista que mostra os predicados, agrupados segundo sua "natureza" (p.ex., em relação a mensagens, ao usuário, etc.), e, na figura VI.16, as transições

e suas respectivas fórmulas.

Queremos alertar, ainda, que, às vezes, o modelo (a representação gráfica e o seu equivalente na forma de lista) parece ter algumas inconsistências. Isto se deve ao fato de que, por causa do espaço físico restrito, nem sempre colocamos todos os predicados e fórmulas neste modelo número 3. Assim, para poder verificar a consistência do modelo, deve se procurar as figuras que mostram modelos parciais e que forneceram a base para este modelo número 3.

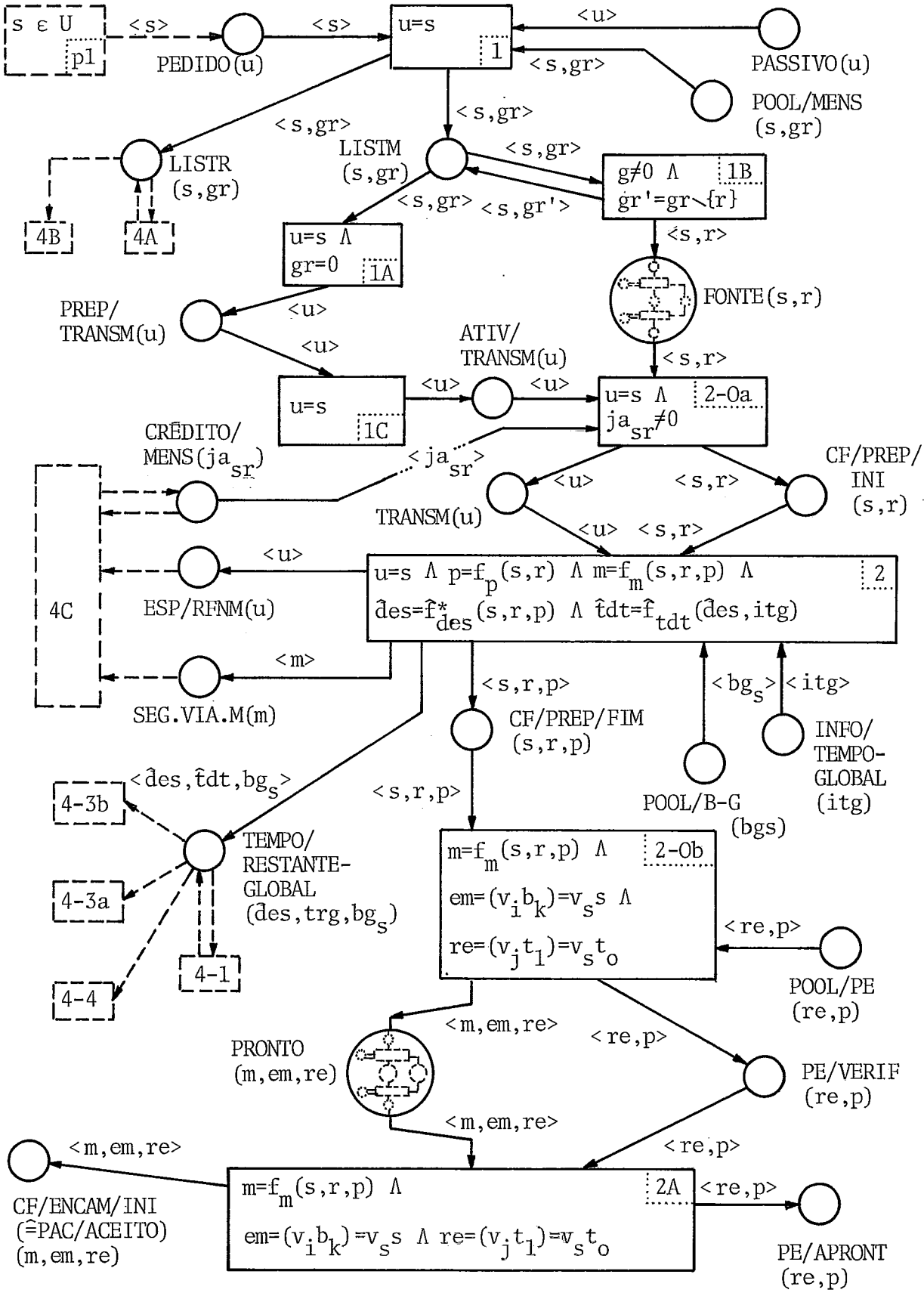


Fig. VI.14.a: Modelo número 3 (transmissão de um pacote simples em redes de computadores, incluindo vértices intermediários, ACK, NAK, 'time-out' e 'check-time'. Parte 1: emissão inicial (baseado nas figuras V.30.a, VI.10, VI.13).

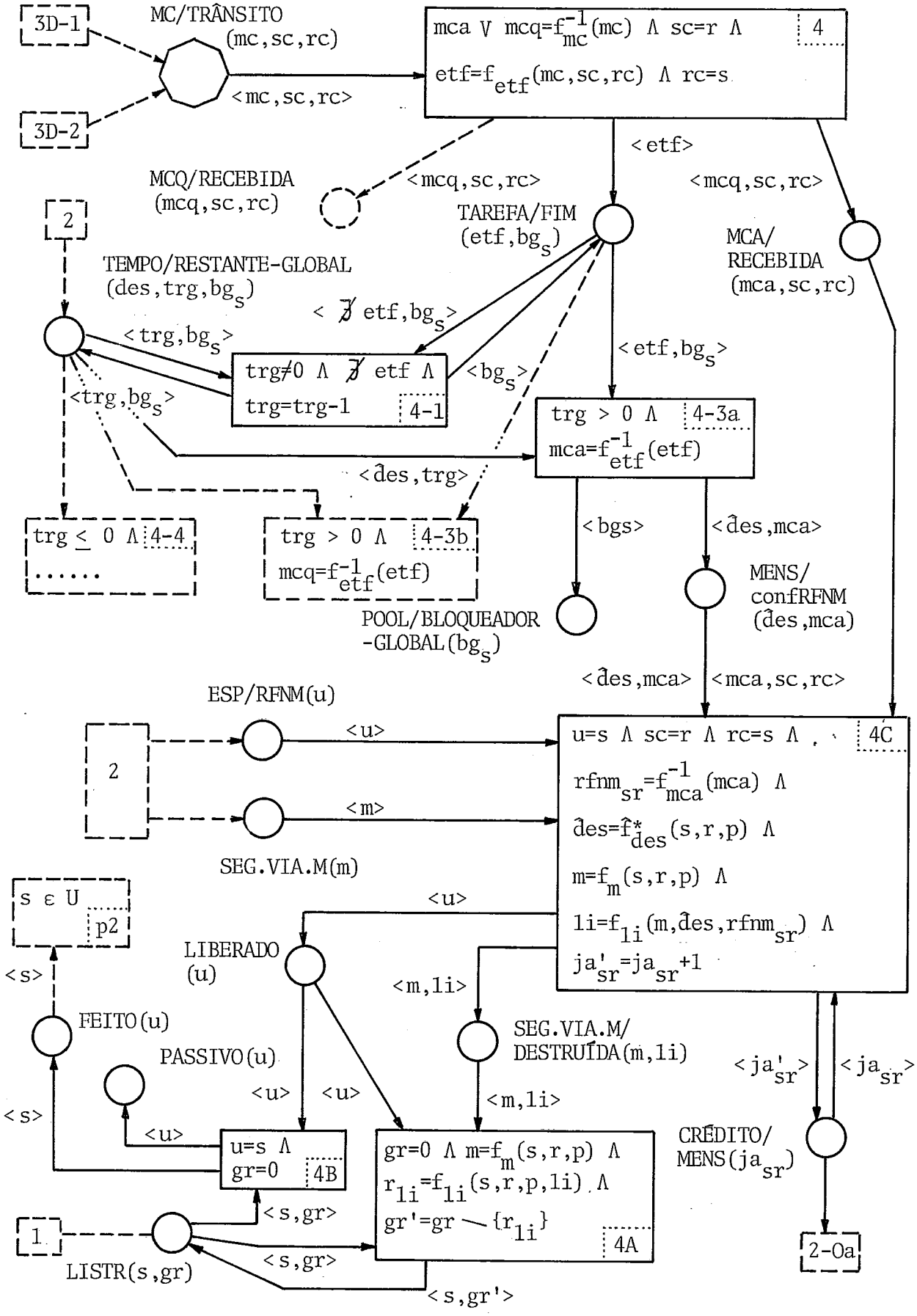


Fig. VI.14.b: Modelo número 3. Parte 2: integração do 'check-time' e resposta ao usuário (baseado nas figs. V.30.a, VI.11, VI.12, VI.13).

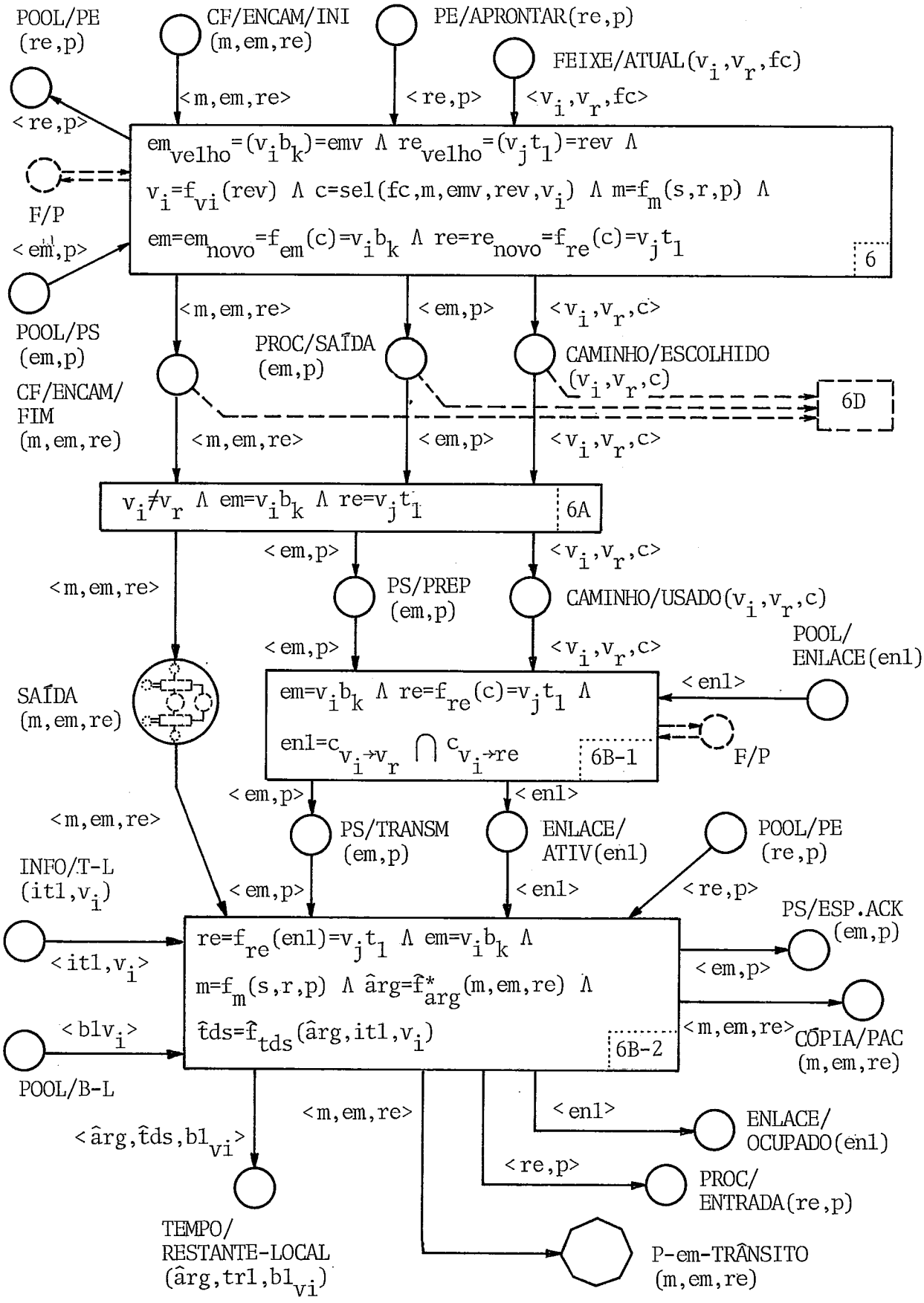


Fig. VI.14.c: Modelo número 3. Parte 3: emissão nos vértices intermediários (baseado nas figuras V.30.b e VI.4).

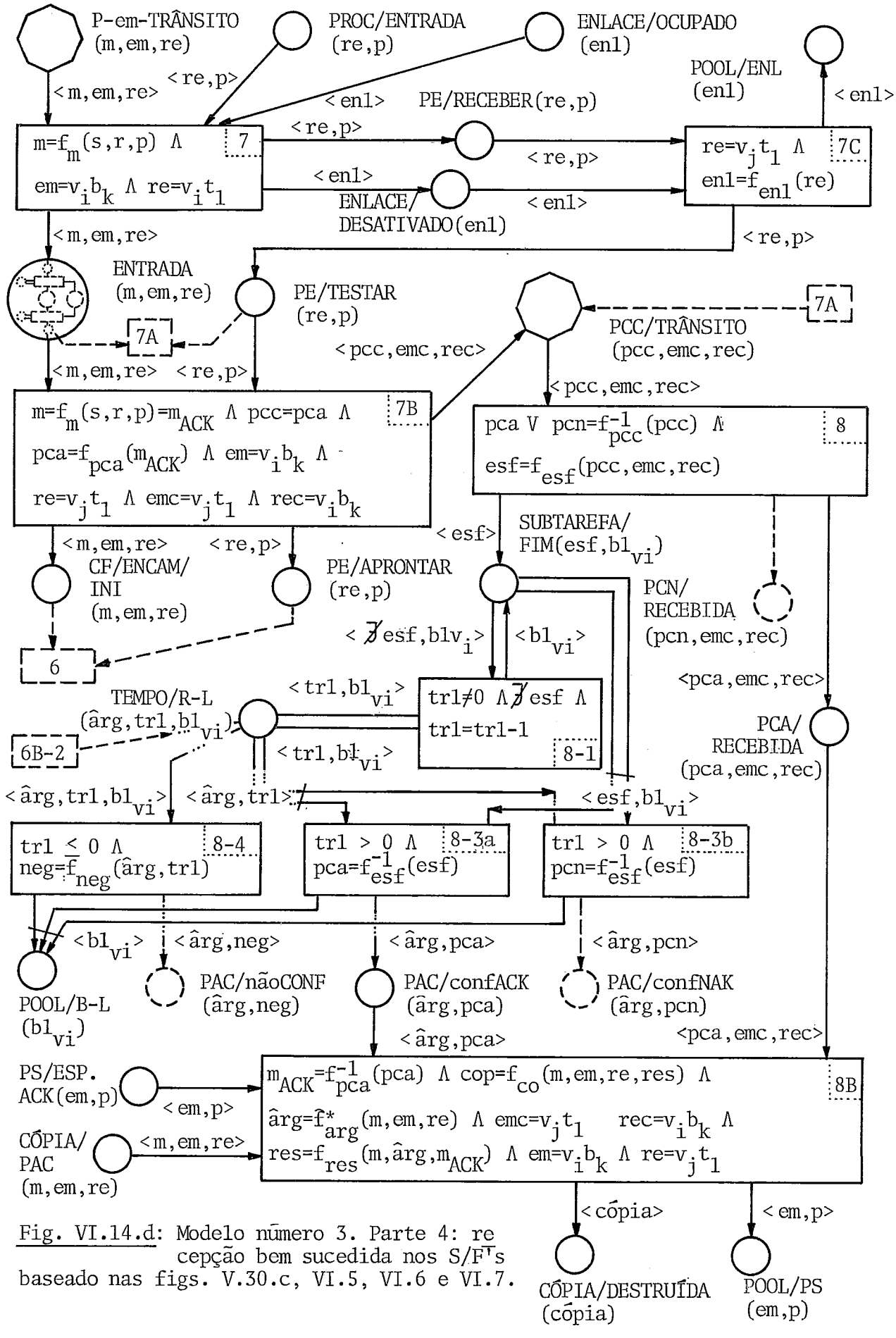


Fig. VI.14.d: Modelo número 3. Parte 4: recepção bem sucedida nos S/F's baseado nas figs. V.30.c, VI.5, VI.6 e VI.7.

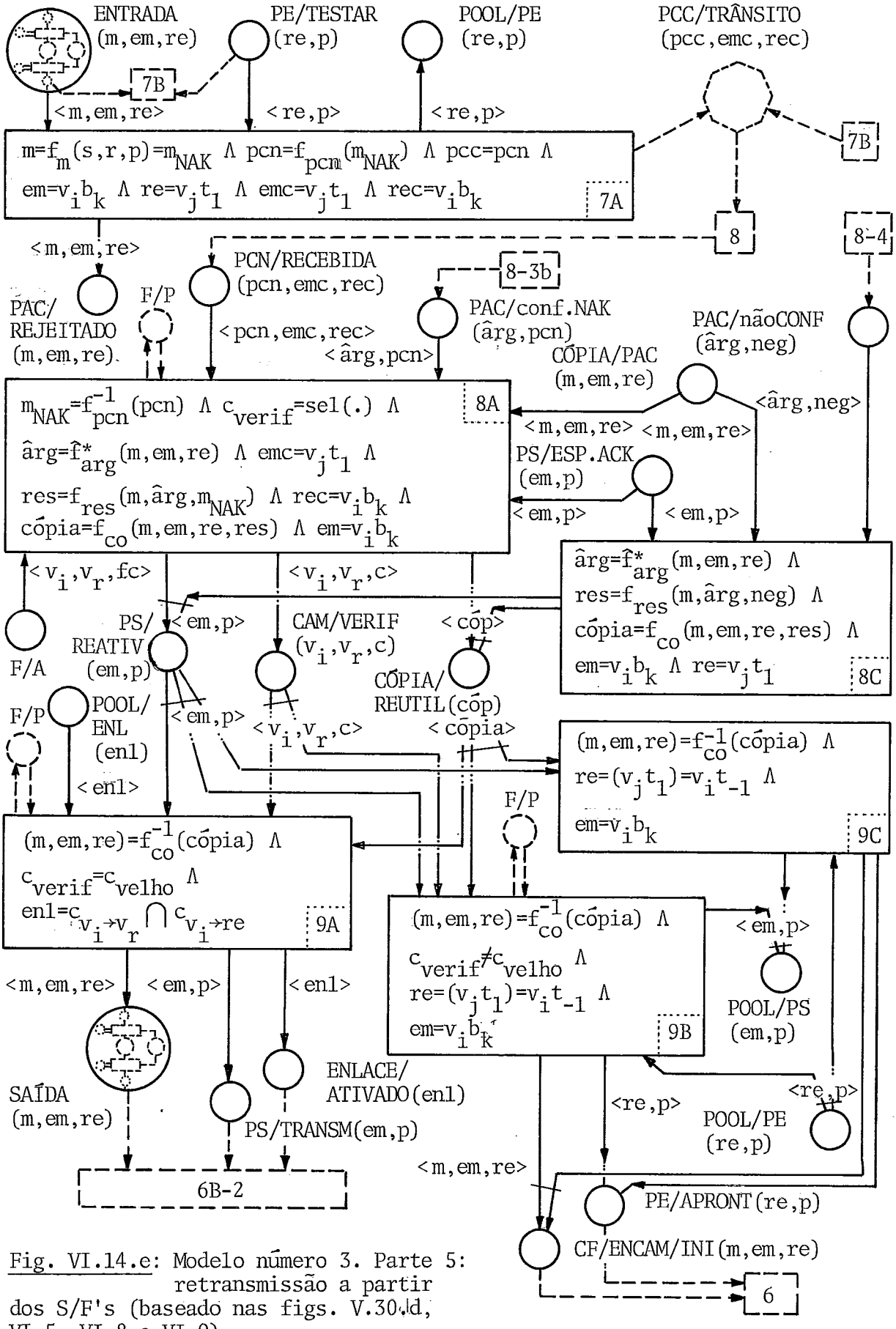


Fig. VI.14.e: Modelo número 3. Parte 5: retransmissão a partir dos S/F's (baseado nas figs. V.30.d, VI.5, VI.8 e VI.9).

