

# UMA FERRAMENTA PARA VALIDAÇÃO DE PROTOCOLOS DE COMUNICAÇÃO

Olival de Gusmão Freitas Júnior

DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO - DSC

CENTRO DE CIÊNCIAS E TECNOLOGIA - CCT

UNIVERSIDADE FEDERAL DA PARAÍBA - UFPB

Abril - 1992

Olival de Gusmão Freitas Júnior

## Uma Ferramenta para Validação de Protocolos de Comunicação

Dissertação apresentada ao Curso de Mestrado em Informática da Universidade Federal da Paraíba, em cumprimento às exigências para obtenção do grau de Mestre.

**Área de Concentração:** Redes de Computadores.

**Orientador:** Wanderley Lopes de Souza.


# Uma Ferramenta para Validação de Protocolos de Comunicação

Olival de Gusmão Freitas Júnior

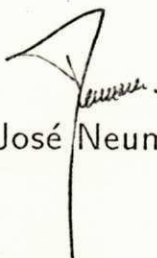
Dissertação apresentada em 10 de Abril de 1992.



Presidente: Wanderley Lopes de Souza (Dr.)



Examinador: Joberto Sérgio Barbosa Martins (Dr.)



Examinador: José Neuman de Souza (M.Sc)

DIGITALIZAÇÃO:

SISTEMOTECA - UFCG

# AGRADECIMENTOS

Gostaria de registrar meu agradecimento ao Professor Wanderley Lopes de Souza, meu orientador, pelo estímulo e apoio indispensável à realização deste trabalho.

Gostaria também de agradecer a todos os funcionários e professores do Departamento de Sistemas e Computação que, direta ou indiretamente, ajudaram-me a realizar este trabalho.

Agradeço ao Departamento de Engenharia da Computação e Automação Industrial (DCA) da Faculdade de Engenharia Elétrica (FEE) da Universidade Estadual de Campinas (UNICAMP) por ter permitido a utilização de seu laboratório e de seus equipamentos na fase final desse trabalho.

Agradeço também aos colegas do curso de mestrado pela amizade.

Quero também agradecer em especial a meus pais, Olival de Gusmão Freitas e Maria Adélia Rapôso Freitas, que sempre me apoiaram e incentivaram.

Agradeço aos colegas Galileu, Alfredo e Leonardo pelo estímulo e pela ajuda dada, no período em que permaneci em Campinas, para a conclusão deste trabalho. Quero também agradecer ao meu primo Romildo pela amizade e apoio.

Agradeço ao auxílio financeiro dado pela Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) e pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq).



"Se você não puder ser um pinheiro no topo da colina,  
seja um arbusto no vale.  
Se não puder ser uma estrada real,  
seja uma vereda.  
Se não puder ser o sol,  
seja uma pequena estrela.  
Não é pelo tamanho que se ganha ou que se perde ...  
Seja o melhor possível naquilo que você é!"

Dedico esta aos meus pais, à minha esposa  
Cléia e ao meu querido filho Olival Neto.

# SUMÁRIO

Este trabalho apresenta uma ferramenta, baseada na técnica Matriz Duologue, para a validação de protocolos de comunicação. Essa ferramenta auxilia o projetista, na fase inicial de design do protocolo, a identificar e corrigir erros básicos, de forma interativa e amigável. Para ilustrar essa técnica e demonstrar a utilidade dessa ferramenta, uma versão simplificada do protocolo X.25 do CCITT é especificada, analisada e validada.

# ABSTRACT

This work presents a tool, based on the Duologue Matriz technique, for the validation of communication protocols. This tool helps the designer, in the initial phase of the protocol design, to identify and to correct basic errors in a interactive and friendly way. To illustrate this technique and to demonstrate the importance of this tool, a simplified version of the CCITT (Consultative Committee for International Telegraph and Telephone) X.25 protocol is specified, analysed and validated.

# Conteúdo

<b>1</b>	<b>Introdução</b>	<b>7</b>
<b>2</b>	<b>Especificação e Validação Formais de Protocolos</b>	<b>10</b>
2.1	Especificação Formal . . . . .	11
2.2	Validação Formal . . . . .	12
2.3	Especificação de um Protocolo através de um par de Grafos . . . . .	14
<b>3</b>	<b>Construção e Validação da Matriz Duologue</b>	<b>17</b>
3.1	Princípio de Validação . . . . .	18
3.2	Generalização do Princípio de Validação . . . . .	20
3.2.1	Deteção de Colisões . . . . .	21
3.2.2	Deteção de Ambigüidades . . . . .	22
3.2.3	Ocorrência de Nonevents . . . . .	24
3.2.4	Tratamento de Ciclos . . . . .	26
3.3	Síntese de um Protocolo de Comunicação . . . . .	27
3.3.1	Generalização do Processo de Síntese . . . . .	29
3.4	Relação Existente entre a Técnica Matriz Duologue e outras Técnicas Correlatas . . . . .	32
<b>4</b>	<b>Uma Ferramenta para Validação de Protocolos</b>	<b>35</b>
4.1	Visão Geral da Ferramenta . . . . .	36
4.2	Matriz de Transição de Estados . . . . .	36
4.3	Geração dos Unilogues . . . . .	38

4.3.1	Primeiro Método para a Geração dos Unilogues . . . . .	38
4.3.2	Segundo Método para a Geração dos Unilogues . . . . .	42
4.3.3	Representação dos Unilogues . . . . .	43
4.4	Geração da Matriz Duologue . . . . .	45
4.5	Geração da Matriz de Validação . . . . .	46
4.6	Correção dos Duologues Errôneos . . . . .	51
<b>5</b>	<b>Protocolo X.25 do CCITT</b>	<b>55</b>
5.1	Modelo de Referência OSI da ISO . . . . .	56
5.2	Descrição do Protocolo X.25 . . . . .	58
5.2.1	Fase de Estabelecimento de Conexão . . . . .	60
5.2.2	Fase de Transferência de Dados . . . . .	62
5.2.3	Fase de Desconexão . . . . .	64
5.3	Mecanismos de Recuperação de Erros . . . . .	65
<b>6</b>	<b>Validação do Protocolo X.25 Simplificado</b>	<b>66</b>
6.1	Especificação Formal de uma Versão Simplificada do Protocolo X.25 do CCITT . . . . .	67
6.2	Validação da Versão Simplificada do Protocolo X.25 . . . . .	68
<b>7</b>	<b>Conclusão</b>	<b>75</b>

# Lista de Figuras

2.1	Processo referente ao protocolo de autorização de acesso simplificado.	15
3.1	Protocolo de estabelecimento de conexão simplificado. . . . .	18
3.2	Protocolo de estabelecimento com um caso de colisão. . . . .	21
3.3	Protocolo de estabelecimento de conexão ambíguo. . . . .	23
3.4	Matrizes duologue e validação. . . . .	23
3.5	Protocolo de estabelecimento sem ambigüidades e sua matriz Val[D].	24
3.6	Protocolo que contém <i>nonevents</i> . . . . .	25
3.7	Grafo que apresenta ciclo. . . . .	26
3.8	Primeiro par de grafos submetido a correção. . . . .	27
3.9	Primeira correção no par de grafos. . . . .	28
3.10	Segundo par de grafos submetido a correção. . . . .	28
3.11	Segunda correção no par de grafos. . . . .	29
3.12	Terceiro par de grafos submetido a correção. . . . .	30
3.13	Novas matrizes duologue e validação. . . . .	30
3.14	Terceira correção no par de grafos. . . . .	31
3.15	Par de grafos analisado. . . . .	33
4.1	Arquitetura geral da ferramenta. . . . .	36
4.2	Registro relativo aos elementos da matriz de transição. . . . .	37
4.3	Código pascal da estrutura da matriz de transição de estados. . . . .	38
4.4	Primeiro algoritmo para a geração dos unilogues. . . . .	42
4.5	Segundo algoritmo para a geração dos unilogues. . . . .	43

4.6	Registro referente a um evento. . . . .	43
4.7	Estrutura dos unilogues. . . . .	44
4.8	Código Pascal da estrutura dos unilogues. . . . .	44
4.9	Código Pascal da estrutura de dados dos duologues. . . . .	45
4.10	Código Pascal do procedimento para a geração da matriz duologue. . . . .	46
4.11	Código Pascal do procedimento para construir um duologue. . . . .	46
4.12	Algoritmo para a geração da matriz de validação. . . . .	47
4.13	Código Pascal para identificação de um duologue não ocorrente. . . . .	47
4.14	Código Pascal para identificação de um duologue bem comportado. . . . .	48
4.15	Código Pascal para identificação de uma situação de impasse . . . . .	49
4.16	Código Pascal para identificação de uma recepção não especificada. . . . .	49
4.17	Código Pascal para identificação de uma colisão mal sucedida. . . . .	50
4.18	Código Pascal para identificar duologues que contém nonevents. . . . .	50
4.19	Código Pascal para a análise dos duologues que contém <i>nonevents</i> . . . . .	51
4.20	Código Pascal para a correção de um duologue errôneo. . . . .	52
4.21	Código Pascal do procedimento de pós-transmissão. . . . .	52
4.22	Código Pascal do procedimento de pré-recepção. . . . .	53
4.23	Código pascal do procedimento de completude. . . . .	54
5.1	Estrutura geral do modelo RM-OSI. . . . .	56
5.2	Arquitetura em camadas do protocolo X.25. . . . .	58
5.3	Formato geral dos pacotes X.25 PCP. . . . .	59
5.4	Diagrama para a fase de estabelecimento de conexão. . . . .	61
5.5	Fases de uma chamada virtual do protocolo X.25 PCP. . . . .	65
6.1	Grafo inicial do primeiro projeto. . . . .	67
6.2	Tela da matriz de transição de estados. . . . .	68
6.3	Matriz duologue referente aos grafos iniciais. . . . .	69
6.4	Grafos corrigidos. . . . .	71

6.5	Matriz duologue referente aos grafos corrigidos. . . . .	72
6.6	Entidade fonte do protocolo X.25 PCP. . . . .	74



# Lista de Tabelas

2.1	Matriz de transição de estados. . . . .	16
4.1	Matriz de transição de um processo comunicante. . . . .	40
4.2	Procedimentos utilizados no algoritmo para a geração dos unilogues. . . . .	42
4.3	Procedimentos utilizados na geração da matriz duologue. . . . .	46
5.1	Primitivas de estab. de conexão e pacotes do protocolo X.25 PCP. . . . .	61
5.2	Primitivas de transf. de dados e pacotes do protocolo X.25 PCP. . . . .	62
5.3	Primitivas de desconexão e pacotes do protocolo X.25 PCP. . . . .	64
6.1	Eventos do grafo e pacotes correspondentes. . . . .	67
6.2	Conjunto de unilogues do processo A referente ao primeiro projeto. . . . .	69
6.3	Matriz de validação referente aos grafos iniciais. . . . .	70
6.4	Conjunto de unilogues do processo A referente ao segundo projeto. . . . .	71
6.5	Conjunto de unilogues do processo B referente ao segundo projeto. . . . .	72
6.6	Matriz de validação referente aos grafos corrigidos. . . . .	73

# Capítulo 1

## Introdução

Redes de computadores aparecem como uma das respostas à necessidade de se compartilhar recursos computacionais e informações distribuídas geograficamente. Para que isso ocorra, é necessário que os computadores envolvidos interajam cooperativamente.

Os sistemas de comunicação caracterizam-se pela troca de mensagens entre seus componentes. Para que haja uma comunicação eficiente e segura, é necessário que essas mensagens fluam ordenadamente através dos componentes do sistema. O conjunto de regras e convenções bem definidas que garante essa comunicação é denominado de protocolo de comunicação. A descrição desse conjunto de regras é denominada de especificação do protocolo.

A fim de permitir a interconexão de sistemas abertos, a **International Organization for Standardization (ISO)** apresentou uma arquitetura baseada em camadas hierárquicas, denominada de modelo de referência **Open Systems Interconnection (OSI)** em [ISO 83].

O modelo OSI define 7 camadas, sendo que cada camada executa um conjunto de funções. Em cada camada existe uma interface com a camada superior e outra com a camada inferior. O objetivo de uma camada (N) é prover serviços para a camada (N+1), utilizando os serviços oferecidos pela camada (N-1). Uma vez definidas as funções de cada uma dessas camadas, diferentes implementações podem ser desenvolvidas, utilizando-se diversos tipos de tecnologias, desde que essas implementações

forneçam os serviços especificados.

Os primeiros protocolos de comunicação foram especificados informalmente através de linguagens naturais. Com o aumento da complexidade dos protocolos, constatou-se que esse tipo de especificação causava ambigüidades e inconsistências. Houve uma evolução para especificações semi-formais, onde uma linguagem natural é associada a diagramas de estados e/ou tabelas de transição.

Entretanto, ficou constatado que os problemas relativos às ambigüidades e inconsistências persistiam. Em particular, ficou comprovado que mesmo utilizando-se especificações semi-formais, erros, que poderiam ser detectados e corrigidos nas fases iniciais do ciclo de desenvolvimento de um protocolo, podem proliferar por todas as implementações, tornando o trabalho de depuração extremamente difícil e de custo elevado.

Portanto, é imprescindível que a especificação de um protocolo seja concisa e totalmente livre de ambigüidades. Para atingir tal objetivo, especificações formais, realizadas através de uma **Técnica de Descrição Formal (TDF)**, devem ser produzidas. Além disso, as especificações formais permitem a análise rigorosa, passo a passo, das diversas etapas do ciclo de desenvolvimento de um protocolo.

O projeto de um protocolo de comunicação inicia-se pela formulação de intenções, onde são definidas as principais características do serviço e do protocolo. Em seguida, essas características são transcritas numa linguagem natural associada a diagramas de estados e/ou tabelas de transição, constituindo-se nas especificações semi-formais. Na etapa seguinte são geradas as especificações formais, utilizando-se uma técnica de descrição formal.

A especificação formal de um protocolo não garante que o protocolo venha a operar da maneira desejada. Isto porque erros podem ter sido introduzidos na especificação ou a especificação pode estar incompleta.

A validação do *design* de um protocolo é uma atividade fundamental, que visa assegurar a correção da concepção do protocolo. O termo validação é empregado em qualquer atividade de análise, que tenha por objetivo garantir ou mesmo aumentar a confiabilidade da especificação, do *design* ou da implementação de um dado protocolo.

O objetivo principal deste trabalho foi desenvolver uma ferramenta amigável e interativa, para auxiliar o projetista, na fase inicial do *design* de um protocolo, a identificar e corrigir erros básicos. Essa ferramenta baseia-se na técnica Matriz Duologue proposta em [Zafi 78]. Nessa técnica, um protocolo é especificado através de um par de grafos orientados e a partir desses grafos uma matriz, denominada matriz duologue, é gerada. A atividade de análise é realizada tendo-se como base essa matriz. Para demonstrar a utilidade dessa ferramenta foram validados vários protocolos, sendo que o mais significativo foi uma versão simplificada do protocolo X.25 do CCITT<sup>1</sup>.

O capítulo 2 apresenta as vantagens do uso de técnicas de descrição formal, enfatiza a importância da atividade de validação e apresenta a especificação de um protocolo através de um par de grafos. O capítulo 3 descreve os conceitos fundamentais da técnica Matriz Duologue e a construção e validação dessa matriz são exemplificadas através de um protocolo de estabelecimento de conexão simplificado. No capítulo 4 é apresentada a ferramenta de validação, dando ênfase à sua arquitetura e aos principais algoritmos envolvidos. No capítulo 5 é descrito o protocolo X.25 do CCITT. No capítulo 6 é especificada e validada, com auxílio da ferramenta, uma versão simplificada do protocolo X.25 do CCITT. Finalmente, no capítulo 7 são apresentadas as conclusões.

---

<sup>1</sup>Consultative Committee for International Telegraph and Telephone

## Capítulo 2

# Especificação e Validação Formais de Protocolos

As especificações informais de sistemas de comunicação causam ambigüidades que dificultam o desenvolvimento desses sistemas. A utilização de especificações semi-formais constituíram-se num avanço, porém não permitiam uma análise rigorosa das diversas etapas do ciclo de desenvolvimento de um protocolo. O uso de especificações formais faz com que sejam reduzidas as possibilidades de erros e o risco de serem obtidas implementações inconsistentes, uma vez que o desenvolvimento do sistema basear-se-á, sobretudo, na sua especificação formal.

## 2.1 Especificação Formal

No final da década de 70, TDFs começaram a ser propostas para as especificações formais de serviços e protocolos de comunicação. Segundo o modelo utilizado, essas técnicas podem ser classificadas em [LoSt 88]:

- **técnicas baseadas em modelos de transição** - as TDFs baseadas em modelos de transição são eficazes para a descrição dos aspectos de controle dos protocolos (fases de inicialização, de estabelecimento e de encerramento de conexão). Para protocolos mais complexos, com um grande número de estados, tais TDFs tendem a ocasionar o fenômeno denominado explosão de estados, sobretudo durante as atividades de análise das especificações. As TDFs mais conhecidas, baseadas em modelo de transição, são: Redes de Petri, Máquinas de Estados Finitas (MEFs) e Linguagens Formais;
- **técnicas baseadas em linguagens de programação** - as linguagens de programação também podem ser utilizadas para a especificação de serviços e protocolos. Isto porque a execução de um protocolo é normalmente seqüencial, podendo ser descrita através de um algoritmo, que por sua vez pode ser especificado de forma clara e precisa através de uma linguagem de programação de alto nível (*e.g.*, Pascal, C e ADA). As TDFs baseadas em linguagens de programação são eficazes para a descrição dos aspectos relativos à transferência e às estruturas de dados dos protocolos;
- **técnicas híbridas** - as técnicas híbridas normalmente são frutos de uma combinação das duas técnicas anteriores. Os aspectos de controle de um protocolo podem ser especificados utilizando-se um modelo de transição, enquanto que as variáveis e outros parâmetros podem ser especificados numa linguagem de alto nível e manipulados por procedimentos também descritos nessa linguagem.

A concepção e a utilização de TDFs, para a especificação de serviços e protocolos de comunicação, têm sido motivo de grande interesse por parte de pesquisadores, de fabricantes de computadores e de órgãos internacionais de padronização.

Durante o ciclo de desenvolvimento de um protocolo de comunicação, as especificações formais, além de facilitar o próprio desenvolvimento do protocolo, devem ser referências confiáveis para as atividades de validação e implementação. Para que esses objetivos possam ser alcançados, é necessário que a TDF utilizada, possua:

- **um alto grau de abstração**, no sentido de independência em relação aos métodos de implementação e no sentido de omissão, em qualquer etapa da especificação, dos detalhes irrelevantes;
- **poder de expressão**, isto é, suas construções devem fornecer meios para expressar comunicação, sincronização e concorrência;
- **poder de análise**, ou seja, disponha de um modelo matemático, que permita a verificação formal das propriedades desejadas para os objetos que estão sendo especificados.

