ASPECTOS DE METODOLOGIA DE GERAÇÃO DE SEQUÊNCIAS DE TESTES PARA PROTOCOLOS DE COMUNICAÇÃO DE DADOS

DENIS GABOS
ESCA Engenharia de Sistemas de Controle e Automação

STEFANIA STIUBIENER
Departamento de Engenharia de Eletricidade,
ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

I. INTRODUCÃO

Em um ambiente OSI* (*), testar uma determinada implementação de protocolos significa verificar se ela oferece os serviços especificados, utilizando corretamente os serviços inferiores e realizando a comunicação com as entidades pares nos outros sistemas. A figura 1.1 apresenta, portanto, a arquitetura conceitual de um testador.

A arquitetura real do testador tomará diversas formas, dependendo dos detalhes da implementação sob teste (IST) a ser testada, o que determinará o método a ser utilizado. (ref.1).

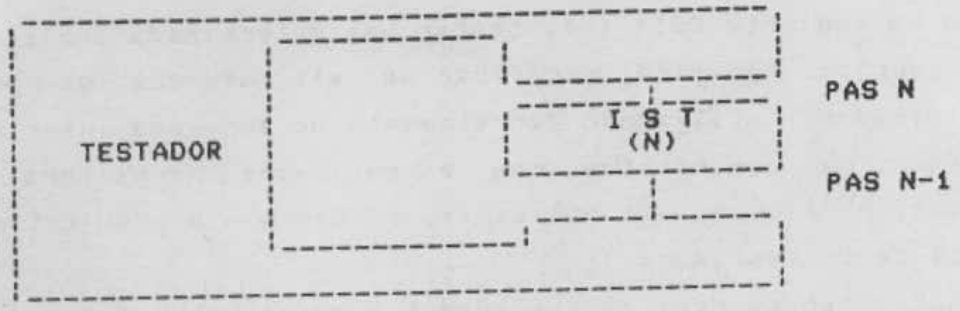
O assunto "Teste de Conformidade" pode ser dividido e estudado sob dois aspectos distintos e quase independentes: a arquitetura real do testador e as sequências de eventos que serão aplicadas nas interfaces da IST a fim de verificar o seu comportamento.

A arquitetura real dependerá de que aspecto do protocolo ou sistema está em teste e das restrições de acesso ao Sistema onde se encontra a IST.

Quanto às sequências de eventos que serão utilizadas nos testes, elas podem ser obtidas a partir da arquitetura conceitual e transposta para a real definida, resolvendo-se os problemas de sincronismo que possam aparecer, como veremos.

Historicamente, até a algum tempo atrás, alguns centros de pesquisa e/ou universidades se destacaram no desenvolvimento e implementação de arquiteturas reais de testadores, como o NPL ("National Phisical Laboratory") na Inglaterra (ref.2), o NBS ("National Bureau of Standards") americano (ref.3), o CREI-Politécnico de Milão (ref.4) ou o GMD ("Gesellschaft fur Mathematik und Datenfernverarbeitung") alemão (ref.5), enquanto que outros concentraram seus esforços nos estudos sobre sequências de teste.

(*) - Entende-se por OSI* tanto os padrões OSI da ISO como as recomendações das séries X e T da CCITT [ref. 1].



IST : Implementação Sob Teste.
PAS : Ponto de Acesso aos Serviços.

FIGURA 1.1: Arquitetura Conceitual de Teste

II. SITUAÇÃO NA PADRONIZAÇÃO DO PROCESSO DE AVALIAÇÃO DE CONFORMIDADE

O objetivo do trabalho da ISO* é padronizar o Processo de Avaliação de Conformidade, que consiste, no seu entender, no processo completo de desempenhar todas as atividades de teste de conformidade necessárias para verificar a conformidade de uma implememtação ou de um sistema a uma ou mais recomendações ISO* a serem avaliadas [ref.1].

Nesse trabalho de padronização a ISO* efetuou uma compilação da experiência acumulada no assunto pelos grupos e centros de pesquisa que vêm trabalhando no assunto.

Padronizar esse processo significa definir a metodologia de trabalho, o método de teste e o conjunto de procedimentos a serem executados. O ponto de partida de trabalho é a recomendação X.290 [ref.1], dividida basicamente em duas partes:

Parte I -Conceitos gerais, onde é estabelecida a terminologia básica comum, o ambiente e método gerais de trabalho e é descrito o processo de avaliação de conformidade;

Parte II -É um guia prático para a especificação de Conjuntos Abstratos de Teste.

Na figura 2.1 está o esquema do processo de avaliação de conformidade [ref.1].

Ao ser realizada uma implementação de protocolo a partir de suas recomendações, algumas escolhas de opções são efetuadas.

Portanto, o Processo de Avaliação de Conformidade parte de dois documentos fornecidos pelo implementador além da implementação propriamente dita:

