UNIVERSIDADE ESTADUAL DE CAMPINAS

FACULDADE DE ENGENHARIA DE CAMPINAS

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

Este exemplar corresponde à redação final da tese defendida por Eduardo Teixeira Gomide e aprovada pela Comissão Julgadora em 09/07/86

Mauricio Ferreira Magalhães

DESENVOLVIMENTO DE UM AMBIENTE DE SIMULAÇÃO PARA PROJETO DE SISTEMAS DE TEMPO REAL

EDUARDO TEIXEIRA GOMIDE

ORIENTADOR: MAURÍCIO FERREIRA MAGALHÃES

TESE APRESENTADA À FACULDADE DE ENGENHARIA DE CAMPINAS / FEC / UNICAMP COMO PARTE DOS REQUISITOS EXIGIDOS PARA A OBTENÇÃO DO TÍTULO DE MESTRE EM CIÊNCIAS.

JULHO - 1986

À MEUS PAIS

MEUS AGRADECIMENTOS:

. A Maurício pela valiosa orientação e estímulo;

. Aos colegas da Universidade pelo apoio e colaboração;

. E a todos os amigos que direta ou indiretamente me auxiliaram na realização desse trabalho.

A realização deste trabalho contou com o apoio financeiro:

. da COORDENAÇÃO DE APERFEIÇOAMENTO DE PESSOAL DE NÍVEL SUPERIOR - CAPES;

. da FUNDAÇÃO DE AMPARO À PESQUISA DO ESTADO DE SÃO PAULO - FAPESP.

INDICE

Capítulo 1 - INTRODUÇÃO.....1

Capítulo 2 - SIMULAÇÃO DE SISTEMAS.....3

2.1 - Modelamento de Sistemas.....4

2.2 - Ferramentas de modelamento.....5

2.2.1 - Características da rede de PETRI....5

2.2.1.1 - Estrutura da rede de Petri.....6

2.2.2 - Características do GMB.....10

2.2.2.1 - Domínio de controle.....10

2.2.2.2 - Domínio dos dados.....14

2.2.2.3 - Domínio da interpretação.....17

2.3 - Aspectos da simulação.....21

2.3.1 - Técnicas de análise e simulação.....23

Capítulo 3 - O MEIO DE SIMULAÇÃO.....25

3.1 - Utilização do simulador GMB*.....25

3.1.1 - Variáveis de simulação.....26

3.1.2 - Comandos do simulador.....28

3.2 - Funcionamento do simulador GMB*.....29

3.2.1 - Ativação e terminação de um nó de controle.....29

3.3 - Estrutura do simulador GMB*.....31

3.3.1 - Domínio de controle do modelo.....36

3.3.1.1 - Descrição dos nós do grafo de controle.....36

3.3.2 - Domínio dos dados do modelo.....37

3.3.2.1 - Especificação dos armazenadores..38

3.3.3 - Interpretação dos módulos que constituem o simulador GMB*.....42

Capítulo 4 - EXEMPLO DE UTILIZAÇÃO DO SIMULADOR GMB* PARA A VALIDAÇÃO DE PROJETOS.....92

4.1 - Simulação visando a validação do grafo de controle.....93

4.2 - Simulação visando a validação do grafo de dados.....109

4.3 - Simulação visando a validação do modelo em um ambiente distribuido.....124

Capítulo 5 - CONCLUSÃO.....145

Apêndice A - EXEMPLO DE UTILIZAÇÃO E APLICAÇÃO DO SIMULADOR GMB*.....147

A.1 - Simulação do grafo de controle.....147

A.1.1 - Definição dos domínios no simulador.....149

A.1.2 - Execução da Simulação.....159

A.2 - Simulação do modelo observando o comportamento dos armazenadores.....160

A.2.1 - Definição dos domínios no simulador.....160

A.2.2 - Execução da simulação.....166

A.3 - Simulação Visando a validação do modelo em um ambiente distribuído.....167

A.3.1 - Definição dos domínios no simulador.....170

A.3.2 - Execução da simulação.....175

REFERÊNCIAS BIBLIOGRÁFICAS.....183

RESUMO

O simulador GMB* faz parte de um ambiente para projeto e programação de aplicações em tempo-real em desenvolvimento no Setor de Automação Industrial do Departamento de Engenharia Elétrica / FEC / UNICAMP.

No projeto e na implementação do simulador GMB* procurou-se enfatisar os aspectos de transportabilidade, estando o mesmo, atualmente, disponível em sistemas operacionais compatíveis com o CP/M e com o MS-DOS. A característica de modularidade foi também privilegiada permitindo a simulação de modelos de grandes dimensões em microcomputadores de oito e dezesseis 'bits'. Para a interação com o usuário, o simulador dispõe de uma interface 'homem-máquina' amigável que possibilita uma fácil manipulação do modelo da aplicação.