## 2.2 Validação Formal

A validação é uma atividade que pode ser exercitada ao longo do ciclo de desenvolvimento de um protocolo. As atividades de validação analisam uma entidade (especificação, *design* ou implementação), buscando garantir-lhe uma maior confiabilidade.

As técnicas de validação podem ser compreendidas em três categorias principais: **verificação, simulação e teste**.

O objetivo principal da simulação e do teste é o mesmo. Nos dois casos, uma entidade é exercitada visando a detecção de erros. Na simulação, a entidade é executada como uma caixa branca (acesso ao seu interior), enquanto que no teste a entidade é executada como uma caixa preta (acesso somente às interfaces).

A grande vantagem da simulação e do teste é que podem ser aplicados a protocolos complexos. Entretanto, estes não podem fornecer resultados definitivos, em relação à ausência de erros, já que não podem ser aplicados exaustivamente.

O objetivo principal da verificação é analisar uma entidade, utilizando algum tipo de raciocínio lógico, visando provar se esta possui ou não determinadas propriedades que lhe são requeridas. Por exemplo, é desejável a ausência de:

- **estados de impasse:** estado global, com o meio vazio, a partir do qual não é possível transmitir novas mensagens;
- **interações não executáveis:** recepção e/ou transmissão de mensagens que não podem ocorrer em condições normais de operação;
- **transbordamento nos canais:** ocorre quando o número de mensagens num canal excede a sua capacidade pré-definida;
- **crescimento ilimitado:** ocorre quando a taxa de transmissão do processo fonte é superior à taxa de recepção do processo destino;
- **laços indesejáveis:** laços de execução que não contribuem para a progressão efetiva do processo.

É desejável também que o protocolo possua:

- **vivacidade:** existência de um estado de retorno que possa ser alcançado a partir de qualquer estado global válido;
- **estabilidade:** a partir de qualquer estado global o sistema sempre retorna, após um número finito de transições, ao ciclo normal de operação;
- **término apropriado:** a execução do sistema termina no estado global final previsto.

A grande vantagem da verificação é que pode fornecer resultados definitivos a respeito das propriedades verificadas. Entretanto, para a sua aplicação, normalmente é necessário impor limitações aos modelos utilizados, o que dificulta o seu emprego em protocolos complexos.

O tipo de propriedade a ser verificada depende da TDF empregada na especificação do protocolo e do tipo de Técnica de Verificação Formal (TVF) a ser utilizada. Frequentemente, quando a TDF é baseada em modelos de transição, as TVFs



utilizadas são baseadas em análise de alcançabilidade, enquanto que se a TDF é baseada em linguagens de programação, as TVFs utilizadas são baseadas em provas de programas. Quando a TDF é híbrida a TVF também o será [WeZa 80].

Uma determinada TVF normalmente permite verificar algumas das propriedades requeridas a um protocolo. A verificação será tanto mais abrangente quanto maior for o número de propriedades investigadas. Portanto, no ciclo de desenvolvimento de um protocolo, várias TVFs podem ser empregadas a fim de outorgar uma maior confiabilidade ao protocolo.

Na fase inicial de *design* de um protocolo, o projetista tende a cometer erros básicos. Por exemplo: mensagens emitidas por uma entidade de protocolo que não são recebidas pela entidade par; as entidades não são corretamente inicializadas; o protocolo não termina apropriadamente; *etc.*. Para corrigir tais erros e auxiliar o projetista nessa fase inicial, TDFs e TVFs mais simples podem ser empregadas. Visto que são técnicas tratáveis, em termos de complexidade, o processo de detecção e correção de erros pode ser mais facilmente automatizado.

## 2.3 Especificação de um Protocolo através de um par de Grafos

A aplicação de grafos para a resolução de problemas teve início em 1736, quando Euler utilizou-os para solucionar o problema da ponte de Königsberg [HoSa 87]. Desde então, os grafos têm sido utilizados numa ampla variedade de aplicações: análise sintática, representação de automâtos, análise de circuitos elétricos, *etc.*

Um grafo  $G$  é representado pela dupla  $G = (V, A)$ , onde  $V$  representa um conjunto de vértices (ou nós) e  $A$  representa um conjunto de arcos (ou arestas) que ligam os vértices. Cada arco em  $G$  é especificado por um par de vértices.

Com respeito à ordenação dos vértices, os grafos podem ser orientados e não-orientados. O segundo tipo não tem ordenação especial para representar um determinado arco. Assim sendo, os pares de vértices  $(v_1, v_2)$  e  $(v_2, v_1)$  representam o mesmo arco. Num grafo orientado, cada arco é representado por um par  $(v_1, v_2)$ ,

onde  $v_1$  representa a extremidade inicial do arco, enquanto que  $v_2$  representa a extremidade final ou terminal do arco. Portanto, num grafo orientado,  $(v_1, v_2)$  e  $(v_2, v_1)$  representam dois arcos diferentes.

Um protocolo de comunicação pode ser definido através das interações entre dois processos comunicantes. Esses processos, por sua vez, podem ser descritos através de grafos orientados. Desta forma, cada um dos processos comunicantes está associado a um desses grafos. Os nós do grafo orientado podem representar os estados de um processo e os arcos podem indicar as possíveis transições de estados.

A unidade básica de comunicação entre os processos é o evento. Para representar eventos; os arcos orientados podem ser rotulados com números inteiros. A transmissão de um evento é representada por um número inteiro negativo, enquanto que a sua recepção é representada pelo mesmo valor positivo. A transmissão ou a recepção de um evento provoca uma transição no grafo. As transições que não são provocadas por transmissões ou recepções de eventos são denominadas *nonevents*.

O grafo da Figura 2.1 representa um dos processos de um protocolo de autorização de acesso simplificado. As mensagens entre os processos são representadas por números inteiros. Nesse exemplo, a mensagem Acesso Requisitado é representada pelo evento do tipo 1, enquanto que a mensagem Acesso Concedido é representada pelo evento do tipo 3.

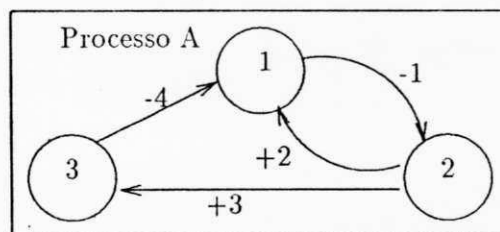


Figura 2.1: Processo referente ao protocolo de autorização de acesso simplificado.

Em relação ao exemplo da Figura 2.1, o rótulo  $(-1)$ , entre os nós 1 e 2, indica um evento do tipo 1 sendo transmitido para um outro processo. O rótulo  $(+3)$ , entre os nós 2 e 3, indica que essa transição somente pode ocorrer quando o evento correspondente for recebido, ou seja, quando o outro processo transmitir o evento do tipo 3.

Uma vez obtido os grafos orientados, correspondentes aos processos comunicantes, é conveniente representá-lo por algum tipo de estrutura de dados. Por exemplo, esses grafos podem ser representados através de matrizes de transição de estados.

Uma matriz é um arranjo bidimensional formado de  $n$  linhas e  $m$  colunas. A matriz de transição de estados de um grafo  $G$  é um arranjo bidimensional  $n \times m$ . A partir dessa matriz, pode-se facilmente determinar se existe um arco que liga quaisquer dois nós  $i$  e  $j$ . Uma transição do estado  $i$  para o estado  $j$  é causada por um evento transmitido ou recebido. As linhas dessa matriz representam os estados vigentes, enquanto que as colunas representam os próximos estados, que podem ser alcançados quando ocorre uma transição no grafo. Essas transições são provocadas pelas transmissões e/ou recepções de eventos. Diferentes tipos de eventos podem provocar uma mesma transição.

A Tabela 2.1 representa a matriz de transição de estados correspondente ao grafo da Figura 2.1. Essa matriz indica todas as possíveis transições que podem ocorrer nesse grafo. Os elementos dessa matriz representam os eventos que são transmitidos e/ou recebidos.

	1	2	3
1	$\lambda$	-1	$\lambda$
2	+2	$\lambda$	+3
3	-4	$\lambda$	$\lambda$

Tabela 2.1: Matriz de transição de estados.

Cada elemento  $a_{ij}$ , dessa matriz de transição de estados, corresponde à transmissão ou à recepção de eventos do nó  $i$  para o nó  $j$ , sendo que  $\lambda$  indica que não existe um arco ligando esses nós. A numeração das linhas dessa matriz representa os estados vigentes, enquanto que a numeração das colunas representa os próximos estados.

## Capítulo 3

# Construção e Validação da Matriz Duologue

Na técnica Matriz Duologue um protocolo é modelado por um par de processos comunicantes, sendo que cada processo é representado por um grafo orientado. A partir desse par de grafos, matrizes de transição de estados são obtidas e a partir dessas matrizes uma matriz de validação pode ser gerada.

Após a detecção dos erros de *design*, através da matriz de validação, tais erros podem ser corrigidos, de acordo com determinadas condições pré-estabelecidas, e os grafos podem ser reprojitados. Isso caracteriza um procedimento de síntese para os protocolos.

### 3.1 Princípio de Validação

Para ilustrar esse princípio, é utilizado um protocolo de estabelecimento de conexão simplificado. O par de grafos relativo a esse protocolo é apresentado na Figura 3.1.

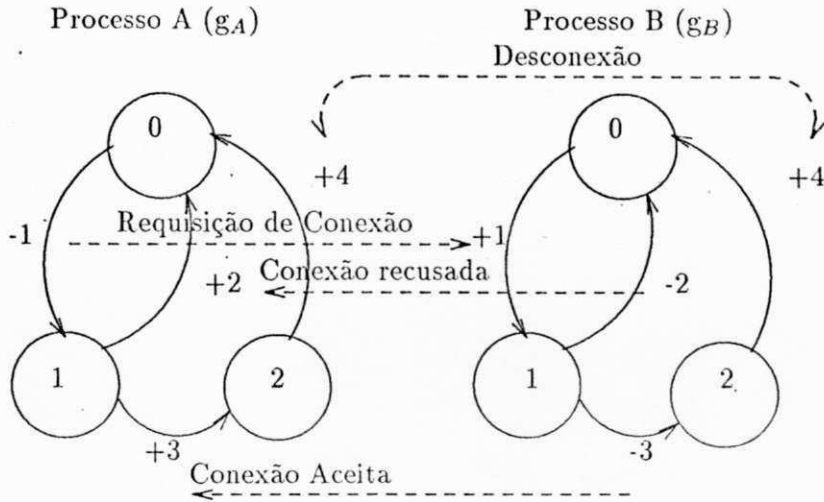


Figura 3.1: Protocolo de estabelecimento de conexão simplificado.

Em relação ao exemplo da Figura 3.1, é assumido que os processos começam e terminam no estado inicial 0. Levando-se em consideração somente as seq"uências de eventos que iniciam e terminam no estado 0, sem atravessá-lo, obtém-se:

Processo A :  $\{-1,+2\}$  ;  $\{-1,+3,+4\}$

Processo B :  $\{+1,-2\}$  ;  $\{+1,-3,+4\}$

Essas seq"uências, obtidas por inspeção dos grafos, são chamadas unilogues. Os conjuntos de todos os unilogues referentes aos grafos  $g_A$  e  $g_B$  são:

$$S_A = [A_1 = \{-1,+2\} ; A_2 = \{-1,+3,+4\}]$$

$$S_B = [B_1 = \{+1,-2\} ; B_2 = \{+1,-3,+4\}]$$

Um duologue é formado pela combinação de dois unilogues pertencentes a diferentes grafos (e.g.,  $[A_1, B_1]$ ). O conjunto de todos os duologues é obtido através do produto cartesiano  $S_A \times S_B$ . Em relação ao exemplo da Figura 3.1, o conjunto de duologues é:

$$S_A \times S_B = \{[A_1, B_1], [A_1, B_2], [A_2, B_1], [A_2, B_2]\}$$

Dispondo esse conjunto de duologues na forma de uma matriz, obtém-se a matriz duologue:

$$D = \begin{pmatrix} [A_1, B_1] & [A_1, B_2] \\ [A_2, B_1] & [A_2, B_2] \end{pmatrix}$$

Existem três tipos principais de duologues : **bem comportado**, **não ocorrente** e **errôneo**.

Um duologue é classificado como **bem comportado** se:

- os eventos transmitidos por um unilogue são recebidos pelo outro unilogue.

Isto significa que:

- o processo A não pode receber um evento antes que esse tenha sido transmitido pelo processo B e vice-versa;
- o processo A deve ser capaz de receber todos os eventos transmitidos pelo processo B, mantendo-se a ordem de transmissão, e vice-versa;
- o processo A deve sempre estar em um estado a partir do qual possa receber o próximo evento e o mesmo deve acontecer com o processo B.

- o duologue sempre termina, isto é, o par de estados iniciais é sempre atingido.

Um duologue é classificado como **não ocorrente** se:

- o duologue nunca pode ser iniciado ou
- qualquer tentativa de executá-lo leva, invariavelmente, à execução de um outro duologue.

Um duologue é classificado como **errôneo** se ele é mal comportado e ocorrente, ou seja, se ele não pertence às duas classes anteriormente citadas.

Para verificar se um protocolo contém erros básicos de *design*, aplica-se uma função de validação aos elementos da matriz duologue.

A cada duologue  $[A_i, B_k]$  pode-se aplicar uma função de validação *Val*. Essa função retorna:

- +1 se o duologue é bem comportado;
- 0 se o duologue é não ocorrente;
- -1 se o duologue é errôneo.

Em relação ao exemplo da Figura 3.1, tem-se:

- o duologue  $[A_1, B_1] = [\{-1, +2\}; \{+1, -2\}]$  é bem comportado;
- o duologue  $[A_1, B_2] = [\{-1, +2\}; \{+1, -3, +4\}]$  é não ocorrente, pois ao tentar executá-lo, executa-se o duologue  $[A_2, B_2] = [\{-1, +3, +4\}; \{+1, -3, +4\}]$ ;
- o duologue  $[A_2, B_1] = [\{-1, +3, +4\}; \{+1, -2\}]$  é não ocorrente, pois ao tentar executá-lo, executa-se o duologue  $[A_1, B_1] = [\{-1, +2\}; \{+1, -2\}]$ ;
- o duologue  $[A_2, B_2] = [\{-1, +3, +4\}; \{+1, -3, +4\}]$  é errôneo, pois não foi prevista a transmissão do evento 4 e o protocolo é levado a uma situação de impasse.

Aplicando-se a função de validação  $Val$ , a cada duologue da matriz  $D$ , obtém-se a matriz de validação  $Val[D]$ .

$$Val[D] = \begin{pmatrix} +1 & 0 \\ 0 & -1 \end{pmatrix}$$

Uma vez obtida a matriz de validação, o projetista deve concentrar sua atenção apenas nos duologues errôneos, ou seja, nos elementos -1 dessa matriz. A matriz  $Val[D]$  acima indica que há um duologue errôneo. Foi, portanto, cometido um erro de *design* no protocolo de estabelecimento de conexão simplificado referente à Figura 3.1.

### 3.2 Generalização do Princípio de Validação

Para generalizar o princípio de validação, é necessário considerar os seguintes casos particulares, que podem ocorrer quando da especificação do protocolo através de um par de grafos: colisões, ambigüidades, nonevents e ciclos.

### 3.2.1 Detecção de Colisões

As colisões ocorrem quando os processos comunicantes envolvidos transmitem *simultaneamente* eventos.

No tratamento das colisões geralmente é considerado que os processos operam assincronamente, ou seja, todas as mensagens transmitidas pelo processo A podem ser recebidas pelo processo B, após decorrido algum intervalo de tempo (e vice-versa). Caso seja prevista a recepção *simultânea* dos eventos que colidiram, o problema da colisão estará resolvido.

Um exemplo de um protocolo, onde ocorre uma colisão, é apresentado na Figura 3.2.

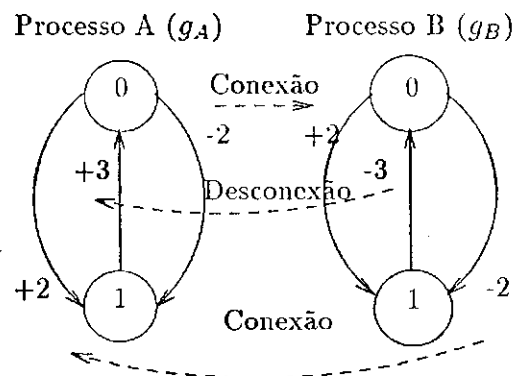


Figura 3.2: Protocolo de estabelecimento com um caso de colisão.

Em relação ao exemplo da Figura 3.2, os unilogues referentes aos grafos  $g_A$  e  $g_B$  são:

$$\text{Processo A: } [A_1 = \{-2, +3\}; A_2 = \{+2, +3\}]$$

$$\text{Processo B: } [B_1 = \{+2, -3\}; B_2 = \{-2, -3\}]$$

Considerando-se os duologues, relativos ao par de grafos da Figura 3.2, tem-se que no duologue  $[A_1, B_2] = [\{-2, +3\}; \{-2, -3\}]$  ambos os processos transmitem *simultaneamente* o evento 2, provocando uma colisão. Esse duologue é ocorrente e uma vez que não foi prevista a recepção *simultânea* do evento 2, ele também é mal comportado. Portanto, o duologue  $[A_1, B_2]$  é errôneo.



Existem casos de colisões que envolvem eventos e *nonevents*. Para esses casos, a técnica Matriz Duologue apresenta uma solução que será abordada na seção 3.3.1, que trata do processo de síntese de um protocolo.

### 3.2.2 Detecção de Ambiguidades

Um protocolo não apresenta ambiguidades se, a qualquer instante, o conhecimento da situação da entidade local é suficiente para determinar a situação da entidade remota e vice-versa.

Por exemplo, considerando-se a situação de um protocolo representada por estados, se existe algum estado alcançável por uma das entidades que possa coexistir, estavelmente, com mais de um estado da outra entidade, o protocolo é considerado ambíguo.

No caso da técnica Matriz Duologue, a situação de um protocolo é determinada através das seqüências de interações (duologues). Portanto, um protocolo será considerado ambíguo, se para uma mesma seqüência de eventos (unilogues) de um processo existe mais de uma seqüência de eventos possível para o outro processo.

Pode-se determinar as ambiguidades de um protocolo, através da matriz de validação, da seguinte forma:

- se a  $i$ ésima linha ( $k$ ésima coluna) da matriz de validação contém mais de um elemento +1, então o comportamento do processo B (processo A) é ambíguo em relação ao processo A (processo B), quando o unilogue  $A_i$  ( $B_k$ ) é executado.

Um exemplo de um protocolo de estabelecimento de conexão ambíguo é apresentado na Figura 3.3.