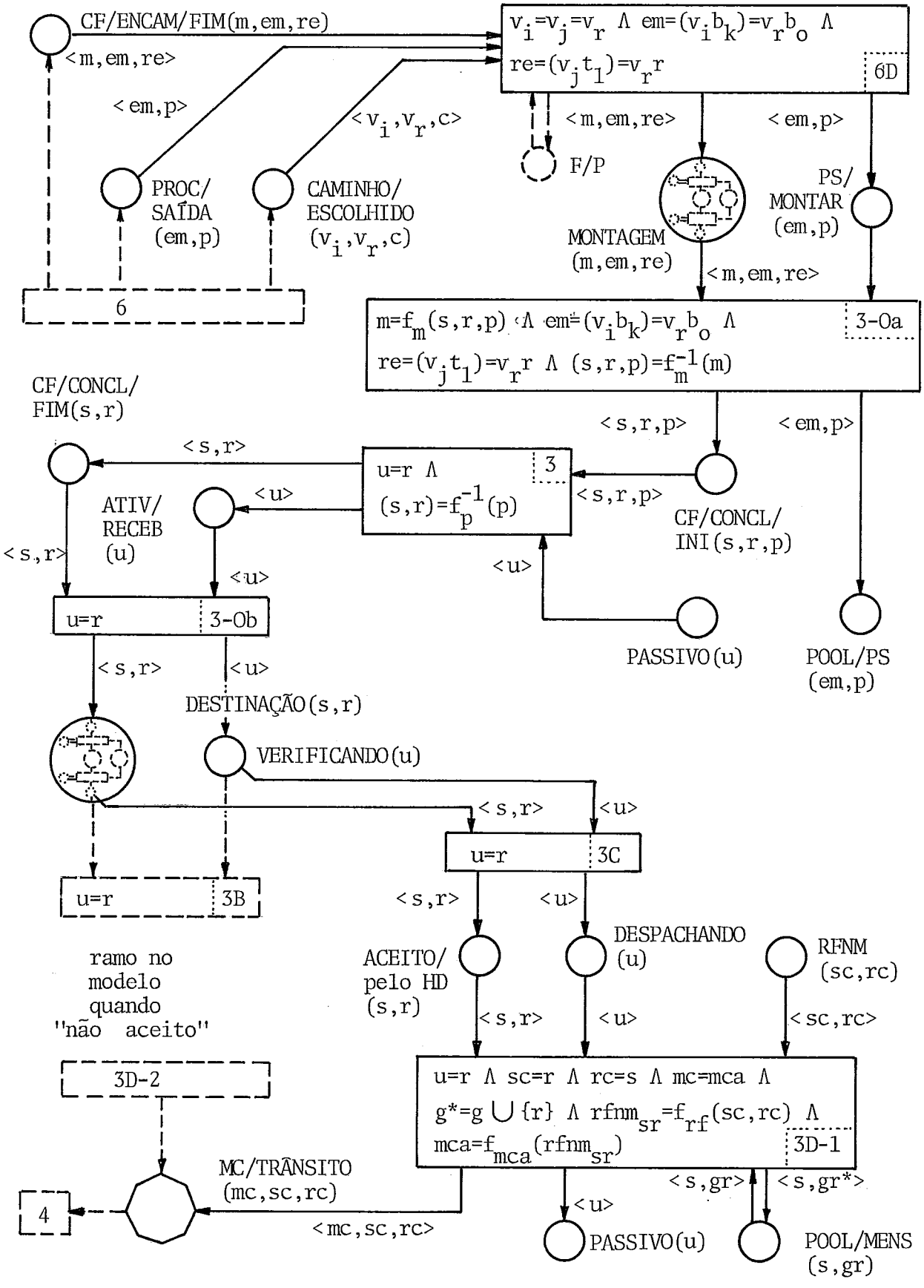


Fig. VI.14.f: Modelo número 3. Parte 6: Recepção no vértice/destinação (baseado nas figuras V.30.e e VI.11).

predicados: MENSAGEM	relação causal com as transições: - ≙ precondição	trans	+ ≙ poscondição	trans
ACEITO/peloHD (s,r)	- <s,r>	3D-1	+ <s,r>	3C
CÓPIA/DESTRUÍDA (cópia)	indefinido		+ <cópia>	8B
CÓPIA/PAC (m,em,re)	- <m,em,re> - <m,em,re> - <m,em,re>	8A 8B 8C	+ <m,em,re>	6B-2
CÓPIA/REUTILIZADA (cópia)	- <cópia> - <cópia> - <cópia>	9A 9B 9C	+ <cópia>	8A 8C
DESTINAÇÃO (s,r)	- <s,r> - <s,r>	3B 3C	+ <s,r>	3-Ob
ENTRADA (m,em,re)	- <m,em,re> - <m,em,re>	7A 7B	+ <m,em,re>	7
FONTE (s,r)	- <s,r>	2-Oa	+ <s,r>	1B
LISTM (s,gr)	- <s,gr> - <s,gr>	1A 1B	+ <s,gr> + <s,gr'>	1 1B
LISTR (s,gr)	- <s,gr> - <s,gr>	4A 4B	+ <s,gr> + <s,gr'>	1 4A
MONTAGEM (m,em,re)	- <m,em,re>	3-Oa	+ <m,em,re>	6D
PAC/ACEITO (m,em,re)	veja: CF/ENCAM/INI		veja: CF/ENCAM/INI	
PAC/REJEITADO (m,em,re)	indefinido		+ <m,em,re>	7A
P-em-TRÂNSITO (m,em,re)	- <m,em,re>	7	+ <m,em,re>	6B-2
POOL/MENSAGEM (s,gr)	- <s,gr> - <s,gr>	1 3D-1	+ <s,gr*>	3D-1
PRONTO (m,em,re)	- <m,em,re>	2A	+ <m,em,re>	2-Ob
SAÍDA (m,em,re)	- <m,em,re>	6B-2	+ <m,em,re> + <m,em,re>	6A 9A
SEG.VIA.M (m)	- <m>	4C	+ <m>	2
SEG.VIA.M/DESTRUÍDA (m,li)	- <m,li>	4A	+ <m,li>	4C

Fig. VI.15.a: Lista dos principais predicados do modelo número 3 (figura VI.14), agrupados segundo as suas diferentes "naturezas". Parte 1: MENSAGEM

predicados: MENSAGEM--UCP	relação causal com as transições: - ≙ precondição	trans	+ ≙ poscondição	trans
CF/CONCL/INI (s,r,p)	- <s,r,p>	3	+ <s,r,p>	3-Oa
CF/CONCL/FIM (s,r)	- <s,r>	3-Ob	+ <s,r>	3
CF/ENCAM/INI (≙ PAC/ACEITO) (m,em,re)	- <m,em,re>	6	+ <m,em,re> + <m,em,re> + <m,em,re> + <m,em,re>	2A 7B 9B 9C
CF/ENCAM/FIM (m,em,re)	- <m,em,re> - <m,em,re>	6A 6D	+ <m,em,re>	6
CF/PREP/INI (s,r)	- <s,r>	2	+ <s,r>	2-Oa
CF/PREP/FIM (s,r,p)	- <s,r,p>	2-Ob	+ <s,r,p>	2
predicados: USUÁRIOS				
ATIV/RECEBER(u)	- <u>	3-Ob	+ <u>	3
ATIV/TRANSM(u)	- <u>	2-Oa	+ <u>	1C
DESPACHANDO(u)	- <u>	3D-1	+ <u>	3C
ESP/RFNM(u)	- <u>	4C	+ <u>	2
FEITO(u)	- <s>	p2	+ <s>	4B
LIBERADO(u)	- <u> - <u>	4A 4B	+ <u>	4C
PASSIVO(u)	- <u> - <u>	1 3	+ <u> + <u>	3D-1 4B
PEDIDO(u)	- <s>	1	+ <s>	p1
PREP/TRANSM(u)	- <u>	1C	+ <u>	1A
TRANSMITIR(u)	- <u>	2	+ <u>	2-Oa
VERIFICANDO(u)	- <u> - <u>	3C 3B	+ <u>	3-Ob

Fig. VI.15.b: Lista dos principais predicados do modelo número 3 (figura VI.14), agrupados segundo as suas diferentes "naturezas".
Parte 2: MENSAGEM - UCP
Parte 3: USUÁRIOS

predicados: PROCESSADOR/ENTR.	relação causal com as transições: - ≙ precondição	trans	+ ≙ poscondição	trans
PE/APRONTAR (re,p)	- <re,p >	6	+ <re,p> + <re,p> + <re,p> + <re,p>	2A 7B 9B 9C
PE/RECEBER (re,p)	- <re,p >	7C	+ <re,p>	7
PE/TESTAR (re,p)	- <re,p > - <re,p >	7A 7B	+ <re,p>	7C
PE/VERIFICAR (re,p)	- <re,p >	2A	+ <re,p>	2-Ob
POOL/PE (re,p)	- <re,p > - <re,p > - <re,p > - <re,p >	2-Ob 6B-2 9B 9C	+ <re,p> + <re,p>	6 7A
PROC/ENTRADA (re,p)	- <re,p >	7	+ <re,p>	6B-2
predicados: PROCESSADOR/SAÍDA				
POOL/PS (em,p)	- <em,p>	6	+ <em,p> + <em,p> + <em,p> + <em,p>	3-Oa 8B 9B 9C
PROC/SAÍDA (em,p)	- <em,p> - <em,p>	6A 6D	+ <em,p>	6
PS/ESP.ACK (em,p)	- <em,p> - <em,p> - <em,p>	8A 8B 8C	+ <em,p>	6B-2
PS/MONTAR (em,p)	- <em,p>	3-Oa	+ <em,p>	6D
PS/PREPARANDO (em,p)	- <em,p>	6B-1	+ <em,p>	6A
PS/REATIVANDO (em,p)	- <em,p> - <em,p> - <em,p>	9A 9B 9C	+ <em,p> + <em,p>	8A 8C
PS/TRANSMITINDO (em,p)	- <em,p>	6B-2	+ <em,p> + <em,p>	6B-1 9A

Fig. VI.15.c: Lista dos principais predicados do modelo número 3 (figura VI.14), agrupados segundo as suas diferentes naturezas.
Parte 4: PROCESSADORES DE ENTRADA
Parte 5: PROCESSADORES DE SAÍDA

predicados: CAMINHOS, ENLACES	relação causal com as transições: - ≙ precondição	trans	+ ≙ poscondição	trans
CAMINHO/ESCOLHIDO (v_i,v_r,c)	- <v_i,v_r,c> - <v_i,v_r,c>	6A 6D	+ <v_i,v_r,c>	6
CAMINHO/VERIFICADO (v_i,v_r,c)	- <v_i,v_r,c> - <v_i,v_r,c>	9A 9B	+ <v_i,v_r,c>	8A
CAMINHO/USADO (v_i,v_r,c)	- <v_i,v_r,c>	6B-1	+ <v_i,v_r,c>	6A
ENLACE/ATIVADO (enl)	- <enl> - <enl>	6B-2 9A	+ <enl>	6B-1
ENLACE/DESATIVADO (enl)	- <enl>	7C	+ <enl>	7
ENLACE/OCUPADO (enl)	- <enl>	7	+ <enl>	6B-2
FEIXE/ATUAL (v_i,v_r,fc)	- <v_i,v_r,fc> - <v_i,v_r,fc>	6 8A	indefinido neste nível do modelo	
POOL/ENLACE (enl)	- <enl> - <enl>	6B-1 9A	+ <enl>	7C
predicados: MENS. de CONTROLE				
MCA/RECEBIDA (mca,sc,rc)	- <mca,sc,rc>	4C	+ <mca,sc,rc>	4
MCQ/RECEBIDA (mcq,sc,rc)	indefinido		+ <mcq,sc,rc>	4
MC/TRÂNSITO (mc,sc,rc)	- <mc,sc,rc>	4	+ <mc,sc,rc>	3D-1
PCA/RECEBIDO (pca,emc,rec)	- <pca,emc,rec>	8B	+ <pca,emc,rec>	8
PCC/TRÂNSITO (pcc,emc,rec)	- <pcc,emc,rec>	8	+ <pcc,emc,rec> + <pcc,emc,rec>	7A 7B
PCN/RECEBIDO (pcn,emc,rec)	- <pcn,emc,rec>	8A	+ <pcn,emc,rec>	8
RFNM(sc,rc)	- <sc,rc>	3D-1	indefinido	

Fig. VI.15.d: Lista dos principais predicados do modelo número 3 (figura VI.14), agrupados segundo as suas diferentes "naturezas".
Parte 6: CAMINHOS E ENLACES
Parte 7: MENSAGENS DE CONTROLE

predicados: AUXILIARES	relação causal com as transições: - ≙ precondição	trans	+ ≙ poscondição	trans
CRÉDITO/MENSAGEM (ja_{sr})	- $<ja_{sr}>$	2-0a	+ $<ja'_{sr}>$	4C
	- $<ja_{sr}>$	4C		
INFO/TEMPO-GLOBAL (itg)	- <itg>	2	indefinido	
INFO/TEMPO-LOCAL (itl,v_i)	- $<itl,v_i>$	6B-2	indefinido	
MENS/conf.RFNM (âes,mca)	- <âes,mca>	4C	+ <âes,mca>	4-3a
PAC/conf.ACK (ârg,pca)	- <ârg,pca>	8B	+ <ârg,pca>	8-3a
PAC/conf.NAK (ârg,pcn)	- <ârg,pcn>	8A	+ <ârg,pcn>	8-3b
PAC/nãoCONF (ârg,neg)	- <ârg,neg>	8C	+ <ârg,neg>	8-4
POOL/BLOQUEADOR-GLOBAL(bg_s)	- $<bg_s>$	2	+ $<bg_s>$	4-3a
POOL/BLOQUEADOR-LOCAL(bl_{vi})	- $<bl_{vi}>$	6B-2	+ $<bl_{vi}>$	8-3a
			+ $<bl_{vi}>$	8-3b
			+ $<bl_{vi}>$	8-4
SUBTAREFA/FIM (esf,bl_{vi})	- $<\not\exists$ esf,$bl_{vi}>$	8-1	+ <esf>	8
	- $<esf,bl_{vi}>$	8-3a	+ $<bl_{vi}>$	8-1
	- $<esf,bl_{vi}>$	8-3b		
TAREFA/FIM (etf,bg_s)	- $<\not\exists$ etf,$bg_s>$	4-1	+ <etf>	4
	- $<etf,bg_s>$	4-3a	+ $<bg_s>$	4-1
	- $<etf,bg_s>$	4-3b		
TEMPO/RESTANTE-GLOBAL (âes,trg,bg_s)	- <trg,bg_s>	4-1	+ <âes,tdt,bg_s>	2
	- <âes,trg>	4-3a	+ <trg',bg_s>	4-1
	- <âes,trg>	4-3b		
TEMPO/RESTANTE-LOCAL (ârg,trl,bl_{vi})	- <trl,bl_{vi}>	8-1	+ <ârg,tds,bl_{vi}>	6B-2
	- <ârg,trl>	8-3a	+ <trl',bl_{vi}>	8-1
	- <ârg,trl>	8-3b		
	- <ârg,trl,bl_{vi}>	8-4		

Fig. VI.15.e: Lista dos principais predicados do modelo número 3 (figura VI.14), agrupados segundo as suas diferentes "naturezas". Parte 8: AUXILIARES.

	T R A N S I Ç Õ E S :
nº	fórmulas
p1	$s \in U$
p2	$s \in U$
1	$u=s$
1A	$u=s \wedge gr=0$
1B	$gr \neq 0 \wedge gr'=gr \setminus \{r\}$
1C	$u=s$
2	$u=s \wedge p=f_p(s,r) \wedge m=f_m(s,r,p) \wedge \hat{d}es=\hat{f}^*_{des}(s,r,p) \wedge \hat{t}dt=\hat{f}_{tdt}(\hat{d}es,itg)$
2-Oa	$u=s \wedge ja_{sr} \neq 0$
2-Ob	$m=f_m(s,r,p) \wedge em=(v_j b_k)=v_s s \wedge re=(v_j t_l)=v_s t_o$
2A	$m=f_m(s,r,p) \wedge em=(v_i b_k)=v_s s \wedge re=(v_j t_i)=v_s t_o$
3	$u=r \wedge (s,r)=f_p^{-1}(p)$
3-Oa	$m=f_m(s,r,p) \wedge em=(v_i b_k)=v_r b_o \wedge re=(v_j t_l)=v_r r \wedge (s,r,p)=f_m^{-1}(m)$
3-Ob	$u=r$
3B	$u=r$
3C	$u=r$
3D-1	$u=r \wedge g^*=g \cup \{r\} \wedge mca=f_{mca}(rfnm_{sr}) \wedge sc=r \wedge rc=s \wedge mc=mca \wedge rfnm_{sr}=f_{rf}(sc,rc)$
4	$sc=r \wedge rc=s \wedge mca \vee mcq=f_{mc}^{-1}(mc) \wedge etf=f_{etf}(mc,sc,rc)$
4-1	$trg \neq 0 \wedge \nexists\, etf \wedge trg=trg-1$
4-3a	$trg > 0 \wedge mca=f_{etf}^{-1}(etf)$
4-3b	$trg > 0 \wedge mcq=f_{rtf}^{-1}(etf)$
4-4	$trg=0 \wedge$ ----
4A	$gr \neq 0 \wedge gr'=gr \setminus \{r_{1i}\} \wedge m=f_m(s,r,p) \wedge r_{1i}=f_{1i}(s,r,p,1i)$
4B	$gr=0 \wedge u=s$
4C	$u=s \wedge sc=r \wedge rc=s \wedge rfnm_{sr}=f_{mca}^{-1}(mca) \wedge \hat{d}es=\hat{f}^*_{des}(s,r,p) \wedge m=f_m(s,r,p) \wedge 1i=f_{1i}(m,\hat{d}es,rfnm_{sr}) \wedge ja'_{sr}=ja_{sr}+1$

Fig. VI.16: Tabela das transições do modelo número 3 (figura VI.14), e suas principais fórmulas, envolvidas diretamente no controle de fluxo.

no.	T R A N S I Ç Õ E S : fórmulas
6	$em_{velho}=(v_i b_k)=emv \wedge re_{velho}=(v_j t_1)=rev \wedge m=f_m(s,r,p) \wedge$ $v_i=f_{vi}(rev) \wedge c=sel(fc,m,emv,rev,v_i) \wedge [fcp'=fcp \cup (fc \setminus \{c\})] \wedge$ $em=em_{novo}=f_{em}(c)=v_i b_k \wedge re=re_{novo}=f_{re}(c)=v_j t_1$
6A	$v_i \neq v_r \wedge em=v_i b_k \wedge re=v_j t_1$
6B-1	$em=v_i b_k \wedge re=f_{re}(c)=v_j t_1 \wedge [fcp'=fcp \cup \{c\}] \wedge$ $enl=c_{v_i \to v_r} \cap c_{v_i \to re}$
6B-2	$re=f_{re}(enl)=v_j t_1 \wedge em=v_i b_k \wedge m=f_m(s,r,p) \wedge$ $\hat{a}rg=\hat{f}^*_{arg}(m,em,re) \wedge \hat{t}ds=\hat{f}_{tds}(\hat{a}rg,itl,v_i)$
6D	$v_i=v_j=v_r \wedge [fcp'=fcp \cup \{c\}] \wedge em=(v_i b_k)=v_r b_o \wedge$ $re=(v_j t_1)=v_r r$
7	$m=f_m(s,r,p) \wedge em=v_i b_k \wedge re=v_j t_1$
7A	$m=f_m(s,r,p)=m_{NAK} \wedge pcn=f_{pcn}(m_{NAK}) \wedge pcc=pcn \wedge$ $em=v_i b_k \wedge re=v_j t_1 \wedge emc=v_j t_1 \wedge rec=v_i b_k$
7B	$m=f_m(s,r,p)=m_{ACK} \wedge pca=f_{pca}(m_{ACK}) \wedge pcc=pca \wedge$ $em=v_i b_k \wedge re=v_j t_1 \wedge emc=v_j t_1 \wedge rec=v_i b_k$
7C	$re=v_j t_1 \wedge enl=f_{enl}(re)$
8	$pca\ pcn=f^{-1}_{pcc}(pcc) \wedge esf=f_{esf}(pcc,emc,rec)$
8-1	$trl \neq 0 \wedge \nexists\ esf \wedge trl'=trl-1$
8-3a	$trl>0 \wedge pca=f^{-1}_{esf}(esf)$
8-3b	$trl>0 \wedge pcn=f^{-1}_{esf}(esf)$
8-4	$trl=0 \wedge neg=f_{neg}(\hat{a}rg,trl)$
8A	$emc=v_j t_1 \wedge rec=v_i b_k \wedge em=v_i b_k \wedge m_{NAK}=f^{-1}_{pcn}(pcn) \wedge$ $cópia=f_{co}(m,em,re,res) \wedge [fcp'=fcp \cup (fc \setminus \{c\})] \wedge$ $c_{verif}=sel(fc,pcn,emc,rec,v_i=f_{vi}(pcn,em)) \wedge$ $\hat{a}rg=\hat{f}_{arg}(m,em,re) \wedge res=f_{res}(m,\hat{a}rg,m_{NAK})$

Fig. VI.16: continuação): Tabela das transições .....

T R A N S I Ç Õ E S :	
no.	fórmulas
8B	$emc=v_j t_1 \wedge rec=v_i b_k \wedge em=v_i b_k \wedge m_{ACK}=f_{pca}^{-1}(pca) \wedge re=v_j t_1 \wedge$ $cópia=f_{co}(m,em,re,res) \wedge \hat{a}rg=\hat{f}^{*}_{arg}(m,em,re) \wedge$ $res=f_{res}(m,\hat{a}rg,m_{ACK})$
8C	$\hat{a}rg=\hat{f}^{*}_{arg}(m,em,re) \wedge res=f_{res}(m,\hat{a}rg,neg) \wedge em=v_i b_k \wedge$ $cópia=f_{co}(m,em,re,res) \wedge re=v_j t_1$
9A	$c_{verif}=c_{velho} \wedge [fcp'=fcp \cup \{c\}] \wedge (m,em,re)=f_{co}^{-1}(cópia) \wedge$ $enl=c_{v_i \to v_r} \cap c_{v_i \to re}$
9B	$c_{verif} \neq c_{velho} \wedge [fcp'=fcp \cup \{c\}] \wedge em=v_i b_k \wedge$ $re=(v_j t_1)=v_i t_{-1} \wedge (m,em,re)=f_{co}^{-1}(cópia)$
9C	$em=v_i b_k \wedge re=(v_j t_1)=v_i t_{-1} \wedge (m,em,re)=f_{co}^{-1}(cópia)$

Fig. VI.16: (continuação-final): Tabela das transições .....

VI.5 - Observações finais sobre os "tempos" envolvidos no controle de fluxo.

Em princípio, cada vez que estamos esperando uma resposta vinda de um lugar diferente ou distante do atual, deveríamos incluir uma verificação com base no tempo passado.

Agora, se esta informação esperada significa somente uma informação complementar, podemos, eventualmente, relaxar esta exigência. Como exemplo, podemos citar o caso de investigações sobre o desempenho normal global de uma rede, onde podemos sempre usar uma estimativa para obter um próximo valor ou uma interpolação para obter valores intermediários que faltam, sob hipótese de que não haja modificações extremas no sistema; estas, por sua vez, serão transmitidas num outro nível, ou seja, não representarão uma resposta, propriamente dita, à nossa investigação sobre o funcionamento normal.

Entretanto, se a resposta aguardada é indispensável para garantir a continuidade de alguma seqüência de operações, esta ferramenta de vigilância deve ser, em todo caso, incluída já que, em casos extremos, podemos, através da propagação dos efeitos de um 'deadlock' local, chegar a um colapso total do sistema. Mas a inclusão desta ferramenta não precisa ser feita em todos os lugares de uma determinada seqüência de operações; bastaria estimar o tempo necessário para que esta seqüência mencionada fosse executada e verificar este tempo. O problema que surge, obviamente, é saber que parte da seqüência falhou quando houve excesso de tempo. Como já foi mencionado no capítulo II, não sabemos, por exemplo no caso de um 'time-out' entre vértices vizinhos, se o pacote se perdeu ou somente se atrasou, ou se o tempo de verificação no vértice/receptor foi longe demais ou, até, se a confirmação levou mais tempo que previsto.

O que tentamos mostrar, neste capítulo, é que a inclusão desta vigilância é somente em parte trivial, ou seja, que somente a parte da verificação do tempo transcorrido entre o "início e fim" de alguma seqüência de operações é bastante simples.

Já a parte complementar, chamada por nós de CONSEQÜÊNCIAS, é algo bem mais difícil porque, repetimos, não sabemos a causa exata do 'time-out' ou do 'check-time'. Conseqüentemente, a nossa reação deve ser baseada em hipóteses (sobre o que aconteceu), utilizando experiências acumuladas do passado para poder tomar, eventualmente, a decisão certa.

Achamos que estes dois exemplos são o bastante para compreender o problema e voltaremos aos diversos outros 'time-outs', como, por exemplo, aquele relacionado ao tempo de espera, no vértice/destinação, da confirmação RFNM vindo do host/destinação (veja fig. II.2), somente no caso em que temos que modelar certas decisões diretamente baseadas nestes 'time-outs' em questão.

C A P Í T U L O VII

Idéias sobre a modelagem, na base de 'PrT-Nets', da transmissão de mensagens do tipo multipacotes em redes de computadores.

VII.1 - Introdução.

(Observação: uma versão preliminar deste capítulo foi publicada como relatório técnico, SCHWARZ (37)).

Investigando problemas dentro do mundo da telemática, isto é, problemas em relação à convivência das tecnologias da computação e das comunicações, vemos que a parte essencial, o suporte indispensável, são as redes de computadores.

Para modelar, então, os problemas destas redes, principalmente quando se trata dos problemas dinâmicos, precisa-se alguma técnica, adicionalmente às teorias de filas e de grafos, que permita modelar decisões, concorrências, etc. Encontramos esta técnica na 'General Net Theory' que contém uma ferramenta poderosa, a técnica de 'predicate/transition-nets' (PrT-Nets'), que pode ser considerada, como foi mencionado no capítulo III, como sendo um Petri-Net de 1ª ordem.

Usando esta técnica, obtivemos um modelo preliminar, o modelo número 1 no capítulo IV, que tentou explicar a transmissão entre usuários e que tratou a subrede de comunicações como sendo um único recurso. A obtenção deste primeiro modelo, ainda bem carente de detalhes, serviu, principalmente, para estudar a utilidade da técnica de 'PrT-Nets' em relação às redes de computadores e, também, para se familiarizar com esta nova técnica.

O modelo seguinte, denominado de modelo número 2, obtido no capítulo V, considerou a inclusão dos detalhes da subrede de comunicação, em particular, os vértices intermediários, já investigando as conseqüências básicas em relação a mensagens de controle do tipo ACK ('acknowledgement') e NAK ('negative ACK'). Este modelo obtido foi completado, no capítulo VI, com idéias sobre tempos envolvidos se restringindo, porém, ao 'time-out', no nível intervértice, e ao 'check-time', no nível 'end-to-end'. Isto resultou no modelo número 3.