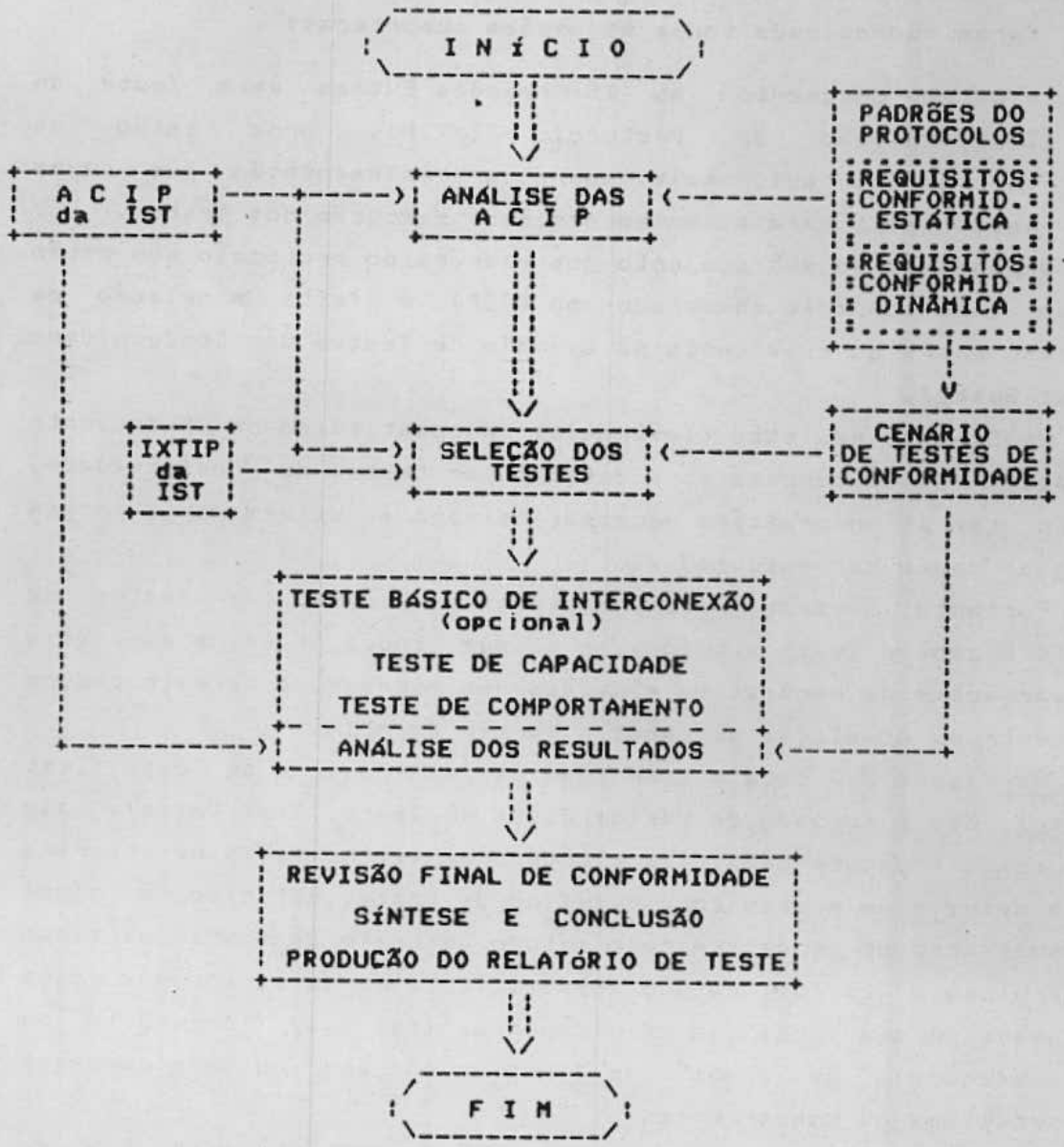


FIGURA 2.1: Processo de Avaliação de Conformidade

- Um documemto com as Afirmacões de Conformidade da Implementação do Protocolo (ACIP) onde são enumeradas de uma forma padronizada todas as opções suportadas;

- Outro documento, as Informações Extras para Teste da Implementação do Protocolo (IXTIP), onde estão as informações adicionais sobre a implementação que sejam necessárias para a implementação e execução dos testes.

Dependendo do sub conjunto dos padrões do protocolo que estão sendo suportados (e enumerados no ACIP) é feita a seleção de testes entre os existentes no Cenário de Testes de Conformidade (Test Suite).

Um Cenário de Teste (Test Suite) é constituído de um conjunto completo de sequências para desempenhar testes de conformidade, junto com as informações necessárias para se determinar a ordem na qual devem ser executadas.

Portanto, a especificação abstrata do Cenário de Testes de Conformidade é feita a priori, e possue todos os casos possíveis de variações de parâmetros e opções dos padrões, e permite chaves de entrada e seleção de casos a partir da ACIP.

Na figura 2.2 está a estrutura de um Cenário de Teste (Test Suite). Ele é formado de vários Casos de Teste (Test Cases), que são especificações completas e independentes de ações necessárias para atingir um propósito específico de teste, definido no nível de abstração de um determinado método abstrato de teste, partindo e terminando em um estado estável (isto é, um estado que possa ser mantido quase que indefinidamente, tal qual "inoperante" ou "transferência de dados") e involvendo uma ou mais conexões consecutivas ou concorrentes.

Os Casos de Teste são constituídos de PASSOS que por sua vez são constituídos por EVENTOS. Os Casos de Teste podem ser organizados em GRUPOS identificados para que sejam utilizados em outros Cenários de Teste.

Portanto, cada Cenário de Teste é composto de uma ou mais sequências de teste com seus casos condicionais e veredictos finais. Esses veredictos podem ser: falha, passou ou inconclusivo.

No anexo C da Parte II da recomendação X.290 é apresentada uma notação (TTCN: Tree and Tabular Combined Notation) para especificação abstrata das sequências de teste.