Através da utilização do simulador pode-se validar várias fases de um projeto de aplicação em tempo-real, tendo em vista a dinâmica do modelo e o fluxo de dados do mesmo.

1 - INTRODUÇÃO

Este estudo faz parte de um conjunto de trabalhos de pesquisa em desenvolvimento no Setor de Computação e Automação Industrial do Departamento de Engenharia Elétrica da UNICAMP. O grupo de trabalho se dedica, atualmente, a pesquisas para a obtenção de conhecimentos e ferramentas utilizadas em aplicações de controle de processos em tempo real.

Dentre os trabalhos desenvolvidos, ressalta-se a máquina GMB*, proposta por De Martino /07/. A máquina GMB* é baseada no modelo GMB* /07/, que é utilizado pelo simulador GMB*.

O simulador GMB* surgiu da necessidade de se ter uma ferramenta para o auxílio no desenvolvimento dos projetos do setor e, particularmente, tratando-se da máquina GMB*, o simulador teria como função a execução de testes dinâmicos no modelo da máquina. Com o desenvolvimento do trabalho, observou-se que as características do simulador permitem a simulação de modelos de diversos aplicativos, pois tais características oferecem uma boa flexibilidade de utilização ao usuário.

O simulador, atualmente, pode funcionar tanto em sistemas operacionais compatíveis com o CP/M 80 (oito bits), quanto em sistemas operacionais compatíveis com o MS-DOS (dezesseis bits).

Este trabalho é composto por seis capítulos e um apêndice, distribuídos da seguinte forma:

- no capítulo 2, são discutidas, brevemente, as técnicas de modelamento e simulação;

- no capítulo 3, descreve-se o meio de simulação através das estruturas do simulador e os procedimentos utilizados para a sua implementação;

- no capítulo 4, demonstra-se a simulação de uma aplicação modelada pelo GMB*;

- no capítulo 5, são apresentadas as conclusões e as perspectivas;

- no apêndice A, demonstra-se, através da simulação de um modelo de uma aplicação, a forma de utilização do simulador GMB*.

2 - SIMULAÇÃO DE SISTEMAS

Um sistema pode ser definido de diversas formas. No contexto deste trabalho um sistema será visto como um conjunto de componentes distintos que possuem múltiplas interações. Para que haja a interação entre os componentes de um dado sistema, é necessário que exista um sincronismo nas atividades em execução. Assim, no projeto de sistemas, a análise da comunicação e da sincronização deve ser criteriosa, pois, durante uma comunicação, a transferência de informações de um componente para outro somente se estabelecerá após a sincronização das atividades existentes. Essa necessidade de sincronismo pode fazer com que um componente fique no estado de espera, aguardando a sincronização com o outro. Durante a elaboração do projeto, um outro aspecto que deve ser estudado é a possibilidade da ocorrência de um 'deadlock' entre os componentes, pois um dado componente pode necessitar de uma informação proveniente de um outro e esse, por sua vez, somente pode fornecer tal informação após receber uma comunicação do primeiro.

É necessária a elaboração de estudos detalhados para evitar a implementação de sistemas que não satisfaçam as especificações ou evitar a implementação de sistemas cujas especificações são inadequadas.

2.1 - Modelamento de sistemas

O estudo de um sistema pode ser realizado por intermédio de diversas técnicas. Em alguns casos, há a possibilidade de se efetuar experimentos no próprio sistema. Entretanto, em muitos outros existe um interesse em se obter informações sobre o sistema antes de sua construção. Nesses casos, uma solução que pode ser as vezes adotada é a construção de protótipos. Todavia, tal solução apresenta alguns aspectos negativos, pois, além de serem excessivamente caros e despenderem um elevado tempo de trabalho, os protótipos nem sempre podem ser construídos, visto que há sistemas que devido às suas naturezas não podem ser representados por intermédio dessa técnica. A solução usualmente escolhida é efetuar os estudos em sistemas por intermédio de modelos.

O modelo pode ser visto como um conjunto de informações que retrata o comportamento de um sistema ou parte desse. Dessa forma, além de representar um sistema, é atribuída ao modelamento a capacidade de simplificação do mesmo.

Deve-se salientar que a utilização das técnicas de modelamento é dificultada quando o sistema a ser modelado possui atividades concorrentes implicando na necessidade do estabelecimento de sincronismo entre as atividades. Associado a esse fato, observa-se a existência de um indeterminismo na computação paralela, devido ao processamento concorrente dos

passos dos diversos processos. Esse indeterminismo é refletido no resultado final do processamento, uma vez que esse depende do instante de iniciação e terminação de cada processo.