O que falta, ainda, em relação a esta primeira fase da modelagem, é a inclusão de mensagens do tipo multipacotes, o que faremos neste capítulo VII, chegando, assim, a um modelo básico (o modelo número 4) que represente, bastante fielmente, as idéias realizadas na primeira versão do ARPANET. Esta primeira versão, mesmo sendo substituída hoje por versões subseqüentes, reune, em nossa opinião, todas as idéias básicas importantes para lidar com redes de computadores.

Este capítulo VII se propõe, então, a apresentar a seqüência dos eventos quando se trata de mensagens do tipo multipacotes (seção VII.2), para, depois, na seção VII.3, investigar os problemas da modelagem no sentido das interfaces entre usuário e subrede de comunicação, da identificação (indexação) dos pacotes na subrede e, finalmente, da remontagem da mensagem original nos vértices/destinação.

VII.2 - A seqüência dos eventos para a transmissão de mensagens do tipo multipacotes.

VII.2.a - Introdução.

Já fizemos três modelos que mostraram a transmissão de mensagens do tipo pacote simples: um foi o modelo simplificado, o número 1 (figura IV.9), outro, o modelo número 2 (figura V.30), foi a sua extensão no sentido de incluir vértices intermediários, ACK e NAK, e, finalmente, o modelo número 3 (figura VI.14), que tratou também da verificação dos tempos 'time-out' e

'check-time'.

O que devemos fazer agora é modificar estes modelos obtidos para poder tratar, em vez de somente mensagens pequenas que não excedem o tamanho máximo de um pacote, mensagens grandes que serão divididas em até oito pacotes de tamanho padronizado. Assim, uma mensagem

$$MSG \doteq \{PAC_1, PAC_2, \ldots, PAC_8\} \text{ , onde}$$

existe a restrição que o tamanho máximo da mensagem MSG não deve exceder, p.ex., oito vezes o tamanho padronizado de pacotes PAC; esta padronização se refere, por enquanto, unicamente a uma determinada subrede de comunicação (p.ex., 1008 bits no ARPANET, como relatado por KLEINROCK (14)).

A pergunta que surge é a seguinte: Quem decide de que tipo será a mensagem e quem vai dividí-la em pacotes?

Para resolver estes pontos temos, em princípio, três candidatos potenciais: o usuário, o 'host'/fonte e o vértice/fonte. Investigando as potencialidades de cada um, chegamos às seguintes conclusões:

- Achamos que não deve ser o usuário que divide a mensagem, mas é certamente ele que classifica a mensagem. Assim, ele define, obedecendo ao critério do tamanho máximo permitido para mensagens (p.ex., 8064 bits no ARPANET), se a mensagem é do tipo simples, interativa, tempo real, arquivo, prioritária, etc. Nós, por enquanto, levamos em consideração somente duas classificações, ou seja, mensagens simples (menor ou igual ao tamanho de um pacote padronizado) e mensagens do tipo multipacotes (tamanho do PAC < MSG $\leq$ 8xPAC);

- Feita a classificação pelo usuário, achamos que o 'host'/fonte é o lugar certo para acatar a decisão sobre o tipo da mensagem. Assim, quando o H/F recebeu uma mensagem do tipo, digamos, "arquivo" (classificado como "multipacotes"), deve ser

ele que divide a mensagem em partes menores, ou seja, em pre pacotes. Isto inclui, obrigatoriamente, uma identificação inconfundível, para cada prepacote, no sentido de preservar as informações sobre a qual mensagem pertence e que lugar ele ocupa nela;

- Existe alguma controvérsia sobre o que deve ser feito pelo H/F e pelo vértice/fonte, respectivamente. Para encontrar uma saída em relação aos dois pontos mais polêmicos, a transparência para o usuário e a sobrecarga da subrede, propomos, para nosso estudo, o seguinte: o usuário, repetimos isto, classifica a mensagem e respeita o seu tamanho máximo; o 'host'/fonte, como já foi mencionado, divide a mensagem em pacotes menores, ou seja, em prepacotes, e garante a identificação, em relação à mensagem, dos diversos prepacotes; finalmente, o vértice/ /fonte, que já faz parte da subrede de comunicação e que não deve, portanto, ser sobrecarregado com tarefas que fogem da responsabilidade direta da transmissão, deve se encarregar da preparação dos prepacotes (empacotar) de uma maneira tal que possam atravessar a subrede como sendo unidades autônomas . Por isso, cada pacote da mensagem dividida deve se tornar um "verdadeiro" PACOTE no sentido da subrede, ou seja, deve conter, além da identificação em relação à mensagem original, os endereços do (usuário)/emissor e receptor e, também, eventuais aditivos para fins de controle interno da subrede.

Tendo colocado este nosso posicionamento, podemos, primeiramente, resumir as diferenças básicas em relação à transmissão de pacotes simples, para, depois, na seção VII.3, modelar estes novos aspectos.

VII.2.b - O esquema simplificado, sem os detalhes dos vértices intermediários, da transmissão de uma mensagem do tipo multipacote.

Em relação à descrição do esquema para pacotes simples (figura II.2) podemos destacar as seguintes diferenças quando se trata de mensagens do tipo multipacotes (figura II.4):

Mostrou-se, então, que a chegada da mensagem ao vértice/fonte, ocorre já em forma de multipacotes; vale destacar que esta chegada dos prepacotes (pp) acontece de uma maneira seqüencial de acordo com a divisão da mensagem original e que, já no V/F, será feito o empacotamento, ou seja, o tratamento dos prepacotes para que se tornem PACOTES no sentido da subrede de comunicação.

A realização da transmissão é feita, individualmente, para cada PACOTE que, aliás, podem tomar caminhos diferentes na subrede de comunicação e assim, eventualmente, chegar em ordem diferente daquela obedecida na emissão.

E, finalmente, do mesmo modo que a subrede recebeu a mensagem, ou seja, os prepacotes (que formam esta mensagem) em ordem seqüencial, ela deve entregar os pacotes numa seqüência tal que formem exatamente a mensagem original ('re-assembly' ou remontagem).

Estas idéias podem ser confiridas na figura II.4.

VII.2.c - O esquema completo da transmissão de mensagens do tipo multipacotes, a saber, incluindo os vértices intermediários com ACK, NAK e 'time-out'.

A adaptação que levou à figura II.4 também foi usada para atualizar a anatomia completa da transmissão de um pacote simples (figura II.5) no sentido de permitir a transmissão de mensagens do tipo multipacotes. Assim, foi esboçado:

- o acesso da mesma maneira como foi mostrada na figura II.4 em torno de H/F e V/F;

- a transmissão dos pacotes (que formam a mensagem), escolhendo um deles, digamos $PAC_{n,i}$, como sendo representativo, exatamente como na figura original (fig. II.5), exceto a indexação;

- a entrega ao usuário tal como o esquema na figura II.4 indicou.

Observando esta adaptação na figura II.6, sentimos que precisamos, provavelmente, somente adaptações do tipo indexação para modificar o modelo, número 3, na base de 'PrT-Net', obtido no capítulo VI.

VII.3 - Modelagem, na base de 'PrT-Nets', da transmissão de mensagens do tipo multipacotes.

VII.3.a - Introdução.

A nossa preocupação principal foi, até agora, mostrar como modelar a transmissão de pacotes simples através de uma rede de comutação de pacotes. Conseguimos mostrar idéias como, p.ex., separar o nível do usuário, em geral, do nível da subrede (capítulo IV). Neste último nível foi modelado, entre outros, o uso repetitivo do esquema gráfico que representa os vértices intermediários (capítulo V).

Agora, em relação a mensagens do tipo multipacotes, acontece que a versão 1 do ARPANET sempre permitiu a transmissão de mensagens multipacotes, mas nós achamos mais conveniente introduzir este conceito na modelagem somente nesta fase de modelagem por considerar aquele esquema que trata de pacotes simples como sendo a base para qualquer tipo de transmissão.

Deste modo, podemos usar os modelos anteriormente obtidos (figura V.30 e figura VI.14) e adaptá-los neste novo sentido.

VII.3.b - Investigação dos problemas associados às mensagens do tipo multipacotes.

VII.3.b.1 - Interface entre um usuário externo e a rede de comutação de pacotes.

Como, até agora, não foi necessário nos preocuparmos muito em distinguir entre os conceitos para MENSAGENS e PACOTES, usamos, às vezes, um ou outro destes termos para especificar mensagens cujo tamanho sempre cabia no formato de um pacote sim-

ples. Mas agora, querendo incluir mensagens com tamanho superior a do de um pacote, achamos oportuno rever as definições para usuário, mensagens, grupo, pacote, etc.

Podemos dizer, que "usuário externo", em geral, pode ser um processo industrial, comercial ou computacional, um terminal, etc., o que, considerando a nomenclatura padronizada sobre redes de computadores, coincide com os níveis de protocolos acima do nível três.

Agora, para nós, o pedido de algum usuário externo u, (u ε U), chegou, via a transição externa p1, ao predicado PEDIDO(u), sendo a partir daqui considerado simplesmente um USUÁRIO no sentido da modelagem. Fisicamente, este pedido (predicado PEDIDO(u)) pode representar o fato que um hospedeiro/fonte, um 'front-end' ligado ao canal s de vértice/fonte ou, até, um terminal inteligente ligado diretamente ao vértice/fonte com capacidades compatíveis com as necessidades exigidas, deseja enviar mensagens a um ou mais outros usuários da rede.

Assim, a condição presente no predicado PEDIDO(u) significa que algum usuário u, (u ε U), tem mensagens para diversos outros receptores a enviar. No modelo anterior, juntamos à esta precondição as outras precondições contidas em PASSIVO(u) e POOL/MENSAGEM(s,gr) para que a transição 1 possa ser disparada. Agora, para melhor podes distinguir entre um usuário EMISSOR e RECEPTOR temos que fazer uma ressalva com respeito à definição do predicado PASSIVO(u).

Usando a definição básica de U = $\{u_1, u_2, \ldots, u_W\}$, que dizia que os u_i's são os usuários conectados em torno da rede de computadores em questão, observe-se que estes são os chamados usuários externos. Assim, u, (u ε U), pode ser somente considerado válido como sendo uma precondição para iniciar (pedir) uma emissão de mensagens o que, em nosso caso, coincide com a transição 1 no modelo.

Já o predicado PASSIVO(u) deveria representar o USUÁRIO, em nosso sentido, ou seja, indicar o responsável pela emissão que este usuário externo pediu. Assim, achamos melhor chamar este predicado de POOL/EMISSOR(s) seguindo as definições já mencionadas no capítulo IV, a saber,

$$s \in EE, \text{ onde } EE = EndE = \{s_1, s_2, \ldots, s_W\},$$

o que indica, que cada usuário pode se tornar emissor (sendo a informação "endereço do emissor" contida numa parte da mensagem). Assim, fica mais claro agora, na modelagem, que não existe empecilho para que cada usuário possa ser um emissor s, ($s \in EE$), e, ao mesmo tempo, também um receptor r, ($r \in ER$), o que, nos modelos anteriores, não foi mostrado de maneira explícita.

Esta idéia, também, reflete bem melhor a situação atual em sistemas automatizados que, em geral, permitem que as funções de entrada e saída funcionassem ao mesmo "tempo" (ao menos, a partir de um certo nível lógico) o que, aliás, corresponde aos sistemas de comunicação que quase sempre estão baseados em transmissões do tipo FULL-DUPLEX.

A associação, no modelo, no sentido de que um usuário se "torna" emissor, é feita através de uma fórmula na transição 1, $s=f_s(u)$, que substitui a fórmula anteriormente empregada, u=s. Ao mesmo tempo em que se faz esta associação, devemos separar as mensagens que este emissor quer enviar.

Estas MENSAGENS estão agrupadas em gr, ($gr \in GR$), indicando, assim, que existe, para cada usuário, um grupo de receptores potenciais. O que deve ficar claro é que a configuração de cada gr é variável, ou seja, que depende dos receptores desejados no momento da colocação do pedido.

Do ponto de vista da modelagem podemos, então, usar um predicado POOL/MENSAGEM(s,gr) que representaria estas MENSAGENS agrupadas.

Ao mesmo tempo em que "acordamos" o usuário/emissor, colocamos o grupo das mensagens, associadas a este emissor, em circulação. Esta inicialização, ou, em termos do modelo, a "inscrição" deste grupo no predicado LISTM(s,gr), pode ser realizada pelo disparo da transição 1 que, também, coloca o emissor em um estado ativo, representado pelo predicado PREP/TRANSM(s); assim, obtivemos o par "PREP/TRANSM(s)-LISTM(s,gr)" que representa a associação entre um grupo de mensagens (de s para gr) e o responsável por elas.

Estas primeiras idéias podem ser observadas na figura VII.1.

VII.3.b.2 - Divisão de uma MENSAGEM em prepacotes e inscrição dos mesmos no vértice/fonte.

Conforme o que falamos na seção VII.2.a, será o 'host'/fonte que faz a divisão da mensagem em prepacotes, seguido pelo tratamento no vértice/fonte para obter PACOTES independentes. Isto significa, do ponto de vista da modelagem, que a mensagem deveria chegar ao predicado FONTE(s,r), que é representativo do canal de entrada (canal s) do vértice/fonte, já numa forma que indica a divisão em prepacotes.

Para realizar isto, intercalaremos, entre os predicados envolvidos, ou seja, entre LISTM(.) e FONTE(.), um "novo" predicado, digamos LISTMP(s,r,gp), que representaria esta divisão. Assim, a seqüência no modelo, ao que concerne as mensagens, pode ser descrita da seguinte maneira:

- do predicado POOL/MENSAGEM(s,gr) tirar as MENSAGENS do emissor s e inscrever, de uma maneira "agrupada", numa lista intermediária, predicado LISTM(s,gr), onde s, (s ε EE), é o emissor, e gr, (gr ε GR), são os receptores agrupados. Deste modo, uma tupla <s,gr> significa, implicitamente, o conjunto das mensagens de s para determinados r_i's e, conseqüentemente, a tupla <s,r> indica uma mensagem de s para um determinado r deste grupo gr;

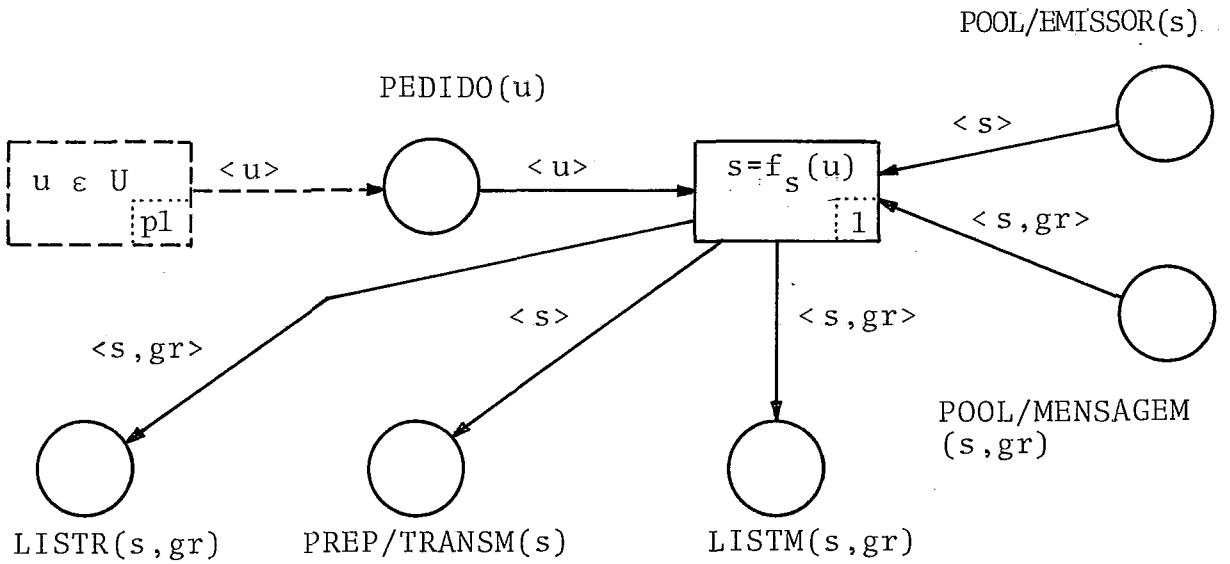


Fig. VII.1: Modelo parcial em torno da transição 1, mostrando a associação entre um pedido de um usuário para realizar a emissão de um grupo de mensagens.

- transferir as mensagens, do predicado LISTM(s,gr) para o novo predicado mencionado, LISTMP(s,r,gp). Esta transferência, feita individualmente para cada mensagem, inclui a divisão de uma determinada mensagem em prepacotes. Indicamos isto através da fórmula $gp=f_{gp}(s,r)$, na transição 1A, onde $gp=\{p_1,p_2,\ldots,p_8\}$, o que representa o grupo de prepacotes que resultam desta divisão. Ao mesmo tempo, temos que "dividir" a responsabilidade de s no sentido que, em vez de ter um único responsável pelo grupo das mensagens, deveria existir um grupo de (sub)responsáveis, um para cada mensagem. Este fato mostramos, na transição 1C, através da expressão $sr=f_{sr}(s,r)$;

- a inscrição no predicado FONTE(.) ocorre de uma maneira bem similar àquela mostrada no modelo número 3, ou seja, que cada prepacote começa ser tratado como sendo (quase) individual. Assim, a tupla $<s,r,p_i>$ identifica o i-ésimo pacote p_i de uma mensagem de s para r, representada até agora pela tupla <s,r>. Conseqüentemente, o predicado deve ser do grau três, a saber, FONTE(s,r,p).

Observando esta seqüência, vemos que conseguimos preservar quase a mesma estrutura básica do modelo número 3 e que, a par-

tir do novo predicado FONTE(s,r,p), podemos realmente usar o resultado obtido anteriormente. Veja esta parte na figura VII.2.

O próximo passo é o "empacotamento", ou seja, tornar o prepacote um PACOTE no sentido da subrede de comunicação. Para isto, basta "ajustar" as inscrições nos diversos elementos do grafo empregado, o 'PrT-Net', com a finalidade que elas representassem, em vez de uma mensagem, MSG_n, agora um PACOTE, $PAC_{n,i}$, desta mensagem. Faremos isto na seção VII.3.b.3.

VII.3.b.3 - Transformação dos prepacotes em PACOTES (no sentido da subrede de comunicação) e as modificações causadas no modelo anterior.

Primeiramente, devemos colocar a pergunta: O que significa PACOTE no sentido da subrede? Sabemos, que PACOTE é o nome dado a uma mensagem que obedece certos padrões em relação ao formato. Um destes padrões está, por exemplo, ligado ao tamanho máximo permitido (p.ex., 1008 bits por pacote no ARPANET). Esta exigência já foi obedecida na entrada da rede, representada, no modelo, pela fórmula da transição 1A, ou seja, $gp=f_{gp}(s,r)$, eliminando, assim, a expressão $p=f_p(s,r)$ na transição 2.

Resolvido isto, podemos nos concentrar em um outro padrão, a saber, que o PACOTE deva ser capaz de "viajar" de uma maneira independente na rede, mas que, adicionalmente, ele deva conter informações suficientes para que, na destinação, se possa recompor a mensagem original. Para conseguir isto, se coloca dentro do pacote, além das informações de controle, os endereços da fonte e da destinação e, também, o número da seqüência em relação à divisão da mensagem original. Como, nesta versão, somente circula uma mensagem por vez entre algum par de usuários, equivalente a um crédito igual a UM, estas informações serão o suficiente para remontar a mensagem.

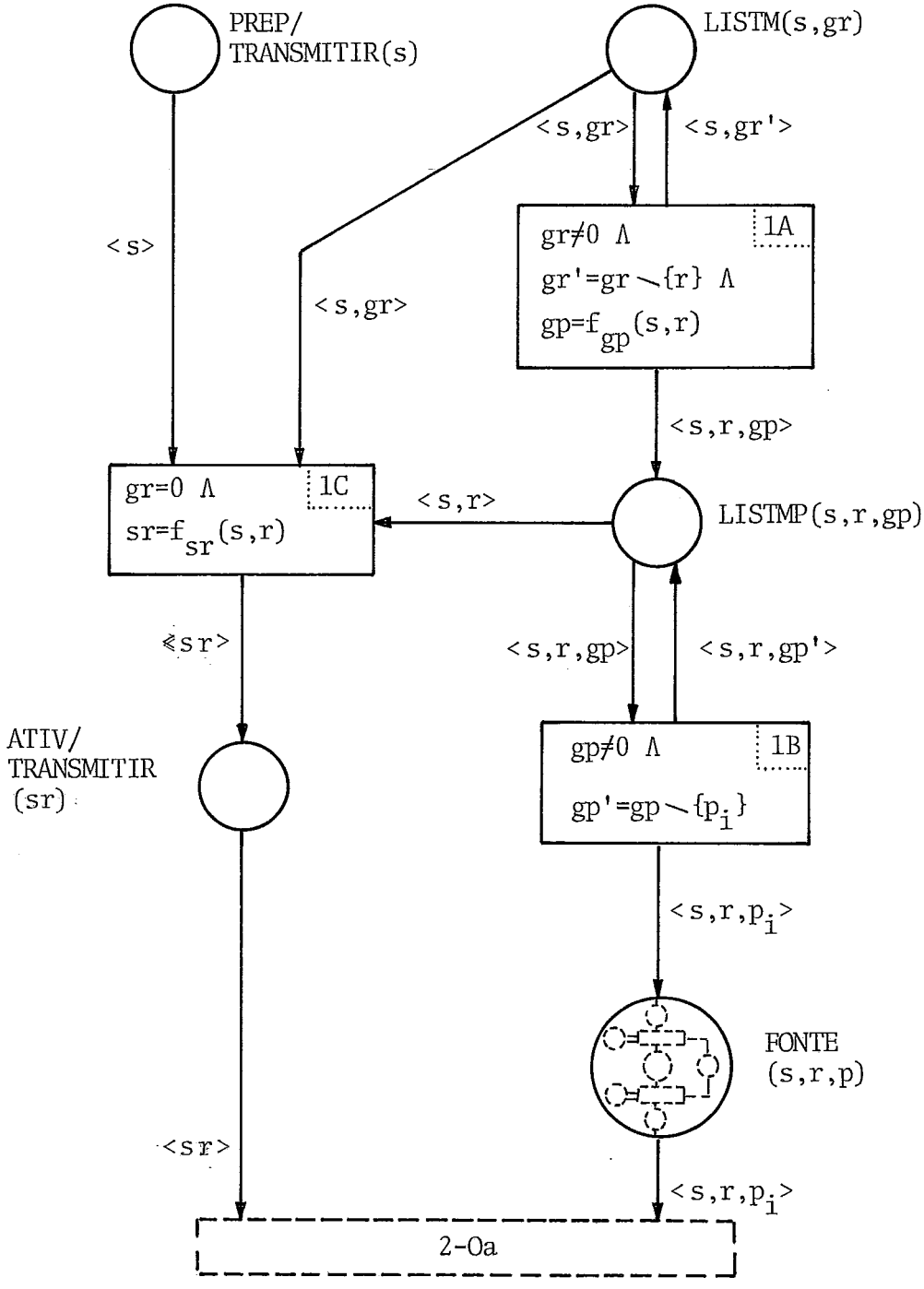


Fig. VII.2: Modelo parcial para mostrar a divisão de uma mensagem em prepacotes e a inscrição dos mesmos no predicado FONTE(s,r,p), representativo do canal de entrada no vértice/fonte.