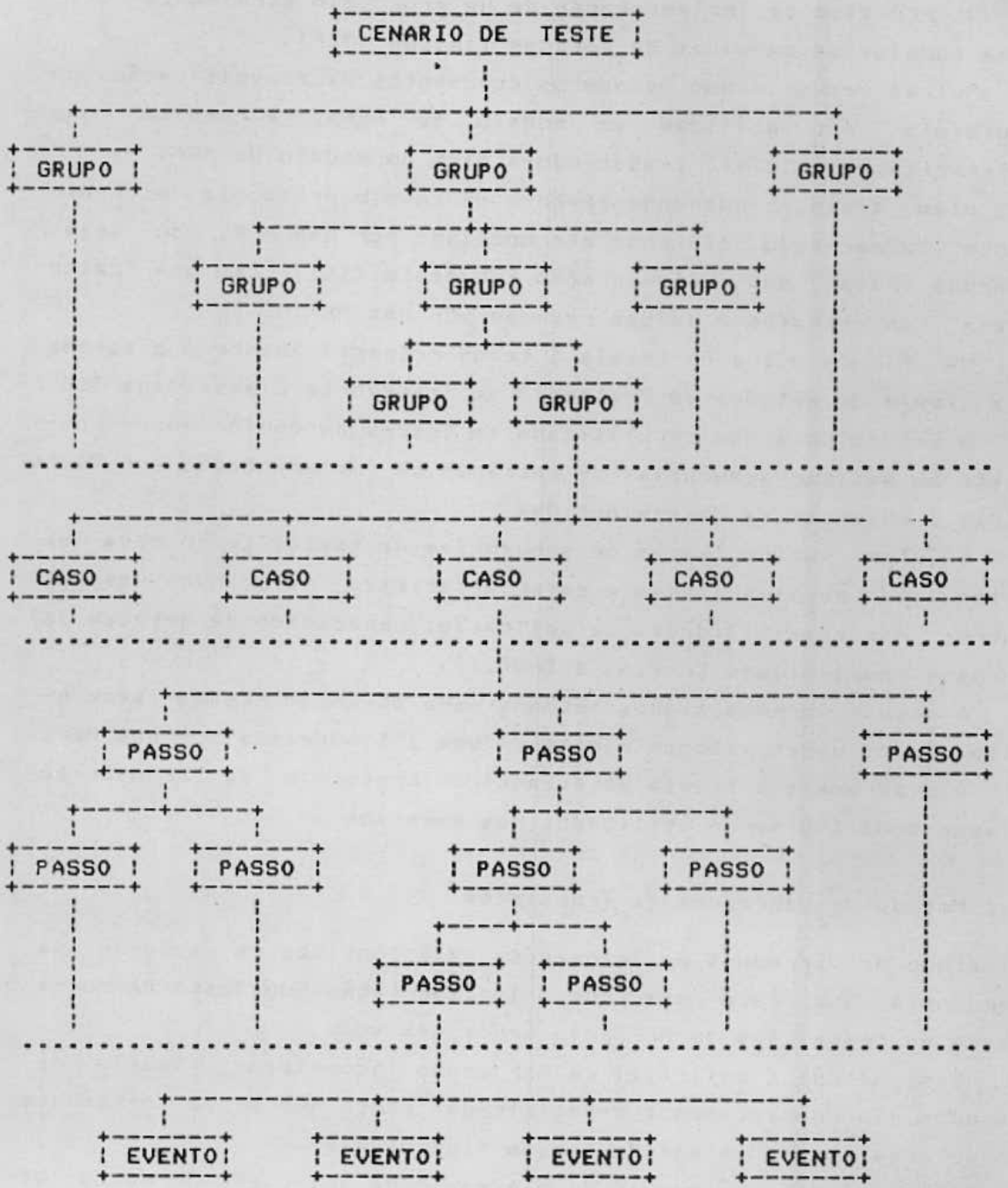


FIGURA 2.2: Estrutura de um Conjunto de Teste

III. SEQUÊNCIAS DE TESTES: ABORDAGEM INICIAL

O processo de implementação de um protocolo geralmente utiliza modelos de máquinas de estados finitos (MEF).

Muitas vezes, mesmo quando os documentos de especificação do protocolo não utilizam um modelo de MEF, é feita uma interpretação inicial, traduzindo-a para um modelo de MEF.

Além disso, independentemente de como o protocolo foi realmente implementado, ele pode ser modelado por uma MEF, ou seja, podemos tratar uma implementação sob teste (IST) como uma "caixa preta" com entradas e saídas regidas por uma MEF.

Na figura 3.1 e na tabela 1 temos respectivamente o diagrama e a tabela de estados do Protocolo de Transporte Classe 0 da ISO.

A metodologia básica utilizada em testes de conformidade consiste em aplicar sequências de entradas na implementação sob teste (IST) e observar as saídas obtidas.

Existem vários tipos de sequências de teste, tendo cada uma delas suas peculiaridades e características, diferindo umas das outras nas possibilidades de aplicação, capacidade de deteção de erros e complexidade [ref.6] e [ref.7].

A seguir veremos alguns métodos para obtenção dessas sequências. Todos esses métodos partem de uma IST modelada por uma MEF.

O diagrama e a tabela de estados do protocolo de transporte classe 0 da ISO serão utilizados nos exemplos.

3.1 Método de Varredura de Transições

O método de Varredura de Transições (VT) consiste em elaborar uma sequência que faça com que a Implementação Sob Teste percorra todas as transições da MEF pelo menos uma vez.

Esse método é aplicável em MEF mesmo incompletas (Tabela de Estados não completamente especificada) tanto que sejam fortemente conectadas (todos os estados sejam alcançáveis).

A sequência a seguir é um exemplo da aplicação do método VT sobre o Protocolo de Transporte classe 0 da ISO:

Entrada:		CR		T_Dreq		CC		DT		DR		CR		CR
Estado:	1		2		1		1		1		2		1	
Entrada:		T_Creq		DT		CR		T_Cresp		T_DTreq		DT		T_Dreq
Estado:	3		1		1		2		4		4		4	
Entrada:		CR		CC		CR		DT		CR		DR		T_Creq
Estado:	1		2		1		2		2		1		2	
Entrada:		DR		T_Creq		CC		CR		CR				
Estado:	1		3		3		4		1		2			

ESTADO > / ENTRADA	1 LIVRE	2 ESPERA CONF. DO USUÁRIO	3 ESPERA CONF. DA REDE	4 TRANSFERÊNCIA DE DADOS
T_Creq	3/CR	-	-	-
T_Dreq	-	1/DR	-	1/N_Dreq
T_Cresp	-	4/CC	-	-
T_DTreq	-	-	-	4/DT
CR	2/T_Cind	1/ERR	-	1/ERR
CC	1/-	1/ERR	4/T_Cconf	-
DT	1/-	1/ERR	1/-	4/T_DTind
DR	1/-	1/ERR	1 /T_Dind / N_Dreq	1/N_Dreq
N_Dind	-	-	-	1/T_Dind
N_Rind	-	-	-	1/T_Dind