Em relação ao exemplo da Figura 3.3, os conjuntos de todos os unilogues referentes aos grafos  $g_A$  e  $g_B$  são:

Processo A :  $[A_1 = \{-2, +2, +3\}; A_2 = \{+2, -2, +3\}]$

Processo B :  $[B_1 = \{+2, -2, -3\}; B_2 = \{-2, +2, -3\}]$

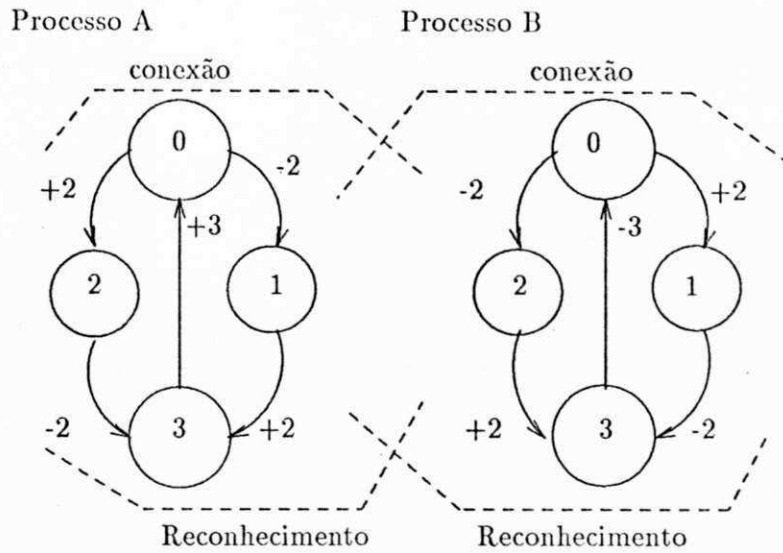


Figura 3.3: Protocolo de estabelecimento de conexão ambíguo.

As matrizes duologue e validação, referentes aos grafos da Figura 3.3, são apresentadas na Figura 3.4.

$$D = \begin{pmatrix} [A_1, B_1] & [A_1, B_2] \\ [A_2, B_1] & [A_2, B_2] \end{pmatrix}$$

$$Val[D] = \begin{pmatrix} +1 & +1 \\ 0 & +1 \end{pmatrix}$$

Figura 3.4: Matrizes duologue e validação.

Em relação a esse exemplo, a matriz de validação resultante contém dois elementos +1 na sua primeira linha. Conseqüentemente, quando o processo A executa o unilogue  $A_1$ , ele não sabe se o processo B está executando o unilogue  $B_1$  ou  $B_2$ . Portanto, o comportamento do processo B é ambíguo em relação ao processo A. Da mesma forma, os dois elementos +1 na segunda coluna identificam uma ambigüidade do processo A em relação ao processo B.

Neste caso específico, as ambigüidades são eliminadas introduzindo um novo evento (evento de reconhecimento). Além disso, é necessário decidir qual dos processos comunicantes terá prioridade, caso ocorra uma colisão. A colisão ocorre quando o processo A e o processo B transmitem *simultaneamente* o evento 2. Em relação ao

exemplo da Figura 3.3, a colisão é decidida em favor do processo A. O par de grafos corrigidos e sua matriz de validação são apresentadas na Figura 3.5.

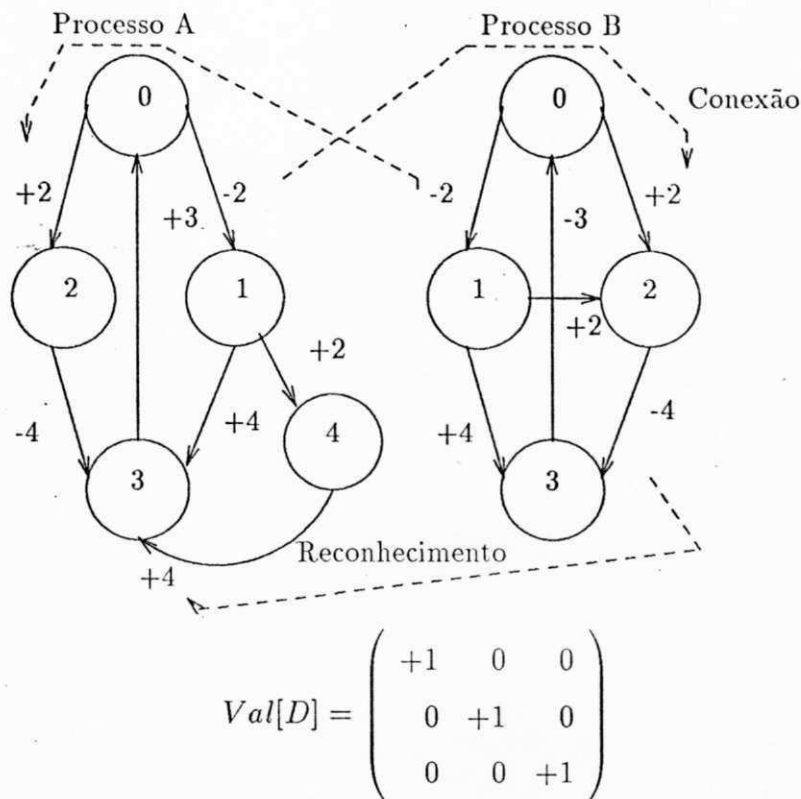


Figura 3.5: Protocolo de estabelecimento sem ambigüidades e sua matriz  $Val[D]$ .

Ambigüidades não representam necessariamente erros de *design*. Dependendo do protocolo que está sendo projetado, os processos envolvidos podem conviver com ambigüidades, desde que não seja importante para uma entidade conhecer, a todo instante, a situação da entidade parceira.

### 3.2.3 Ocorrência de Nonevents

Na técnica Matriz Duologue, a especificação de um protocolo é baseada na descrição dos eventos trocados entre os processos. As transições de um processo, que são provocadas por eventos que não são fruto de uma transmissão ou recepção envolvendo o outro processo, são tratadas como *nonevents* (e.g., temporização). Num grafo orientado, os *nonevents* são representados por 0.

Um exemplo de um protocolo, onde ocorre *nonevents*, é apresentado na Figura 3.6.

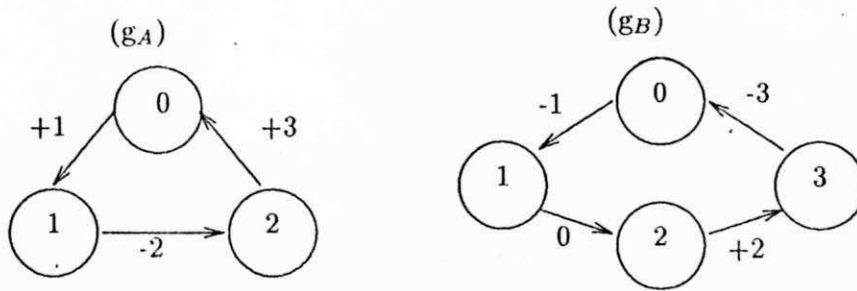


Figura 3.6: Protocolo que contém *nonevents*.

Em relação ao exemplo da Figura 3.6, os unilogues referentes aos grafos  $g_A$  e  $g_B$  são:

Processo A :  $[A_1 = \{+1, -2, +3\}]$

Processo B :  $[B_1 = \{-1, 0, +2, -3\}]$

Uma vez que o duologue  $[A_1, B_1]$  é único e pode ser iniciado, então ele é ocorrente. Resta saber se ele é bem comportado. Dois casos devem ser analisados:

- após a ocorrência da interação 1, se  $g_B$  executar o *nonevent* antes da chegada do evento 2, o duologue será bem comportado;
- após a ocorrência da interação 1, se  $g_B$  não executar o *nonevent* antes da chegada do evento 2, esse evento não será recebido e, portanto, o duologue será mal comportado e, conseqüentemente, errôneo.

A forma de correção desses duologues é abordada na seção 3.3.1 e faz parte do processo de síntese de um protocolo.

### 3.2.4 Tratamento de Ciclos

Grafos que apresentam ciclos, que fazem parte de seqüências de eventos que começam e terminam no estado inicial, podem, potencialmente, gerar unilogues de comprimento infinito. Para evitar esse problema, uma possível solução é considerar tamanhos de unilogues suficientemente grandes, possibilitando que os ciclos sejam atravessados pelo menos uma única vez por um ou mais unilogues. Para evitar erros de *design* em um ciclo, pode-se impor que qualquer ciclo em um dos grafos orientado deve ter um correspondente ciclo no outro grafo orientado.

Um exemplo de um grafo orientado, onde ocorre ciclo, é apresentado na Figura 3.7.

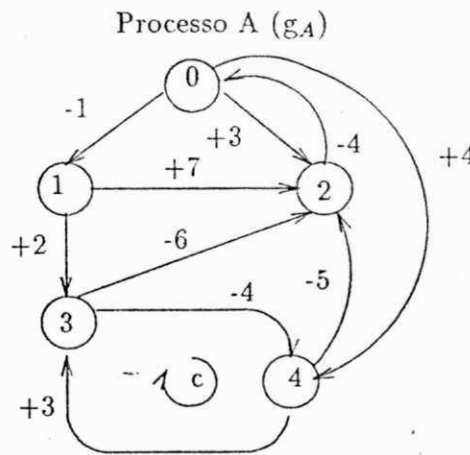


Figura 3.7: Grafo que apresenta ciclo.

Em relação ao grafo da Figura 3.7, para se obter uma boa validação é necessário considerar unilogues de comprimento sete. Esse número é suficiente já que o ciclo é atravessado uma única vez em todos os casos possíveis. O comprimento máximo dos unilogues é determinado através da seguinte fórmula:

$$TamMaxUnilogue = Numero\_de\_Vertices + \sum_{c \in Ciclo} Tam(c)$$

Uma vez identificados os duologues errôneos, pode-se analisar e tentar corrigir esses duologues. A técnica Matriz Duologue prescreve algumas condições para a correção de um duologue errôneo.

### 3.3 Síntese de um Protocolo de Comunicação

A matriz de validação identifica a presença de duologues errôneos que serão corrigidos. Essa correção pode ser realizada aplicando-se algumas condições básicas.

Uma primeira correção pode ser realizada aplicando-se, ao duologue errôneo, a condição de pós-transmissão:

- quando um duologue é executado, então para todas as transmissões de eventos, se um unilogue  $A_i$  (unilogue  $B_k$ ) contém uma transmissão de um evento, então essa transmissão deve corresponder a uma subsequente recepção desse evento no unilogue  $B_k$  (unilogue  $A_i$ ).

A condição de pós-transmissão pode ser aplicada aos grafos da Figura 3.8.

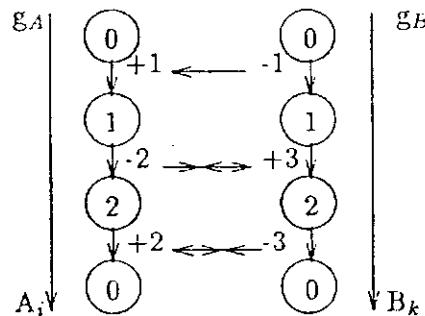


Figura 3.8: Primeiro par de grafos submetido a correção.

Considerando-se o duologue

$$[A_i, B_k] = [\{+1, -2, +2\}, \{-1, +3, -3\}]$$

ele é mal comportado, pois o unilogue  $A_i$  não foi projetado para receber o evento 3 transmitido pelo unilogue  $B_k$ . Da mesma forma, o unilogue  $B_k$  não foi projetado para receber o evento 2 transmitido pelo unilogue  $A_i$ .

Aplicando-se a condição de pós-transmissão no duologue  $[A_i, B_k]$ , obtém-se:

$$[A_i, B_k] = [\{+1, -2, +3\}, \{-1, +2, -3\}]$$

Portanto, após a aplicação dessa condição, o unilogue  $B_k$  foi corrigido para receber o evento 2 transmitido pelo unilogue  $A_i$ , e o unilogue  $A_i$  foi corrigido para receber

o evento 3 transmitido pelo unilogue  $B_k$ . Os grafos corrigidos são apresentados na Figura 3.9.

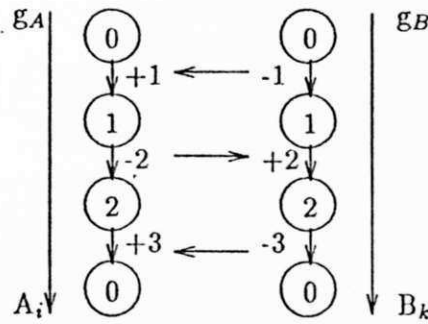


Figura 3.9: Primeira correção no par de grafos.

Se um duologue errôneo não se enquadra na condição de pós-transmissão, então deve-se aplicar a condição de pré-recepção:

- quando um duologue é executado, então para todas as recepções de eventos, se o unilogue  $A_i$  (unilogue  $B_k$ ) contém uma recepção de um evento, então essa recepção deve corresponder a uma prévia transmissão desse evento pelo unilogue  $B_k$  (unilogue  $A_i$ ).

A condição de pré-recepção pode ser aplicada aos grafos da Figura 3.10.

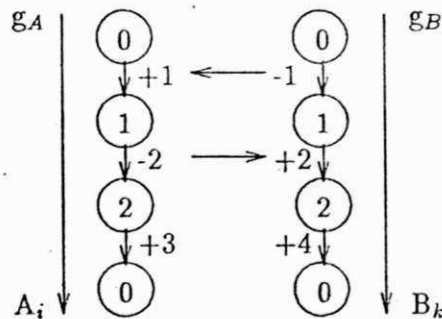


Figura 3.10: Segundo par de grafos submetido a correção.

Considerando-se o duologue

$$[A_i, B_k] = [\{+1, -2, +3\}, \{-1, +2, +4\}]$$

este satisfaz a condição de pós-transmissão. Todavia, ele continua a ser mal comportado, já que esse duologue não termina devido a uma situação de impasse:

o processo  $A_i$  espera a recepção do evento 3, enquanto que o processo  $B_k$  espera a recepção do evento 4.

Aplicando-se a condição de pré-recepção ao duologue  $[A_i, B_k]$ , obtém-se:

$$[A_i, B_k] = \{ \{+1, -2, +3, -4\}, \{-1, +2, -3, +4\} \}$$

Portanto, após a aplicação dessa condição, o unilogue  $A_i$  foi corrigido para transmitir o evento 4 e o unilogue  $B_k$  foi corrigido para transmitir o evento 3. Os grafos corrigidos são apresentados na Figura 3.11.

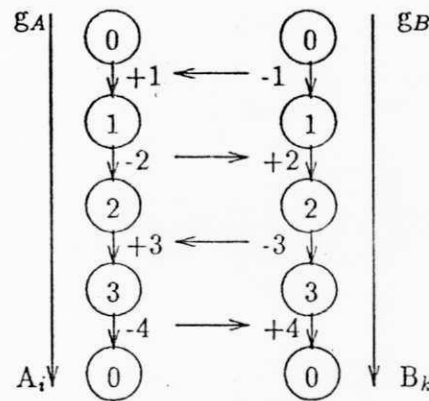


Figura 3.11: Segunda correção no par de grafos.

### 3.3.1 Generalização do Processo de Síntese

Uma última correção pode ser realizada em um duologue errôneo, quando este está sujeito a certas restrições envolvendo tempo. Essa correção é baseada na condição de completude:

- para todas as transmissões de eventos em um duologue  $[A_i, B_k]$ , se o unilogue  $A_i$  (unilogue  $B_k$ ) transmite um evento  $n$ , que pode colidir com um nonevent do unilogue  $B_k$  (unilogue  $A_i$ ), então o processo  $B$  (processo  $A$ ) deve ser projetado para receber  $n$  independentemente da ocorrência do nonevent



A condição de completude pode ser aplicada aos grafos da Figura 3.12.

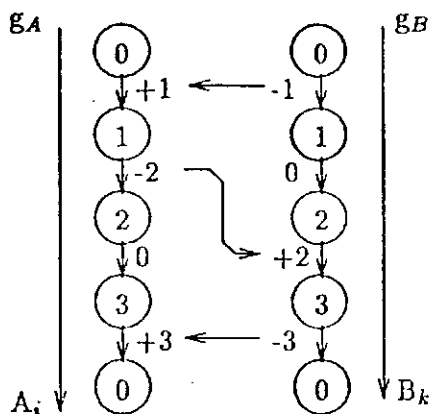


Figura 3.12: Terceiro par de grafos submetido a correção.

Considerando-se o duologue

$$[A_i, B_k] = [\{+1, -2, 0, +3\}, \{-1, 0, +2, -3\}]$$

ele satisfaz as condições de pós-transmissão e pré-recepção. Entretanto, se o unilogue  $A_i$  transmite o evento 2 antes que o unilogue  $B_k$  deixe o estado 1, esse evento pode encontrar o unilogue  $B_k$  ainda no estado 1 e, portanto, este não será recebido. O mesmo ocorre com a transmissão do evento 3 pelo unilogue  $B_k$ .

Aplicando-se a condição de completude ao duologue  $[A_i, B_k]$ , novos unilogues devem ser gerados para realizar a correção:

$$\text{Processo A : } [A_1 = \{+1, -2, 0, +3\}; A_2 = \{+1, -2, +3, 0\}]$$

$$\text{Processo B : } [B_1 = \{-1, 0, +2, -3\}; B_2 = \{-1, +2, -3\}]$$

As novas matrizes duologue e validação são apresentadas na Figura 3.13.

$$D = \begin{pmatrix} [A_1, B_1] & [A_1, B_2] \\ [A_2, B_1] & [A_2, B_2] \end{pmatrix} \text{Val}[D] = \begin{pmatrix} 0 & 0 \\ 0 & +1 \end{pmatrix}$$

Figura 3.13: Novas matrizes duologue e validação.