No modelo anterior usamos a fórmula $m=f_m(s,r,p)$ para, por um lado, simplificar as inscrições na representação gráfica, e, por outro lado, representar uma mensagem já empacotada (no sentido descrito no parágrafo anterior). Agora, neste modelo, podemos juntar estas duas idéias (empacotar e simplificar) numa só expressão, obtendo

$$PAC = f_{pac}(s,r,p_i) \ .$$

Esta expressão substituirá, em todo o modelo (exceto em relação à cópia da mensagem guardada no V/F), a antiga fórmula $m=f_m(s,r,p)$. Usaremos, também, em vez da inscrição <m> nos arcos, a tupla <PAC>, exceto, novamente, no caso em que se trata da cópia da MENSAGEM, representado pelo predicado SEG.VIA.M(msg), onde usamos a tupla <msg>. Em relação a esta cópia, ignorava-se, no modelo anterior, se o m representara uma mensagem ou um pacote, já que não houve distinção entre estes dois termos. Agora, permitindo mensagens do tipo multipacotes, devemos considerar este fato e faremos isto, na transição 2, através da fórmula

$$msg = f_{msg}(s,r,\{p_i\}) \quad ,$$

$$1 \leq i \leq \text{número máximo dos pacotes por mensagem},$$

deixando, entretanto, em aberto se esta função simplesmente juntaria os PACOTES que circularão na subrede ou se faria algo a mais (como, p.ex., "filtrar" somente a informação básica da mensagem).

Mostramos, na fig. VII.3, um exemplo desta adaptação (em torno da transição 2) e remetemos o leitor ao modelo final (figura VII.8) para observar as modificações em sua totalidade, resumidas na tabela da figura VII.4.

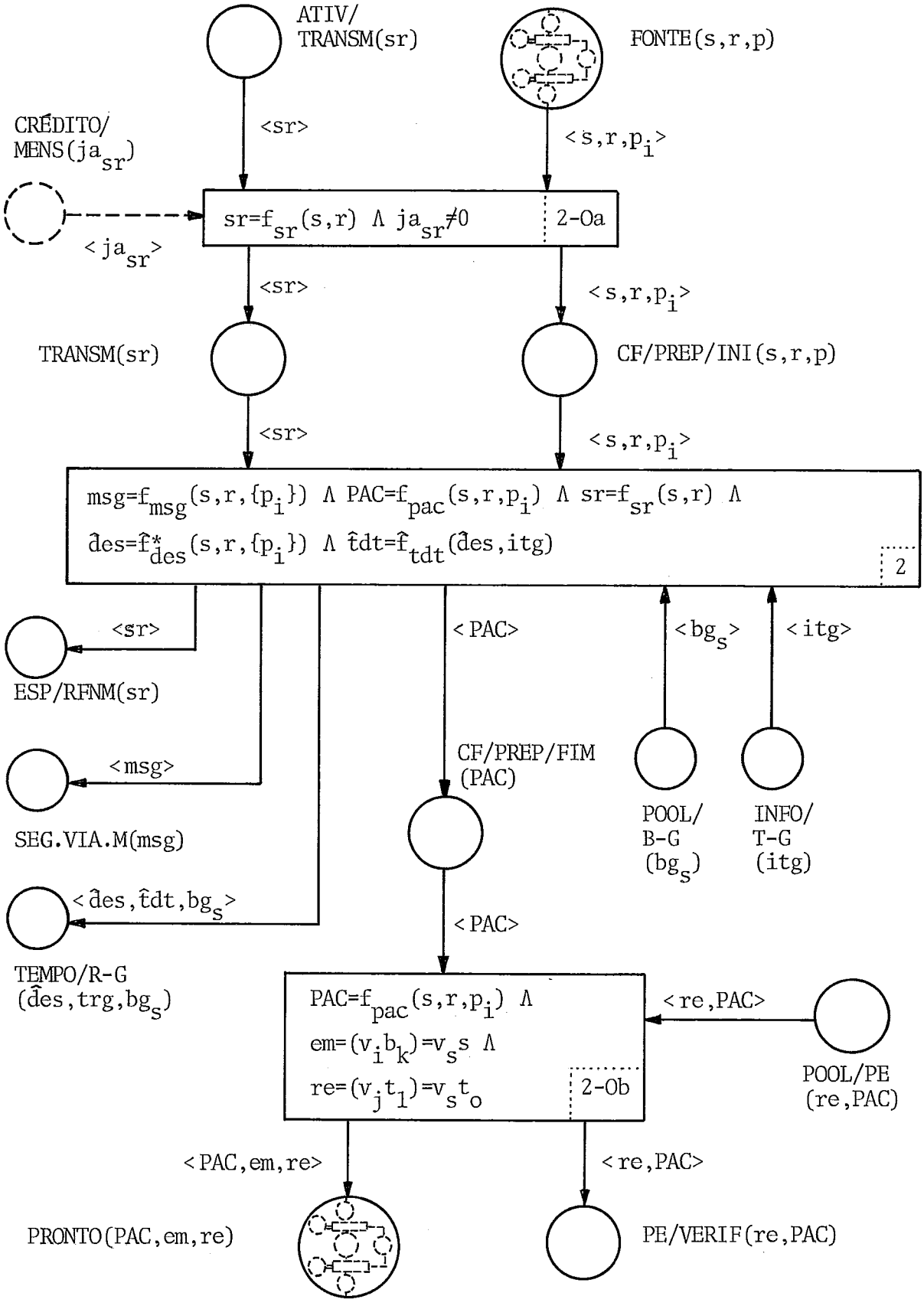


Fig. VII.3: Exemplo da adaptação do modelo anterior a mensagens do tipo multipacotes.

MODELO:	
atual	anterior
$PAC=f_{pac}(s,r,p_i)$	$m=f_m(s,r,p)$
$msg=f_{msg}(s,r,\{p_i\})$	$m=f_m(s,r,p)$
(PAC), <PAC>	(s,r,p), <s,r,p>
(re,PAC), <re,PAC>	(re,p), <re,p>
(em,PAC), <em,PAC>	(em,p), <em,p>
(PAC,em,re)	(m,em,re)
<PAC,em,re>	<m,em,re>
$\hat{des}=\hat{f}^*_{des}(s,r,\{p_i\})$	$\hat{des}=\hat{f}^*_{des}(s,r,p)$
(s), <s>	(u), <u>
(sr), <sr>	(u), <u>
$sr=f_{sr}(s,r)$	u=s
$s=f_s(u)$	u=s

Fig. VII.4: Modificações, em relação ao modelo número 3, obtido no capítulo VI, das inscrições nos arcos e predicados.

VII.3.b.4 - A remontagem da mensagem original a partir dos PACOTES recebidos no vértice/destinação.

Usando uma seqüência similar àquela empregada para dividir uma mensagem, podemos, agora, remontá-la. Assim, podemos dizer, em termos gerais que

- o último vértice S/F no caminho entre V/F e V/D, ou seja, aquele "artificial" que coincide com o vértice/destinação, será responsável por juntar todos os pacotes pertencentes a uma mesma mensagem;

- a UCP deste vértice/destinação é responsável pelo "desempacotamento", ou seja, pela retirada dos aditivos que foram necessários para tornar o pacote autônomo no sentido da subrede;

- o envio dos pacotes, digamos dos pospacotes, ao H/D acontece da mesma forma como a mensagem original (embora ainda dividida em pacotes);

- o host/destinação elimina os últimos "traços" da divisão, manda a mensagem ao usuário/receptor e, como responsável por ele, gera o RFNM associado.

Vejamos, estas idéias, passo a passo, em relação ao modelo na base de 'PrT-Net'.

O último vértice S/F, num caminho entre V/F e V/D, tem canais de entrada como se ele realmente fosse um vértice S/F. Já as saídas (figura V.11) se dividem em três partes: uma parte que leva aos outros vértices vizinhos S/F (canais b_1 até b_Q), uma outra parte que representa um artifício para igualá-la à primeira parte (canal $b_0 \hat{=} r'$) e, finalmente, uma parte que representa o vértice/destinação (canal r). A segunda parte, que interessa agora, é representada pelo predicado MONTAGEM (PAC,em,re) e tem como objetivo principal juntar os pacotes e colocá-los na ordem certa, já que eles podem chegar fora da seqüência original empregada. Também, deve se eliminar possíveis duplicatas, originando de eventuais retransmissões irregulares nos vértices intermediários e, no caso em que faltam pacotes, tomar as providências adequadas (como, p.ex., gerar uma mensagem de controle para investigar esta falha).

Em nosso modelo, que se restringe a indicar somente os acontecimentos acima de um certo nível lógico, esta ação da montagem, ou melhor, da premontagem, é modelada pelo conjunto de predicados MONTAGEM(PAC,em,re), que representa os pacotes envolvidos, e PS/MONTAR(em,PAC), que representa os responsáveis pelos pacotes na subrede de comunicações.

Queremos lembrar que não modelamos, por enquanto, a interdependência destes dois predicados (responsáveis pelo funcionamento da "entrada" de um pacote num vértice) porque ela reflete uma situação num nível físico/lógico não investigada por nós. Mas,

no caso em que se deseja, ao menos, mostrar esta interdependência, pode-se usar, ao lado de S-invariantes simples (usuários, mensagens, etc.), aqueles que relacionam, p.ex., "usuário e mensagens", "processadores e pacotes", "processadores, pacotes e enlaces", etc., (capítulo III.2.f.5).

Para chegar, no modelo, à UCP deste vértice/destinação, passamos pela transição 3-Oa. Esta transição indica o início da separação das ações da subrede daquelas da rede no sentido amplo, ou seja, o responsável pelos pacotes na subrede, representado pela tupla <em,PAC>, volta ao 'pool', predicado POOL/PS(em,PAC), e o usuário/receptor (p.ex., o host/destinação) assumirá, em torno da transição 3, esta responsabilidade.

Assim, a transição 3-Oa (com a expressão $PAC=f_{pac}(s,r,p_i)$, em vez de $m=f_m(.)$) serve, principalmente, para modelar a colocação, pacote por pacote, na UCP, onde serão retiradas as partes de controle, CRC, FLAG, etc. Esta eliminação dos aditivos é indicada pela fórmula, na transição 3, $(s,r,p_i)=f_{pac}^{-1}(PAC)$, que, assim, força a recuperação da parte essencial do pacote PAC, ou seja, a parte que contém "informação" associada à mensagem.

Ao mesmo tempo em que a UCP realiza esta recuperação, precisa-se ativar o responsável, a partir daqui, pela recepção da mensagem. Nas primeiras versões do modelo usamos, para indicar esta ativação, uma expressão bastante vaga, ou seja, u=r, que não mostrou muito claramente o que aconteceu. Nesta versão atual, achamos mais conveniente mostrar a relação mais imediata, a saber, que a ativação depende das informações contidas no pacote PAC que chegou a este vértice/destinação. Assim, a fórmula $r=f_r(PAC)$ indica mais precisamente esta relação e vale lembrar, ainda, que

$$r \in ER, \quad \text{onde} \quad ER = EndR = \{r_1, r_2, \ldots, r_m\} ,$$

o que significa que cada usuário pode se tornar receptor. O uso de um predicado POOL/RECEPTOR(r), em vez de PASSIVO(u), reforça ainda mais esta idéia.

Em princípio, podemos nos contentar com esta modificação no modelo. Acontece, porém, que, no caso em que chegam mensagens (pacotes) de diferentes emissores a este mesmo receptor, haverá uma competição pelo acesso à UCP. O resultado: somente uma mensagem consegue obter um responsável pela recepção (veja a fórmula $r=f_r(PAC)$), obrigando as outras a esperar até que o receptor volte ao 'pool'.

Para evitar isto, usaremos um artifício similar aquele usado para escolher os (sub)responsáveis pelas emissões, ou seja, usaremos, em vez da fórmula mencionada, a seguinte:

$$rs = f_{rs}(PAC,r) \wedge r' = r \setminus \{rs\} .$$

Sem nos preocupar, por enquanto, com o número máximo possível dos (sub)responsáveis e ficando no nível lógico dos acontecimentos, podemos afirmar que, desta maneira, permitimos a recepção e tratamento "simultâneo" de mensagens de vários emissores neste mesmo receptor.

A figura VII.5 ilustra estas idéias.

O próximo ponto a resolver é a verificação do despacho para o host/destinação que representa o usuário/receptor. Uma vez que a UCP conseguiu "filtrar" somente a parte importante do pacote (i.e., a parte que contém INFORMAÇÃO), os pacotes chegam ao canal r que liga o vértice/destinação ao host associado. No modelo, o predicado DESTINAÇÃO(s,r,p) representa este fato, sendo realizado, nele, o agrupamento final dos pacotes pertencentes à mesma mensagem.

Depois, aceitos pelo H/D, pode-se pensar em gerar o RFNM ('request for next message') que, como mostra a expressão antiga $rfnm_{sr}=f_{rf}(sc,rc)$ na transição 3D-1, se deve referir, agora, realmente ao par "s-r" dos usuários.

Isto pode ser visto na figura VII.6 e observe, ainda, que a expressão, na transição 3D-1, $gr^*=gr\bigcup\{rs\}$ significa que o gru-

po dos receptores das mensagens de um determinado emissor está sendo recomposto (transmissão completada), sendo irrelevante quem foi o "subresponsável" pela emissão. Ao que toca a expressão $r^{*}=r \cup \{rs\}$, podemos alertar que será ela, em próximas tentativas de modelagem, um dos elementos responsáveis pela gerência (avaliação, permissões, etc.) de um crédito entre um determinado par "s-r". Sem insistir em detalhes, indicaremos esta idéia, na transição 3D-1 (em vez da antiga expressão $rfnm_{sr}=f_{rf}(.)$), através da fórmula $rfnm_{sr}=f_{rf}(sc,rc,r^{*})$.

Falta ainda mostrar a liberação da cópia da mensagem, guardada no vértice/fonte, e a conseqüente liberação do crédito, em relação ao par "s-r", para poder enviar uma outra mensagem. Aí, não temos muitos problemas: só aplicando as idéias desenvolvidas neste trabalho e executando as conseqüentes pequenas modificações em relação aos termos u (que vira sr) e m (que se torna msg), chegamos ao modelo final mostrado na figura VII.8.

Assim chegamos, finalmente, à geração da resposta ao usuário em resposta ao "pedido" que ele fez. Continuando ignorando a modelagem em torno de ums resposta negativa, temos que ver as condições necessárias para chegar ao predicado FEITO(u) que representaria uma "resposta afirmativa". O que é importante ressaltar é o fato que esta resposta é dada em função do sucesso de TODO o grupo associado ao emissor em questão (sendo isto um conceito que, certamente, deveria ser revisto nas próximas tentativas de modelagem!). Conseqüentemente, a lista dos receptores do emissor s (predicado LISTR(s,gr)) é esvaziada em função dos receptores "satisfeitos" que, por sua vez, são liberados através da informação que a transmissão foi bem sucedida (isto é, basicamente, quando chegar o RFNM associado).

Observe as idéias dos últimos dois parágrafos na figura VII.7 e preste atenção às modificações nas diversas inscrições que, embora graficamente pequena, têm um significativo bastante importante no contexto de mensagens do tipo multipacote.

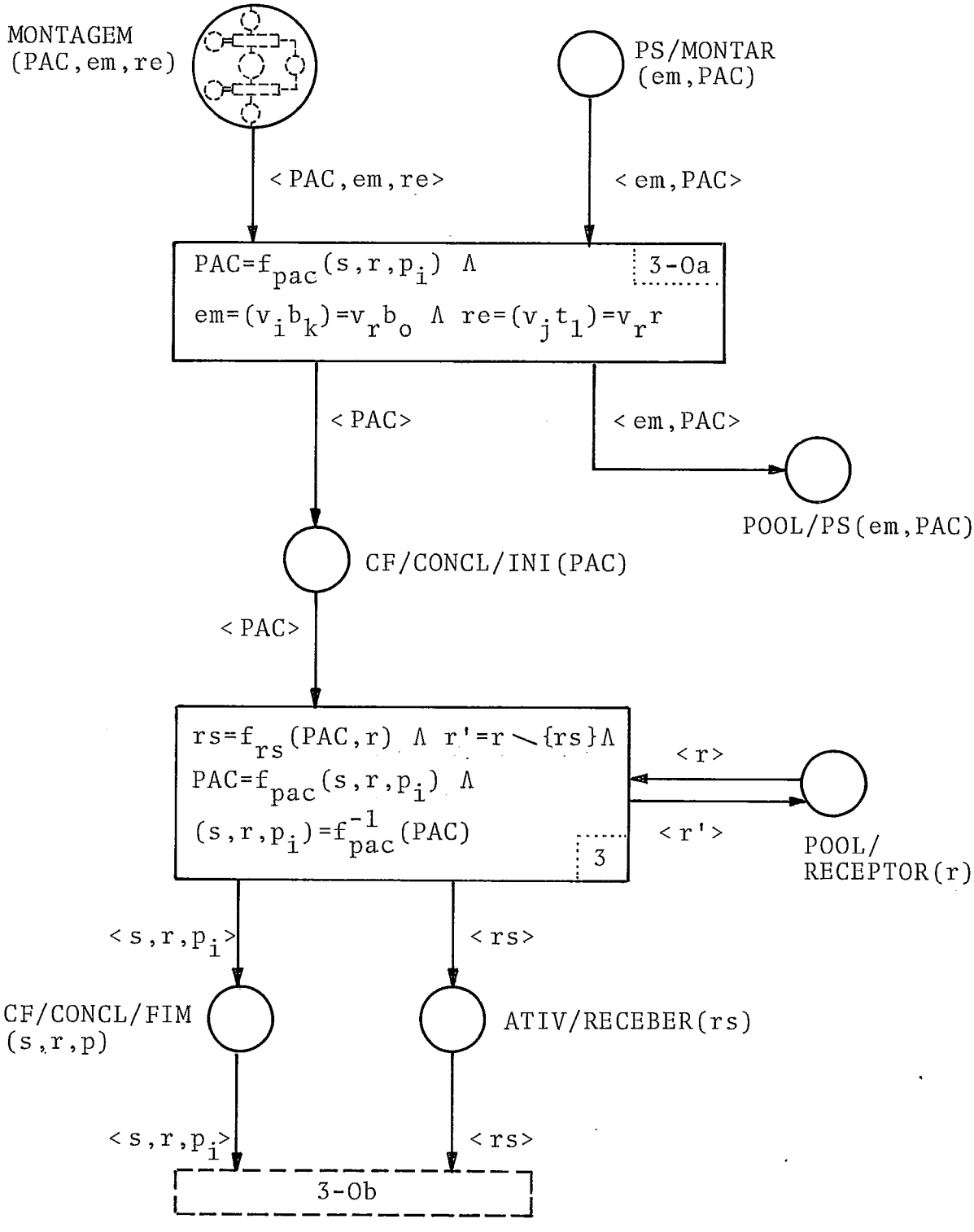


Fig. VII.5: Modelo modificado parcial em torno da remontagem de uma mensagem no vértice/destinação.

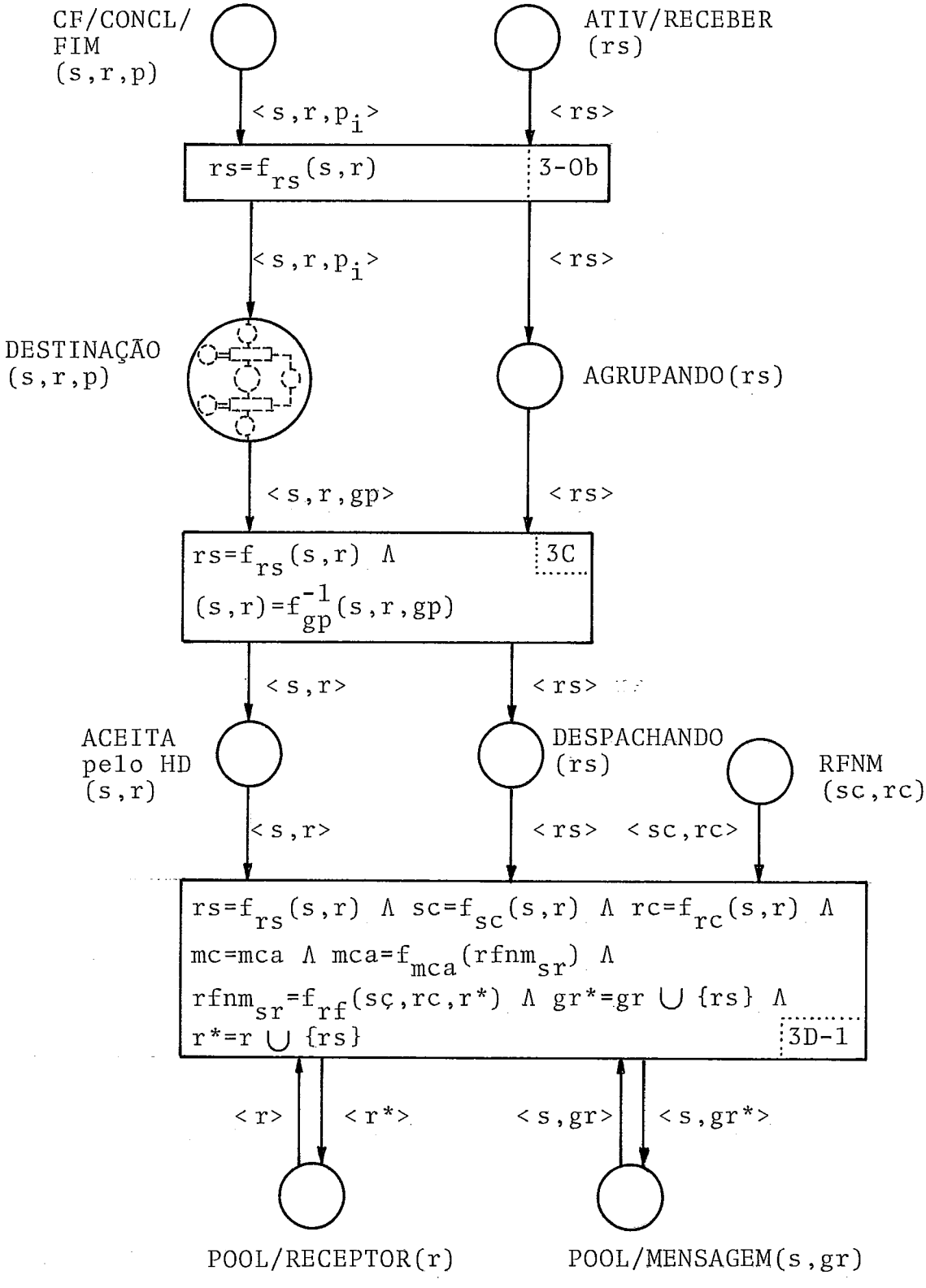


Fig. VII.6: Modelo modificado parcial em torno do despacho final para o usuário/receptor.