Neste trabalho, teve-se como objetivo o desenvolvimento de um ambiente de simulação que tratasse sistemas que apresentam características de concorrência e/ou paralelismo. Para tanto, buscou-se na literatura ferramentas de modelamento que se adaptassem a essas necessidades.

2.2 - Ferramentas de modelamento

Na literatura, encontram-se descritas diversas ferramentas que estão sendo utilizadas para o modelamento de sistemas que apresentam características de concorrência e/ou paralelismo.

Neste item, é efetuada a análise da Rede de Petri /20/ e o Graph Model of Behavior - GMB /29/, uma vez que essas ferramentas possuem características adequadas para o modelamento de tais sistemas.

2.2.1 - Características da Rede de Petri

A rede de Petri começou a ser desenvolvida por Carl Adam Petri /20/, sendo que seu uso foi gradativamente aprimorado no decorrer do tempo. O estudo da rede de Petri foi desenvolvido em duas direções: estudo da teoria pura e estudo da teoria aplicada da rede de Petri.

A teoria aplicada está dirigida para o modelamento, análise e

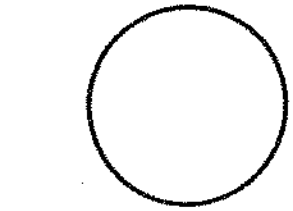
obtenção de dados de sistemas. Por outro lado, a teoria pura se preocupa com o desenvolvimento de ferramentas básicas, técnicas e conceitos necessários para a aplicação.

2.2.1.1 - Estrutura da rede de Petri

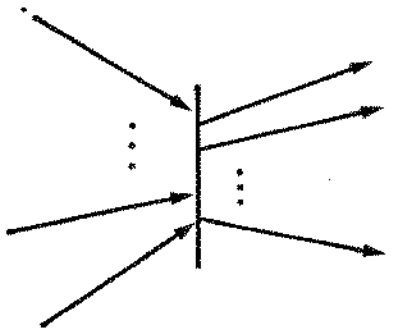
Uma rede de Petri é composta de quatro partes: um conjunto de lugares 'P', um conjunto de transições 'T', uma função de entrada 'I' e uma função de saída 'O'.

A definição teórica da rede é, sem dúvida, uma definição mais formal. Entretanto, este estudo é limitado ao tratamento da representação gráfica da estrutura da rede de Petri, pois, dessa forma, pode-se ilustrar os conceitos da teoria mais facilmente.

O grafo da rede de Petri é composto, em parte, pelos elementos primitivos ilustrados na figura 2.2.1.1. Esses elementos primitivos são interligados no grafo por intermédio de arcos. Tais interligações estabelecem o sentido em que a rede deverá ser executada.



lugar;



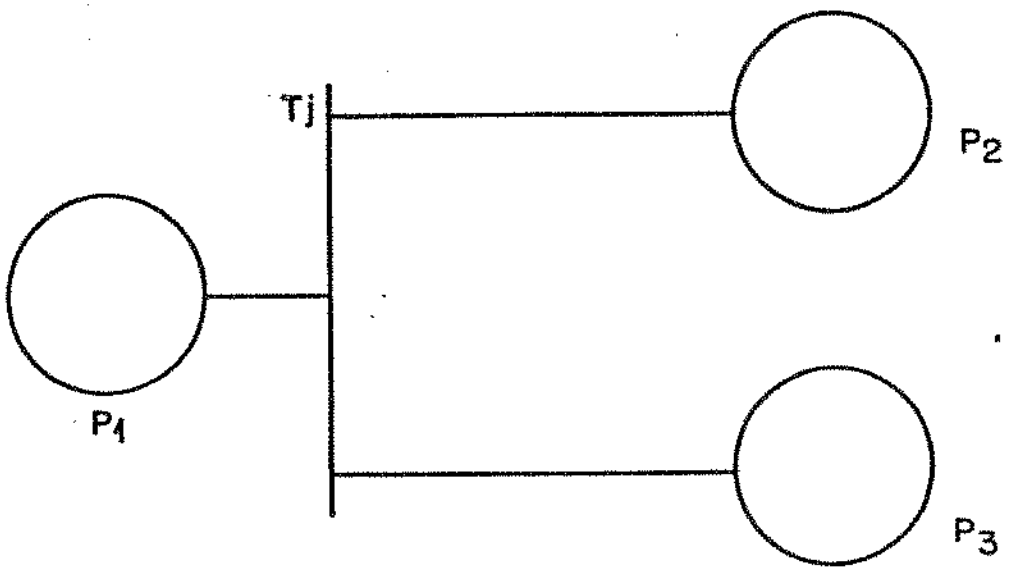
transição.