Em relação a essas matrizes, têm-se:

- no duologue  $[A_1, B_1] = [\{+1, -2, 0, +3\}; \{-1, 0, +2, -3\}]$ , se o unilogue  $A_1$  transmite o evento 2 antes que o unilogue  $B_1$  possa receber esse evento, então deve-se considerar a possibilidade da execução de outro duologue. Portanto, esse

duologue é não ocorrente, pois ao tentar executá-lo, executa-se o duologue  $[A_1, B_2] = \{ \{+1, -2, 0, +3\}; \{-1, +2, -3\} \}$ ;

- no duologue  $[A_1, B_2] = \{ \{+1, -2, 0, +3\}; \{-1, +2, -3\} \}$ , se o unilogue  $B_2$  transmite o evento 3 antes que o unilogue  $A_1$  possa receber esse evento, então deve-se considerar a possibilidade da execução de outro duologue. Portanto, esse duologue é não ocorrente, pois ao tentar executá-lo, executa-se o duologue  $[A_2, B_2] = \{ \{+1, -2, +3, 0\}; \{-1, +2, -3\} \}$ ;
- no duologue  $[A_2, B_1] = \{ \{+1, -2, +3, 0\}; \{-1, 0, +2, -3\} \}$ , se o unilogue  $A_2$  transmite o evento 2 antes que o unilogue  $B_1$  possa receber esse evento, então deve-se considerar a possibilidade da execução de outro duologue. Portanto, esse duologue é não ocorrente, pois ao tentar executá-lo, executa-se o duologue  $[A_2, B_2] = \{ \{+1, -2, +3, 0\}; \{-1, +2, -3\} \}$ ;
- o duologue  $[A_2, B_2] = \{ \{+1, -2, +3, 0\}; \{-1, +2, -3\} \}$  é bem comportado, pois os eventos transmitidos por um unilogue são sempre recebidos pelo outro unilogue e após a execução desse duologue, o processo B retorna imediatamente ao estado inicial, enquanto que o processo A retorna ao estado inicial após à execução de um *nonevent*.

Os grafos corrigidos, referentes a uma nova especificação desse protocolo, são apresentadas na Figura 3.14.

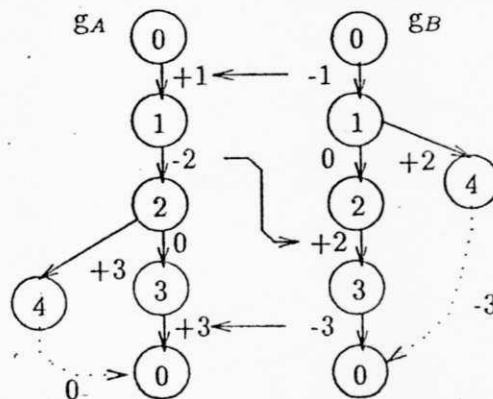


Figura 3.14: Terceira correção no par de grafos.

### 3.4 Relação Existente entre a Técnica Matriz Duologue e outras Técnicas Correlatas

Baseada na técnica Matriz Duologue, outras técnicas correlatas surgiram: Diagrama de Fase[West 78] e Pertubação[WeZa 80].

A técnica Diagrama de Fase é uma implementação da técnica Matriz Duologue, onde cada duologue é representado graficamente através de um diagrama de fase. Esse diagrama permite uma clara representação da ordem relativa das transições dos dois processos que executam o duologue.

Na técnica Matriz Duologue, a teoria dos conjuntos e a lógica de predicados são utilizadas para a deteção de erros. Na técnica Diagrama de Fase, a partir da representação gráfica obtida, dois conjuntos ordenados de inteiros positivos são gerados. A partir desses conjuntos é que os erros são detectados.

A técnica Diagrama de Fase é limitada a protocolos que não contenham ciclos, restringindo a análise já que tais ciclos são comuns em protocolos complexos.

A técnica de Pertubação é utilizada para gerar a partir dos grafos, a árvore de alcançabilidade relativa ao protocolo. Através da análise dessa árvore é que erros de *design* são detectados e posteriormente corrigidos.

Cada nó da árvore de alcançabilidade guarda, para cada processo comunicante, o seu estado vigente no grafo e os eventos que são transmitidos e/ou recebidos. A raiz dessa árvore corresponde ao estado inicial do sistema (estados iniciais dos processos nos grafos).

Para a determinação dos estados alcançáveis a partir do estado inicial, este é perturbado através da execução das possíveis transições pertinentes aos processos envolvidos. Para cada transição, individualmente executada, um novo nó da árvore é gerado. Esse procedimento é repetido para cada nó obtido, até que as folhas da árvore sejam atingidas.

Uma vez obtida a árvore de alcançabilidade, quatro tipos de erros de *design* podem ser identificados: estados de impasse, recepções não especificadas, interações não executáveis e estados ambíguos.

Um estado de impasse ocorre quando transições de estado não são mais possíveis a partir de um estado global vigente (diferente do estado inicial) e não existem mensagens em trânsito.

Uma recepção não especificada ocorre quando uma recepção, que pode ocorrer, não é especificada no *design*. Na ausência de procedimentos adequados de recuperação, recepções não especificadas podem causar, aos processos envolvidos, uma entrada em algum estado desconhecido, através de uma transição não especificada no *design*.

Uma interação não executável está presente quando o projetista inclui transmissões e recepções de mensagens, que não podem ocorrer em condições normais de operação. Uma interação não executável é equivalente ao código morto em um programa.

Um estado ambíguo existe quando um estado de um processo pode coexistir estavelmente com vários estados diferentes de outros processos comunicantes.

Na técnica Matriz Duologue, os erros provocados por estados de impasse e recepções não especificadas estão implicitamente relacionados aos duologues errôneos. Como foi visto na seção anterior, as ambigüidades de um protocolo são também detectáveis através da matriz de validação. Para ilustrar esses fatos, os grafos relativos à Figura 3.15 são analisados.

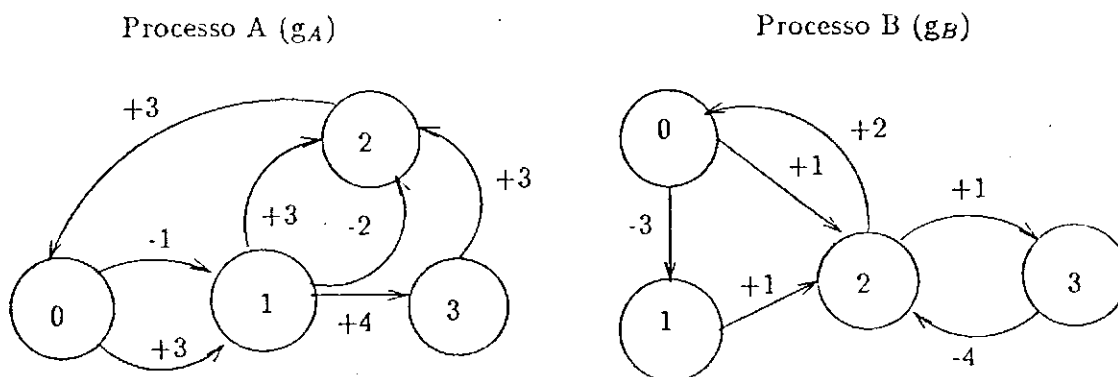


Figura 3.15: Par de grafos analisado.

A partir desse par de grafos, é obtida a matriz de validação e em seguida é verificado se os erros detectados, utilizando-se a técnica Matriz Duologue, são os mesmos que estão apresentados em [WeZa 80].

Em relação ao exemplo da Figura 3.15, tem-se:

- o duologue  $[A_1, B_2] = [\{+3, +3, +3\}; \{-3, +1, +2\}]$  é errôneo, já que o processo A espera a transmissão do evento 3 e o processo B espera a transmissão do evento 1. Portanto, esse tipo de erro corresponde a uma situação de impasse. O mesmo ocorre em relação aos duologues  $[A_1, B_3]$ ,  $[A_3, B_1]$ ,  $[A_3, B_2]$ ,  $[A_3, B_3]$ ,  $[A_3, B_4]$ ,  $[A_4, B_1]$ ,  $[A_5, B_1]$ ,  $[A_5, B_4]$ ,  $[A_6, B_2]$  e  $[A_6, B_3]$ ;
- o duologue  $[A_2, B_2] = [\{+3, -2, +3\}; \{-3, +1, +2\}]$  é errôneo, já que não foi prevista a recepção do evento 2 pelo processo B. Portanto, esse tipo de erro corresponde a uma recepção não especificada. O mesmo ocorre em relação aos duologues  $[A_2, B_3]$ ,  $[A_4, B_2]$ ,  $[A_4, B_3]$ ,  $[A_4, B_4]$ ,  $[A_5, B_2]$  e  $[A_5, B_3]$ .

Portanto, resta somente as interações não executáveis para serem detectadas através da técnica Matriz Duologue. Cabe salientar que mesmo na técnica de Pertubação, esse tipo de erro não é automaticamente detectável.

## Capítulo 4

# Uma Ferramenta para Validação de Protocolos

A ferramenta desenvolvida permite a correção dos duologues errôneos do protocolo, através da aplicação das condições discutidas no capítulo anterior. Essa ferramenta utiliza os conceitos da técnica Matriz Duologue para detectar e corrigir erros de *design*, que possam vir a ocorrer em um protocolo de comunicação.

## 4.1 Visão Geral da Ferramenta

A ferramenta foi implementada totalmente na linguagem Pascal, usando o compilador turbo Pascal versão 5.0. A sua arquitetura geral é apresentada na Figura 4.1.

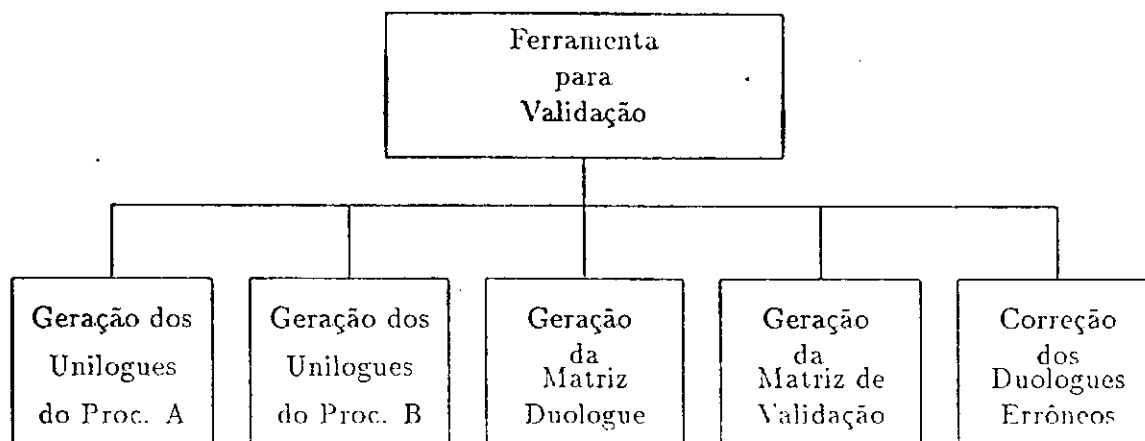


Figura 4.1: Arquitetura geral da ferramenta.

## 4.2 Matriz de Transição de Estados

Uma vez obtido os grafos orientados, que representam os processos comunicantes, o usuário deve fornecer as matrizes de transição de estados. A interface para a entrada dos dados para essas matrizes é bastante amigável e flexível.

A estrutura de dados escolhida, para implementar a matriz de transição de estados, é baseada no método de alocação dinâmica de memória e utiliza listas encadeadas com um registro para cada valor diferente de zero. Isto porque a quantidade de dados a serem armazenados depende da complexidade do protocolo a ser validado.

Uma vez que as matrizes de transição de estados são geralmente esparsas, uma estrutura de dados adequada para essas situações foi utilizada na implementação. A matriz é representada através de listas com ligações múltiplas, com um registro

para cada valor diferente de zero. A grande vantagem dessa implementação é que somente são alocados espaços de memória para as células, que serão referenciadas na matriz de transição de estados.

A estrutura de dados para a matriz de transição de estados é composta de um vetor de apontadores para as linhas e um vetor de apontadores para as colunas. Por exemplo, `LinhaMatriz[1]` contém um apontador para o primeiro registro da linha 1. Da mesma forma, `ColunaMatriz[2]` contém um apontador para o primeiro registro da coluna 2.

Cada registro da matriz de transição de estados (Figura 4.2) tem cinco campos

Linha	Coluna	Eventos	DirElemento
			BaxElemento

Figura 4.2: Registro relativo aos elementos da matriz de transição.

onde:

- o campo `Linha` (do tipo inteiro) corresponde ao número da linha do elemento na matriz de transição de estados;
- o campo `Coluna` (do tipo inteiro) corresponde ao número da coluna do elemento na matriz de transição de estados;
- o campo `Eventos` (do tipo seqüência de caracteres) guarda os eventos e *nonevents* que provocam a transição entre dois nós de um grafo;
- o campo `DirElemento` (do tipo ponteiro) guarda o endereço da próxima linha (em ordem crescente de número de linha), onde ocorre um valor diferente de zero;
- o campo `BaxElemento` (do tipo ponteiro) guarda o endereço da próxima coluna (em ordem crescente de número de coluna), onde ocorre um valor diferente de zero.

A estrutura de dados escolhida para representar a matriz de transição de estados foi implementada da seguinte forma:



```
01 Type
02   ApontadorElem = ^ Elemento;
03   Elemento = Record
04       Linha,
05       Coluna : Integer;
06       Evento : String;
07       Dir_Elem : ApontadorElem;
08       Bax_Elem : ApontadorElem
09   end;
10   LinhaMatriz = Array [1..200] of ApontadorElem;
11   ColunaMatriz = Array [1..200] of ApontadorElem;
```

Figura 4.3: Código pascal da estrutura da matriz de transição de estados.

### 4.3 Geração dos Unilogues

No exemplo relativo à Figura 3.1, os unilogues foram obtidos, pelo projetista, através de uma simples inspeção dos grafos. Para protocolos complexos, torna-se difícil, algumas vezes até impossível, a obtenção dos unilogues manualmente. A teoria dos grafos fornece subsídios para que esses unilogues possam ser obtidos automaticamente.

A ferramenta de validação desenvolvida permite a geração automática dos unilogues segundo dois métodos distintos: o primeiro, proposto em [Zafi 78], utiliza álgebra matricial para a geração dos unilogues; o segundo, proposto neste trabalho, gera os unilogues através de caminhamentos no grafo.

#### 4.3.1 Primeiro Método para a Geração dos Unilogues

Por esse método, o conjunto de todos os unilogues é formado pela união de subconjuntos de unilogues de comprimento fixo. Cada subconjunto é gerado a partir da matriz de transição de estados e cabe ao projetista fornecer o comprimento máximo dos unilogues.

Seja  $[a]$  a matriz de transição de estados de um processo  $A$ . Um elemento  $a_{ik}$  de  $[a]$  representa o conjunto de todos os arcos, que partindo do estado  $i$  chegam ao estado  $k$ . Considerando-se a matriz  $[a]^j$  como sendo  $j$  vezes o produto de  $[a]$  consigo mesma, no sentido teórico dos conjuntos, o elemento  $a_{ik}(j)$  representa o conjunto

dos caminhos de comprimento  $j$ , que saindo do estado  $i$  chegam ao estado  $k$ . Isto é,

$$a_{ik}(j) = \bigcup_{i=1}^n a_{in}(j-1) \times a_{nk}(1) \quad (4.1)$$

Por exemplo,  $a_{00}(j)$  é formado pelo conjunto de todos os caminhos de comprimento  $j$ , que começam e terminam no estado 0. Contudo,  $a_{00}(j)$  pode conter caminhos que atravessam o estado inicial 0. Esses caminhos podem ser eliminados, utilizando-se uma matriz  $[I']$  como fator multiplicativo. A razão para isso é que somente os unilogues, conjunto de caminhos que começam e terminam no estado inicial sem atravessá-lo, devem ser considerados.

A matriz  $[I']$  possui a mesma ordem da matriz de transição de estados  $[a]$ . Todos os elementos da primeira linha da matriz  $[I']$  são nulos, para evitar que arcos de comprimento  $(j-1)$  atravessem o estado inicial 0. Consequentemente, como os arcos de comprimento  $j$  são calculados a partir dos arcos de comprimento  $(j-1)$ , estes também não irão atravessar o estado inicial 0. A Figura abaixo representa uma matriz  $[I']$  de ordem 5.

$$[I'] = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Desta forma, o conjunto de unilogues de comprimento  $j$  pode ser obtido recursivamente através da equação

$$[a']^j = ([a']^{j-1} * [I']).[a] \quad (4.2)$$

onde:

- $[a']^{j-1}$  e  $[a']^j$  são as matrizes que contém os unilogues de comprimento  $(j-1)$  e  $j$  respectivamente;
- $[I']$  é a matriz, de mesma ordem de  $[a]$ , que permite eliminar os unilogues de tamanho  $(j-1)$  que atravessam o estado inicial;

- \* representa uma multiplicação de matrizes e . representa uma concatenação de matrizes;
- $[a']^1 = [a]$ .

Para ilustrar o mecanismo de criação dos unilogues, utiliza-se a matriz de transição de estados  $[a]$  representada na Tabela 4.1.

	0	1	2	3	4
0	$\lambda$	-1	+3	$\lambda$	+4
1	$\lambda$	$\lambda$	-8	+3	$\lambda$
2	-4	$\lambda$	$\lambda$	$\lambda$	$\lambda$
3	$\lambda$	$\lambda$	-6	$\lambda$	-4
4	$\lambda$	$\lambda$	-10	+3	$\lambda$

Tabela 4.1: Matriz de transição de um processo comunicante.

Nessa matriz, o elemento  $a_{ik}$  é o arco que vai do estado  $i$  ao estado  $k$ , sendo que  $\lambda$  representa o conjunto vazio.

A partir da matriz de transição de estados  $[a]$ , pode-se encontrar o conjunto de todos os unilogues de um determinado comprimento, que começam e terminam no estado inicial 0 sem atravessá-lo. Por exemplo, os elementos  $a_{00}(3)$ , relativos à matriz da Tabela 4.1, são obtidos a partir da equação 4.2 considerando-se  $j = 3$ .

$$[a']^3 = ([a']^2 * [I']).[a]$$

sendo que

$$[a] = \begin{pmatrix} 0 & -1 & +3 & 0 & +4 \\ 0 & 0 & -8 & +3 & 0 \\ -4 & 0 & 0 & 0 & 0 \\ 0 & 0 & -6 & 0 & -4 \\ 0 & 0 & -10 & +3 & 0 \end{pmatrix}$$

Utilizando-se novamente a equação 4.2 para  $j = 2$ , tem-se

$$[a']^2 = ([a']^1 * [I']).[a]$$

Portanto

$$[a']^2 = \begin{pmatrix} [+3, -4] & 0 & [-1, -8] [+4, -10] & [-1, +3] [+4, +3] & 0 \\ [-8, -4] & 0 & [+3, -6] & 0 & [+3, -4] \\ 0 & 0 & 0 & 0 & 0 \\ [-6, -4] & 0 & [-4, -10] & [-4, +3] & 0 \\ [-10, -4] & 0 & 0 & [+3, -6] & [+3, -4] \end{pmatrix}$$

$$[a']^2 * [I'] = \begin{pmatrix} 0 & 0 & [-1, -8] [+4, -10] & [-1, +3] [+4, +3] & 0 \\ 0 & 0 & [+3, -6] & 0 & [+3, -4] \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & [-4, -10] & [-4, +3] & 0 \\ 0 & 0 & 0 & [+3, -6] & [+3, -4] \end{pmatrix}$$

Uma vez que

$$a_{00}(3) = \bigcup_{i=1}^n a_{0n}(2) \times a_{n0}(1)$$

$a_{00}(3)$  é o resultado da concatenação da primeira linha de  $[a']^2 * [I']$  com a primeira coluna de  $[a]$ . Logo, o conjunto de todos os unilogues de comprimento 3, que começam e terminam no estado 0, é:

$$a_{00}(3) = [\{-1, -8, -4\}, \{+4, -10, -4\}]$$

O conjunto de todos os unilogues do processo A é

$$S_A = \bigcup_{j=1}^n a_{00}(j) \quad (4.3)$$

onde  $n$  é o comprimento máximo dos unilogues do processo A ( $S_A$ ). Analogamente, o conjunto de todos os unilogues do processo B é

$$S_B = \bigcup_{j=1}^m b_{00}(j) \quad (4.4)$$

onde  $b_{00}(j)$  é derivado da matriz  $[b]$  da mesma forma que  $a_{00}(j)$  foi derivado de  $[a]$ .