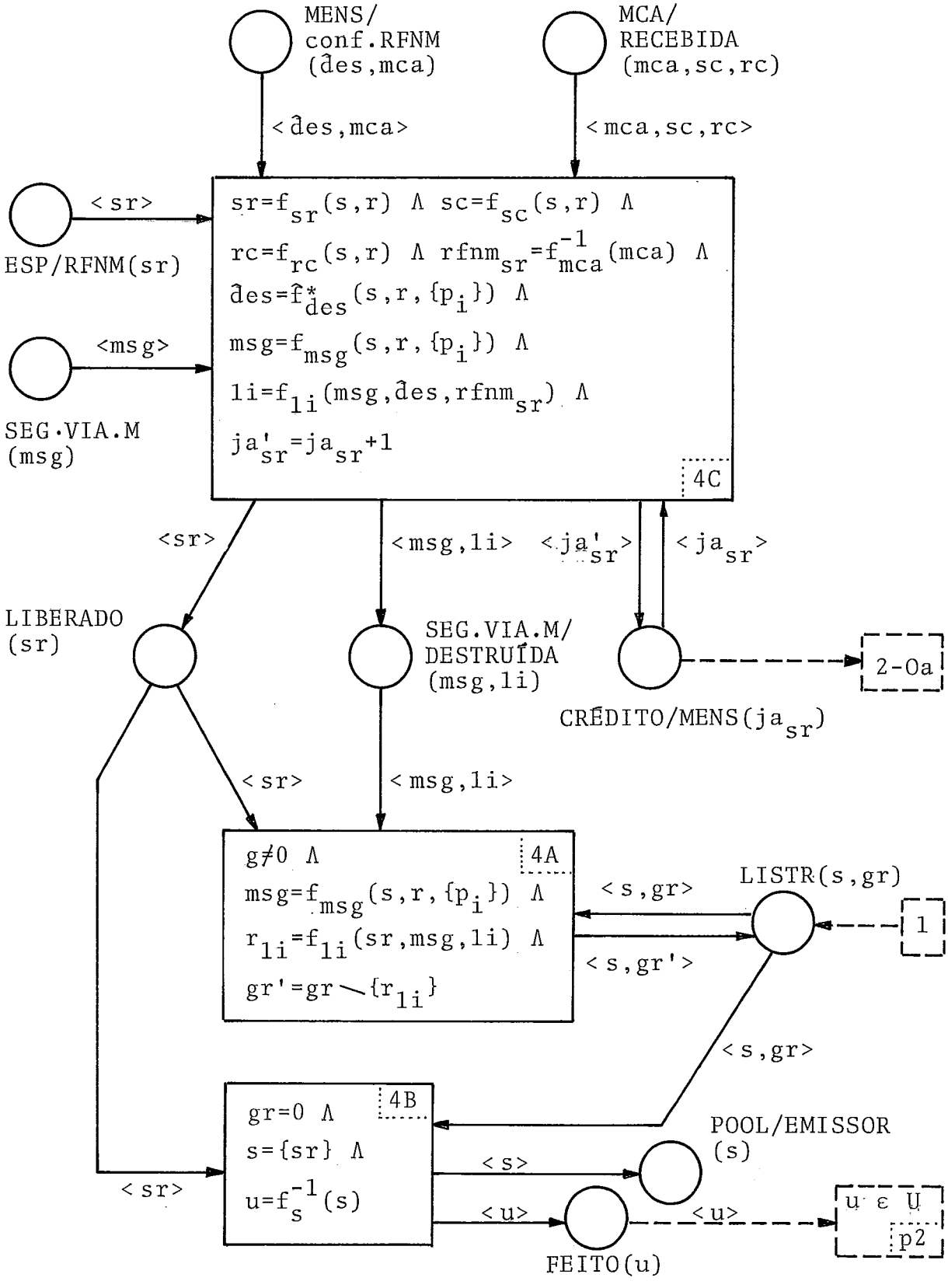


Fig. VII.7: Modelo modificado parcial em torno da liberação das cópias e da geração de uma resposta afirmativa ao usuário.

VII.4 - O modelo número 4, o final deste trabalho, na base de 'PrT-Nets', da transmissão de mensagens do tipo multipacotes numa rede de computadores.

Mostraremos, na figura VII.8, o esquema gráfico do modelo final deste trabalho; na base de 'PrT-Nets', que representa a transmissão de mensagens do tipo multipacotes numa rede de computadores. Este modelo número 4 obtido, usando aquele realizado anteriormente (modelo número 3, figura VI.14) e incluindo as modificações necessárias, representa, bastante fielmente, a situação da versão 1 do ARPANET, ou seja, inclui a modelagem dos vértices intermediários e as conseqüências em relação ao ACK, NAK, 'time-out' e 'check-time', assim como, a partir deste trabalho, os aspectos em relação a mensagens simples ou de multipacotes.

O esquema gráfico é dividido em partes, ou seja,

- fig- VII.8.a - emissão inicial em torno do V/F;
- fig. VII.8.b - integração do 'check-time' e resposta ao usuário;
- fig. VII.8.c - emissão nos vértices intermediários;
- fig. VII.8.d - recepção bem sucedida nos vértices intermediários;
- fig. VII.8.e - retransmissão a partir dos vértices intermediários;
- fig. VII.8.f - recepção e remontagem no vértice/destinação.

Usando a experiência obtida nos capítulos IV até VI, acrescentamos, na figura VII.9, uma lista contendo os principais predicados agrupados segundo as suas "naturezas", seguida pela figura VII.10 que contém uma tabela das transições e suas respectivas fórmulas.

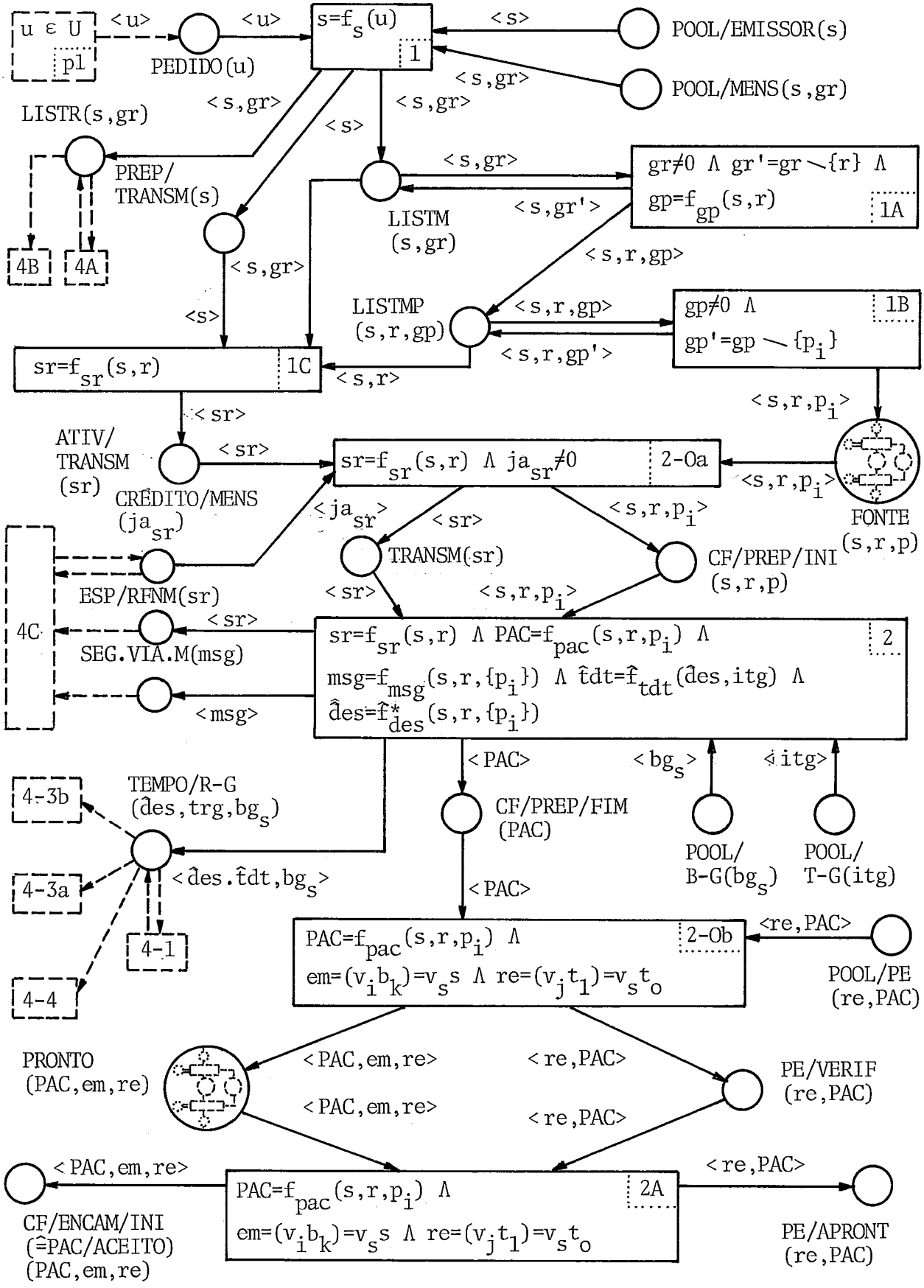


Fig. VII.8.a: Modelo nº 4 (transmissão de mensagens do tipo multipacotes em redes de computadores, incluindo vértices intermediários, ACK, NAK, 'time-out' e 'check-time'). Parte 1: emissão inicial em torno do vértice/fonte.

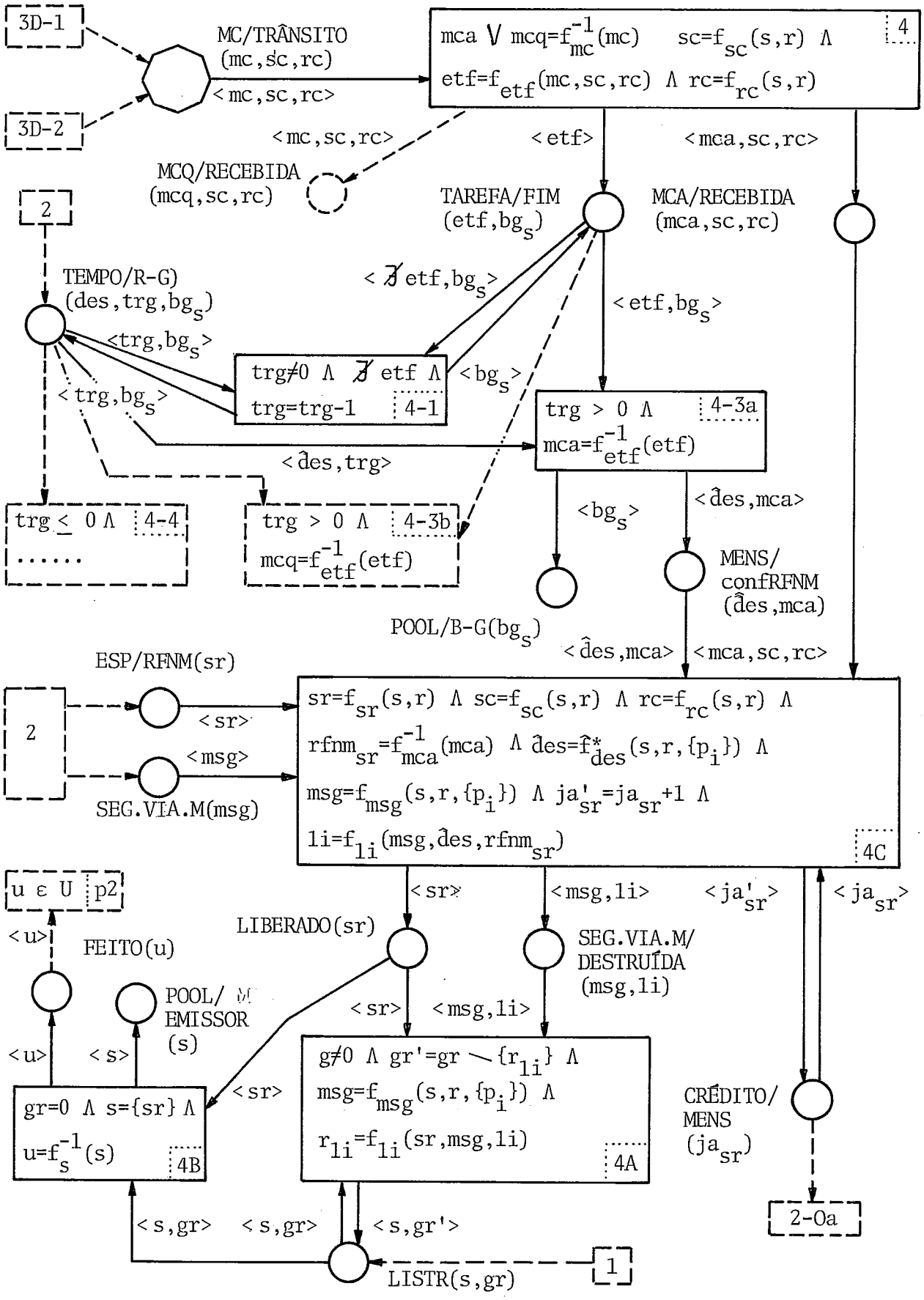


Fig. VII.8.b: Modelo nº 4. Parte 2: integração do 'check-time' e da resposta ao usuário no modelo final.

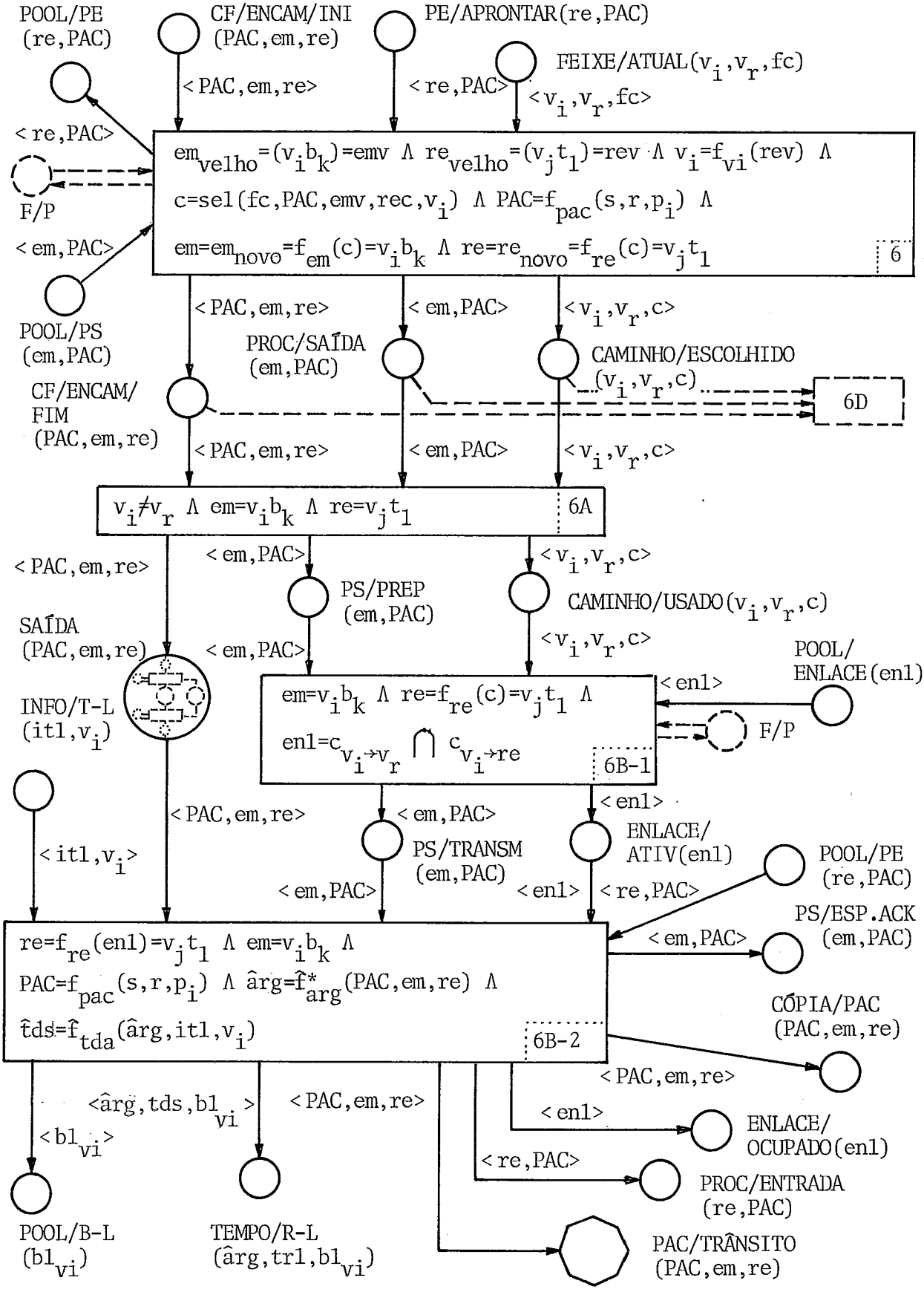


Fig. VII.8.c: Modelo nº 4. Parte 3: emissão nos vértices intermediários.

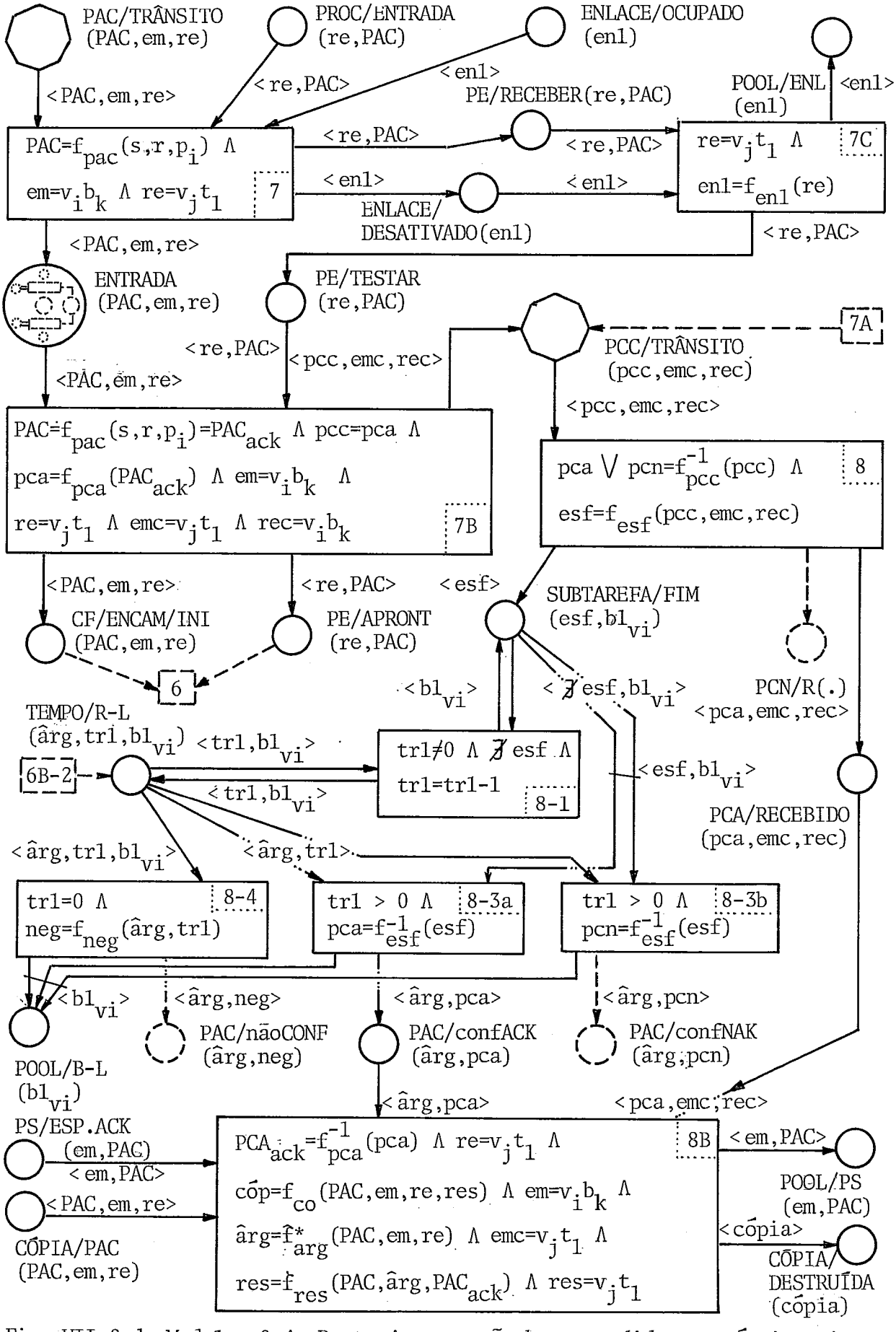


Fig. VII.8.d: Modelo nº 4. Parte 4: recepção bem sucedida nos vértices intermediários.

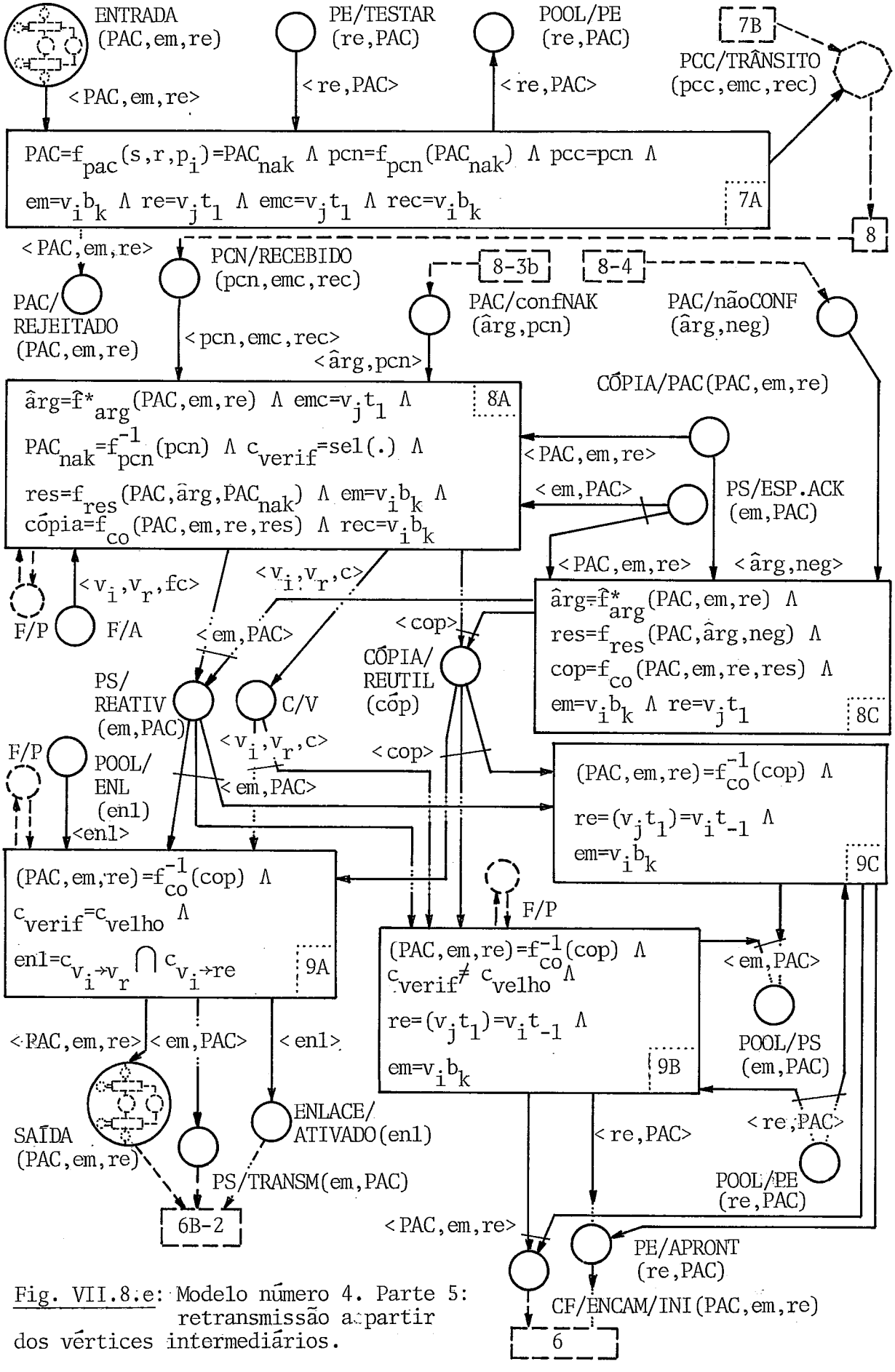


Fig. VII.8.e: Modelo número 4. Parte 5: retransmissão a partir dos vértices intermediários.