FIGURA 2.2.1.1

Elementos primitivos da rede de Petri

O fato de um arco ser proveniente de um lugar 'P1' e estar conectado a uma transição 'Tj' determina que o lugar 'P1' é entrada da transição 'Tj'. Por outro lado, um lugar de saída é definido pela existência de um arco que interliga uma transição a um lugar. A figura (2.2.1.2) ilustra a definição.

Um outro elemento primitivo associado ao grafo da rede de Petri é denominado marca, representado no grafo por um ponto. As marcas são responsáveis pela execução da rede, determinando a evolução do modelo.



Onde: P1 - lugar de entrada da transição Tj;

P2 e P3 - lugares de saídas da transição Tj.

INTERLIGAÇÃO DOS ELEMENTOS DA REDE DE PETRI
FIGURA 2.2.1.2

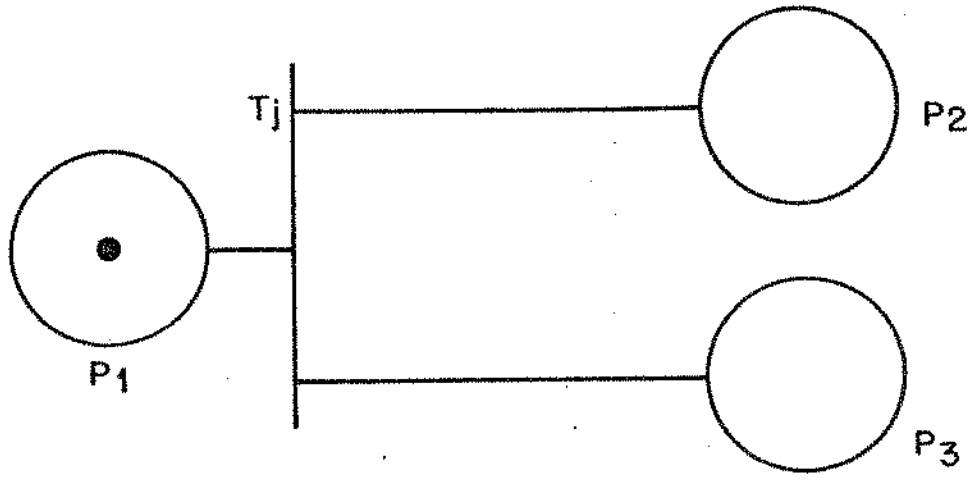


FIGURA 2.2.1.3

REPRESENTAÇÃO DA MARCA NO GRAFO

Para que ocorra a execução da rede, torna-se necessário que as transições sejam disparadas. Para tanto, os lugares de entrada de uma determinada transição devem conter tantas marcas quanto for o número de arcos que interligam esse lugar à transição. Uma vez disparada uma transição, os seus lugares de saída recebem um número de marcas igual ao número de arcos que interligam a transição ao lugar.

Observando-se o exemplo da figura (2.2.1.3), nota-se que a marca existente em 'P1' possibilita o disparo da transição 'T1', o que implica que marcas irão aparecer em 'P2' e 'P3'.

Maiores detalhes relativamente à rede de Petri podem ser encontrados na referência /20/.

2.2.2 - Características do GMB

O modelo Graph Model of Behavior -GMB- foi desenvolvido na Universidade da California, Los Angeles -UCLA. Esse modelo é composto de três domínios: domínio de controle, domínio de dados e domínio da interpretação que, interrelacionados, descrevem o comportamento de um sistema.

2.2.2.1 - Domínio de controle

O domínio de controle, responsável pela descrição do fluxo de controle, consiste de nós que representam eventos e arcos que descrevem a precedência no relacionamento entre eventos.

Cada nó possui uma expressão lógica de entrada que dita as condições sobre as quais o nó será ativado. Um "OR" (+) em uma expressão lógica de entrada possibilita a ativação do nó, uma vez que um dos arcos relacionados tenha recebido uma marca para efetuar a iniciação. Um "AND" (*) em uma expressão lógica de entrada obriga a que todos os arcos relacionados já tenham recebido uma marca para que o nó seja ativado.

Cada nó possui uma expressão lógica de saída que dita os arcos através dos quais os nós colocam marcas após sua terminação. Um "OR" implica que pelo menos um dos arcos deverá receber marca. Um "AND" implica que todos os arcos relacionados devem receber marca. Observa-se que os arcos pertencentes a expressões lógicas de entrada e/ou de saída de diversos nós são denominados arcos complexos. Os arcos de entrada e os arcos de saída podem ser relacionados tanto por expressões do tipo "OR" quanto "AND".