O mecanismo para a obtenção do conjunto de unilogues de um processo segue uma ordenação seqüencial e contém os procedimentos indicados na Tabela 4.2.

Procedimento	Significado
Multiplica	Multiplica a matriz $[a']^{j-1}$ pela matriz $[I']$
Concatena	Concatena, semelhantemente à multiplicação, os elementos não nulos da matriz resultante de Multiplica com os elementos não nulos de $[a]$
Seleciona	Concatena os elementos não nulos da primeira linha da matriz resultante de Concatena com os elementos não nulos de $[a]$

Tabela 4.2: Procedimentos utilizados no algoritmo para a geração dos unilogues.

O mecanismo, para a obtenção do conjunto de unilogues, pode ser descrito através do algoritmo apresentado na Figura 4.4.

```

Algoritmo CriaUnilogues
Início
  Leia (MaxLenUnilogue)
  Multiplica ([a]*[I'],MatResult)
  Para K = MaxLenUnilogue, dec K de 1, ate 2, faça
    Se K > 2
      Entao
        . Concatena (MatResult.[a],[a'])
        . Multiplica ([a']*[I'],MatResult)
      Senao Seleciona (MatResult,[a])
  ArmazenaUnilogues(MatResult)
Fim

```

Figura 4.4: Primeiro algoritmo para a geração dos unilogues.

### 4.3.2 Segundo Método para a Geração dos Unilogues

Por esse método, os caminhos que começam e terminam no vértice inicial  $S_0$  são determinados percorrendo-se o grafo. Esse caminhamento é realizado de forma recursiva e à medida que o vértice  $S_0$  é atingido, a partir de um vértice  $S_n$ , esse caminho (unilogue) é armazenado. Então, a partir de um sucessor de  $S_n$ , a busca de um novo caminho é iniciada.

Cada vértice de um grafo conhece todos os seus sucessores e para que este retorne ao seu predecessor é necessário que todos os seus sucessores sejam alcançados.

A partir do vértice inicial  $S_0$ , caminhos são formados de maneira incremental, adicionando-se um dos sucessores de  $S_0$  em cada instante. Desta forma, todos os possíveis caminhos saindo de  $S_0$  são explorados. Recursivamente, esse procedimento é aplicado a um novo sucessor escolhido, até que o vértice inicial  $S_0$  seja novamente alcançado. A Figura 4.5 descreve o algoritmo utilizado para a geração automática dos unilogues.

Algoritmo AchaUnilogues

Início

Se  $V = \text{Vertice\_Inicial}$

Entao Armazena\_Unilogue

Senao

Para todo  $S$  pertencente a  $\text{Sucessor}(V)$  Faca

Se  $\text{Num\_Ocorr}(S, CC) < 2$

Entao

. Coloca  $S$  no Caminho\_Corrente

. Chama AchaUnilogues( $S$ )

. Retira  $S$  do Caminho\_Corrente

Fim entao

Fim.

Figura 4.5: Segundo algoritmo para a geração dos unilogues.

Para que os ciclos presentes num grafo sejam atravessados uma única vez, o número máximo de vezes que um determinado vértice pode ser visitado foi fixado em dois. O procedimento AchaUnilogues realiza uma marcação que controla o número de vezes que um vértice de um grafo é visitado.

### 4.3.3 Representação dos Unilogues

Cada unilogue é representado por uma lista encadeada, onde cada elemento dessa lista tem os campos

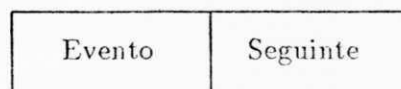


Figura 4.6: Registro referente a um evento.

onde seguinte é um apontador para o próximo evento do unilogue.

O conjunto de unilogues de um processo comunicante é, por sua vez, representado através de um vetor de apontadores, onde cada apontador aponta para o início de

uma lista encadeada. Cada lista encadeada corresponde a um determinado unilogue desse processo.

Uma vez conhecido o número de unilogues dos processos comunicantes, todos os unilogues irão compartilhar o vetor de apontadores. A utilização de uma estrutura geral para representar esses unilogues, permite que qualquer unilogue possa ser referenciado através de um único índice desse vetor. Uma vez conhecida a posição do unilogue, pode-se ter acesso aos seus eventos, pois esse índice é um apontador para o início da lista encadeada. A Figura 4.7 representa a estrutura dos unilogues.

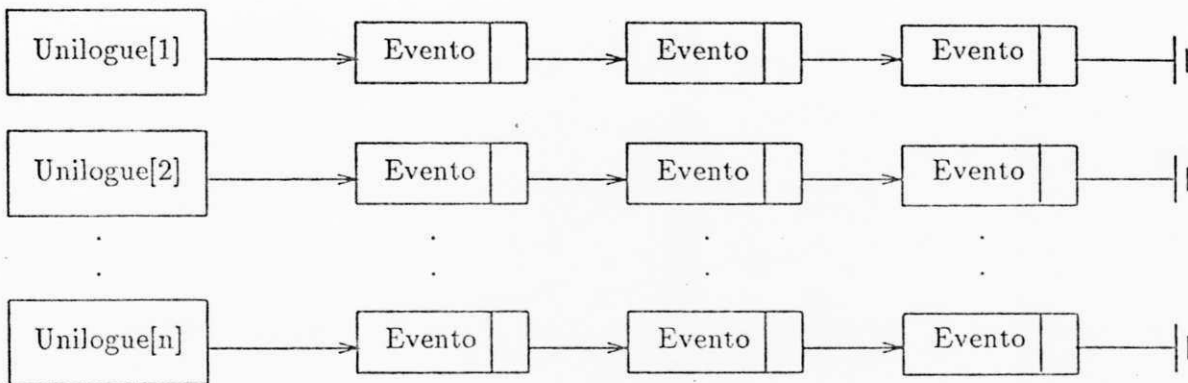


Figura 4.7: Estrutura dos unilogues.

O código Pascal que representa a estrutura dos unilogues está descrito na Figura 4.8.

```

01 Type
02   ApontUnilogue = ^ Elemento;
03   Elemento = Record
04       Evento : Integer;
05       Seguinte : ApontUnilogue
06   end;
07   Unilogue = Array [1..200] of ApontUnilogue;

```

Figura 4.8: Código Pascal da estrutura dos unilogues.

Para que esse vetor de apontadores seja compartilhado entre os processos comunicantes, é necessário conhecer o número de unilogues do processo A ( $N_a$ ) e do processo B ( $N_b$ ). Por exemplo, suponha que um processo A tenha 3 unilogues e que um processo B tenha 4. Logo, Unilogue[1] é um apontador para o primeiro unilogue do processo A. Da mesma forma, Unilogue[5] é um apontador para o segundo



unilogue do processo B.

## 4.4 Geração da Matriz Duologue

A partir dos unilogues referentes aos processos comunicantes A e B, obtidos através da utilização de um dos algoritmos vistos na seção anterior, é montada a matriz duologue D. A matriz D é formada através do produto cartesiano do conjunto de unilogues do processo A com o conjunto de unilogues do processo B.

A idéia de usar uma estrutura geral, para representar o conjunto de unilogues de um protocolo, facilita a geração da matriz D. Como um duologue é formado pela combinação de um unilogue de cada grafo, é necessário, portanto, apenas dois índices do vetor de apontadores para formar um duologue. Cada índice permite o acesso a um unilogue de cada grafo.

Cada duologue da matriz D é representado através de uma cadeia de caracteres, formada pela concatenação de um unilogue de cada grafo. O código Pascal que representa a estrutura de dados dos duologues está descrito na Figura 4.9.

```
01 Type
02   ApontDuologue = ^ Duologue;
03   Duologue = Record
04       Linha,
05       Coluna : Integer;
06       Duol : String;
07       Proximo : ApontDuologue
08   End;
09   ListaDuologue = ApontDuologue;
```

Figura 4.9: Código Pascal da estrutura de dados dos duologues.

O mecanismo para a geração da matriz duologue segue o esquema do procedimento descrito na Figura 4.10.

```
01 Procedure CriaMatrizDuologue;
02
03   Begin
04     For I := 1 To Na Do
05       For J := Na + 1 To Na + Nb Do
06         Begin
07           ConstroiDuologue (I, J, S);
```



```

08     InereNaLista (ListaDuologue, I, J, S)
09     End
10 End;

```

Figura 4.10: Código Pascal do procedimento para a geração da matriz duologue.

O procedimento *CriaMatrizDuologue* utiliza uma subrotina *ConstroiDuologue* que cria os duologues a partir de um unilogue de cada grafo (Figura 4.11).

```

01 Procedure ConstroiDuologue (I, J: Integer; Var S: String);
02 Begin
03     ConcatenaEventosUnilogue ( I, S1);
04     ConcatenaEventosUnilogue ( J, S2);
05     S := '[' + S1 + ',' + S2 + ']'
06 End;

```

Figura 4.11: Código Pascal do procedimento para construir um duologue.

Além da subrotina acima, o procedimento *CriaMatrizDuologue* utiliza as subrotinas indicadas na Tabela 4.3.

Procedimento	Significado
ConcatenaEventosUnilogues	Concatena todos os eventos de um unilogue, armazenando-os numa cadeia de caracteres S. A variável I identifica esse unilogue no vetor de apontadores.
InereNaLista	Inere um determinado duologue em uma lista encadeada.

Tabela 4.3: Procedimentos utilizados na geração da matriz duologue.

## 4.5 Geração da Matriz de Validação

A função de validação, apresentada no capítulo 3, é aplicada a todos os elementos da matriz duologue D. Como resultado, obtém-se a matriz de validação Val[D].

A matriz Val[D] é uma matriz de inteiros, cujos elementos podem assumir os valores +1, 0 ou -1, que representam um duologue bem comportado, não ocor-

rente ou errôneo respectivamente. O mecanismo para a geração da matriz Val[D] é descrito, genericamente, na Figura 4.12.

```

Início
  Para I = 1 .. Na Faca
    Para J = Na + 1 .. Na + Nb Faca
      início
        ClassificaDuologue (I, J, Predicado)
        Caso Predicado do tipo
          (* duologue bem comportado *)
          "BC": VetorVal [I,J] = +1
          (* duologue nao ocorrente *)
          "ND": VetorVal [I,J] = 0
          (* duologue erroneo *)
          "ER": VetorVal [I,J] = -1
      fim
  fim
Fim

```

Figura 4.12: Algoritmo para a geração da matriz de validação.

O algoritmo utilizado, na implementação do procedimento ClassificaDuologue, pode ser dividido em quatro etapas: identificação dos duologues que não podem ser iniciados; identificação dos duologues bem comportados; identificação dos duologues potencialmente errôneos e identificação dos duologues errôneos. Os duologues que contém *nonevents* são considerados um caso especial e tratados separadamente.

A divisão nessas etapas facilita a geração da matriz de validação, na medida em que as avaliações são realizadas em ordem crescente de complexidade.

Um duologue que não pode ser iniciado é não ocorrente. Isso pode ser facilmente detectado, comparando-se a primeira interação de cada duologue. A Figura 4.13 descreve a identificação de tais duologues.

```

01  Begin
02    If (Unilogue[i]^evento > 0) and (Unilogue[j]^evento > 0)
03      Then
04        VetorVal[i,j] := 0  (* duologue nao ocorrente *)
05  End;

```

Figura 4.13: Código Pascal para identificação de um duologue não ocorrente.

A identificação dos duologues bem comportados é dividida em duas sub-etapas. A primeira sub-etapa analisa se os eventos transmitidos por um unilogue são recebidos pelo outro unilogue, enquanto que a segunda considera a possibilidade da ocorrência de colisões bem sucedidas.

Uma colisão é considerada bem sucedida, quando ambos os processos transmitem *simultaneamente* eventos e, após decorrido algum intervalo de tempo, é prevista a recepção desses eventos.

Após a execução de um duologue bem comportado, ambos os processos devem retornar ao estado inicial, sendo que todos os eventos transmitidos pelo processo A devem ter sido recebidos pelo processo B e vice-versa. A Figura 4.14 descreve a identificação de tais duologues.

```

01 Begin
02   ApontUnilProcA := Unilogue[i];
03   ApontUnilProcB := Unilogue[j];
04   While (ApontUnilProcA <> Nil) and (ApontUnilProcB <> Nil) Do
05     begin
06       Aux := ApontUnilProcA;
07       Tmp := ApontUnilProcB;
08       If Abs(Unilogue[i]^evento) = Abs(Unilogue[j]^evento)
09         Then begin
10           ApontUnilProcA := ApontUnilProcA^.seguinte;
11           ApontUnilProcB := ApontUnilProcB^.seguinte
12         end
13       Else
14         If (ApontUnilProcA^.evento < 0) and (ApontUnilProcB^.evento < 0)
15           and (ApontUnilProcA^.evento = Tmp^.evento*(-1))
16           and (ApontUnilProcB^.evento = Aux^.evento*(-1))
17         Then begin
18           ApontUnilProcA := Aux^.seguinte;
19           ApontUnilProcB := Tmp^.seguinte
20         end
21       end;
22       If ApontUnilProcA = Nil and ApontUnilProcB = Nil
23         Then VetorVal[i,j] := +1 (* duologue bem comportado *)
24     End;

```

Figura 4.14: Código Pascal para identificação de um duologue bem comportado.

Um duologue é considerado potencialmente errôneo se é inicialmente mal comportado. Ele será considerado errôneo se for também ocorrente. Portanto, a identificação desses dois tipos de duologues passa pela identificação dos duologues não ocorrentes.

Nas duas últimas etapas do procedimento ClassificaDuologue, um duologue é considerado não ocorrente, se qualquer tentativa de executá-lo leva, invariavelmente, à execução de um outro duologue.

Na identificação dos duologues potencialmente errôneos, três casos são considerados: situação de impasse, recepção não especificada e colisão mal sucedida. Uma

vez que esses casos são fontes potenciais de erros de *design*, eles são analisados separadamente.

Uma situação de impasse ocorre quando ambos os processos esperam *simultaneamente* recepções de eventos. Isso pode ser detectado comparando-se as interações de cada duologue. A Figura 4.15 descreve a identificação de tais duologues.

```

01 Begin
02   If (ApontUnilProcA^.evento > 0) and (ApontUnilProcB^.evento > 0)
03     Then
04       begin
05         ProcuraDuologueAlternativo(ApontUnilProcA, ApontUnilProcB, Achou);
06         If Achou Then VetorVal[i,j] := 0 (* duologue nao ocorrente *)
07         Else VetorVal[i,j] := -1 (* duologue erroneo *)
08       end
09 End;
```

Figura 4.15: Código Pascal para identificação de uma situação de impasse

Uma recepção não especificada ocorre quando um dos processos transmite um evento, cuja recepção não foi prevista pelo outro processo envolvido. A Figura 4.16 descreve a identificação de tais duologues.

```

01 Begin
02   While (ApontUnilProcA <> Nil) and (ApontUnilProcB <> Nil) Do
03     begin
04       Aux := ApontUnilProcA^.seguinte;
05       Tmp := ApontUnilProcB^.seguinte;
06       If ((ApontUnilProcA^.evento < 0) and (ApontUnilProcB^.evento > 0))
07         or ((ApontUnilProcA^.evento > 0) and (ApontUnilProcB^.evento < 0))
08         and (ApontUnilProcA^.evento <> Tmp^.evento*(-1))
09         and (ApontUnilProcB^.evento <> Aux^.evento*(-1))
10       Then begin
11         ProcuraDuologueAlternativo(ApontUnilProcA, ApontUnilProcB, Achou);
12         If Achou Then VetorVal[i,j] := 0 (* duologue nao ocorrente *)
13         Else VetorVal[i,j] := -1 (* duologue erroneo *)
14       end;
15       ApontUnilProcA := ApontUnilProcA^.seguinte;
16       ApontUnilProcB := ApontUnilProcB^.seguinte
17     end
18 End;
```

Figura 4.16: Código Pascal para identificação de uma recepção não especificada.

Uma colisão mal sucedida ocorre quando os processos transmitem *simultaneamente* eventos, cujas recepções não são previstas. Isso pode ser facilmente detectado comparando-se as interações de cada duologue. A Figura 4.17 descreve a identificação de tais duologues.

UFPA/BIBLIOTECA/PRAI

```

01 Begin
02   While (ApontUnilProca <> Nil) and (ApontUnilProcB <> Nil) Do
03     begin
04       Aux := ApontUnilProca^.seguinte;
05       Tmp := ApontUnilProcB^.seguinte;
06       If ((ApontUnilProca^.evento < 0) and (ApontUnilProcB^.evento < 0))
07         and (ApontUnilProca^.evento <> Tmp^.evento*(-1))
08         and (ApontUnilProcB^.evento <> Aux^.evento*(-1))
09       Then begin
10         ProcuraDuologueAlternativo(ApontUnilProca, ApontUnilProcB, Achou);
11         If Achou Then VetorVal[i,j] := 0 (* duologue nao ocorrente *)
12         Else VetorVal[i,j] := -1 (* duologue erroneo *)
13       end;
14       ApontUnilProca := ApontUnilProca^.seguinte;
15       ApontUnilProcB := ApontUnilProcB^.seguinte
16     end
17 End;
```

Figura 4.17: Código Pascal para identificação de uma colisão mal sucedida.

A presença de *nonevents* num duologue altera a seqüência normal de análise das interações envolvidas. Uma vez detectado um *nonevent* (Figura 4.18), uma análise particular para esse tipo de caso é desencadeada.

```

01 Begin
02   ApontUnilProca := Unilogue[i];
03   ApontUnilProcB := Unilogue[j];
04   While (ApontUnilProca <> Nil) and (ApontUnilProcB <> Nil) Do
05     begin
06       If (ApontUnilProca^.evento = 0) or (ApontUnilProcB^.evento = 0)
07       Then AnalisaDuologue(Unilogue[i], Unilogue[j]);
08       ApontUnilProca := ApontUnilProca^.seguinte;
09       ApontUnilProcB := ApontUnilProcB^.seguinte
10     end
11 End;
```

Figura 4.18: Código Pascal para identificar duologues que contém *nonevents*.