TABELA 1: Tabela de Estados Protocolo de Transporte Classe 0 da ISO

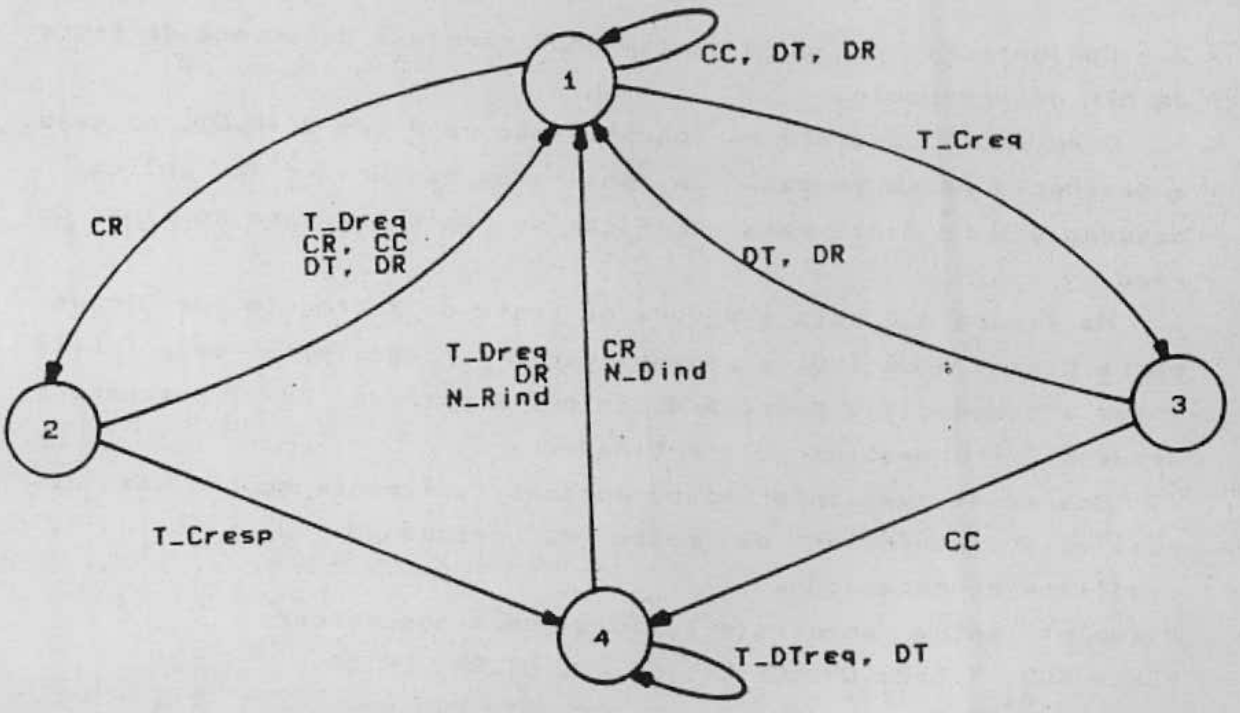


FIG. 3.1: Protocolo de Transporte Classe 0 da ISO Diagrama de Estado

Entrada: T_Cresp DR CR T_Cresp N_Dind CR T_Cresp
Estado: 2 4 1 2 4 1 2 4

Entrada: N_Rind
Estado: 4 1

Devido ao fato que a sequência é elaborada a fim de exatamente cobrir todas e apenas as transições possíveis, e que isso é feito através do modelo de MEF da IST, podemos ver que esse método tem a capacidade de detectar erros funcionais, mas não detecta erros de transição ou de estados perdidos ou em excesso devido a falhas na construção dos diagramas de estados.

3.2 Método W

O método W é baseado na concatenação de duas sequências:

1. Sequência W: distingue o comportamento entre quaisquer 2 pares de estados, ou seja, no caso de protocolos é uma sequência que produz uma saída diferente para cada estado em que a MEF se encontra;

2. Conjunto P: conjunto de caminhos parciais da árvore de teste da MEF do protocolo.

O método W consiste na concatenacão de P com W (P.W), ou seja, a sequência percorre cada um dos ramos da árvore e aplica a sequência W no final para verificar se realmente está no final do ramo.

Na figura 3.2 está a árvore de teste do Protocolo de Transporte Classe 0 da ISO, e podemos verificar facilmente pela tabela 1 que a sequência W desse protocolo é a entrada DR ("Disconnect Request") transmitida pela entidade par.

Com essas duas informações podemos facilmente montar uma das possíveis sequências de teste do método W. O símbolo "." significa concatenação.

Exemplo: (entre parêntesis estão algumas observações)

P.W = (DR, T_Creq.DR, CR.DR, CC.DR, DT.DR, DR.DR, (2)
(0) (1) (1)
CR.(T_Dreq, T_Cresp, CR, CC, DT, DR).DR, (3)
T_Creq.(CC, DT, DR).DR, (4)
CR.T_Cresp.(T_Dreq, T_DTreq, CR, DT, DR, (5)
N_Dind, N_Rind).DR)

OBSERVAÇÕES:

(0): O primeiro caminho de P é a sequência vazia;

(1): Antes de ir para o segundo nível de ramos da árvore de teste, devem ser testados todos os ramos do primeiro nível, mesmo os intermediários para chegar aos outros níveis;

(2): Sequência que percorre todos os ramos do primeiro nível;

(3): Sequência que percorre todos os ramos do segundo nível, entrando pelo estado 2. Como exemplo, aplicando a concatenação, essa parte da sequência ficaria: CR.T_Dreq.DR, CR.T_Cresp.DR, CR.CR.DR, CR.CC.DR, CR.DT.DR, CR.DR.DR;

(4): Sequência que percorre todos os ramos do segundo nível entrando pelo estado 3;

(5): Sequência que percorre todos os ramos do último nível da árvore.

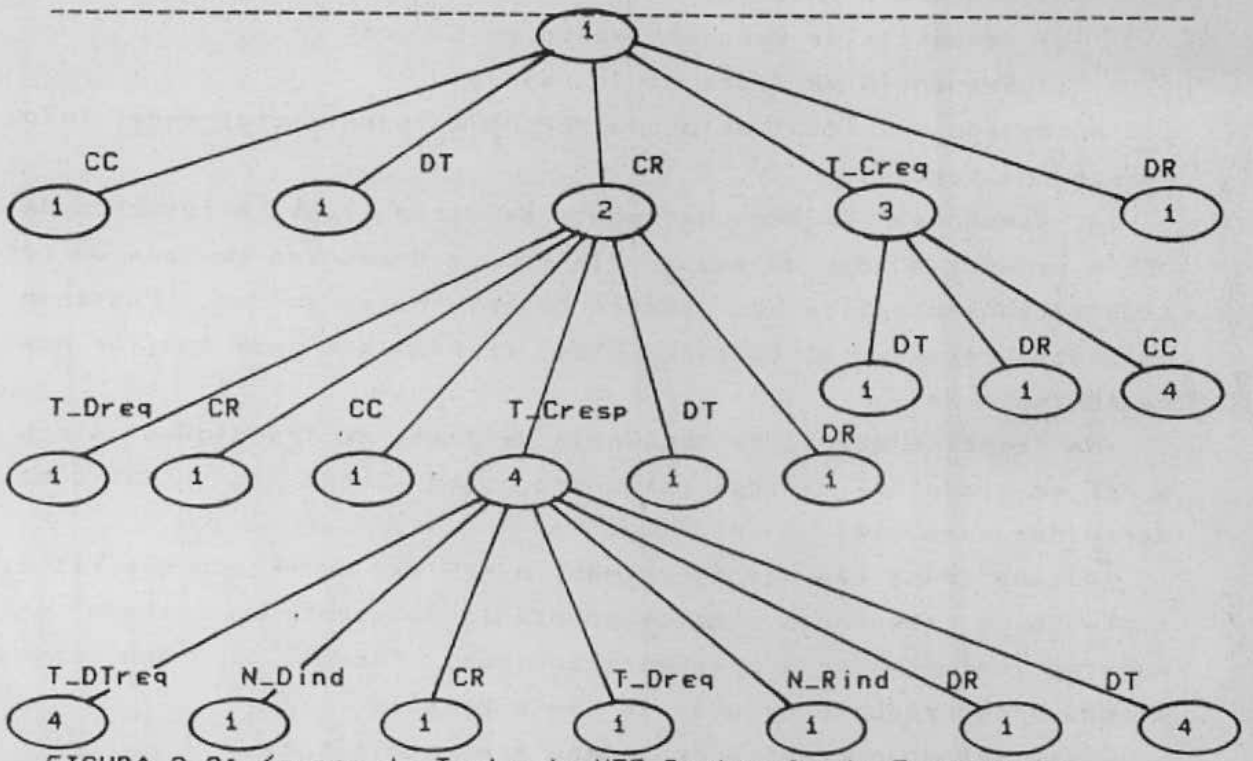


FIGURA 3.2: Árvore de Teste da MEF Protocolo de Transporte Classe 0 da ISO

A única restrição à aplicação desse método está no fato que não é garantida a existência da sequência W em MEF incompletas.