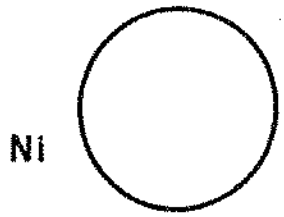
Nesse contexto, um grafo de controle é definido como uma descrição estática das sequências de controle possíveis, sendo que o estado do controle é representado pela distribuição de marcas nos arcos do grafo e, havendo a exclusão mútua de eventos, ela será modelada por intermédio da utilização dos arcos complexos.

Quando um nó é ativado, as marcas que o habilitam são eliminadas e, após o término da execução do processador

controlado associado, novas marcas são criadas e colocadas nos arcos de saída. Esse fluxo de marcas estabelece uma representação dinâmica do comportamento do sistema. Observa-se que somente após o término da ativação de um dado nó é que esse poderá ser novamente ativado.

Uma outra importante característica do grafo de controle é o fato de não poder ser modelado permitindo má terminação, isto é, após o término da execução de um grafo, as marcas existentes nos arcos de controle devem estar distribuídas de forma a possibilitar uma nova execução.

A figura 2.2.2.1 ilustra os elementos primitivos do grafo de controle.



nó de controle;

Ai

arco de controle;

● pi

marca.

FIGURA 2.2.2.I

Elementos primitivos do grafo de controle

2.2.2.2 - Domínio dos dados

O domínio dos dados descreve os pontos de transformação dos dados, sendo que alguns são vinculados ao grafo de controle.

O grafo de dados é composto por quatro elementos primitivos:

a - Armazenador de Dados;

b - Processador Controlado;

c - Processador Não Controlado;

d - Arco de Dados.

Os armazenadores de dados, como o próprio nome sugere, são utilizados para o armazenamento dos dados que estão sendo manipulados.

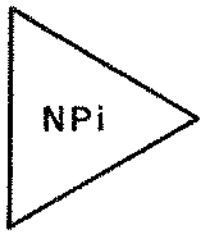
Os processadores controlados são transformadores de dados, cuja ativação depende da iniciação do nó de controle associado. A finalização da execução do processador controlado causa a respectiva terminação do nó de controle associado.

Os processadores não controlados são transformadores de dados cujo controle de ativação e terminação independe do grafo de controle, ficando estes vinculados às alterações dos conteúdos dos armazenadores associados a sua entrada.

Os arcos de dados constituem as ligações entre os processadores e armazenadores existentes no grafo de dados. Tais ligações estabelecem os caminhos possíveis para o fluxo de dados.

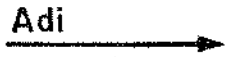
A figura 2.2.2.2 ilustra a representação gráfica dos elementos primitivos do grafo de dados.

processador controlado;



processador não controlado;

armazenador;



arco de dados.

FIGURA 2.2.2.2

Elementos primitivos do grafo de dados

2.2.2.3 - Domínio da Interpretação

A interpretação é utilizada para definir os formatos de dados armazenados e transportados pelos arcos de dados, assim como na definição das transformações efetuadas pelos processadores sobre os dados.

O domínio da interpretação é constituído por diversas primitivas que, combinadas, geram o contexto global das definições e transformações dos dados. A linguagem na qual essas primitivas são definidas não é especificada pelo modelo. Utilizando-se uma linguagem de definição que possa vir a ser executada pelo processador físico facilitar-se-á a concepção final do projeto.

Com o intuito de se obter um modelo com características mais adequadas às aplicações em sistemas de controle de processos em tempo real, foram propostas por DE MARTINO /07/ alterações no modelo GMB, criando-se com isso um modelo denominado GMB*.

2.2.3 - Características do GMB*

O GMB* é uma extensão do GMB. As modificações propostas por DE MARTINO /07/ restrigem-se a:

- inserção de prioridades nas marcas dos arcos do grafo de controle;
- permissão da marcação dos arcos de entrada do grafo

de controle e da troca dos dados através dos arcos de dados externos, durante a execução do modelo:

- eliminação do processador não controlado do grafo de dados.

As figuras 2.2.3.1 e 2.2.3.2 ilustram uma aplicação, utilizando-se o modelo GMB*. Observa-se que o grafo de controle é composto de quatro nós e oito arcos de controle, sendo que o grafo de dados possui quatro processadores controlados, dois arcos de dados e um armazenador. No estado inicial do grafo de controle, os arcos A1 e A5 estão marcados e o armazenador do grafo de dados se encontra vazio.