Em função da posição dos *nonevents* no duologue e levando-se em consideração, implicitamente, o tempo para o disparo de uma transição com *nonevent*, o procedimento *AnalisaDuologue* classifica se o duologue é bem comportado, não ocorrente ou errôneo. A Figura 4.19 descreve o procedimento de análise dos duologues que contém *nonevents*.

```

01 Begin
02   Aux := Unilogue[i];
03   Temp := Unilogue[j]; Loop := true;
04   while (Aux<>Nil) and (Temp<>Nil) and Loop Do
05     begin
06       Aux1 := Aux^.seguinte;
07       Temp1 := Temp^.seguinte;
```

```

08     if ((Unilogue[i]^evento = 0) and (Unilogue[j]^evento)) or
09         ((Unilogue[j]^evento = 0) and (Unilogue[i]^evento > 0))
10     then begin VetorVal[i,j] := 0;
11                Loop := false
12     end
13     else if (Abs(Aux^.evento)=Abs(Temp^.evento)) and
14             (aux^.evento<>Temp^.evento)
15     then begin Aux := Aux^.seguinte;
16                Temp := Temp^.seguinte
17     end
18     else if ((Temp^.evento=0) and (Aux^.evento<0)) or
19             ((aux^.evento=0) and (Temp^.evento<0))
20     then Recepcao_Nao_Especificada
21     else if ((Aux^.evento=0) and (Temp^.evento>0) and
22             (Aux1^.evento)) or ((Temp^.evento=0) and
23             (Aux^.evento) and (Temp1^.evento))
24     then Situacao_de_Impasse
25     else if ((Aux^.evento<0) and (Temp^.evento<0)) or
26             ((Aux^.evento<0) and (Temp^.evento=0) and
27             (Temp1^.evento)) or ((Temp^.evento<0) and
28             (Aux1^.evento=0) and (Aux1^.evento<0))
29     then Caso_de_Colisao
30     else if (Aux=Nil) and (Temp^.evento=0)
31     then Temp := Temp^.seguinte
32     else if (Temp=Nil) and (Aux^.evento=0)
33     then Aux := Aux^.seguinte
34     end;
35     if (Aux=Nil) and (Temp=Nil) then VetorVal[i,j] := +1
36 End;

```

Figura 4.19: Código Pascal para a análise dos duologues que contém *nonevents*

## 4.6 Correção dos Duologues Errôneos

Quando um erro de *design* é detectado na matriz Val[D], a ferramenta identifica a posição do duologue errôneo e tenta automaticamente corrigi-lo, de acordo com as condições apresentadas no capítulo 3. Caso essa correção não seja possível, os duologues errôneos são apresentados ao projetista.

O mecanismo de correção de duologues é implementado através de um procedimento, que determina o tipo de correção que deve ser aplicado a cada duologue errôneo. Esse procedimento está descrito na Figura 4.20.

```

01 Procedure SinteseDuologues;
02   Var I, J : integer;
03   Begin
04     For I := 1 to Na do
05       For J := Na + 1 to Na + Nb do
06         If Vetor_Val[I,J] = -1
07         then begin

```



```

08         CorrigiDuologue (I, J, Condicao);
09         MostraEventosTela (A, Unilogue[I]);
10         MostraEventosTela (B, Unilogue[J])
11     end
12 End;

```

Figura 4.20: Código Pascal para a correção de um duologue errôneo.

O procedimento `CorrigiDuologue` recebe um unilogue de cada processo comunicante e faz a sua correção aplicando uma das seguintes condições: pós-transmissão, pré-recepção ou completude. Esse procedimento retorna o tipo da condição aplicada ao duologue.

A condição de pós-transmissão é empregada para corrigir os casos de recepções não especificadas. O procedimento que implementa essa condição é apresentado na Figura 4.21.

```

01 Procedure Pos_Transmissao;
02   Begin
03     Aux := Unilogue[i];
04     Tmp := Unilogue[j];
05     While Aux <> Nil and Tmp <> Nil Do
06       begin
07         Aux1 := Aux^.seguinte;
08         Tmp1 := Tmp^.seguinte;
09         If (Abs(Aux^.evento) = Abs(Tmp^.evento))
10           and (Aux^.evento <> Tmp^.evento)
11         Then begin Aux := Aux^.seguinte;
12                   Tmp := Tmp^.seguinte
13               end
14         Else
15           If ((Abs(Aux^.evento) = Abs(Aux1^.evento)) and
16              (Aux^.evento <> Aux1^.evento)) and
17              ((Abs(Tmp^.evento) = Abs(Tmp1^.evento)) and
18              (Tmp^.evento <> Tmp1^.evento))
19           Then If (Aux^.evento < 0) and (Tmp1^.evento < 0)
20                 Then begin Temp := Aux1^.evento;
21                           Aux1^.evento := Tmp^.evento;
22                           Tmp^.evento := Temp
23                       end
24                 Else begin Temp := Tmp1^.evento;
25                           Tmp1^.evento := Aux^.evento;
26                           Aux^.evento := Temp
27                       end
28           end
29       end
30     end
31   end
32 End;

```

Figura 4.21: Código Pascal do procedimento de pós-transmissão.

A condição de pré-recepção é empregada para corrigir as situações de impasse. O procedimento que implementa essa condição é apresentado na Figura 4.22.

```

01 Procedure Pre_Recepcao;
02   Begin
03     Aux := Unilogue[i]; Tmp := Unilogue[j];
04     While Aux <> Nil and Tmp <> Nil Do
05       begin
06         Aux1 := Aux^.seguinte;
07         Tmp1 := Tmp^.seguinte;
08         If (Abs(Aux^.evento) = Abs(Tmp^.evento))
09           and (Aux^.evento <> Tmp^.evento)
10         Then begin
11           Aux := Aux^.seguinte;
12           Tmp := Tmp^.seguinte
13         end
14         Else If (Aux^.evento > 0) and (Tmp^.evento > 0)
15         Then begin
16           Event1 := Aux^.evento * (-1);
17           Event2 := Tmp^.evento * (-1);
18           New(NovoElem1);
19           Tmp1^.seguinte := NovoElem1;
20           NovoElem1^.evento := Event1;
21           New(NovoElem2);
22           NovoElem2^.seguinte := Aux^.seguinte;
23           Aux^.seguinte := NovoElem2;
24           NovoElem2^.evento := Event2;
25         end
26       end
27   End;

```

Figura 4.22: Código Pascal do procedimento de pré-recepção.

A correção de completude é empregada para corrigir os duologues errôneos que contém *nonevents*. O procedimento que implementa essa condição é apresentado na Figura 4.23.

```

01 Procedure Completude(ApontUnilogue);
02   Begin
03     Aux := ApontUnilogue; Temp := ApontUnilogue;
04     Loop := true;
05     While Loop Do
06       begin
07         Aux1 := Aux^.seguinte;
08         Tmp1 := Aux^.seguinte;
09         If ApontUnilogue^.evento = 0
10         Then begin
11           LiberaEvento(ApontUnilogue);
12           Loop := false
13         end
14         Else
15           If Tmp1^.evento = 0
16           Then begin
17             LiberaEvento(Tmp1);
18             Loop := false
19           end
20         Else begin
21           While Aux1^.seguinte <> Nil Do
22             begin

```



```
23         ApontUnilogue := Aux1;
24         Aux1 := Aux1^.seguinte
25     end;
26     If Aux1^.evento = 0
27     Then begin
28         LiberaEvento(Aux1);
29         Loop := false
30     end
31     end;
32     Tmp1 := Tmp1^.seguinte
33 end;
34 CriaNovoUnilogue
35 End;
```

Figura 4.23: Código pascal do procedimento de completude.

Correções, que são muito inerentes ao tipo de protocolo que está sendo concebido, não são realizadas automaticamente pela ferramenta. Por exemplo, as colisões mal sucedidas. Isto porque, para tal tipo de erro, é necessário decidir em favor de qual processo a colisão será corrigida, definir novas interações e novos estados.

As correções efetuadas baseiam-se em princípios genéricos e para casos particulares podem não ser ótimas. Cabe ao projetista aceitá-la ou propor uma outra correção.

## Capítulo 5

# Protocolo X.25 do CCITT

A fim de prevenir o desenvolvimento de interfaces incompatíveis entre redes públicas de diferentes países, o CCITT apresentou um padrão internacional de protocolos, que mais tarde foi incorporado ao modelo de referência **Open Systems Interconnection (OSI)** da **International Organization for Standardization (ISO)** [ISOb 83]. Esse padrão é conhecido como X.25. Atualmente, a recomendação X.25 compreende os padrões relativos às três camadas inferiores do modelo OSI.

## 5.1 Modelo de Referência OSI da ISO

Devido à grande complexidade dos problemas envolvidos no projeto de protocolos, os especialistas da ISO e do CCITT organizaram as diversas funções dos componentes de um sistema em camadas. Isso resultou no modelo RM-OSI que estabelece 7 camadas e os respectivos protocolos para cada sistema aberto, conforme a Figura 5.1.

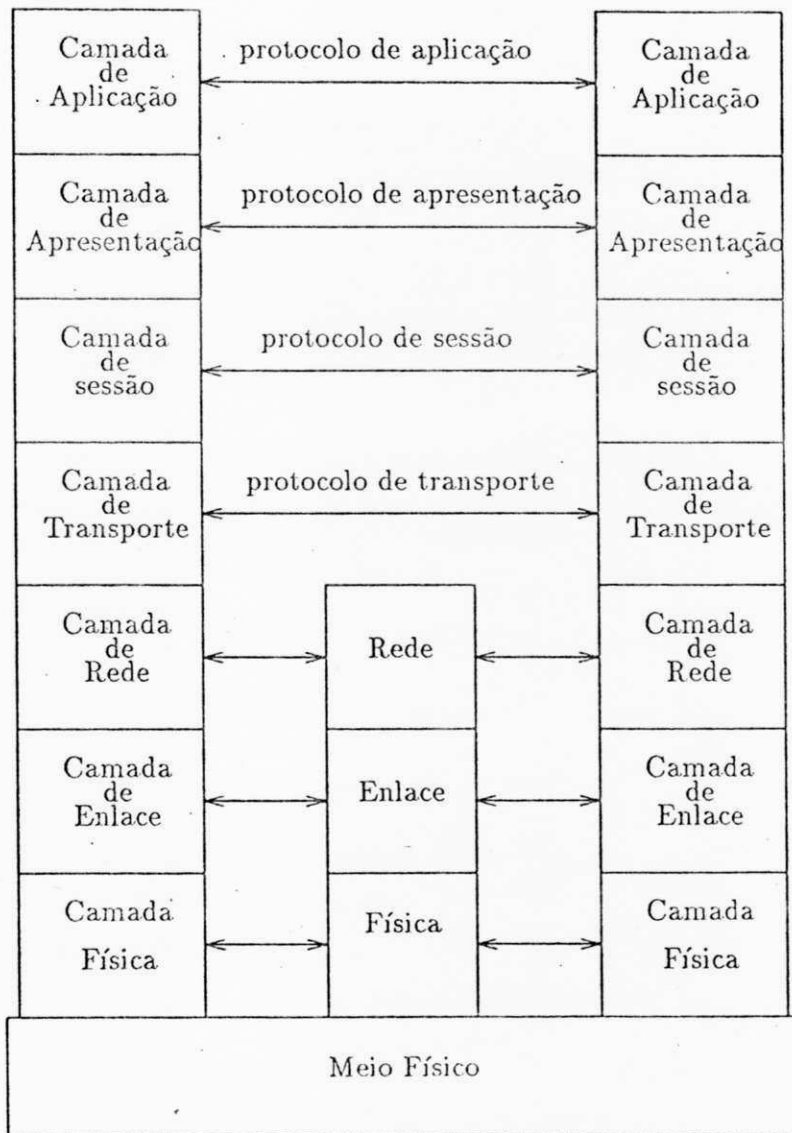


Figura 5.1: Estrutura geral do modelo RM-OSI.

O propósito de cada camada é oferecer serviços à camada imediatamente su-

perior, utilizando-se dos serviços oferecidos pela camada imediatamente inferior. Resumidamente, as camadas no modelo RM-OSI podem ser definidas da seguinte forma:

- **física** - tem como objetivo prover as características mecânicas, elétricas, funcionais e os procedimentos para ativar, manter e desativar conexões físicas para a transmissão de bits entre entidades da camada de enlace. Essa camada permite o envio de uma cadeia de bits, sem que haja uma preocupação com o significado desses bits;
- **enlace** - tem como objetivo detectar e, possivelmente, corrigir erros que possam ocorrer na camada física, convertendo um canal de transmissão não confiável em um canal confiável, para o uso das entidades da camada de rede. Isto é conseguido através da partição da seqüência de bits em quadros, inserindo em cada um alguma forma de redundância para detecção de erros;
- **rede** - tem como objetivo estabelecer e liberar conexões de rede entre um par de entidades da camada de transporte. Além disso, a camada de rede efetiva funções de roteamento e controle de congestionamento, associadas ao estabelecimento e operação da conexão;
- **transporte** - tem como objetivo oferecer uma comunicação fim a fim confiável, através da transferência transparente de dados entre entidades da camada de sessão. Outra função importante dessa camada é a multiplexação de conexões;
- **sessão** - tem como objetivo organizar e sincronizar o diálogo e gerenciar a troca de dados entre entidades da camada de apresentação;
- **apresentação** - tem como objetivo realizar transformações nas estruturas dos dados, antes de seu envio à camada de sessão. Essa camada resolve problemas de sintaxe entre sistemas abertos;
- **aplicação** - tem como objetivo servir aos usuários que querem se comunicar através do modelo RM-OSI.

Os protocolos pertencentes às camadas física, enlace e rede constituem os protocolos de baixo nível e os demais constituem os protocolos de alto nível.

## 5.2 Descrição do Protocolo X.25

O protocolo X.25 define a interface entre o hospedeiro, chamado Equipamento Terminal de Dados (ETD), e a rede, chamada de Equipamento de Circuitos de Dados (ECD). Além disso, o X.25 define o formato e o significado das informações trocadas através da interface ETD-ECD. A comunicação, usando a arquitetura em camadas do protocolo X.25, é apresentada na Figura 5.2.

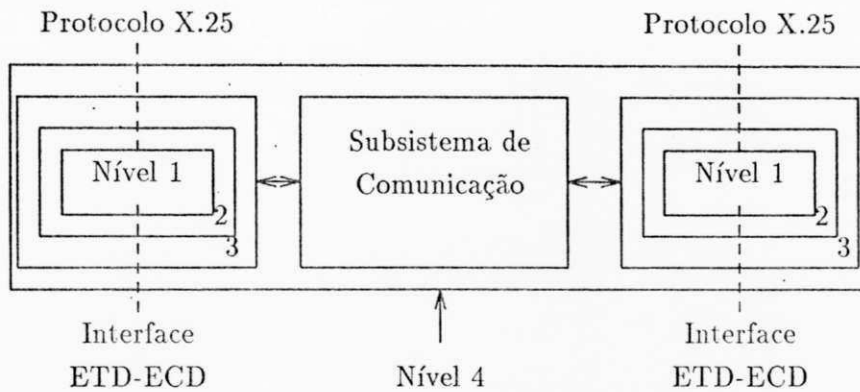


Figura 5.2: Arquitetura em camadas do protocolo X.25.

O primeiro nível da recomendação X.25 é constituído pelas recomendações X.21 e X.21 bis e tem como objetivo fundamental a especificação das características mecânicas, elétricas, funcionais e procedurais para operar a transmissão de circuitos de dados [Hals 88].

O segundo nível do X.25 tem como função assegurar uma comunicação confiável entre ETD-ECD. Os protocolos usados nesse nível são LAP (*Link Access Procedure*) e LAPB (*Link Access Procedure Balanced*).

Os três níveis da recomendação X.25 são independentes um do outro e os dois níveis inferiores podem ser substituídos por qualquer protocolo que desempenhe a mesma função.

Para distinguir o protocolo relativo à camada 3 dos protocolos das camadas 1 e 2, este é denominado de X.25 PCP<sup>1</sup>.

A camada de rede gerencia as conexões entre dois ETDs. Duas formas de co-

<sup>1</sup>Isto é, protocolo da camada de pacotes.

nexões são oferecidas: chamadas virtuais e circuitos virtuais permanentes. O protocolo X.25 PCP define a sinalização necessária entre ETD-ECD, para que seja possível estabelecer e liberar chamadas virtuais, e define também o controle das transferências de dados.

O protocolo X.25 PCP é definido em termos de [CCIT 88]:

- procedimentos para estabelecimento de chamadas virtuais e circuitos virtuais permanentes;
- formatos dos pacotes associados a esses procedimentos;
- procedimentos e formatos dos pacotes para facilidades opcionais do usuário e outras facilidades especificadas pelo ETD.

O protocolo X.25 PCP é usado para fornecer um serviço orientado-a-conexão para um ETD conectado a uma rede pública de comutação de pacotes. Esse protocolo define diferentes formatos de pacotes. Esses pacotes são classificados em pacotes de dados e de controle. A maioria desses pacotes são de controle.

Um pacote de dados tem o campo de informação de comprimento variável. O formato geral dos pacotes X.25 PCP é ilustrado na Figura 5.3.

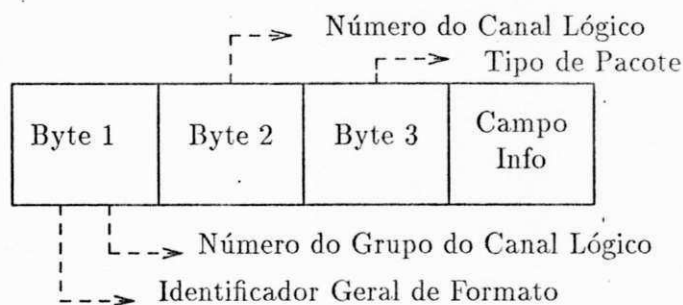


Figura 5.3: Formato geral dos pacotes X.25 PCP.

Em relação à Figura 5.3, o campo de informação pode conter informação de controle ou de dados de comprimento variável. Para distinguir um pacote de dados de um pacote de controle existe um bit, situado no octeto 3, que assumirá o valor 0 nos pacotes de dados e o valor 1 nos pacotes de controle. Esse bit é conhecido como **qualificador**.

Embora os três níveis do protocolo X.25 tenham, normalmente, significado local, há uma facilidade que permite que o nível 3 tenha significado fim a fim. Isto é possível graças a um bit especial presente no cabeçalho dos pacotes. Esse bit deverá assumir o valor 1 quando o ETD fonte requerer uma confirmação fim a fim.