Podemos ver que a abrangência do método W é muito maior que a do método VT, mas em contra partida a complexidade do método e da sequência final é bem maior.

3.3 Método das Sequências de Verificacão

Esse método utilizará em uma das suas partes uma sequência chamada de Sequência de Distinção (Xd) que é uma sequência de entrada para a qual a saída produzida é diferente para cada estado inicial.

No caso de protocolos de comunicação de dados essa sequência geralmente se confunde com a sequência W (ítem 3.2) pois o único comportamento visível que caracteriza W são as saídas do protocolo.

O método das sequências de verificação é constituido por três partes:

a) Sequência Inicial;

b) Sequência de Reconhecimento de Estado;

c) Sequência de Teste de Transições.

A sequência Inicial coloca a MEF no estado inicial onde irão começar os testes.

A sequência de Reconhecimento de Estado testa a resposta da MEF à sequência Xd, ou seja, ela coloca a máquina em cada um de seus estados e aplica Xd, sempre observando as saídas. Passando por essa etapa, o testador tem certeza que pode confiar nas respostas à Xd.

Na terceira etapa, a sequência de Teste de Transições coloca a MEF em cada um de seus estados e testa todas as transições definidas para ele.

A cada transição que é testada, o MEF sai do estado sob teste, portanto a sequência precisa colocá-la novamente no estado em questão para testar a próxima transição. Podemos ver que esse método pode produzir sequências bem extensas.

Esse método necessita de alguns pré-requisitos:

a)que a MEF seja fortemente conectada;

b)que ela possua uma sequência Xd;

c)que os erros sejam constantes.

A sequência de verificação detecta qualquer erro funcional e de transição, entretanto não consegue detectar estados extras (devido à implementação) ou perdidos (não implementados).

3.4 Comparação

Na tabela 2 estão alguns resultados do tamanho de sequências

quando aplicadas em alguns protocolos exemplos.

Cabem ainda algumas observações com relação à tabela:

a) X.25-A é o protocolo X.25 acrescido de um Estado de Erro do Usuário, que não existe na norma CCITT;
b) X.25-B é o mesmo X.25 sem esse estado, e portanto, não completamente especificado. Entretando ele possue uma Entrada/Saida adicional "Read State" para indicação de erros de usuário;
c) No método W, o tamanho da sequência depende bastante do tamanho da sequência W;
d) A sequência final do método de sequências de verificação depende bastante de Xd. No caso do exemplo dado no ítem 3.2.3, além de Xd ser curta, ela sempre colocava a MEF no estado 1, o que simplificou bastante o retorno ao estado sob teste.

	TRANSPORTE	X.25-A	X.25-B	U.K.TRANS.
ESTADOS	4	9	8	10
ENTRADAS	10	7	8	10
TIPO	COMPRIMENTO MÁXIMO DE SEQUÊNCIAS			
VT	81	362	273	1.129
W	92	611	216	4.505
CS	122	-	380	-

TABELA 2: COMPARAÇÃO DE MÉTODOS DE OBTENSÃO DE SEQUÊNCIAS DE TESTE

IV. PROBLEMAS ESPECÍFICOS DE TESTES DE PROTOCOLOS

4.1. Sincronismo

A arquitetura que geralmente é utilizada para implementar testadores se caracteriza por ser um sistema distribuído onde não há uma comunicação direta entre o Diretor de Testes (que chamaremos testadores T) e o Respondedor de Testes (que chamaremos respondedor R).

Além disso, tanto T como R são MEF também e possuem entradas e saídas.

Portanto, implementar um testador significa implementar as

MEF's de T e R e sincronizá-las de maneira que trabalhando juntas realizem as sequências de entrada estudadas nos ítens anteriores.

Entretanto, não existe comunicação direta entre T e R; toda troca de mensagens ocorre através da própria implementação sob teste (I). Na figura 4.1 temos um modelamento em filas para estudarmos a troca de mensagens entre entidades T,I e R.

Em [7] é criada uma maneira de representar interações entre entidades através das filas, por exemplo, uma mensagem que passa de T para I pode ser representada tendo T como ponto de vista (S (ti), S de "SEND") ou (R (it)).

Portanto qualquer sequência de entradas e saídas de I pode ser representada em função de T ou R para fins de estudo.

Chamaremos de Sequência Básica de Interação (SBI) àquelas que definem uma sequência de transições em T, R e I, ou seja, sequências que não podem ser divididas em outras menores. Como exemplo podemos ter a recepção em I através de T de CR ("Call Request") (R (it)) e a consequente transmissão de I para R (S (ir)) de T_Cind. Temos então a SBI R (it) S (ir).

Diz-se que T ou R terá um problema de sincronismo quando, dada uma sequência constituída por 2 SBI, T ou R não participa da primeira SBI e tem que transmitir algo na segunda SBI, ou seja, ao ser implementada a MEF de T ou R não é possível obter um evento que dispare a participação de T ou R na segunda SBI.

Exemplos de SBI com problemas de sincronismo:

a) R (ir) + R (it) :problema para T;

b) R (ir) S (ir) + R (it) S (it) :também problema para T.

Portanto, um trabalho muito importante a ser feito é verificar se uma dada sequência de teste, desenvolvida por qualquer um dos três métodos vistos nos ítens anteriores, não possui problemas de sincronismo.

Em [7] são mensionados e indicados algorítimos para verificação se um determinado protocolo possui ou não problemas intrínsicos de sincronismo. Quando isso ocorre, significa que não é possível obter uma sequência de teste sem problemas de sincronismo.