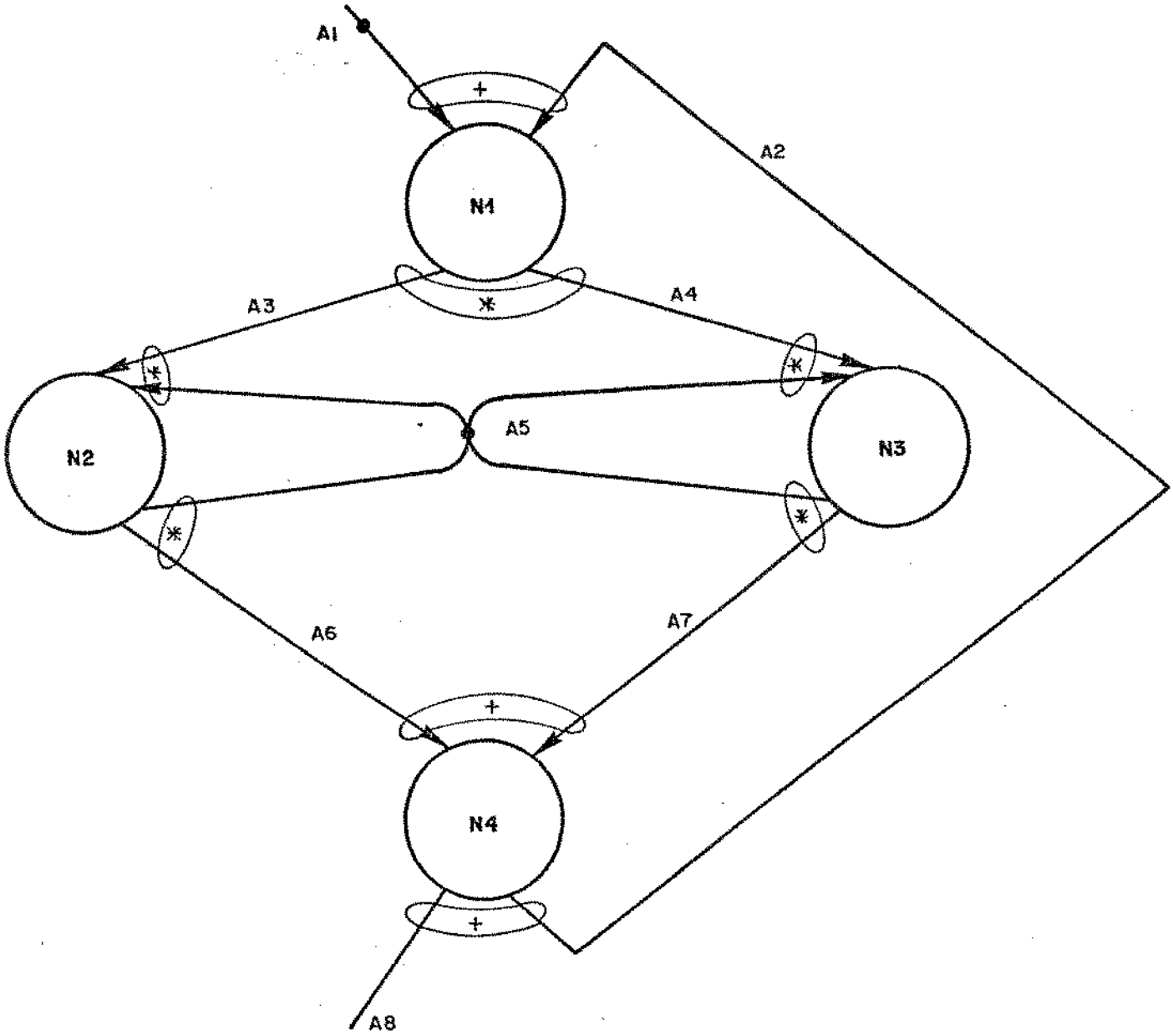


FIGURA 2.2.3.1
GRAFO DE CONTROLE

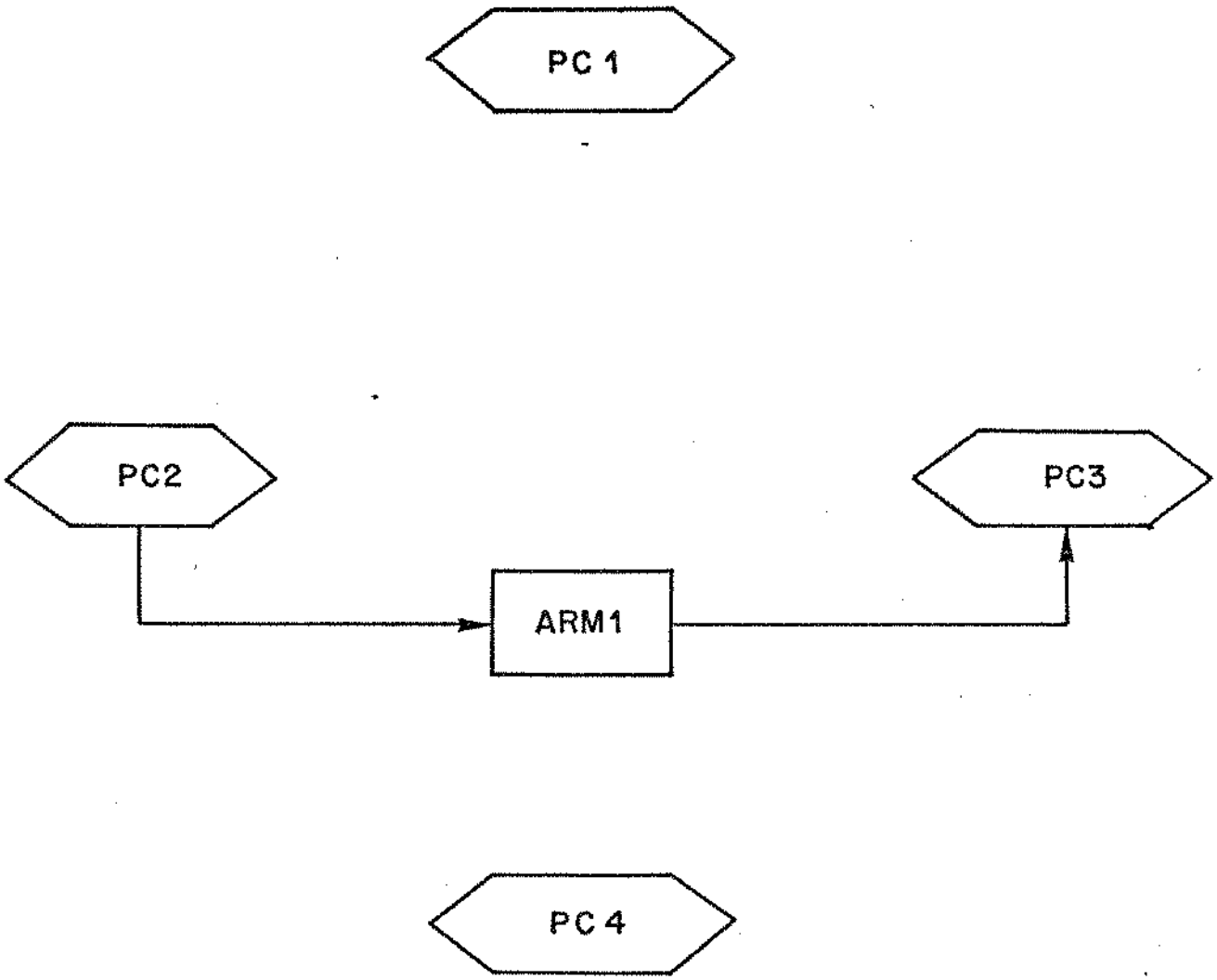


FIGURA 2.2.3.2
GRAFO DE DADOS

2.3 - Aspectos da Simulação

O desenvolvimento de projetos com a utilização de técnicas de modelamento, análise e simulação, é geralmente composto pelas três etapas ilustradas na figura (2.3.1). Para o desenvolvimento de uma certa aplicação, faz-se o seu modelamento através da utilização de ferramentas tais como as discutidas nos itens anteriores. Obtido o modelo, passa-se para a fase de análise, objetivando a observação de seu comportamento. Caso ocorra alguma anomalia, efetuam-se as alterações pertinentes no modelo e reinicia-se o processo de análise. Esse procedimento terá sua execução repetida até que o modelo se apresente correto.