### 5.2.1 Fase de Estabelecimento de Conexão

Uma chamada virtual é uma associação temporária entre dois ETDs. O estabelecimento de uma conexão é iniciado quando um ETD fonte envia um pacote CallRequest ao seu ECD. Ao receber esse pacote, a sub-rede entregará ao ECD de destino o pacote IncomingCall que, por sua vez, irá entregá-lo ao seu ETD. Se o ETD de destino aceitar o pedido de conexão, este deverá enviar o pacote CallAccepted ao seu ECD. Ao receber esse pacote, a sub-rede entregará ao ECD fonte o pacote CallConnected. Quando o ETD fonte receber esse pacote, a conexão estará estabelecida. Se por uma razão qualquer uma chamada virtual não pode ser estabelecida, a sub-rede enviará um pacote de desconexão ClearIndication ao ETD fonte [Tane 89].

Quando uma entidade local da camada rede recebe de seu usuário uma primitiva N\_Connect.request, essa primitiva é mapeada no pacote CallRequest, para que esta possa ser transmitida à entidade remota. Por outro lado, quando a entidade remota recebe o pacote IncomingCall, o seu usuário é notificado através da primitiva N\_Connect.indication. Da mesma forma, quando a entidade remota recebe uma primitiva N\_Connect.response de seu usuário, essa primitiva é mapeada no pacote CallAccepted. Quando a entidade local recebe o pacote CallConnected, o seu usuário é notificado através da primitiva N\_Connect.confirm. A Tabela 5.1 descreve, para a fase de estabelecimento de conexão da camada de rede, o relacionamento entre as primitivas e os pacotes do protocolo X.25 PCP.

Cada ETD conectado a uma rede pública de dados tem assinalado um número único de identificador de canal lógico. Esse número pode ser identificado no cabeçalho de cada pacote. A partir desse número a sub-rede encaminhará o pacote ao ETD de destino.

O ETD fonte pode escolher um canal lógico ocioso qualquer para identificar a

Primitivas	Pacotes
N_Connect.request	CallRequest
N_Connect.indication	IncomingCall
N_Connect.response	CallAccepted
N_Connect.confirm	CallConnected

Tabela 5.1: Primitivas de estab. de conexão e pacotes do protocolo X.25 PCP.

conexão. Se este número está sendo usado no ETD de destino, o ECD de destino deverá substituí-lo por outro, antes de entregar o pacote ao ETD de destino. Pode acontecer que ambos, ETD fonte e ECD de destino, escolham o mesmo número de canal lógico *simultaneamente*, terminando em uma colisão de chamada. Neste caso, o protocolo X.25 PCP especifica que a chamada de saída do ETD fonte é encaminhada e a chamada de entrada do ECD de destino é cancelada. Para diminuir as chances de uma colisão de chamada, o ETD fonte escolhe as chamadas de saída mais utilizáveis e o ECD de destino escolhe as chamadas de entrada menos utilizáveis.

A recomendação X.25 fornece um diagrama de estados que descreve o comportamento de um ETD-ECD para a fase de estabelecimento de uma chamada virtual. A Figura 5.4 apresenta o diagrama para a fase de estabelecimento de conexão (Call Setup).

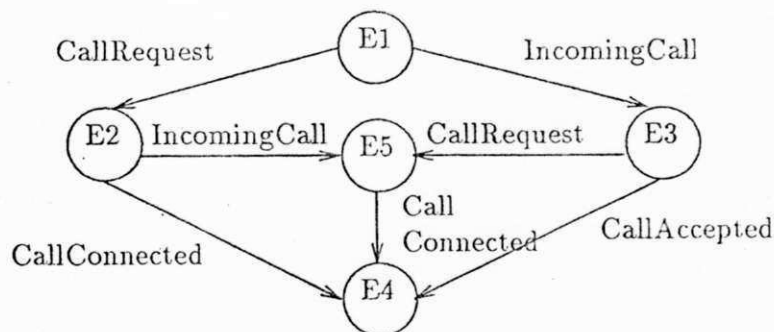


Figura 5.4: Diagrama para a fase de estabelecimento de conexão.

Em relação à Figura 5.4, inicialmente a interface está no estado E1 (estado inicial). A transmissão de um pacote CallRequest ou a recepção de um pacote IncomingCall provoca uma transição do estado E1 para o estado E2 (ETD esperando) ou E3 (ECD esperando) respectivamente. A partir do estado E2 pode-se alcançar



diretamente o estado E4 (fase de transferência de dados) pela recepção do pacote CallConnected. O estado E4 somente é atingido pelo ETD, quando este recebe do ECD o pacote CallConnected. Por outro lado, a transmissão de um pacote CallAccepted provoca uma transição do estado E3 para o estado E4. A partir dos estados E2 ou E3 pode ocorrer uma colisão de chamada. Isto acontece quando os dois ETDs solicitam, *simultaneamente*, o estabelecimento de uma conexão. Neste caso, o ETD que receber em primeiro lugar o pacote CallConnected da sub-rede ingressará na fase de transferência de dados.

### 5.2.2 Fase de Transferência de Dados

Após o estabelecimento da conexão, qualquer um dos ETDs poderá iniciar a transferência de dados através do envio do pacote Data ao seu ECD. Ao receber esse pacote, o ECD de destino irá entregá-lo ao seu ETD. Durante a fase de transferência de dados, é possível a transmissão *full-duplex* de informação.

Esses pacotes de dados são numerados em seqüência pelo ETD fonte. O ETD de destino poderá controlar o fluxo de pacotes de dados através do mecanismo de janela deslizante. Se o ETD ou o ECD receber um pacote de dados com numeração fora da janela, isto será considerado um erro local e a chamada virtual será *rêsetada*, ou seja, a numeração dos pacotes de dados voltará ao início.

Quando uma entidade local recebe uma primitiva N\_Data.request de seu usuário, essa primitiva é mapeada no pacote Data para que este possa ser transmitido à entidade remota. Por outro lado, quando a entidade remota recebe o pacote Data, o seu usuário é notificado através da primitiva N\_Data.indication. A Tabela 5.2 descreve, para a fase de transferência de dados da camada de rede, o relacionamento entre as primitivas de serviço e os pacotes do protocolo X.25 PCP.

Primitivas	Pacotes
N_Data.request	ETD_Data
N_Data.indication	ECD_Data

Tabela 5.2: Primitivas de transf. de dados e pacotes do protocolo X.25 PCP.

O tamanho máximo de cada pacote de dados em uma rede de comutação de pacotes é limitado a 128 octetos de dados, para assegurar um tempo de resposta aceitável. Entretanto, se um ETD deseja transmitir uma mensagem de comprimento superior à 128 octetos, esta deve ser dividida em um número apropriado de pacotes de dados. Em seguida, cada pacote será enviado, separadamente, pela rede de comutação de pacotes.

O algoritmo de controle de fluxo é baseado no mecanismo de janela deslizante. Para implementar esse mecanismo, todos os pacotes de dados contêm um número de seqüência enviada  $P(S)$  e um número de seqüência recebida  $P(R)$ . Como esses números são incrementados em módulo 8, seus ciclos vão de 0 a 7. O  $P(R)$  contido em cada pacote de dados informa o fluxo de pacotes de dados na direção oposta. Alternativamente, se não existem pacotes de dados esperando a transmissão na direção oposta, o  $P(R)$  pode ser enviado pelo receptor num pacote ReceiveReady (RR). A estação receptora pode controlar o ritmo de entrega de pacotes de dados, mediante a geração de confirmação. As confirmações podem ser enviadas nos pacotes RR, ou sobre os próprios pacotes de dados através do mecanismo de janela deslizante.

Para cada direção, o primeiro pacote de dados de um canal lógico é determinado por  $P(S) = 0$ . Cada pacote de dados, na mesma direção, é incrementado em uma unidade no  $P(S)$ . O número de pacotes relacionado com a mesma chamada, que pode ser enviado sem reconhecimento em cada direção, é limitado pelo tamanho da janela.

Para controlar o fluxo de pacotes de dados é preciso determinar o número máximo de buffers necessário à manipulação de cada pacote. Na prática, o número de buffers requerido, para suprir todas as chamadas que podem ser ativadas, é maior que o número total de buffers fornecido. Por esta razão, uma facilidade é fornecida no protocolo para suspender, temporariamente, o fluxo de pacotes de dados associado a uma chamada específica. Essa facilidade é fornecida quando da recepção de um pacote ReceiveNotReady (RNR) num canal lógico. Cada pacote RNR contém um  $P(R)$ , que define um novo limite inferior da janela para esse canal. Na recepção de um pacote RNR, o emissor deve suspender a transmissão de pacotes até que o receptor esteja pronto para receber novos pacotes de dados.

### 5.2.3 Fase de Desconexão

Durante as fases de estabelecimento de conexão e transferência de dados, qualquer um dos ETDs pode iniciar a fase de desconexão (liberação da chamada virtual). O início dessa fase ocorre quando um dos ETDs envia o pacote ClearRequest ao seu ECD. Ao receber esse pacote, a sub-rede entregará ao ECD de destino o pacote ClearIndication que, por sua vez, irá entregá-lo ao seu ETD. O ETD de destino deverá responder ao seu ECD com o pacote ClearConfirmation. Ao receber esse pacote, o ECD fonte irá entregá-lo ao seu ETD, completando a desconexão.

Quando uma entidade local recebe uma primitiva N\_Disconnect.request de seu usuário, essa primitiva é mapeada no pacote ClearRequest para que este possa ser transmitido à entidade remota. Por outro lado, quando a entidade remota recebe o pacote ClearIndication, o seu usuário é notificado através da primitiva N\_Disconnect.indication. Como resultado do recebimento do pacote ClearIndication, a entidade remota deverá enviar, como reconhecimento, o pacote ClearConfirmation para a entidade local. A Tabela 5.3 descreve, para a fase de desconexão da camada rede, o relacionamento entre as primitivas de serviço e os pacotes do protocolo X.25 PCP.

Primitivas	Pacotes
N_Disconnect.request	ClearRequest
N_Disconnect.indication	ClearIndication

Tabela 5.3: Primitivas de desconexão e pacotes do protocolo X.25 PCP.

A ordenação temporal das primitivas, que representam as três fases de uma chamada virtual do protocolo X.25 PCP, é apresentada na Figura 5.5.

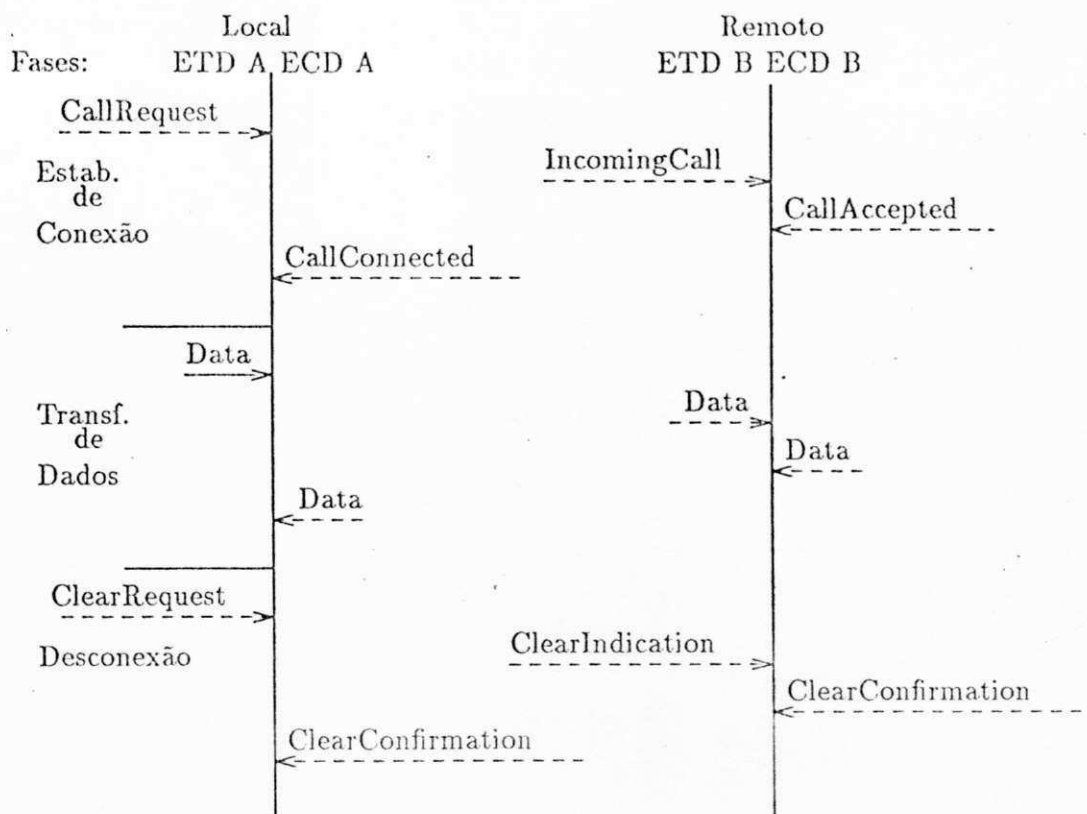


Figura 5.5: Fases de uma chamada virtual do protocolo X.25 PCP.

### 5.3 Mecanismos de Recuperação de Erros

Os mecanismos de recuperação de erros associados ao protocolo X.25 PCP são realizados pelos procedimentos Reset e Restart. O procedimento Reset é usado para reinicializar uma chamada virtual na fase de transferência de dados. O procedimento Restart é utilizado para liberar, simultaneamente, todos os circuitos virtuais em utilização.

Ao receber um pacote de dados fora do limite da janela, o ETD enviará um pacote ResetRequest. Isto faz com que o fluxo de dados seja *reinicializado*. Ao receber esse pacote, a sub-rede entregará ao ECD de destino o pacote ResetIndication que, por sua vez, irá entregá-lo ao seu ETD. Se surgir um problema no diálogo na interface ETD-ECD, que seja necessário liberar todas as chamadas virtuais estabelecidas, a interface poderá fazer uso do procedimento Restart. Isto é realizado pela sub-rede que envia um pacote RestartIndication para a interface ETD-ECD.

## Capítulo 6

### Validação do Protocolo X.25

### Simplificado

Para demonstrar a utilidade da ferramenta descrita no capítulo 4, uma versão simplificada do protocolo X.25 do CCITT é especificada, analisada e validada.

## 6.1 Especificação Formal de uma Versão Simplificada do Protocolo X.25 do CCITT

A recomendação X.25 PCP, descrita no capítulo anterior, fornece os diagramas de estados que descrevem o comportamento de uma interface ETD/ECD durante as fases de uma chamada virtual. A partir desses diagramas são definidos os eventos que serão utilizados para um primeiro projeto do grafo referente a uma entidade local (o grafo referente à entidade remota é similar). Esses grafos representam os processos comunicantes desse protocolo. A relação existente entre os eventos do grafo e os pacotes trocados na interface ETD/ECD é apresentada na Tabela 6.1.

Evento	Pacote	evento	Pacote
-1	CallRequest	+1	IncomingCall
-2	CallAccepted	+2	CallConnected
-3	ETD_Data	+3	ECD_Data
-4	ClearRequest	+4	ClearIndication
-5	ETD_ClearConfirmation	+5	ECD_ClearConfirmation

Tabela 6.1: Eventos do grafo e pacotes correspondentes.

A Figura 6.1 apresenta o grafo inicial, referente ao primeiro projeto da entidade local, que será submetido à ferramenta de validação. Esse grafo inicial descreve o comportamento da entidade local durante as três fases de uma chamada virtual.

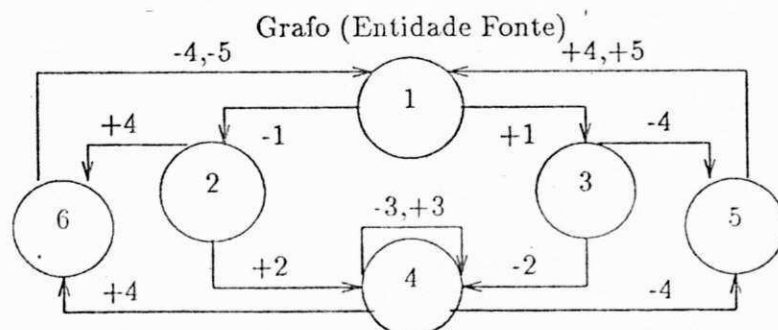


Figura 6.1: Grafo inicial do primeiro projeto.

No capítulo 2 foram introduzidos os conceitos de grafos e como representá-los através de matrizes de transições de estados. Na ferramenta desenvolvida, a interface para a entrada dos dados da matriz de transição de estados é bastante amigável e flexível. A tela da matriz de transição de estados, referente ao grafo inicial, é apresentada na Figura 6.2.

VALPROT - Ferramenta para Validação de Protocolos						Lin :
						Col :
Avança-Pag.	Retorna-Pag.	Fornece-Eventos	Help	Esc-Sair		
Processo A						
	<input type="text" value="1"/>	<input type="text" value="2"/>	<input type="text" value="3"/>	<input type="text" value="4"/>	<input type="text" value="5"/>	<input type="text" value="6"/>
<input type="text" value="1"/>		-1	+1			
<input type="text" value="2"/>				+2	+2	+4
<input type="text" value="3"/>				-2	-4	
<input type="text" value="4"/>				-3,+3	-4	+4
<input type="text" value="5"/>	+4,+5					
<input type="text" value="6"/>	-4,-5					
Esc-Menu Principal		BackSpace-Incluir evento		Enter-Confirmar Evento		

Figura 6.2: Tela da matriz de transição de estados.

## 6.2 Validação da Versão Simplificada do Protocolo X.25

A ferramenta de validação, descrita no capítulo 4, oferece um ambiente que possibilita ao projetista alterar as especificações de um protocolo na sua fase inicial. Essa ferramenta fornece uma resposta imediata em relação ao protocolo que está sendo concebido.

Para validar um protocolo, utilizando-se a técnica Matriz Duologue, o projetista deverá fornecer as matrizes de transição de estados referentes aos processos envolvidos.

No processo de validação dos grafos iniciais foram obtidos os conjuntos de todos os unilogues do protocolo. O conjunto de unilogues, relativo ao processo A, é apresentado na Tabela 6.2 (o conjunto de unilogues relativo ao processo B é similar).