4.2 Variacão de Parâmetros

Os métodos que estudamos nos ítens anteriores testam o comportamento da MEF sem levar em conta a influência que determinados parâmetros podem exercer. Dependendo dos valores em

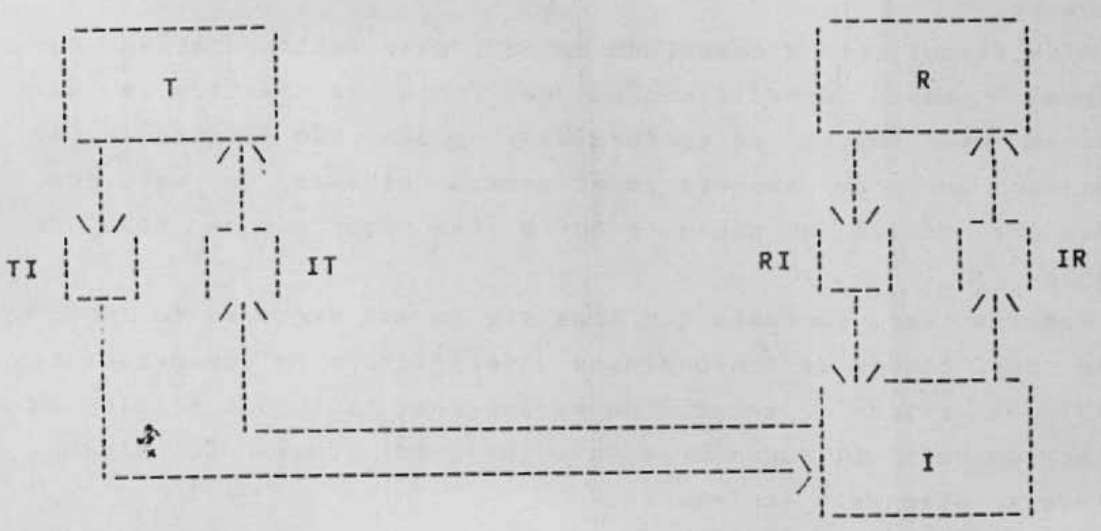
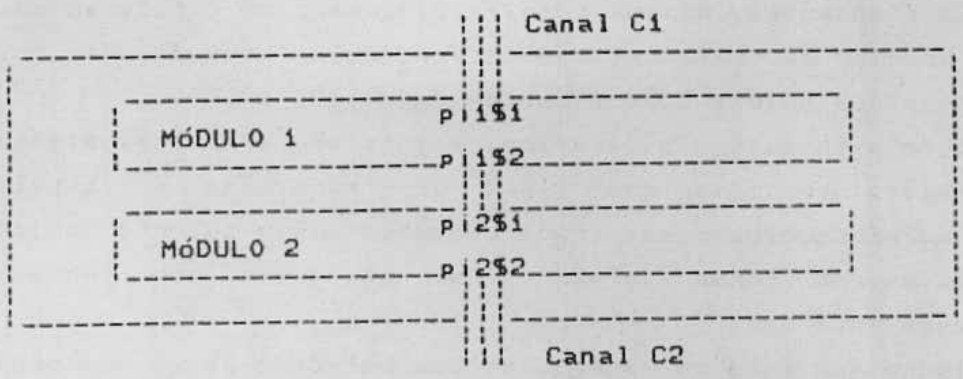


FIGURA 4.1: Modelo de Fila para Troca de Mensagems entre T, I e R



piX$Y: ponto de interação do módulo

FIGURA 4.2: Entidade de Protocolo com 2 Módulos

conjunto desses parâmetros, o comportamento da MEF pode ser bem diferente.

Além disso, como é comentado em [8], duas implementações que tiverem a mesma especificação como ponto de partida e que passaram nos testes de conformidade podem não conseguir se comunicar por não suportarem as mesmas classes de serviços, opções de código ou qualquer outro ítem negociado na fase de conexão.

Podemos ver, portanto que consiste em uma parte muito importante dos testes de conformidade a verificação do comportamento da IST com relação a variação de parâmetros, tanto das primitivas de serviço de nível superior e inferior como também de algumas variáveis internas à implementação.

Um dos métodos que poderia ser utilizado para testar a variação de parâmetros consiste em tratar cada um deles como uma variável de estado adicional. Nesse método a MEF cresce de maneira exponencial, tornando-se impraticável o trabalho de elaboração da sequência de teste.

Em [9] é desenvolvida uma técnica diferente da citada acima, e que se baseia na representação do protocolo através de dois grafos: o de controle e o de fluxo de dados (GC e GFD).
Para chegar a esses grafos, primeiramente são efetuadas algumas transformações em cima dos tipos de transições em Estelle [ref.14] do protocolo. Essas transformações visam obter o conjunto de Transições na Forma Normal (TFN) os quais não possuem ramificações internas.

Na figura 4.2 está um exemplo de uma entidade de um protocolo dividido em módulos, como poderia ser a arquitetura de uma especificação em Estelle.

Na figura 4.3 estão exemplos de declaração de tipo de transição, uma para o módulo 1 e outra para o módulo 2.

Em [9] são apresentados os passos para, a partir das transições chegar até P1 e P2 que são TFN.

No caso do exemplo foram efetuadas as seguintes operações:

1. Eliminação da cláusula "FROM através da criação da variável "state" e a sua colocação no PROVIDED;
2. Eliminação da cláusula TO através da existência da atribuição "state = connecting";
3. Junção do módulo 1 com módulo 2 pois a passagem de um para o

outro ocorre através de um evento interno do ponto de vista de entradas e saídas;

4. Divisão em P1 e P2 devido a duas condições possíveis da variável D, ou seja, no caso de D ocorrerá P1 e no caso de não D ocorrerá P2.

Se aplicarmos todas as transformações possíveis em uma especificação em Estelle do protocolo de transporte classe 0 da ISO, teremos o grafo de controle da figura 4.4. As transições são as TFN que se encontram enumeradas no apêndice de [9].

```
                    Especificação em ESTELLE
(módulo 1)
   WHEN        pi1$1.connect_req
   FROM        idle
   PROVIDED    A and (B or C)
   TO          connecting
   BEGIN
         if D then S1
           else S2
         output pi1$2.event1(connect_req, X+2)
   END;

(módulo 2)
   WHEN        pi2$1.event1(data_unit, lenght)
   PROVIDED    data_unit = connect_req  and  lenght (= 10
   BEGIN
         S3
         output pi2$2.connect_ind
   END;
```

```
                  TRANSIÇÕES NA FORMA NORMAL

(P1)
   WHEN        pi1$1.connect_req
   PROVIDED    state = idle and A and (B or C) and D
               and (X (=8)
   BEGIN
         state = connecting;
         s1;
         s3;
         output pi2$2.connect_ind
   END;

(P2)
   WHEN        pi1$1.connect_req
   PROVIDED    state = idle and A and (B or C) and not(D)
               and (X (=8)
   BEGIN
         state = connecting;
         s2;
         s3;
         output pi2$2.connect_ind
   END;
```