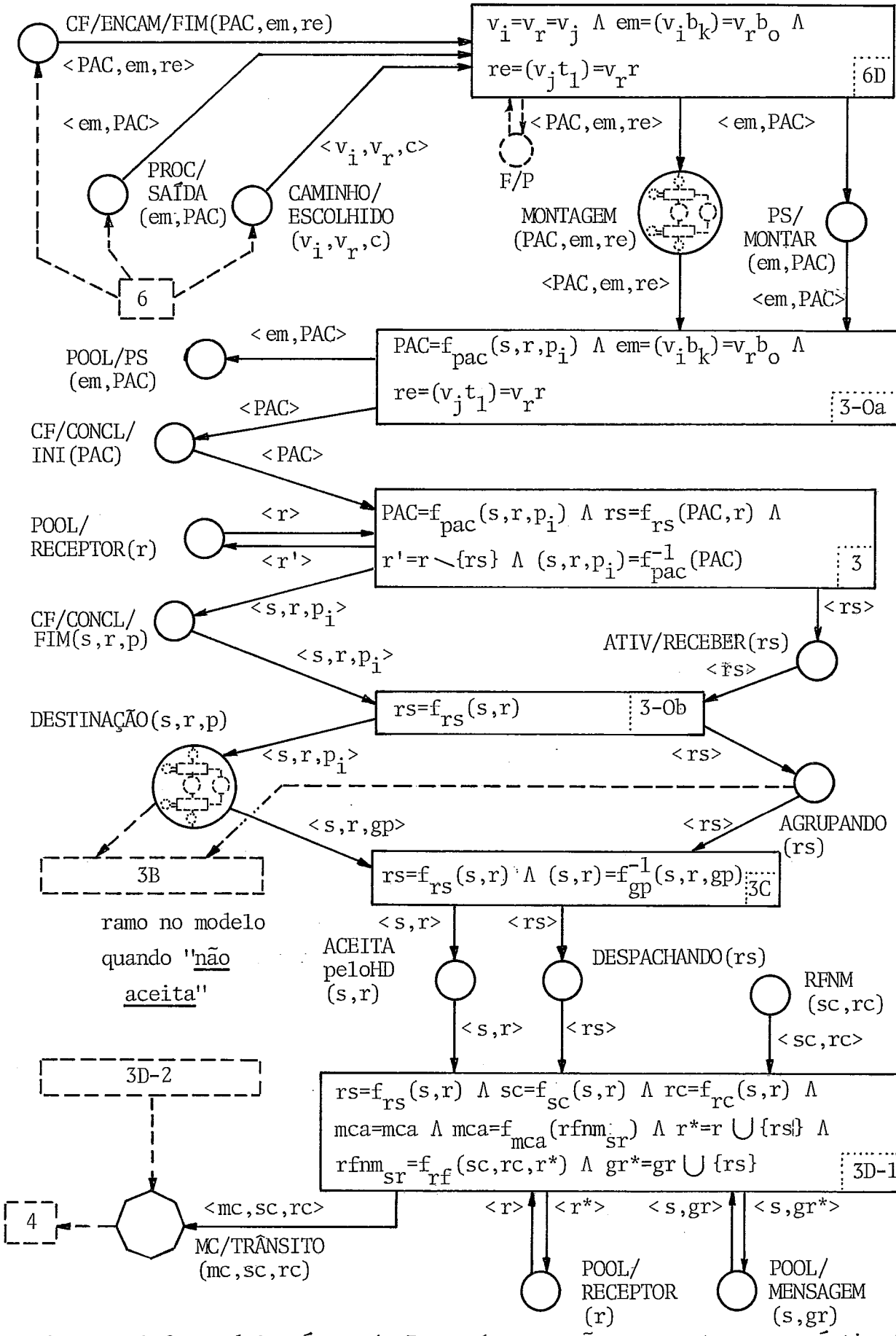


Fig. VII.8.f: Modelo número 4. Parte 6: recepção e remontagem no vértice/ /destinação.

predicados: MENSAGEM, PACOTES	relação causal com as transições: - ≙ precondição	trans	+ ≙ poscondição	trans
ACEITA/peloHD (s,r)	- <s,r>	3D-1	+ <s,r>	3C
CÓPIA/DESTRUÍDA (cópia)	indefinido		+ <cópia>	8B
CÓPIA/PAC (PAC,em,re)	- <PAC,em,re> - <PAC,em,re> - <PAC,em,re>	8A 8B 8C	+ <PAC,em,re>	6B-2
CÓPIA/REUTILIZADA (cópia)	- <cópia> - <cópia> - <cópia>	9A 9B 9C	+ <cópia> + <cópia>	8A 8C
DESTINAÇÃO (s,r,p)	- <s,r,p_i>	3C	+ <s,r,p_i>	3-Ob
ENTRADA (PAC,em,re)	- <PAC,em,re> - <PAC,em,re>	7A 7B	+ <PAC,em,re>	7
FONTE (s,r,p)	- <s,r,p_i>	2-Oa	+ <s,r,p_i>	1B
LISTM (s;gr)	- <s,gr> - <s,gr>	1A 1C	+ <s,gr> + <s,gr'>	1 1A
LISTMP (s,r,gp)	- <s,r,gp> - <s,r>	1B 1C	+ <s,r,gp> + <s,r,gp'>	1A 1B
LISTR (s,gr)	- <s,gr> - <s,gr>	4A 4B	+ <s,gr> + <s,gr'>	1 4A
MONTAGEM (PAC,em,re)	- <PAC,em,re>	3-Oa	+ <PAC,em,re>	6D
PAC/ACEITO (PAC,em,re)	veja: CF/ENCAM/INI		veja: CF/ENCAM/INI	
PAC/REJEITADO (PAC,em,re)	indefinido		+ <PAC,em,re>	7A
PAC/TRÂNSITO (PAC,em,re)	- <PAC,em,re>	7	+ <PAC,em,re>	6B-2
POOL/MENSAGEM (s,gr)	- <s,gr> - <s,gr>	1 3D-1	+ <s,gr*>	3D-1
PRONTO (PAC,em,re)	- <PAC,em,re>	2A	+ <PAC,em,re>	2-Ob
SAÍDA (PAC,em,re)	- <PAC,em,re>	6B-2	+ <PAC,em,re> + <PAC,em,re>	6A 9A
SEG.VIA.M (msg)	- <msg>	4C	+ <msg>	2
SEG.VIA.M/ DESTRUÍDA(msg,li)	- <msg,li>	4A	+ <msg,li>	4C

Fig. VII.9.a: Lista dos principais predicados do modelo número 4 (figura VII.8), agrupados segundo as suas diferentes "naturezas". Parte 1: MENSAGEM, PACOTES.

predicados: PACOTES - UCP	relação causal com as transições: - ≙ precondição	trans	- ≙ poscondição	trans
CF/CONCL/INI (PAC)	- <PAC>	3	+ <PAC>	3-Oa
CF/CONCL/FIM (s,r,p)	- <s,r,p_i>	3-Ob	+ <s,r,p_i>	3
CF/ENCAM/INI (≙PAC/ACEITO) (PAC,em,re)	- <PAC,em,re>	6	+ <PAC,em,re> + <PAC,em,re> + <PAC,em,re> + <PAC,em,re>	2A 7B 9B 9C
CF/ENCAM/FIM (PAC,em,re)	- <PAC,em,re> - <PAC,em,re>	6A 6D	+ <PAC,em,re>	6
CF/PREP/INI (s,r,p)	- <s,r,p_i>	2	+ <s,r,p_i>	2-Oa
CF/PREP/FIM (PAC)	- <PAC>	2-Ob	+ <PAC>	2
predicados: USUÁRIOS				
AGRUPANDO(rs)	- <rs>	3C	+ <rs>	3-Ob
ATIV/RECEBER(rs)	- <rs>	3-Ob	+ <rs>	3
ATIV/TRANSM(s)	- <s>	2-Oa	+ <s>	1C
DESPACHANDO(rs)	- <rs>	3D-1	+ <rs>	3C
ESP/RENM(sr)	- <sr>	4C	+ <sr>	2
FEITO(u)	- <u>	p2	+ <u>	4B
LIBERADO(sr)	- <sr>	4A	+ <sr>	4C
PEDIDO(u)	- <u>	1	+ <u>	p1
POOL/EMISSOR(s)	- <s>	1	+ <s>	4B
POOL/RECEPTOR(r)	- <r> - <r*>	3 3D-1	+ <r'> + <r>	3 3D-1
PREP/TRANSM(s)	- <s>	1C	+ <s>	1
TRANSMITIR(sr)	- <sr>	2	+ <sr>	2-Oa

Fig. VII.9.b: Lista dos principais predicados do modelo número 4, (figura VII.8), agrupados segundo as suas diferentes "naturezas":
Parte 2: PACOTES - UCP
Parte 3: USUÁRIOS

predicados: PROCESSADOR/ENTR.	relação causal com as transições: - ≙ precondição	trans	+ ≙ poscondição	trans
PE/APRONTAR (re,PAC)	- <re,PAC>	6	+ <re,PAC> + <re,PAC> + <re,PAC> + <re,PAC>	2A 7B 9B 9C
PE/RECEBER (re,PAC)	- <re,PAC>	7C	+ <re,PAC>	7
PE/TESTAR (re,PAC)	- <re,PAC> - <re,PAC>	7A 7B	+ <re,PAC>	7C
PE/VERIFICAR (re,PAC)	- <re,PAC>	2A	+ <re,PAC>	2-Ob
POOL/PE (re,PAC)	- <re,PAC> - <re,PAC> - <re,PAC> - <re,PAC>	2-Ob 6B-2 9B 9C	+ <re,PAC> + <re,PAC>	6 7A
PROC/ENTRADA (re,PAC)	- <re,PAC>	7	+ <re,PAC>	6B-2
predicados: PROCESSADOR/SAÍDA				
POOL/PS (em,PAC)	- <em,PAC>	6	+ <em,PAC> + <em,PAC> + <em,PAC> + <em,PAC>	3-Oa 8B 9B 9C
PROC/SAÍDA (em,PAC)	- <em,PAC> - <em,PAC>	6A 6B	+ <em,PAC>	6
PS/ESP.ACK (em,PAC)	- <em,PAC> - <em,PAC> - <em,PAC>	8A 8B 8C	+ <em,PAC>	6B-2
PS/MONTAR (em,PAC)	- <em,PAC>	3-Oa	+ <em,PAC>	6D
PS/PREPARANDO (em,PAC)	- <em,PAC>	6B-1	+ <em,PAC>	6A
PS/REATIVANDO (em,PAC)	- <em,PAC> - <em,PAC> - <em,PAC>	9A 9B 9C	+ <em,PAC> + <em,PAC>	8A 8C
PS/TRANSMITINDO (em,PAC)	- <em,PAC>	6B-2	+ <em,PAC> + <em,PAC>	6B-1 9A

Fig. VII.9.c: Lista dos principais predicados do modelo número 4 (figura VII.8) agrupados segundo as suas diferentes "naturezas":
Parte 4: PROCESSADORES DE ENTRADA
Parte 5: PROCESSADORES DE SAÍDA.

predicados: CAMINHOS, ENLACES	relação causal com as transições: - ≙ precondição	trans	+ ≙ poscondição	trans
CAMINHO/ESCOLHIDO (v_i,v_r,c)	- <v_i,v_r,c> - <v_i,v_r,c>	6A 6D	+ <v_i,v_r,c>	6
CAMINHO/VERIFICADO (v_i,v_r,c)	- <v_i,v_r,c> - <v_i,v_r,c>	9A 9C	+ <v_i,v_r,c>	8A
CAMINHO/USADO (v_i,v_r,c)	- <v_i,v_r,c>	6B-1	+ <v_i,v_r,c>	6A
ENLACE/ATIVADO (enl)	- <enl>	6B-2	+ <enl> + <enl>	6B-1 9A
ENLACE/DESATIVADO (enl)	- <enl>	7C	+ <enl>	7
ENLACE/OCUPADO (enl)	- <enl>	7	+ <enl>	6B-2
FEIXE/ATUAL (v_i,v_r,fc)	- <v_i,v_r,fc> - <v_i,v_r,fc>	6 8A	indefinido neste nível do modelo	
POOL/ENLACE (enl)	- <enl> - <enl>	6B-1 9A	+ <enl>	7C
predicados: MENS. de CONTROLE				
MCA/RECEBIDA (mca,sc,rc)	- <mca,sc,rc>	4C	+ <mca,sc,rc>	4
MCQ/RECEBIDA (mcq,sc,rc)	indefinido		+ <mcq,sc,rc>	4
MC/TRÂNSITO (mc,sc,rc)	- <mc,sc,rc>	4	+ <mc,sc,rc>	3D-1
PCA/RECEBIDO (pca,emc,rec)	- <pca,emc,rec>	8B	+ <pca,emc,rec>	8
PCC/TRÂNSITO (pcc,emc,rec)	- <pcc,emc,rec>	8	+ <pcc,emc,rec> + <pcc,emc,rec>	7A 7B
PCN/RECEBIDO (pcn,emc,rec)	- <pcn,emc,rec>	8A	+ <pcn,emc,rec>	8
RFNM(sc,rc)	- <sc,rc>	3D-1	indefinido	

Fig. VII.9.d: Lista dos principais predicados do modelo número 4 (figura VII.8), agrupados segundo as suas diferentes "naturezas":
Parte 6: CAMINHOS E ENLACES
Parte 7: MENSAGENS E PACOTES DE CONTROLE.

predicados: AUXILIARES	relaçao causal com as transições: - ≙ precondição	trans	+ ≙ poscondição	trans
CRÉDITO/MENSAGEM (ja_{sr})	- $<ja_{sr}>$	2-Oa	+ $<ja'_{sr}>$	4C
	- $<ja_{sr}>$	4C		
INFO/TEMPO-GLOBAL (itg)	- <itg>	2	indefinido	
INFO/TEMPO-LOCAL (itl,v_i)	- $<itl,v_i>$	6B-2	indefinido	
MENS/conf.RFNM ($\hat{d}es$,mca)	- $<\hat{d}es$,mca>	4C	+ $<\hat{d}es$,mca>	4-3a
PAC/conf.ACK ($\hat{a}rg$,pca)	- $<\hat{a}rg$,pca>	8B	+ $<\hat{a}rg$,pca>	8-3a
PAC/conf.NAK ($\hat{a}rg$,pcn)	- $<\hat{a}rg$,pcn>	8A	+ $<\hat{a}rg$,pcn>	8-3b
PAC/nãoCONF ($\hat{a}rg$,neg)	- $<\hat{a}rg$,neg>	8C	+ $<\hat{a}rg$,neg>	8-4
POOL/BLOQUEADOR-GLOBAL (bg_s)	- $<bg_s>$	2	+ $<bg_s>$	4-3a
POOL/BLOQUEADOR-LOCAL(bl_{vi})	- $<bl_{vi}>$	6B-2	+ $<bl_{vi}>$	8-3a
			+ $<bl_{vi}>$	8-3b
			+ $<bl_{vi}>$	8-4
SUBTAREFA/FIM (esf,bl_{vi})	- $<\not\exists esf,bl_{vi}>$	8-1	+ <esf>	8
	- $<esf,bl_{vi}>$	8-3a	+ $<bl_{vi}>$	8-1
	- $<esf,bl_{vi}>$	8-3b		
TAREFA/FIM (etf,bg_s)	- $<\not\exists etf,bg_s>$	4-1	+ <etf>	4
	- $<etf,bg_s>$	4-3a	+ $<bg_s>$	4-1
TEMPO/RESTANTE-GLOBAL ($\hat{d}es,trg,bg_s$)	- $<trg,bg_s>$	4-1	+ $<\hat{d}es,\hat{t}dt,bg_s>$	2
	- $<\hat{d}es,trg>$	4-3a	+ $<trg,bg_s>$	4-1
TEMPO/RESTANTE-LOCAL ($\hat{a}rg,trl,bl_{vi}$)	- $<trl,bl_{vi}>$	8-1	+ $<\hat{a}rg,\hat{t}ds,bl_{vi}>$	6B-2
	- $<\hat{a}rg,trl>$	8-3a	+ $<trl,bl_{vi}>$	8-1
	- $<\hat{a}rg,trl>$	8-3b		
	- $<\hat{a}rg,trl,bl_{vi}>$	8-4		

Fig. VII.9.e: Lista dos principais predicados do modelo número 4 (figura VII.8), agrupados segundo as suas diferentes "naturezas": Parte 8: AUXILIARES.

T R A N S I Ç Õ E S	
nº	fórmulas
p1	$u \in U$
p2	$u \in U$
1	$s=f_s(u)$
1A	$gr\neq 0 \wedge gr'=gr \setminus \{r\} \wedge gp=f_{gp}(s,r)$
1B	$gp\neq 0 \wedge gr'=gp \setminus \{p_i\}$
1C	$sr=f_{sr}(s,r) \wedge gr=0$
2	$sr=f_{sr}(s,r) \wedge PAC=f_{pac}(s,r,p_i) \wedge msg=f_{msg}(s,r,\{p_i\}) \wedge$ $\hat{des}=\hat{f}^*_{des}(s,r,\{p_i\}) \wedge \hat{tdt}=\hat{f}_{tdt}(\hat{des},itg)$
2-Oa	$ja_{sr}\neq 0 \wedge sr=f_{sr}(s,r)$
2-Ob	$PAC=f_{pac}(s,r,p_i) \wedge em=(v_i b_k)=v_s s \wedge re=(v_j t_l)=v_s t_o$
2A	$PAC=f_{pac}(s,r,p_i) \wedge em=(v_i b_k)=v_s s \wedge re=(v_j t_l)=v_s t_o$
3	$rs=f_{rs}(PAC,r) \wedge r'=r \setminus \{rs\} \wedge PAC=f_{pac}(s,r,p_i) \wedge$ $(s,r,p_i)=f^{-1}_{pac}(PAC)$
3-Oa	$PAC=f_{pac}(s,r,p_i) \wedge em=(v_i b_k)=v_r b_o \wedge re=(v_j t_l)=v_r r$
3-Ob	$rs=f_{sr}(s,r)$
3C	$rs=f_{rs}(s,r) \wedge (s,r)=f^{-1}_{gp}(s,r,gp)$
3D-1	$rs=f_{rs}(s,r) \wedge gr^*=gr \cup \{rs\} \wedge mca=f_{mac}(rfnm_{sr}) \wedge r^*=r \cup \{rs\} \wedge$ $rc=f_{rc}(s,r) \wedge mc=mca \wedge rfnm_{sr}=f_{rf}(sc,rc) \wedge sc=f_{sc}(s,r)$
4	$sc=f_{sc}(s,r) \wedge rc=f_{rc}(s,r) \wedge mca \vee mcq=f^{-1}_{mc}(mc) \wedge$ $etf=f_{etf}(mc,sc,rc)$
4-1	$trg\neq 0 \wedge \nexists\, etf \wedge trg=trg-1$
4-3a	$trg>0 \wedge mca=f^{-1}_{etf}(etf)$
4-3b	$trg>0 \wedge mcq=f^{-1}_{etf}(etf)$
4-4	$trg=0 \wedge$ ----

Fig. VII.10: Tabela das transições do modelo número 4 (figura VII.8), e suas principais fórmulas, envolvidas diretamente no controle de fluxo.

T R A N S I Ç Õ E S	
nº	fórmulas
4A	$gr \neq 0 \wedge gr' = gr \setminus \{r_{li}\} \wedge msg = f_{msg}(s,r,\{p_i\}) \wedge r_{li} = f_{li}(sr,msg,li)$
4B	$gr = 0 \wedge s = \{sr\} \wedge u = f_s^{-1}(s)$
4C	$sr = f_{sr}(s,r) \wedge sc = f_{sc}(s,r) \wedge rc = f_{rc}(s,r) \wedge ja'_{sr} = ja_{sr} + 1 \wedge$ $rfnm_{sr} = f_{mca}^{-1}(mca) \wedge \hat{des} = \hat{f}^*_{des}(s,r,\{p_i\}) \wedge$ $msg = f_{msg}(s,r,\{p_i\}) \wedge li = f_{li}(msg,\hat{des},rfnm_{sr})$
6	$em_{velho} = (v_i b_k) = emv \wedge re_{velho} = (v_j t_l) = rev \wedge v_i = f_{vi}(rev) \wedge$ $PAC = f_{pac}(s,r,p_i) \wedge c = sel(fc,PAC,emv,rev,v_i) \wedge$ $[fcp' = fcp \cup (fc \setminus \{c\})] \wedge em = em_{novo} = f_{em}(c) = v_i b_k \wedge$ $re = re_{novo} = f_{re}(c) = v_j t_l$
6A	$v_i \neq v_r \wedge em = v_i b_k \wedge re = v_j t_l$
6B-1	$em = v_i b_k \wedge re = f_{re}(c) = v_j t_l \wedge [fcp' = fcp \cup \{c\}] \wedge$ $enl = c_{v_i \to v_r} \cap c_{v_i \to re}$
6B-2	$re = f_{re}(enl) = v_j t_l \wedge em = v_i b_k \wedge PAC = f_{pac}(s,r,p_i) \wedge$ $\hat{arg} = \hat{f}^*_{arg}(PAC,em,re) \wedge \hat{tds} = \hat{f}_{tds}(\hat{arg},itl,v_i)$
6D	$v_i = v_j = v_r \wedge [fcp' = fcp \cup \{c\}] \wedge em = (v_i b_k) = v_r b_o \wedge re = (v_j t_l) = v_r r$
7	$PAC = f_{pac}(s,r,p_i) \wedge em = v_i b_k \wedge re = v_j t_l$
7A	$PAC = f_{pac}(s,r,p_i) = PAC_{nak} \wedge pcn = f_{pcn}(PAC_{nak}) \wedge pcc = pcn \wedge$ $em = v_i b_k \wedge re = v_j t_l \wedge emc = v_j t_l \wedge rec = v_i b_k$
7B	$PAC = f_{pac}(s,r,p_i) = PAC_{ack} \wedge pca = f_{pca}(PAC_{ack}) \wedge pcc = pca \wedge$ $em = v_i b_k \wedge re = v_j t_l \wedge emc = v_j t_l \wedge rec = v_i b_k$
7C	$re = v_j t_l \wedge enl = f_{enl}(re)$

Fig. VII.10 (continuação): Tabela das transições ...