$A_1 = \{+1,-4,+5\}$	$A_{15} = \{+1,-2,+3,-4,+5\}$
$A_2 = \{+1,-4,+4\}$	$A_{16} = \{+1,-2,+3,-4,+4\}$
$A_3 = \{-1,+4,-5\}$	$A_{17} = \{+1,-2,-3,+4,-5\}$
$A_4 = \{-1,+4,-4\}$	$A_{18} = \{+1,-2,-3,+4,-4\}$
$A_5 = \{+1,-2,+4,-5\}$	$A_{19} = \{+1,-2,-3,-4,+5\}$
$A_6 = \{+1,-2,+4,-4\}$	$A_{20} = \{+1,-2,-3,-4,+4\}$
$A_7 = \{+1,-2,-4,+5\}$	$A_{21} = \{-1,+2,+3,+4,-5\}$
$A_8 = \{+1,-2,-4,+4\}$	$A_{22} = \{-1,+2,+3,+4,-4\}$
$A_9 = \{-1,+2,+4,-5\}$	$A_{23} = \{-1,+2,+3,-4,+5\}$
$A_{10} = \{-1,+2,+4,-4\}$	$A_{24} = \{-1,+2,+3,-4,+4\}$
$A_{11} = \{-1,+2,-4,+5\}$	$A_{25} = \{-1,+2,-3,+4,-5\}$
$A_{12} = \{-1,+2,-4,+4\}$	$A_{26} = \{-1,+2,-3,+4,-4\}$
$A_{13} = \{+1,-2,+3,+4,-5\}$	$A_{27} = \{-1,+2,-3,-4,+5\}$
$A_{14} = \{+1,-2,+3,+4,-4\}$	$A_{28} = \{-1,+2,-3,-4,+4\}$

Tabela 6.2: Conjunto de unilogues do processo A referente ao primeiro projeto.

A partir dos unilogues referentes aos processos A e B, é montada a matriz duologue D. A Figura 6.3 apresenta a matriz duologue referente aos grafos iniciais.

$$D = \begin{pmatrix} [A_1, B_1] & [A_1, B_2] & \dots & [A_1, B_{28}] \\ [A_2, B_1] & [A_2, B_2] & \dots & [A_2, B_{28}] \\ [A_3, B_1] & [A_3, B_2] & \dots & [A_3, B_{28}] \\ \dots & \dots & \dots & \dots \\ [A_{28}, B_1] & [A_{28}, B_2] & \dots & [A_{28}, B_{28}] \end{pmatrix}$$

Figura 6.3: Matriz duologue referente aos grafos iniciais.

Aplicando-se a função de validação Val, a cada duologue da matriz D, obtém-se





de um dos processos (no caso, processo B) e as entidades do protocolo deixaram de ter um comportamento simétrico.

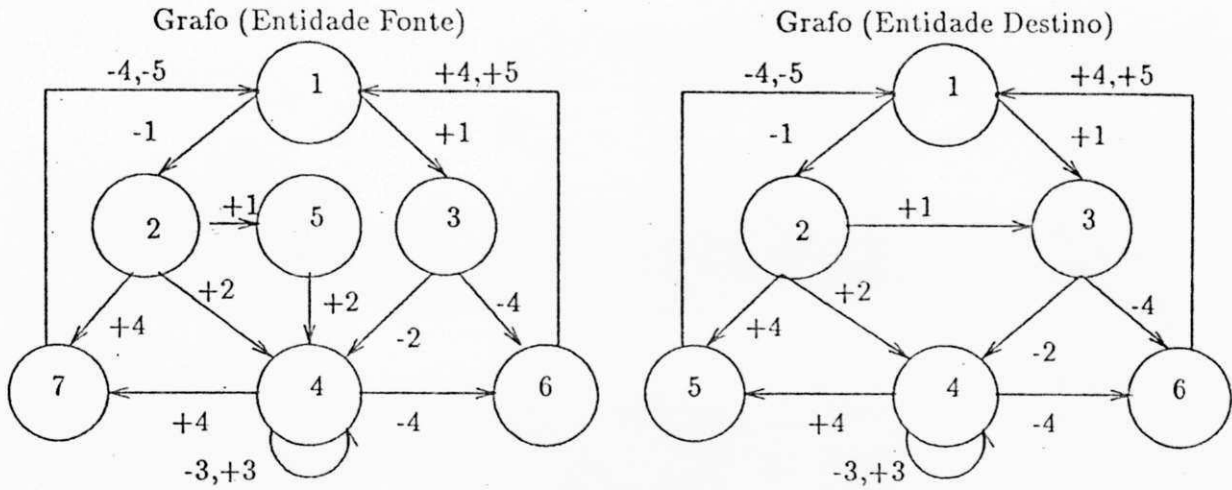


Figura 6.4: Grafos corrigidos.

Para verificar se o problema dessa primeira colisão mal sucedida foi resolvido, pode-se submeter os grafos referentes à Figura 6.4 à ferramenta de validação.

No processo de validação dos grafos corrigidos foram obtidos os conjuntos de todos os unilogues do protocolo. Os conjuntos de unilogues, relativos aos processos A e B, são apresentados nas Tabelas 6.4 e 6.5 respectivamente.

$A_1 = \{+1,-4,+5\}$	$A_{11} = \{-1,+2,-4,+5\}$	$A_{21} = \{-1,+1,+2,+4,-5\}$	$A_{31} = \{-1,+2,-3,-4,+5\}$
$A_2 = \{+1,-4,+4\}$	$A_{12} = \{-1,+2,-4,+4\}$	$A_{22} = \{-1,+1,+2,+4,-4\}$	$A_{32} = \{-1,+2,-3,-4,+4\}$
$A_3 = \{-1,+4,-5\}$	$A_{13} = \{+1,-2,+3,+4,-5\}$	$A_{23} = \{-1,+1,+2,-4,+5\}$	$A_{33} = \{-1,+1,+2,+3,+4,-5\}$
$A_4 = \{-1,+4,-4\}$	$A_{14} = \{+1,-2,+3,+4,-4\}$	$A_{24} = \{-1,+1,+2,-4,+4\}$	$A_{34} = \{-1,+1,+2,+3,+4,-4\}$
$A_5 = \{+1,-2,+4,-5\}$	$A_{15} = \{+1,-2,+3,-4,+5\}$	$A_{25} = \{-1,+2,+3,+4,-5\}$	$A_{35} = \{-1,+1,+2,+3,-4,+5\}$
$A_6 = \{+1,-2,+4,-4\}$	$A_{16} = \{+1,-2,+3,-4,+4\}$	$A_{26} = \{-1,+2,+3,+4,-4\}$	$A_{36} = \{-1,+1,+2,+3,-4,+4\}$
$A_7 = \{+1,-2,-4,+5\}$	$A_{17} = \{+1,-2,-3,+4,-5\}$	$A_{27} = \{-1,+2,+3,-4,+5\}$	$A_{37} = \{-1,+1,+2,-3,+4,-5\}$
$A_8 = \{+1,-2,-4,+4\}$	$A_{18} = \{+1,-2,-3,+4,-4\}$	$A_{28} = \{-1,+2,+3,-4,+4\}$	$A_{38} = \{-1,+1,+2,-3,+4,-4\}$
$A_9 = \{-1,+2,+4,-5\}$	$A_{19} = \{+1,-2,-3,-4,+5\}$	$A_{29} = \{-1,+2,-3,+4,-5\}$	$A_{39} = \{-1,+1,+2,-3,-4,+5\}$
$A_{10} = \{-1,+2,+4,-4\}$	$A_{20} = \{+1,-2,-3,-4,+4\}$	$A_{30} = \{-1,+2,-3,+4,-4\}$	$A_{40} = \{-1,+1,+2,-3,-4,+4\}$

Tabela 6.4: Conjunto de unilogues do processo A referente ao segundo projeto.

$B_1 = \{+1, -4, +5\}$	$B_{15} = \{+1, -2, +3, +4, -5\}$	$B_{29} = \{-1, +2, -3, -4, +5\}$
$B_2 = \{+1, -4, +4\}$	$B_{16} = \{+1, -2, +3, +4, -4\}$	$B_{30} = \{-1, +2, -3, -4, +4\}$
$B_3 = \{-1, +4, -5\}$	$B_{17} = \{+1, -2, +3, -4, +5\}$	$B_{31} = \{-1, +1, -2, +4, -5\}$
$B_4 = \{-1, +4, -4\}$	$B_{18} = \{+1, -2, +3, -4, +4\}$	$B_{32} = \{-1, +1, -2, +4, -4\}$
$B_5 = \{+1, -2, +4, -5\}$	$B_{19} = \{+1, -2, -3, +4, -5\}$	$B_{33} = \{-1, +1, -2, -4, +5\}$
$B_6 = \{+1, -2, +4, -4\}$	$B_{20} = \{+1, -2, -3, +4, -4\}$	$B_{34} = \{-1, +1, -2, -4, +4\}$
$B_7 = \{+1, -2, -4, +5\}$	$B_{21} = \{+1, -2, -3, -4, +5\}$	$B_{35} = \{-1, +1, -2, +3, +4, -5\}$
$B_8 = \{+1, -2, -4, +4\}$	$B_{22} = \{+1, -2, -3, -4, +4\}$	$B_{36} = \{-1, +1, -2, +3, +4, -4\}$
$B_9 = \{-1, +2, +4, -5\}$	$B_{23} = \{-1, +2, +3, +4, -5\}$	$B_{37} = \{-1, +1, -2, +3, -4, +5\}$
$B_{10} = \{-1, +2, +4, -4\}$	$B_{24} = \{-1, +2, +3, +4, -4\}$	$B_{38} = \{-1, +1, -2, +3, -4, +4\}$
$B_{11} = \{-1, +2, -4, +5\}$	$B_{25} = \{-1, +2, +3, -4, +5\}$	$B_{39} = \{-1, +1, -2, -3, +4, -5\}$
$B_{12} = \{-1, +2, -4, +4\}$	$B_{26} = \{-1, +2, +3, -4, +4\}$	$B_{40} = \{-1, +1, -2, -3, +4, -4\}$
$B_{13} = \{-1, +1, -4, +5\}$	$B_{27} = \{-1, +2, -3, +4, -5\}$	$B_{41} = \{-1, +1, -2, -3, -4, +5\}$
$B_{14} = \{-1, +1, -4, +4\}$	$B_{28} = \{-1, +2, -3, +4, -4\}$	$B_{42} = \{-1, +1, -2, -3, -4, +4\}$

Tabela 6.5: Conjunto de unilogues do processo B referente ao segundo projeto.

A partir dos unilogues referentes aos processos A e B é montada a matriz duologue D. A Figura 6.5 apresenta a matriz duologue referente aos grafos corrigidos.

$$D = \begin{pmatrix} [A_1, B_1] & [A_1, B_2] & \dots & [A_1, B_{42}] \\ [A_2, B_1] & [A_2, B_2] & \dots & [A_2, B_{42}] \\ [A_3, B_1] & [A_3, B_2] & \dots & [A_3, B_{42}] \\ \cdot & \cdot & \dots & \cdot \\ [A_{40}, B_1] & [A_{40}, B_2] & \dots & [A_{40}, B_{42}] \end{pmatrix}$$

Figura 6.5: Matriz duologue referente aos grafos corrigidos.

Aplicando-se a função de validação Val, a cada duologue da matriz D, obtém-se a matriz de validação Val[D]. A Tabela 6.6 apresenta a matriz de validação referente aos grafos corrigidos.



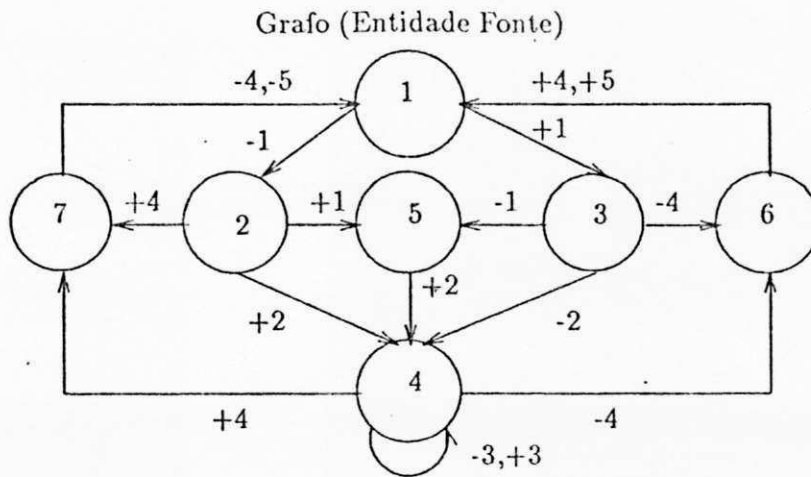


Figura 6.6: Entidade fonte do protocolo X.25 PCP.

Quanto à segunda correção, os projetistas do protocolo X.25 PCP optaram por descartar o pacote de dados enviado (ETD\_Data). Ou seja, tal erro de *design* não foi levado em consideração.

Esse exemplo, justifica a solução adotada na implementação da ferramenta, no sentido de não corrigir automaticamente os casos de colisões mal sucedidas, já que existem várias correções possíveis para esse caso em particular.

# Capítulo 7

## Conclusão

Devido à crescente aceitação do modelo OSI e de seus protocolos, validar a implementação desses protocolos tem se tornado uma atividade muito importante e de intensa pesquisa.

A validação de uma implementação de um protocolo fica comprometida se a especificação, que deu origem a essa implementação, contém erros. Mesmo em relação ao modelo OSI, já ficou comprovado que ocorreram erros de *design* em certos protocolos (*e.g.*, X.21 e X.25), que obrigam, hoje, as diversas implementações existentes a conviver com tais erros.

A validação do *design* é uma atividade fundamental, no contexto do ciclo de vida de um protocolo, pois busca avaliar a concepção conjuntamente com a especificação do protocolo e evitar que erros de *design* se propaguem para as demais fases do seu ciclo de vida.

Uma vez que há uma tendência do projetista, na fase inicial de *design*, de cometer erros básicos, é importante que técnicas de especificação e validação não muito complexas sejam utilizadas numa fase inicial, a fim de facilitar a automatização do processo de validação. Essa automatização é importante, na medida em que fornece ao projetista uma resposta imediata com respeito ao protocolo que está sendo concebido.

Neste trabalho houve uma preocupação de desenvolver uma ferramenta amigável e interativa, que atendesse aos requisitos acima explicitados.

A partir da especificação de um protocolo, através de um par de grafos, que por sua vez são representados por matrizes de transição de estados, essa ferramenta realiza funções de geração, análise e síntese.

Na primeira etapa, para ambos os grafos, todos os caminhos que começam e terminam no estado inicial (unilogues) são gerados.

Na segunda etapa, após a construção do conjunto de seqüências de interações (matriz duologue), essas seqüências são analisadas e os seguintes erros básicos de *design* podem ser detectados: situações de impasse, recepções não especificadas, perdas de transmissões, colisões mal sucedidas e ambigüidades.

Na última etapa, os seguintes tipos de erros, que foram detectados na etapa anterior, são corrigidos automaticamente: situações de impasse, recepções não especificadas e perdas de transmissões.

Erros relativos às colisões mal sucedidas não são corrigidos pela ferramenta e deixados a cargo do projetista, pois são muito inerentes ao protocolo que está sendo concebido. As ambigüidades também não são corrigidas, pois nem sempre representam erros de *design*.

A ferramenta desenvolvida foi utilizada para validar vários protocolos, entre eles, uma versão simplificada do protocolo X.25 PCP. Nas várias validações realizadas ficou comprovado que, freqüentemente, são cometidos vários erros básicos de *design* (concepção ou especificação).

Os resultados obtidos comprovaram a tese inicial de que uma ferramenta, utilizando uma técnica relativamente simples, pode ser extremamente útil ao projetista.

É importante ressaltar que as técnicas de validação descritas neste trabalho, apesar de existirem a mais de uma década, continuam sendo bastante empregadas na validação de protocolos de comunicação.

Dando prosseguimento a este trabalho, estão sendo realizados esforços no sentido de incorporar à ferramenta outras formas de correção automática, para os erros detectados. O objetivo é o de aperfeiçoar os procedimentos de síntese executados pela ferramenta.



# Bibliografia

- [BaDa 80] D.L.A. Barber, D.W. Davies, W.L. Price, C.M. Solomonides, "*Computer Networks and Their Protocols*," Jonh Wiley & Sons Ltd, First Edition, 1980.
- [BoMu 79] Bondy, J.A., U.S.R. Murty, "*Graph Theory with Applications*," Second Edition, 1979.
- [CCIT 88] CCITT Recomendation X.223, "*Use of X.25 to provide the OSI connection-mode network service for CCITT applications*," IXth Plenary Assembly, Vol. VIII, Fascicle VIII.5, 1988.
- [Furt 73] L.A. Furtado, "*Teoria dos Grafos*," Série Ciência da Computação, Primeira Edição, 1973.
- [Hals 88] Halsall, F., "*Data Communications, Computer Networks and OSI*," Addison-Wesley Publishing Company, Second Edition, 1988.
- [HoSa 87] E. Horowitz, S. Sahni, "*Fundamentos de Estruturas de Dados*," Editora Campus, Terceira Edição, 1987.
- [ISOa 83] ISO IS 7185, "*Programming Language Pascal*," First Edition, 1983.



- [ISO 83] ISO IS 7498, "Information Processing Systems - Basic Reference Model for Open Systems Interconnection," First Edition, 1983.
- [LoSo 90] W. Lopes de Souza, J.N. Souza, "Simulation of Protocol Formal Specifications," *Anais da International Conference on Local Area Networks*, Madras (India), pp. 174-188, Jan. 1990.
- [LoSt 88] W. Lopes de Souza, S. Stiubiener, "Especificação, Verificação e Testes de Protocolos," *Revista da Sociedade Brasileira de Telecomunicações*, Vol 3, pp. 1-20, Dez. 1988.
- [AnSa 86] J. Antão B.M., J.P. Sauvé, W.F. Giozza, J. Marinho de A., "Protocolos de Alto Nível e Avaliação de Desempenho," McGraw-Hill, Primeira Edição, 1986.
- [Souz 90] J.N. de Souza, "Uma metodologia para validação, através de simulação, de especificações formais de protocolos de comunicação," Tese de Mestrado relativa ao Curso de Mestrado do DSC/CCT/UFPb, Campina Grande (Pb), 1990.
- [Tane 89] Tanenbaum, Andrew S., "Computer Networks," Segunda Edição, Prentice-Hall International, 1989.
- [Taro 86] Tarouco, L.M.R., "Redes de Computadores Locais e de Longa Distância," Primeira Edição, McGraw-Hill, 1986.
- [Zafi 78] P. Zafiropulo, "Protocol Validation by Duologue-Matrix Analysis," *IEEE - Transactions on Communications*, Vol. COM-26, No. 8, pp. 1187-1194, Aug. 1978.

- [West 78] C.H. West, "Automated Technique of Communications Protocol Validation," *IEEE - Transactions on Communications*, Vol. COM-26, No. 8, pp. 1271 - 1275, Aug. 1978.
- [WeZa 80] C.H. West, P. Zafiropulo, "Towards Analyzing and Synthesizing Protocols," *IEEE - Transactions on Communications*, Vol. COM-28, No. 4, pp. 651-661, April 1980.