FIGURA 4.3: Exemplo de Transformações

Na figura 4.5, a título de exemplo, está uma parte do grafo de fluxo de dados. Ele é constituido basicamente por três tipos de nós: a) as variáveis internas da entidade; b) as primitivas de entrada e saída que alimentam e utilizam as variáveis e c) as

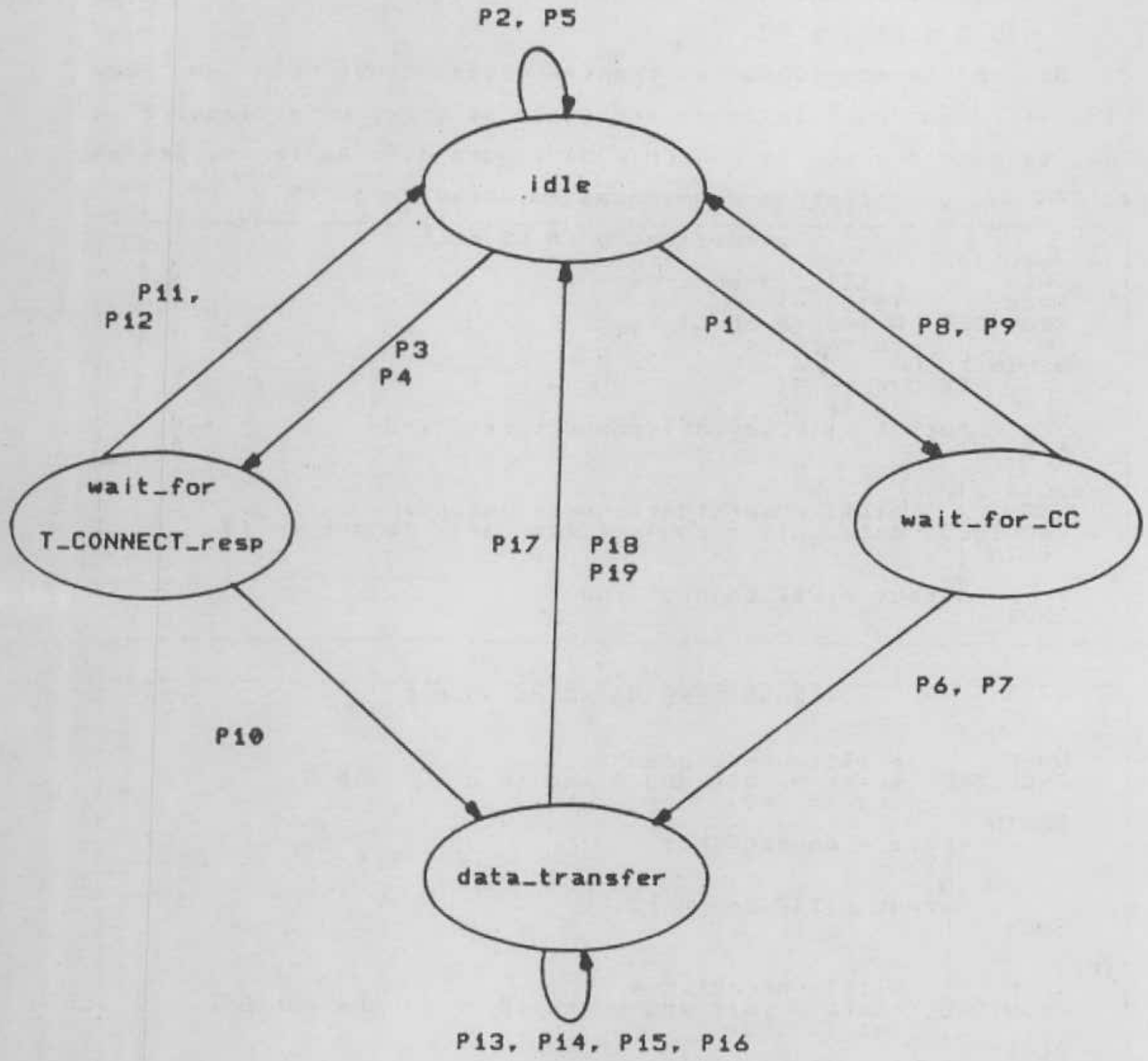


FIGURA 4.4: GRAFO DE CONTROLE Protocolo de Transporte classe 0 da ISO

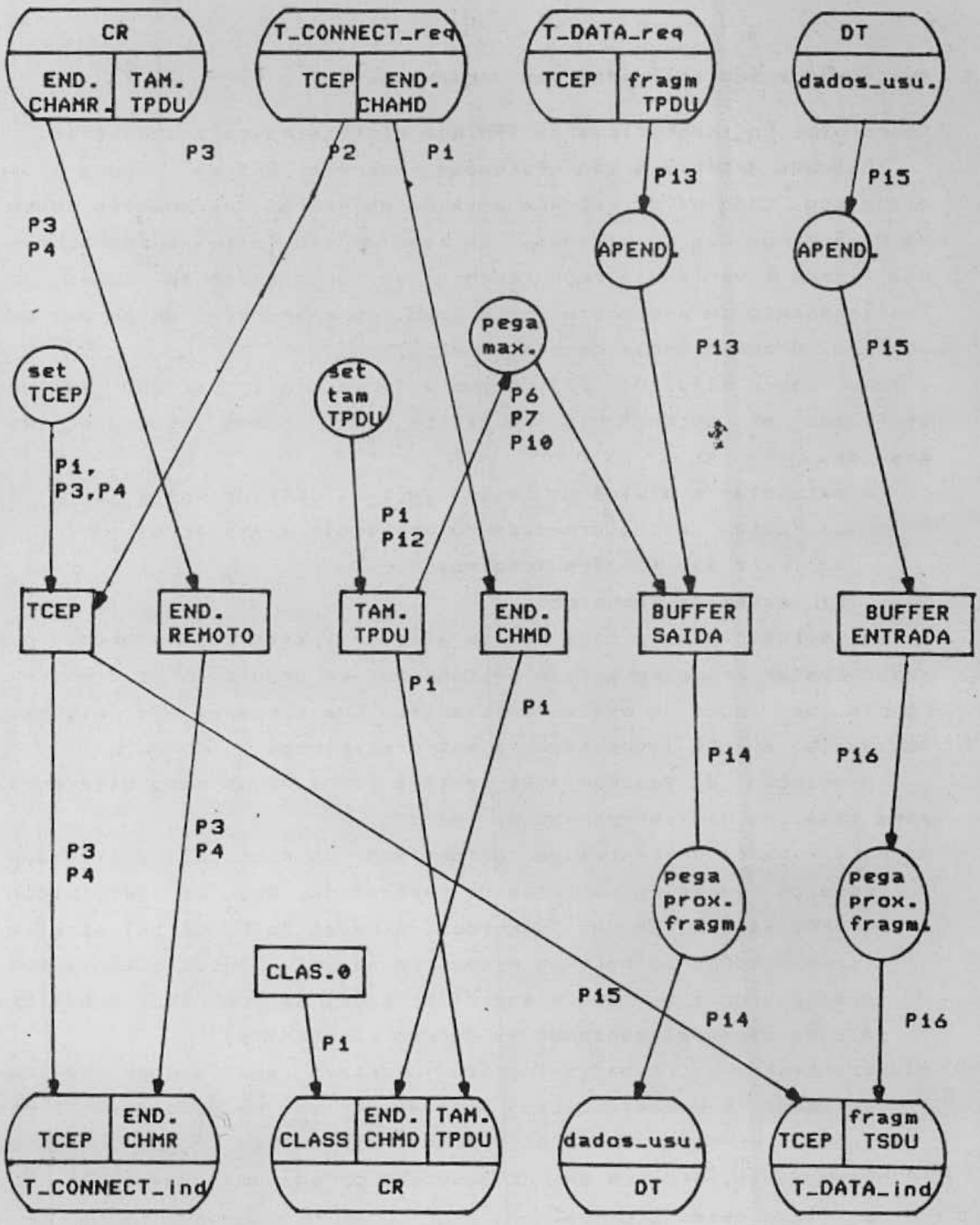


FIGURA 4.5: GRAFO DE FLUXO DE DADOS

Protocolo de Transporte classe 0 da ISO

funções que são aplicadas nas variáveis.

Os rótulos Pn identificam as TFN que efetuam essas associações.

Algumas operações são efetuadas sobre o GFD da figura 4.5, dividindo cada nó de entrada e saída em tantos nós quantos forem os parâmetros das primitivas. Em seguida são feitos agrupamentos dos ramos e variáveis procurando organizar o grafo em fases do funcionamento do protocolo, tais como, conexão referenciamento de conexão, transferência de dados, etc.

Na figura III.4 de [9] temos a forma final da GFD que é utilizada na metodologia de teste que iremos descrever em seguida.

A metodologia divide os testes em três categorias:

a) Testes de conformidade do protocolo e serviços;

b) Teste das funções informais;

c) Testes de robustez.

O método básico consiste em efetuar testes por blocos, ou seja, testar primeiro a fase de conexão, em seguida a de transferência de dados e assim por diante. Com excessão de algumas variáveis, existe independência entre os blocos.

A seleção da sequência de teste é feita de um modo diferente para cada uma das categorias de teste:

a) Para a parte de testes de conformidade do protocolo e serviços temos os testes de variação de parâmetros. Dado um certo bloco do GFD, escolhe-se uma "sub-tour" através do GC de tal maneira a cobrir todos os rótulos presentes naquele bloco. Então monta-se a sequência de modo a exercitar todas as ocorrências básicas de cada variável mantendo as outras constantes;

b) Para testar as chamadas funções informais (ex: funções de codificação e decodificação) pode ser criado um modelo de falhas através da enumeração das possíveis falhas e os efeitos observáveis. Através do GC deve ser criada uma sequência que verifique esses efeitos.

c) Para efetuar testes de robutez a sequência deve possuir todas as entradas possíveis em cada estado. Portanto a sequência deve colocar a MEF em cada um dos estados do GC, testar todas as entradas e desconectar.

V. CONCLUSÕES E TENDÊNCIAS