T R A N S I Ç Õ E S	
nº	fórmulas
8	$pca \vee pcn=f^{-1}_{pcc}(pcc) \wedge esf=f_{esf}(pcc,emc,rec)$
8-1	$trl\neq 0 \wedge \not\exists esf \wedge trl=trl-1$
8-3a	$trl>0 \wedge pca=f^{-1}_{esf}(esf)$
8-3b	$trl>0 \wedge pcn=f^{-1}_{esf}(esf)$
8-4	$trl=0 \wedge neg=f_{neg}(\hat{a}rg,trl)$
8A	$emc=v_j t_1 \wedge rec=v_i b_k \wedge em=v_i b_k \wedge PAC_{nak}=f^{-1}_{pcn}(pcn) \wedge$ $cópia=f_{co}(PAC,em,re,res) \wedge [fcp'=fcp \cup (fc \setminus \{c\})] \wedge$ $c_{verif}=sel(fc,pcn,emc,rec,v_i=f_{vi}(pcn,em)) \wedge$ $\hat{a}rg=\hat{f}^{*}_{arg}(PAC,em,re) \wedge res=f_{res}(PAC,\hat{a}rg,PAC_{nak})$
8B	$emc=v_j t_1 \wedge rec=v_i b_k \wedge em=v_i b_k \wedge PAC_{ack}=f^{-1}_{pca}(pca) \wedge$ $re=v_j t_1 \wedge cópia=f_{co}(PAC,em,re,res) \wedge \hat{a}rg=\hat{f}^{*}_{arg}(PAC,em,re) \wedge$ $res=f_{res}(PAC,\hat{a}rg,PAC_{ack})$
8C	$\hat{a}rg=\hat{f}^{*}_{arg}(PAC,em,re) \wedge res=f_{res}(PAC,\hat{a}rg,neg) \wedge em=v_i b_k \wedge$ $cópia=f_{co}(PAC,em,re,res) \wedge re=v_j t_1$
9A	$c_{verif}=c_{velho} \wedge [fcp'=fcp \cup \{c\}] \wedge (PAC,em,re)=f^{-1}_{co}(cópia) \wedge$ $enl=c_{v_i \to v_r} \cap c_{v_i \to re}$
9B	$c_{verif}\neq c_{velho} \wedge [fcp'=fcp \cup \{c\}] \wedge em=v_i b_k \wedge$ $re=(v_j t_1)=v_i t_{-1} \wedge (PAC,em,re)=f^{-1}_{co}(cópia)$
9C	$em=v_i b_k \wedge re=(v_j t_1)=v_i t_{-1} \wedge (PAC,em,re)=f^{-1}_{co}(cópia)$

Fig. VII.10 (continuação-final): Tabela das transições ...

VII.5 - Observações finais sobre o modelo número 4.

Com este capítulo VII se encerra a primeira fase da modelagem, na base de 'PrT-Nets', dos problemas do controle de fluxo em redes de computadores. Estamos conscientes que falta ainda muito a fazer, mas achamos que obtivemos uma boa base para poder continuar este trabalho. Pensamos em melhorar o modelo nos seguintes sentidos:

- atualizando a estrutura gráfica em si, tentando atingir uma representatividade fiel de algum sistema existente (p.ex., o sistema que a EMBRATEL está implantando);

- aumentar a maneabilidade do modelo para poder executar modificações mais rápidas e eficientemente. Isto implica no uso de sistemas automatizados como, p.ex., um terminal gráfico interativo e um 'plotter', acoplados a um sistema computacional;

- usar a estrutura (gráfica) do modelo para conduzir investigações sobre a dinâmica do sistema modelado.

Queremos alertar que, até este momento, estamos somente na fase de um modelo que representa problemas estáticos, ou seja, que permite investigações em relação a situações mais ou menos previsíveis. Assim podemos, na verdade, modelar as precondições, poscondições, decisões, etc., relacionadas a estas situações previsíveis, mas não teríamos muitas chances em descobrir interações sutis na estrutura distribuída representada pelo modelo 'PrT-Net'.

Conseqüentemente, não devemos perder de vista este objetivo das investigações da dinâmica que, achamos nós, será indispensável no momento em que a estrutura distribuída dos computadores será considerada como sendo somente uma ferramenta para permitir o controle, vigilância e condução de sistemas industriais grandes e complexos no sentido distribuído, hierarquico e adaptativo.

C A P Í T U L O VIII

Observações finais: resumo, sugestões e conclusões.

VIII.1 - Resumo.

Falando, primeiramente, sobre o que foi conseguido com este trabalho, podemos dizer, em termos gerais, que conseguimos compreender bastante bem os acontecimentos numa rede de computadores graças à tentativa da modelagem dos mesmos.

Isto representou um dos pontos mais interessantes durante todo este estudo, ou seja, que a técnica de 'PrT-Net' permite a modelagem em qualquer nível, sempre com o grau desejado de pormenorização.

Mais ainda, como o formalismo usado pode variar desde inscrições verbais até fórmulas lógicas bastante complexas, pode-se obter, mais facilmente, um rascunho inicial de um modelo para, depois, pouco a pouco, penetrar mais profundamente no ambiente de um sistema real (que, em nosso caso, é uma rede de computadores) e, assim, realizar um modelo que é extremamente representativo dos acontecimentos do sistema em questão.

Achamos também, que a nossa maneira de atacar o problema da modelagem foi acertada. O que queremos dizer com isto, é, que continuamos a defender a idéia que se deve começar, em qualquer problema de modelagem, com uma visão geral do sistema para não se arriscar a perder de vista os objetivos finais e globais do sistema como um todo.

Foi esta uma das razões, como mencionado no CAPÍTULO I, que nos levou a fazer, antes de começar em usar a técnica de 'PrT-Net', bastante estudos preliminares para nos familiarizar com

os problemas específicos de redes de computadores. Estes trabalhos iniciais já mostraram a sua utilidade, pois, além de fornecer a base para uma compreensão melhor de redes para nós mesmos, eles facilitaram o ensino destes tópicos na graduação e pós-graduação, dependendo, obviamente, do nível empregado na apresentação destes assuntos.

Assim, um dos trabalhos iniciais de SCHWARZ (26) serviu para começar a "sentir" redes de computadores, outros, de SCHWARZ (27) e PALMA (19), serviram para reconhecer a utilidade, na modelagem, da teoria de grafos.

Também tocamos um pouco os assuntos de redes locais, GROJSGOLD (09), e das regras de comunicação, em especial os protocolos do chamado nível dois. Neste último estudo, feito por MUSCHELLACK (18), iniciamos, aliás, uma primeira tentativa de usar grafos especiais, que foi, em particular, o Petri-Net, para fins de modelagem de protocolos.

Já os estudos seguintes, feitos por SCHWARZ (30,31,32), nos conduziram diretamente para os problemas dinâmicos em redes. Em particular, SCHWARZ (30) investigou os problemas do controle de fluxo o que, parcialmente, serviu para compor o capítulo II deste trabalho presente. Já os outros dois trabalhos, SCHWARZ (31,32), embora previstos, inicialmente, para serem usados neste trabalho, servirão como uma das bases para a sua continuação, quando pretendemos incluir, no modelo obtido aqui, os problemas de controle de acesso em redes MHT (redes com meios heterogêneos de transmissão), ou seja, em redes que usam, além das linhas terrestres, também enlaces via satélites.

Como já dissemos, o estudo feito por SCHWARZ (30), sobre o controle de fluxo, serviu como base para compor o CAPÍTULO II. Escolhemos, sem grande justificativa, o controle simples em canais, com ACK, NAK e 'time-out', e a versão 1 do controle 'end-to-end', do ARPANET, para representar o controle de fluxo

em redes de computadores.

Uma vez isolados os métodos de controle de fluxo desejados, abrimos o CAPÍTULO III para apresentar um método novo, surgido há pouco. Esta técnica é chamada de 'predicate/transition-net' e representa, basicamente, um Petri-Net de primeira ordem. Como mostrado por GENRICH e LAUTENBACH (08) e RICHTER (23), esta técnica de 'PrT-Net' se mostrou muito útil na modelagem de sistemas (como, p.ex., em banco de dados, comunicação em escritórios, etc.) que contêm, inerentemente, um grande grau de complexidade, concorrência e conflitos, o que resulta numa multitude de decisões a conduzir. Então, nada mais natural do que usar esta técnica 'PrT-Net' para modelar os acontecimentos dinâmicos em redes de computadores.

Assim, a partir do CAPÍTULO IV, usamos esta técnica para modelar o controle de fluxo, empregando um procedimento de "pequenos passos", e aproveitando todos os nossos conhecimentos adquiridos nos estudos iniciais mencionados, incluindo o estudo da técnica de 'PrT-Net'. Fizemos um primeiro modelo que, basicamente, representa o controle 'end-to-end', ignorando, porém, por enquanto, os vértices intermediários.

O modelo obtido neste capítulo IV serviu como base e, como vimos no decorrer deste trabalho, em cada capítulo subseqüente se acrescentou algumas idéias a mais sem, contudo, modificar a estrutura básica deste modelo inicial.

Conseguimos, assim, no CAPÍTULO V, incluir o problema mais importante, ou seja, a consideração dos vértices intermediários. Para obter isto, tivemos que "abrir" os vértices para poder tratar os diferentes canais de entrada e saída de uma maneira quase independente. Mostramos como se pode, usando 'PrT-Nets', modelar filas em relação ao aspecto de administração do espaço disponível nas mesmas.

A inclusão, no modelo, dos problemas ACK e NAK foi uma conseqüência direta do tratamento dos vértices intermediários, isto

é, que entrou aqui, no capítulo V, o problema do controle de fluxo em canais entre vértices vizinhos.

Como este controle em canais inclui também aspectos do ponto de vista de 'time-out', achamos por bem acrescentar, no CAPÍTULO VI, este fato no modelo obtido no capítulo anterior. Também incluímos, no mesmo modelo, a questão do 'check-time', uma espécie de 'time-out' no nível 'end-to-end', avisando, porém, que existem ainda outros tempos envolvidos no controle de fluxo, não modelados, por enquanto.

A modelagem foi concluída, no CAPÍTULO VII, com a tentativa de mostrar como é bastante simples modificar o modelo obtido para considerar, também, mensagens do tipo multipacotes, ou seja, mensagens grandes que podem ser divididas em até oito pacotes simples.

VIII.2 - Sugestões de desenvolvimentos futuros em relação ao modelo obtido neste trabalho.

As principais sugestões podem ser divididas em duas partes, classificadas, informalmente, da seguinte maneira:

- MELHORAMENTOS do modelo de controle de fluxo existente, como, por exemplo, através da inclusão, na modelagem, dos problemas de mensagens de controle, de ferramentas específicas, de complementos para os predicados indefinidos, dos enlaces via satélite, etc.;

- uso de SISTEMAS COMPUTACIONAIS para facilitar, primeiramente, a modelagem em si e, segundo, para poder investigar a dinâmica do sistema representado pelo modelo.

Tendo agrupados estas sugestões, podemos falar um pouco mais sobre cada um destes grupos sem contudo, tentar explicá-los exaustivamente.

Melhoramentos do modelo existente.

Este grupo é aquele que representa a continuação mais direta deste trabalho. Já mencionamos, direta ou indiretamente, em vários trechos desta apresentação, que queremos chegar a um modelo que represente o estágio da última versão do controle 'end-to-end' do ARPANET.

Para conseguir isto, precisamos, em primeiro lugar, modelar as MENSAGENS DE CONTROLE em geral. Já iniciamos algo a este respeito quando modelamos a transmissão de informações do tipo ACK, NAK e RFNM, respectivamente. Contudo, não modelamos, explicitamente, como criar estas mensagens, como transmití-las, como controlar o fluxo delas através da rede, etc.

Precisa-se, também, distinguir se uma mensagem de controle é gerada pela subrede de comunicação ou se ela vem de "fora", se ela implica num controle de fluxo privilegiado (p.ex., sendo prioritária nos caminhos normais, exigindo caminhos especiais, etc.), se ela pode ser incorporada em mensagens comuns, etc.

Vemos, então, que, antes de obter resultados em relação às versões dois e três do controle 'end-to-end' da ARPANET, como descritas por KLEINROCK (14) e resumidas por SCHWARZ (30), precisa-se resolver este problema de mensagens de controle, já que nestas versões mencionadas haverá reserva explícita, e antecipada, de alguns recursos da subrede.

Outro tipo de melhoramentos está intimamente ligado à modelagem de três FERRAMENTAS: o roteamento, a alocação dos 'buffers' e a adaptação do crédito, considerados por SCHWARZ (30) como sendo os utensílios principais para otimizar o controle de fluxo.

Pode-se, eventualmente, considerar a ALOCAÇÃO DOS 'BUFFERS' como sendo nada mais que uma ferramenta local; entretanto, isto somente é verdade até um certo ponto, ou seja, até o momento em

que, por exemplo, o 'buffer' de uma fila estiver cheio e que temos que tomar uma decisão (como, por exemplo, bloquear o acesso a este vértice, desviar o tráfego, etc.). Diríamos até, que esta decisão deve ser tomada antes que a fila esteja no limite de sua capacidade. Já estudamos, também, os métodos de alocação dos espaços apresentados por KERMANI (13), para vértices/destinação, e por KAMOUN e KLEINROCK (12), para vértices intermediários, respectivamente. O que falta agora a fazer, é a integração destes métodos em nosso modelo.

Se a alocação dos 'buffers' é uma ferramenta prevista para, principalmente, fins locais, não se pode dizer o mesmo da ADAPTAÇÃO DO CRÉDITO. Não importa, se queremos adaptar o crédito entre vértices vizinhos, ou entre vértices 'end-to-end'; em cada caso há sempre um emissor, responsável pelas permissões de crédito, e um receptor que fornece as informações básicas para calcular estas permissões. Conseqüentemente, estas informações devem passar pela rede como sendo mensagens de controle, incluindo um tratamento privilegiado das mesmas. O estudo de KERMANI (13) mostrou a problemática do cálculo do crédito em si; cabe agora a nós, depois de alguns estudos preliminares já feitos, incorporar esta ferramenta no resultado de nossa modelagem.

Grande parte dos problemas de controle de fluxo está, realmente, situada em cada um dos vértices (veja o problema dos 'buffers') e, também, na relação destes vértices entre si (veja o problema do crédito). Tendo dado um tratamento adequado a estes problemas, podemos nos preocupar com um outro tipo de recurso que é a capacidade dos diferentes caminhos ou rotas na rede. Como caminhos englobam não somente os enlaces entre vértices vizinhos mas, também, os vértices nas extremidades destes enlaces, precisamos um tipo de ROTEAMENTO que "deveria saber", por hipótese, tudo o que acontece e, também, o que acontecerá, eventualmente, na rede.

O estudo dos métodos existentes sobre roteamento, feito por SCHWARZ (30), mostrou que, devido à importância deste roteamento ou 'routing', existe uma multitude de métodos apresentados na

literatura. A título de exemplo podemos citar pesquisas feitas (que resultaram até em métodos implantados) por FULTZ (05) e KLEINROCK (14), ou por YUM e SCHWARTZ (40) que fizeram estudos comparativos de diversos métodos adaptativos.

Aparece, neste momento, a palavra ADAPTATIVA que é o problema chave para poder resolver o roteamento. Precisa-se, então, sempre no vértice em questão, que escolhe a rota mais adequada, informações em relação à situação no próprio vértice, nos vértices vizinhos e, ainda, nos outros vértices; isto pode ser feito através de mensagens de controle, privilegiada ou não, que circulam na rede. Já fizemos uma indicação tímida da escolha do caminho a partir de um feixe de caminhos disponíveis, mas pouco foi realmente dito sobre as informações necessárias para fazer isto. Estamos nos referindo aos predicados do tipo INFO/TEMPOS que, segundo hipótese, contêm estas informações, mas ficou em aberto como elas lá chegaram.

Como último ponto destes possíveis melhoramentos básicos, queremos menciocar a inclusão, na rede de computadores, de ENLACES VIA SATÉLITE.

Em estudos preliminares, feitos por SCHWARZ (31,32), foram investigados vários métodos de como obter acesso a este enlace em questão. Como resultado sentimos, que a inclusão, no modelo, de um enlace via satélite não pode ser simplesmente resolvida pela escolha "deste caminho" no momento do roteamento. Entram aí, infelizmente, de ponto de vista da modelagem e do controle em si, problemas associados ao atraso e à vazão que são grandezas maiores que em linhas terrestres; isto implica, no momento da escolha do caminho para transmissão, retransmissão e confirmação, em considerações especiais, associadas, não somente à origem e à destinação destas mensagens, mas, sobretudo, ao tipo (p.ex., mensagens grandes, curtas, interativas, etc.), à classe (p.ex., mensagens do usuário, de confirmação, de controle, informações sobre o estado da rede, etc.) e eventuais prioridades.

Entretanto, os problemas não se resumem somente a estes pontos; existe uma diferença fundamental relativa ao acesso às redes. Em redes com linhas terrestres há, normalmente, pontos físicos distribuídos na rede (representados pelos vértices/fonte) onde se pode competir, isoladamente, usando, por exemplo, sistemas de filas, para obter acesso à rede. Assim, pode-se dizer que cada um destes pontos de acesso é independente um do outro (pelo menos numa primeira análise) e que estamos trabalhando, nestes pontos de acesso, somente com um eixo abstratro, o tempo. Já em enlaces via satélite (ou, também, em sistemas de rádio-difusão), o ponto físico escolhido para obter acesso à rede, não pode ser considerado isoladamente. Estamos competindo, nestes pontos, para obter acesso a um espaço abstrato, formado pelo tempo e pelas freqüências disponíveis.

Seria, então, interessante de considerar algum método, como, por exemplo, aquele de BORGONOVO e FRATTA (02,03) e, através de sua incorporação no nosso modelo, tentar descobrir como resolver, além dos problemas mencionados, a sincronização global exigida por todos estes métodos de acesso aos enlaces via satélite.

Uso de sistemas computacionais.

Durante o desenvolvimento do modelo sobre o controle de fluxo em redes de computadores, sentimos, em várias ocasiões, a falta de um sistema de apoio automatizado. O que queremos dizer com isto é, em primeiro lugar, que, devido à complexidade crescente do modelo, a maneabilidade do modelo em si começa a se tornar cada vez mais difícil. E, segundo, fazer investigações de ponto de vista da consistência do modelo, do comportamento dinâmico, etc., se tornam quase impossíveis.

Tratando, primeiramente, da MANEABILIDADE do modelo, sabemos, da teoria de grafos, que cada grafo pode ser representado dentro de um computador. Para conseguir isto, existem vários métodos e um deles, baseado na matriz de incidência, proposto por GENRICH e LAUTENBACH (08), mostra que o grafo do tipo PrT ('PrT-Net') não é uma exceção. Vimos, no capítulo III, a repre-

sentação, na base de uma matriz de incidência (figura III.29) de um exemplo bastante trivial de um 'PrT-Net' (figura III.28). Entretanto, na aplicação em relação ao modelo simplificado do capítulo IV (figura IV.9), esta matriz, na sua forma original, já não serviu mais de uma maneira adequada. Isto se deve ao fato, que esta matriz é, primeiramente, muito esparsa, o que resultaria num desperdício em relação ao espaço, e, segundo, que, para estruturas grandes de 100 ou mais predicados, dezenas de transições (com fórmulas cada vez mais sofisticadas) e inscrições mais complexas, a probabilidade de fazer, no tratamento manual, algum erro (na indexação, nas multiplicidades, nas inscrições, etc.) cresce bastante. Foi esta uma das razões que nos levou a não usar, neste trabalho, a matriz propriamente dita, mas uma variante que faz um agrupamento dos predicados segundo as suas naturezas, completada por uma lista das transições com suas respectivas fórmulas.

Achamos, que estas listas podem ser facilmente introduzidas num computador e que este sistema automatizado, tendo uma estrutura relacional, poderia indicar todas as implicações de uma possível mudança de qualquer detalhe no modelo. Para facilitar a compreensão global do modelo dever-se-ia usar, adicionalmente, um sistema gráfico que permita, por um lado, obter, de uma maneira automatizada, esboços do sistema total ou somente parte dele e, também, por outro lado, um tipo de paginação num terminal gráfico.

Tendo melhorado, deste modo, em muito a maneabilidade do modelo, podemos nos dedicar aos métodos de verificação da CONSISTÊNCIA DINÂMICA do modelo. Queremos lembrar, que o modelo obtido neste trabalho foi somente verificado (mas, também, não exaustivamente) em relação aos grupos de predicados, ou seja, foi verificado que as condições de um determinado grupo não se "perdem" no modelo. Também tentamos conceber o modelo no sentido de que cada informação (mensagens ou pacotes) sempre tenha associado um responsável pela emissão ou recepção, e que uma transmissão somente pode ocorrer se há algum enlace ('link') disponível.

Então, esta consistência deveria ser testada, primeiramente, de um modo preliminar, ou seja, que o modelo satisfaz as exigências para um único usuário e uma única mensagem (o que fizemos manualmente de modo parcial) para, depois, entrar no teste da dinâmica propriamente dita. Este teste incluirá a verificação de todas as marcações iniciais, de todas as combinações possíveis, de todas as limitações existentes, etc., ou seja, a partir de qualquer marcação inicial permitida, deve ser testado se o modelo satisfaz às "invariantes" (veja o capítulo III) em cada instante da progressão das marcações através do modelo. O 'PrT-Net' que fornece a base para um modelo estático, se transforma, segundo RICHTER (23), neste momento, num 'PrT-System' que, por sua vez, representa toda a dinâmica do sistema sobre a estrutura modelada do 'PrT-Net'.

Achamos, então, que ficou evidente que este tipo de teste, que revelaria os verdadeiros problemas (conflitos, concorrências, bloqueio perpétuo, etc.), somente pode ser realizado, de um modo satisfatório, por um sistema de apoio computadorizado.

VIII.3. Conclusões.

Podemos afirmar, baseados na experiência adquirida neste trabalho, que a modelagem com 'PrT-Net' permite certamente compreender o funcionamento do sistema modelado em qualquer nível e em qualquer detalhe.

Conseqüentemente, uma vez obtida esta compreensão, pode-se partir para detectar eventuais falhas na concepção do sistema e tomar atitudes coerentes para corrigí-las. Não importa, se o sistema já existe na realidade ou se é, por exemplo, somente um projeto; o ponto importante é, que a modelagem com 'PrT-Nets' representa uma certa forma de especificação de sistema.

Dissemos "certa forma",porque o tipo de especificação, inerente à ferramenta empregada por nós (os 'PrT-Nets'), deve ser considerada como sendo um complemento indispensável, em relação a certas definições e especificações feitas de uma maneira informal. A ferramenta adotada neste trabalho adiciona o poder

de tomada de decisões ao modelo básico.

Considerando que não existem sistemas reais sem deficiências, achamos que estas decisões são importantíssimas porque somente assim podemos conceber sistemas que são, ao menos, tolerantes em relação a falhas ('fault tolerant systems'), ou seja, que tomam determinadas decisões, em locais e momentos certos, para, no extremo, evitar o colapso de um sistema, ou, sem ir tão longe, para melhorar, por exemplo, o seu desempenho.

Surgiu, obviamente, a pergunta de como, por exemplo, detectar imprecisões do sistema.

A primeira possibilidade surge,normalmente, quando se faz a transposição da situação estática do sistema para o modelo, ou seja, na primeira fase da modelagem que podemos chamar de representar corretamente um sistema. Nesta fase se exigem, continuamente, testes concernentes às relações entre a realidade física do sistema com as propriedades básicas da ferramenta para modelagem (em nosso caso, o 'PrT-Net') para garantir a obtenção de um modelo básico que é realmente representativo e consistente do ponto de vista da estática do sistema em questão.

Depois disto, entramos na fase da modelagem dinâmica do sistema e é este ponto que, a nosso ver, exige maiores considerações teóricas. Como o modelo foi concebido para, justamente, facilitar o teste desta parte importante, podemos nos perguntar como investigar a dinâmica do sistema no seu modelo.

Foi mencionado, no decorrer deste trabalho, que, no momento em que se começa usar as chamadas "marcações", a ferramenta 'PrT-Net' se torna um 'PrT-System'. Portanto, a evolução das marcações, através da satisfação das pré- e pós-condições de uma transição e o seu conseqüente disparo, representa a mudança dos estados de 'PrT-System'. Mas como este modelo, por definição, deve sempre, em cada instante, representar o sistema-base, estes estados significam diferentes situações deste sistema, ou seja, a dinâmica do sistema. Então, logicamente, "problemas" na evolução das marcações no modelo (ou seja, no 'PrT-System'), indicam, fortemente, deficiências no sistema original, não ignorando, entretanto, er-

ros ocorridos na modelagem em si.

Agora, cada ferramenta de modelagem tem as suas propriedades particulares e o 'PrT-Net/System' não é uma exceção a isto. Assim, como a nossa ferramenta é, basicamente, o resultado do "mapeamento" de vários Petri-Nets sobre uma única estrutura, surge a pergunta se as propriedades particulares de Petri-Nets, tais como, por exemplo, consistência, alcançabilidade, 'coverability', 'liveness or deadness', o fato de ser conservativo, etc., possam ser aplicadas a 'PrT-Nets'.