O que se pode perceber tanto da distribuição dos grupos de pesquisa quanto da evolução dos trabalhos ao longo do tempo é que, após uma etapa inicial em que alguns se preocuparam mais com a arquitetura em detrimento da sequência de testes e vice-versa, agora está havendo um intercâmbio maior entre esses grupos, preocupados em chegar a uma implementação.

A metodologia apresentada no item 4.2 torna-se complicada e trabalhosa à medida que aumenta a complexidade do protocolo sob teste. A evolução natural no caso é o desenvolvimento de ferramentras automáticas que interpretem as especificações e elaborem os grafos, ao mesmo tempo que auxiliem na execução das etapas de simplificação do grafo de fluxo de dados bem como na geração final da sequência de teste.

VI. BIBLIOGRAFIA

[1]- CCITT - SG VII, "Draft Recommendation X.290 on Testing and Verification of Data Communication Protocols", Contribution 257, COM VII - 257 E, dezembro, 1987.

[2] - RAYNER, D., "A System for Testing Protocol Implementations', in [11], pp.539-554.

[3] - NIGHTINGALE, J.S., " Protocol Testing Using a Reference Implementation", in [11], pp.513-522.

[4] - PALAZZO, S. et alli, "A Layer-Independent Architecture for a Testing System for Protocol Implementation", in [12], pp. 393-406.

[5] - GIEBLER, A., "Testing and Diagnosis Aids for Higher Level Protocols", in [12], pp. 407-420.

[6] - SARIKAYA, B. e BOCHMANN, G.V., "Some Experience With Test Sequence Generation for Protocols", in [12], pp.555-567.

[7] - SARIKAYA, B. e BOCHMANN, G.V., "Synchronization and Specification Issues in Protocol Testing", IEEE Trans.

Comm. Vol. COM-32, n.4, abril 1984, pp 389-395.

[8] - BOCHMANN, G.V., "On the Theorical Power of Some Testing Methods", in [10], pp.15-25.

[9]- SARIKAYA, B., "Test Design for Computer Network Protocols", Tese de Doutorado, School of Computer Science, McGill University, Montreal, março 1984, parte 2, capítulos 4, 5 e 6.

[10]- "Protocol Testing - Towards Proof?" INWG/NPL Workshop, maio 1981, Proceedings, V.2, Testing and Certification.

[11]- "Protocol Specification, Testing and Verification, II", C. Sunshine (ed), IFIP, 1982.

[12]- "Protocol Specification, Testing and Verification, III" WEST, C.H e RUDIAN, H. (ed), IFIP, 1983.

[13]- "Protocol Specification, Testing and Verification V, M. Diaz (ed.). Elsevier Science Publishers B. V. (North-Holland), IFIP, 1986.

[14]- ISO TC97/SC21/WG1, "Information Processing Systems - Open Systems Interconnection - Estelle - A Formal Description Technique Based on an Extended State Transition Model", ISO DP 9074, 1986.