Queremos deixar claro, que não foi objetivo desta tese investigar quais das propriedades básicas (de Petri-Nets) foram mantidas, excluíndo desta restrição, eventualmente, os pontos consistência e de ser conservativo. A este respeito, mencionamos várias vezes, no decorrer deste trabalho, o termo de "S-invariantes" que, justamente, permite testar a consistência das marcações na evolução dinâmica de 'PrT-Nets', sem, contudo, as ter aplicado consistentemente em nosso modelo devido à falta, por enquanto, de uso de sistemas computacionais para poder realizar estes testes. O que fizemos foi a criação de um modelo que garante, ao menos num primeiro nível de modelagem, que o 'PrT-Net' obtido seja conservativo (que os elementos não se "percam" no modelo) e que certos "pares de predicados" foram formalizados para indicar eventuais futuras "S-invariantes".

Vimos, então, que fizemos os primeiros passos para entrar num campo significativo de pesquisa prática e teórica sobre 'PrT-Nets/systems', sobretudo se se quer usar toda a potencialidade de poder de modelagem ('modelling power'), que se refere à aptidão de representar um sistema corretamente, e do poder de decisões ('decision power'), que se refere à potencialidade de analisar o modelo e determinar propriedades do sistema-base em questão.

ANEXO A

Bibliografia

(01) BRAUER, Wilfried (editor); "Net Theory and Applications"; Proceedings of the Advanced Course on General Net Theory of Processes and Systems, Hamburg, October 8-19, 1979; *Lecture Notes in Computer Science*, nº 84, Springer-Verlag, Berlin, Heidelberg, New York, 1980, 537 pgs.;

(02) BORGONOVO, F., L. Fratta; "A New Technique for Satellite Broadcast Channel Communication"; *Proceeding Fifth Data Communication Symposium*, Sept. 1977, Snowbird, Utah, USA, pp. 2-1/2-4;

(03) BORGONOVO, F., L. Fratta; "SRUC: A Technique for Packet Transmission on Multiple Access Channels"; *International Conference on Computer Communications* (ICCC-78), Kyoto, Japan, 26-29 Setp. 1978, pp. 601-607;

(04) DAVIES, D.W. et alii; "*Computer Networks and their Protocols*", John Wiley & Sons, New York, 1979, 487 pgs.;

(05) FULTZ, G.L.; "*Adaptive Routing Techniques for Message Switching Computer-Communication Networks*"; Univ. of California, Los Angeles, (UCLA), Thesis Ph.D., 1972, 418 pgs.; (also: Univ. Microfilms, a XEROX Comp., Ann Arbor, Michigan, code: 75-25,768);

(06) GENRICH, H.J. et alii; "Elements of General Net Theory"; Advanced Course of GNT of Processes and Systems, *Lecture Notes in Computer Science*, nº 84, Springer Verlag, Berlin, 1980, pp. 21-163;

(07) GENRICH, H.J. et alia; "A Dictionary of Some Basic Notions of Net Theory"; Advanced Course of GNT of Processes and Systems, *Lecture Notes in Computer Science*, nº 84, Springer Verlag, Berlin, 1980, pp. 519-535;

(08) GENRICH, H.J., K. Lautenbach; "System Modelling with High--Level Petri-Nets"; *Theoretical Computer Science*, Vol. 13 (1981), pp. 109-136;

(09) GROJSGOLD, A.L. e G. Schwarz; "Um Estudo sobre Redes Locais de Computadores"; *Publ. Didática COPPE(Sistemas)-UFRJ*, PDD-02/80, Rio de Janeiro, 1980, 73 pgs.;

(10) HEART, F.E. et al.; "The Interface Message Processor for the ARPA Computer Network"; *SJCC-1970*, pp. 551-567;

(11) INOSE, H. e T. Saito; "Theoretical Aspects in the Analysis and Synthesis of Packet Communication Networks"; *Proc. IEEE*, Vol. 66, nº 11, Nov. 1978, pp. 1409-1422;

(12) KAMOUN, F. e L. Kleinrock; "Analysis of Shared Finite Storage in a Computer Network Node Environment under General Traffic Conditions"; *IEEE Transaction on Communications*, Vol. COM-28, nº 7, July 1980, pp. 992-1003.

(13) KERMANI, P.; *"Switching and Flow Control Techniques in Computer Communication Networks"*; Computer Science Depto., UCLA-ENG-7802, Febr. 1978, 334 pgs.; (also: publ. as Ph.D. dissertation, UCLA, Dec. 1977);

(14) KLEINROCK, L.; *"Queueing Systems, Volume II: Computer Applications"*; John Wiley & Sons, New York, 1976, 549 pgs.;

(15) KLEINROCK, Leonard; "On Flow Control in Computer Networks"; *International Conference on Communications* (ICC), Toronto, Canada, June 4-7, 1978, Vol. 2, pp. 27.2.1-5;

(16) LAM, S. Simon; "Store-and-Forward Buffer Requirements in a Packet Switching Network"; *IEEE Trans. on Communications*, Vol. COM-23, nº 4, April 1976, pp. 394-403;

(17) MARUYAMA, K.; "Optimization of Mixed-Media Communication Networks"; *Computer Networks*, Vol. 2, 1978, pp. 168-178;

(18) MUSCHELLACK, Erich; *"Proposta de um Padrão de Protocolo de Comunicação"*; Tese de M.Sc., COPPE(Sistemas)--UFRJ, Rio de Janeiro, Brasil, Maio 1979, 165 pgs.; (tb.: Publ. Int. COPPE/UFRJ, PTS-05/79, 1979);

(19) PALMA, W.R.; *"Definição Topológica de Redes de Comunicação de Dados"*; Tese M.Sc., COPPE(Sistemas)-UFRJ, Rio de Janeiro, Brasil, Dez. 1979, 171 pgs.;

(20) PETERSON, James L.; "Petri-Nets"; *Computing Surveys*, Vol. 9, nº 3, Sept. 1977, pp. 223-252;

(21) PETRI, C.A.; "Introduction to General Net Theory"; Advanced Course of GNT of Processes and Systems, *Lecture Notes in Computer Science*, nº 84, Springer Verlag, Berlin, 1980, pp. 1-19;

(22) POUZIN, L.; "Flow Control in Data Networks - Methods and Tools"; *ICCC-1976*, Toronto, Canada, 3-6 Aug., 1976, pp. 467-474;

(23) RICHTER, Gernot; "Netzmodelle fuer die Buerokommunikation"; *Manuscript (interno da GMD-Birlingshofen, Bonn)*, 30.09.82, 88 pgs.; (Teil 1: publicado em Informatic-Spektrum (1983), veja ref. 24);

(24) RICHTER, Gernot; "Netzmodelle für die Bürokommunikation, Teil 1"; *Informatik-Spektrum* (1983)6, pp. 210-220, Springer-Verlag, 1983;

(25) RICHTER, Gernot; Comunicação pessoal; Gesellschaft für Mathematik und Datenverarbeitung (GMD) mbH, Bonn, 6.1.1984;

(26) SCHWARZ, G.; "Redes de Computadores - Uma Análise Qualitativa"; *Publ. interna, COPPE(Sistemas)-UFRJ*, PDD-11/78, 1978, Rio de Janeiro, 115 pgs.;

(27) SCHWARZ, Gerhard; "Redes de Computadores - Um Estudo de Modelos Matemáticos"; *Publ. interna COPPE/UFRJ*, PDD-13/19, Rio de Janeiro, 1979, 188 pgs.;

(28) SCHWARZ, G.; "Redes de Computadores - Programação Matemática e Simulação"; *Publ. interna COPPE(Sistemas)-UFRJ*, PDD-16/79, 1979, Rio de Janeiro, 136 pgs.;

(29) SCHWARZ, Gerhard; "Computadores em Tempo Real: Uma Introdução aos Conceitos Básicos"; *Livro-Texto Interno, COPPE/UFRJ*, PDD-06/81, 500 pp., 1981;

(30) SCHWARZ, Gerhard; "Redes de Computadores - Descrição Qualitativa de Métodos para o Controle de Fluxo em Linhas Terrestres"; *Publ. Did. COPPE/UFRJ*, PDD-02/83, 73 pgs.; 1981;

(31) SCHWARZ, Gerhard; "Redes de Computadores - Descrição Qualitativa de Métodos de Controle de Acesso em Redes de Rádio-Difusão ou com Enlaces via Satélite"; *Publ. Did., COPPE/UFRJ*, PDD-13/82, 67 pgs.; 1981;

(32) SCHWARZ, Gerhard; "Descrição Qualitativa de Alguns Métodos de Controle de Fluxo e de Acesso em Redes de Computadores com Meios Heterogêneos de Transmissão (Redes MHT); *Relatório Didático, COPPE(Sistemas)-UFRJ*, ES-33/84, Fev. 1984, Rio de Janeiro, 34 pgs.;

(33) SCHWARZ, Gerhard; "Uma Introdução a GNT ('General Net Theory') e Grafos do Tipo PrT ('Predicate/Transition-Nets'); *Relatório Técnico, COPPE(Sistemas)-UFRJ*, ES-22/83, Fev. 1983, Rio de Janeiro, Brasil, 62 pgs.;

(34) SCHWARZ, Gerhard; "Modelo Simplificado, Usando 'PrT-Nets', da Transmissão de um Pacote Simpes em Redes de Computadores"; *Relatório de Pesquisa, COPPE(Sistemas)-UFRJ*, ES-23/83, Rio de Janeiro, 43 pgs.;

(35) SCHWARZ, Gerhard; "O Uso de 'PrT-Nets' para Modelar os Acontecimentos nos Vértices Intermediários na Transmissão de Pacotes numa Rede de Computadores"; *Relatório de Pesquisa, COPPE(Sistemas)-UFRJ*, ES-32/84, Janeiro 1984, Rio de Janeiro, 101 pgs.;

(36) SCHWARZ, Gerhard; "Melhoramentos do Modelo Sobre a Transmissão de Pacotes Simples em Redes de Computadores no Sentido de Incluir a Verificação de Tempos"; *Relatório de Pesquisa, COPPE(Sistemas)-UFRJ*, ES-35/84, Março 1984, Rio de Janeiro, 58 pgs.;

(37) SCHWARZ, Gerhard; "Idéias Sobre a Modelagem, na Base de 'PrT-Nets', da Transmissão de Mensagens do Tipo Multipacotes em Redes de Computadores"; *Relatório de Pesquisa, COPPE(Sistemas)-UFRJ*, ES-36/84, Abril 1984, Rio de Janeiro, Brasil, 40 pgs.;

(38) VASCONCELLOS, Eduardo de; "*Análise de Desempenho de Sub-Sistemas de Comunicação de Dados: Aplicação de Modelos de Multifilas*"; Tese M.Sc., COPPE(Sistemas)-UFRJ, Rio de Janeiro, Brasil, Nov. 1982, 209 pgs.;

(39) VOSS, Klaus; "Using Predicate/Transition-Nets to Model and Analyze Distributed Database Systems; *IEEE Transaction on Software Engineering*, Vol. SE-6, nº 6, Nov. 80, pp. 539-544.

(40) YUM, Tak-Shing e Mischa Schwartz; "Comparison of Adaptive Routing Algorithms for Computer Communication Networks"; *NTC-78*, Birmingham, Alabama, Dec. 3-6, 1978, pp. 4.1.1-5;

ANEXO B

Nomenclatura

(Obs.: somente a nomenclatura usada freqüentemente neste trabalho é incluída neste anexo B).

NOMENCLATURA	DEFINIÇÃO
ACK	'Acknowledgement'; para possíveis indexações veja a sigla PAC
A/D	analógico/digital (por exemplo, conversor A/D)
ARG	conjunto de argumentos, $\{arg_1, arg_2, ...\}$, usados quando não se quer ser mais específico em relação aos predicados como, por exemplo, em FILA(arg,k).
arg, arg_i	arg, $arg_i \in$ ARG
âarg	argumento abstrato para "descrever" alguma subtarefa como, por exemplo, a transmissão e confirmação de pacotes entre vértices vizinhos
ARPANET	"Advanced Research Projects Agency (ARPA) Network', associada a 'U.S. Department of Defense (DoD)'
b_k	identificador de um canal de saída de um vértice S/F; usado, principalmente, na combinação $v_i b_k$
bg_s, bg_{vi}	um tipo de condição especial identificável para garantir o bom funcionamento de um contador de tempo; associado ao funcionamento global, para um determinado emissor s, ou local, para um determinado vértice v_i
C	conjunto de caminhos escolhidos entre vértices i e j; $C = \{c_{ij} \mid c_{ij} = sel(fc_{ij}, ...)\}$

c, c_{ij}	c, $c_{ij} \in C$
C/E	'Condition/Event-Systems'; sistemas C/E
CPU	'Central Processing Unit' (veja UCP)
CV	conjunto de caminhos virtuais entre vértices i e j; $CV = \{cv_{ij}\}$
cv, cv_{ij}	cv, $cv_{ij} \in CV$
des	argumento abstrato para "descrever" alguma tarefa; por exemplo, a transmissão e confirmação de mensagens entre vértices/fonte e -/destinação.
e, e_i	e, $e_i \in EE$
EM	conjunto de "emissores" locais, $EM = \{em_1, em_2, \ldots\}$ de uma mensagem empacotada, ou seja, o processador de saída de uma fila de saída b_k associada a um vértice v_i
em, em_i	em, $em_i \in EM$, onde $em = v_i b_k = f$ (emissor local)
EndE=EE	conjunto de endereços de um usuário/emissor $EndE = \{e_1, e_2, \ldots\}$ ou $EndE = \{s_1, s_2, \ldots\}$
EndR=ER	conjunto de endereços de um usuário/receptor $EndR = \{r_1, r_2, \ldots\}$
ENL	conjunto de enlaces entre vértices vizinhos; $ENL = \{enl_1, enl_2, \ldots\}$
enl, enl_i	enl, $enl_1 \in ENL$
E/S	entrada/saída
esf, etf	argumentos usados para indicar que a execução de uma (sub)-tarefa chegou ao fim (usado, por exemplo, em TAREFA/FIM(etf,bg_s))
f(.)	função, em geral
f_p	função de mapeamento com respeito à parte adicional de uma mensagem <s,r> como, por exemplo, $p=f_p(s,r)$, onde $f_p: (UxU \rightarrow P)$
f_p^{-1}	função inversa associada à função f_p

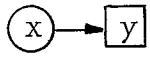
f_m, f_m^{-1}	função (e a sua inversa) de mapeamento para simplificar a denominação de uma mensagem empacotada <s,r,p>; por exemplo, $m=f_m(s,r,p)$, onde $f_m: (UxUxP \rightarrow M)$
FC	conjunto de diferentes agrupamentos de caminhos virtuais (feixes de caminhos); $FC = \{fc_{ij} \mid fc_{ij} = (cv_{ij},s,r)\}$
fc, fc_{ij}	$fc, fc_{ij} \in FC$
GNT	'General Net Theory'
GR	conjunto de receptores r_i de um determinado emissor, agrupados em gr_i; $GR = \{gr_1, gr_2, \ldots\}$
gr, gr_i	$gr, gr_i \in GR$
H/D	hospedeiro/destinação
H/F	hospedeiro/fonte
i, i^T	vetor, e sua transposta, que representa uma S-invariante
itg,itl	argumentos usados em predicados auxiliares (por exemplo, INFO/TEMPO-GLOBAL(itg)) para indicar informações sobre tempos no sentido global ou local
ja_{sr}	argumento usado em predicados auxiliares (por exemplo, CRÉDITO/MENSAGEM(ja_{sr})) para indicar um certo crédito ou janela
K,KP,...	conjunto de argumentos, nos predicados que representam filas, que indicam os lugares ou espaços no 'buffer' da fila; $K = \{k_1, k_2, \ldots\}$, $KP = \{kp_1, kp_2, \ldots\}$, etc.
$k, k_i, kp, kp_i, \ldots$	$k, k_i \in K$ (capacidade, em geral), $kp_1, kp \in KP$ (capacidade em relação ao processador), $ks, ks_2 \in KS$ (capacidade da fila do canal s), etc.
li	liberação de algum recurso
M	conjunto de mensagens entre usuários $M=\{M_1, M_2, \ldots\}$ ou pares de emissor e receptor $M = \{s_1r_1p, s_1r_2p, \ldots\}$

m, m_i	$m, m_i \in M$
MC	conjunto de mensagens de controle, em geral; $MC = \{mc_1, mc_2, \ldots\}$; variantes MCA, MCN, MCQ para indicar mc's afirmativas, negativas, quaisquer
mc, mc_i	$mc, mc_i \in MC$; (variantes mca, mcn, mcq)
MHT	meios heterogêneos de transmissão (por exemplo, redes com MHT)
mod.n	indica, como, por exemplo, no caso z'=z+1,mod.n, que os operadores sempre serão parte de um anel com n elementos
MSG	mensagem, em geral, que chega à rede; uma possível indexação veja com PAC
MUX	um multiplexador lógico que permite o compartilhamento de uma única UCP
NAK	'Negative Acknowledgement'; uma possível indexação veja com PAC
neg	argumento usado em predicados auxiliares (por exemplo, PAC/nãoCONF(ârg,neg)) para indicar um resultado negativo
P	conjunto da parte adicional à mensagem para formar um pacote; $P = \{p_i \mid p_i = f(s,r)\}$
p, p_i	$p, p_i \in P$; atenção: p_i indica também "portas" ou transições externas
PAC	pacote, isto é, uma mensagem obedecendo um determinado formato imposto pela rede de computadores; também usado como argumento em predicados em substituição ao m
PAC_n	pacote; indexação: número da seqüência
PAC_{n_r}	pacote; indexação: número da seqüência avisando que foi retransmissão
$PAC_{n,i}$	pacote; indexação: indica o i-ésimo pacote da n-ésima mensagem

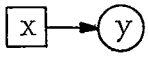
PC	conjunto de pacotes de controle, em geral; PC = {pc_1,pc_2,...}; variantes PCC, PCN, PCA para indicar pc's comuns, negativos, afirmativos
pc,pc_i	pc, pc_i $\in$ PC; (variantes pcc, pcn, pca)
Petri-Net	nome próprio para 'Place-Transition-Net'
PrT-Net	'Predicate/Transition-Net' ou grafos do tipo PrT
PrT-System	representa a dinâmica de um 'PrT-Net'
P/T-Net	'Place/Transition-Net'
r,r_i	r, r_i $\in$ EndR
RC	conjunto de endereços de receptores de mensagens de controle (mensagens geradas internamente na subrede de comunicação); RC = {rc_1,rc_2,...}
rc,rc_i	rc, rc_i $\in$ RC
RE	conjunto de receptores locais de uma mensagem empacotada, ou seja, o processador de entrada de uma fila t_1 associada a um vértice v_j; RE = {re_1,re_2,...}, onde re = $v_j t_1$ = f (receptor local)
re,re_i	re, re_i $\in$ RE
RFNM	'Request-for-next-message'; uma possível indexação veja com PAC
rs	considerando r como sendo a identificação de um usuário/receptor, rs indica a "sua parte" responsável pela recepção de mensagens de um determinado emissor s
S	conjunto de S-elementos ou predicados, representados graficamente por círculos; S = {s_1,s_2,...}
s,s_i	s, s_i $\in$ S ou s, s_i $\in$ EndE (quando se trata dos endereços de um usuário/emissor)
SC	conjunto de endereços de emissores de mensagens de controle (veja MC); SC = {sc_1,sc_2,...}

sc, sc_i	sc, $sc_i \in SC$
sel(.)	função para escolher um caminho; por exemplo, a partir de um feixe disponível
S/F	vértices intermediários do tipo 'store-and--forward'
S-invariante	vetor i de polinômios que garante que $C^T.i=0$ e $i^T.M=i^T.M_o$, onde C representa uma matriz de incidência e M, M_o as marcações globais e iniciais, respectivamente
sr	considerando s como sendo a identificação de um usuário/emissor, sr indica a "sua parte" responsável pelo envio de mensagens de s para um determinado receptor r
T	conjunto de T-elementos ou transições, representados graficamente por retângulos; $T=\{t_1, t_2, \ldots\}$
t, t_i	t, $t_i \in T$
t_1, t_o	identificador de um canal de entrada num vértice S/F; usado, principalmente, na combinação $v_j t_1$
t_{out}	tempo de verificação para receber uma resposta do vértice vizinho
t_{rm}	tempo necessário para executar a remontagem de uma mensagem
$\hat{t}ds, \hat{t}dt$	argumentos usados para iniciar um contador regressivo, representado, por exemplo, pelo predicado auxiliar TEMPO/R-L($\hat{a}rg$,trl,bl_{vi}), ou seja, o tempo determinado (estimado) para a execução de uma subtarefa (descrita por $\hat{a}rg$) ou uma tarefa (descrita por $\hat{d}es$)
trl,trg	argumentos usados em predicados auxiliares (por exemplo, TEMPO/RESTANTE-LOCAL($\hat{a}rg$,trl,bl_{vi})) para indicar o tempo restante local (iniciação com $\hat{t}ds$) ou global (iniciação com $\hat{t}dt$)

U	conjunto de usuários, em geral, $U = \{u_1, u_2, \ldots\}$, usado como argumento nos predicados
u, u_i	$u, u_i \in U$
UCP	unidade central de processamento (veja CPU)
ulm	'upper limit of multiplicity' ou limite superior de multiplicidade usado para indicar a "capacidade" de predicados
V	conjunto de vértices numa subrede de comunicação; $V = \{v_1, v_2, \ldots\}$
v, v_i	$v, v_i \in V$
$v_i b_k$	processador de saída da fila b_k no vértice v_i (veja também em)
$v_j t_1$	processador de entrada na fila t_1 no vértice v_j (veja também re)
V/D	vértice/destinação
V/F	vértice/fonte
○	representa um lugar, predicado ou S-elemento (veja também S)
▭	representa uma transição ou T-elemento (veja também T)
→	representa uma relação causal entre elementos S e T
•	'token' em geral
< >	tupla de grau zero; equivalente a possível passagem de uma condição simples (sem especificação) ou o equivalente de um 'token' em Petri-Nets
$< >_{sr}$	condição simples, mas somente válida para uma determinada situação (por exemplo, no caso de $< >_{sr}$ isto significa uma condição relacionada ao par s-r de usuários)
<u>, <u,m>	tuplas de diferentes graus; usadas como inscrições em arcos onde elas especificam, por exemplo, a relação causal entre predicados e transições adjacentes



remoção de itens de lugares; $(x,y) \in F \cap (S \times T)$, onde $F \cap (S \times T) = Z$, uma 'target relation'



adição de itens a lugares; $(x,y) \in F \cap (T \times S)$, onde $F \cap (T \times S) = Q^{-1}$, uma 'source relation'

Σ estrutura matemática para formar as inscrições em arcos e transições; também "soma" matemática

$\bigcup$ união generalizada ('generalized union' ou 'Vereinigungsmenge')

$\bigcap$ interseção ('intersection' ou 'Durchschnitt')

$\subseteq$ subconjunto ('subset' ou 'Teilmenge')

$\setminus$ representa uma subtração de elementos de conjuntos; por exemplo $gr \setminus \{r\}$

$\in$ usado para indicar que um elemento está contido num conjunto; por exemplo, $x \in X$

$\{\ \}$ indica, normalmente, um conjunto

$\exists$ quantificador existencial ('existencial quantifier' ou 'Seinszeichen')

$\nexists$ negação do quantificador existencial, ou seja, "não existe, por exemplo, resultado..."

$:=$ igualdade por definição

$\wedge$ conjunção ('Konjunktion')

$\vee$ disjunção ('Disjunktion')