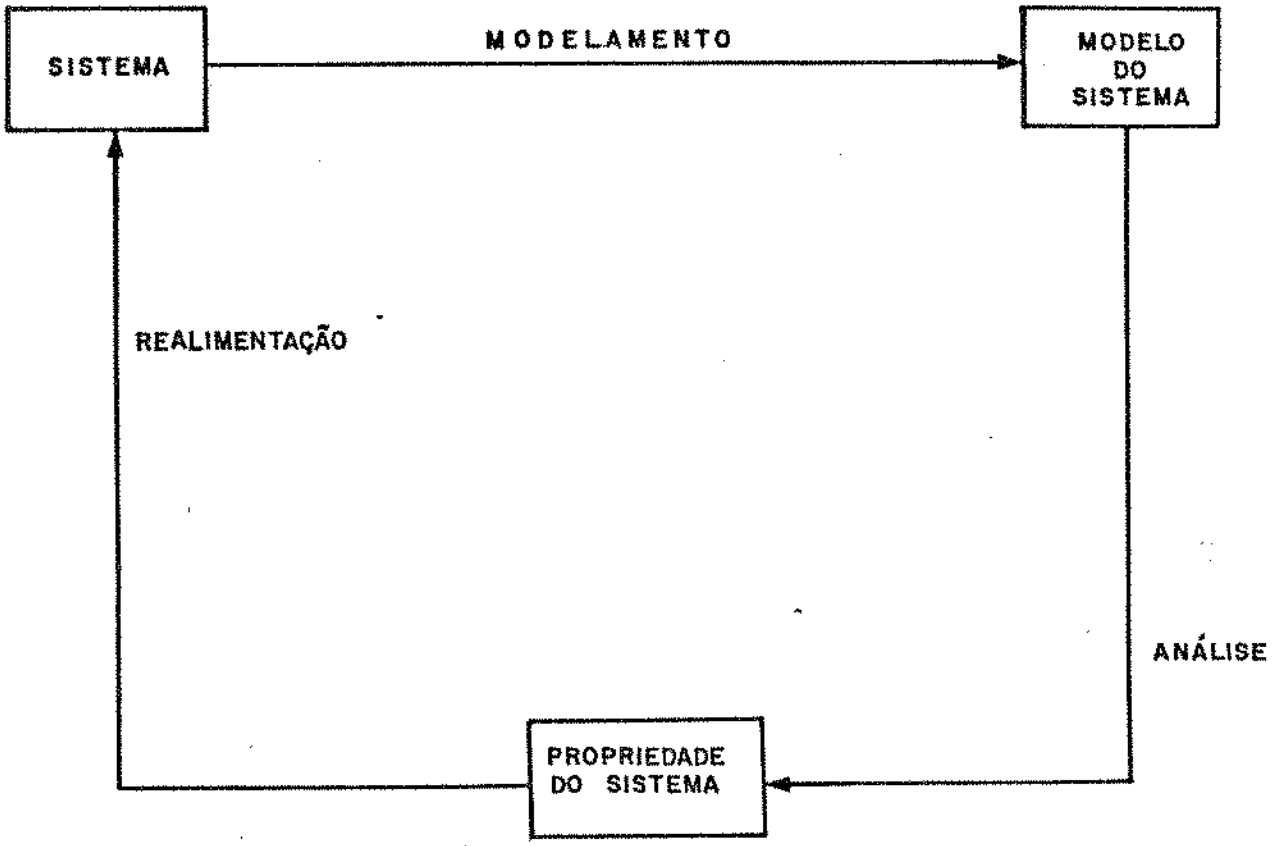


FIGURA 2.3.1
ETAPAS DO DESENVOLVIMENTO DE UM PROJETO

2.3.1 Técnicas de análise e simulação

Para a obtenção da análise de um modelo matemático, pode-se, por vezes, utilizar técnicas analíticas. Entretanto, quando a utilização dessas técnicas não é possível ou desejável, é necessário empregar métodos computacionais para a obtenção das informações referentes ao modelo. Sobre esse aspecto, acrescente-se que a principal diferença entre um método analítico e um método computacional consiste no fato de que o método analítico, ao ser empregado, gera soluções genéricas, enquanto a utilização de métodos computacionais produz soluções restritas.

Por intermédio de uma análise passo a passo, pode-se obter a solução de um conjunto de condições que foram previamente definidas antes da execução de um determinado passo. A associação das diversas soluções obtidas a cada passo se traduz no comportamento global do modelo. Com isso, a utilização desse método possibilita a obtenção, tanto de um estudo geral do comportamento do modelo, quanto de um estudo objetivando analisar as suas características específicas.

Usualmente, os métodos computacionais que manipulam modelos matemáticos dinâmicos recebem a denominação de simulação. No contexto deste trabalho, a simulação é vista como uma técnica que visa ao estudo de um sistema e à solução de eventuais anomalias através da observação da performance de um modelo dinâmico.

Baseado nesse conceito de simulação, desenvolveu-se um

simulador que se encontra descrito mais detalhadamente no capítulo posterior. A concepção dessa ferramenta é baseada no GMB* /07/, pois o GMB* é uma ferramenta de modelamento que possui a propriedade de modelar diversos sistemas, especialmente os sistemas que apresentam características de concorrência e/ou paralelismo entre processos.

Baseado no modelo GMB*, construiu-se o simulador visando à obtenção de uma ferramenta de análise dinâmica que viesse auxiliar o desenvolvimento de projetos de sistemas.

Através da utilização do simulador, o usuário analisa o comportamento do sistema modelado obtendo, dessa maneira, informações, passo a passo, do seu comportamento. Ocorrendo alguma anomalia, o usuário efetua as alterações pertinentes no modelo e reinicia a simulação.

Conclui-se que a utilização do simulador se torna parte importante no desenvolvimento de um projeto, visto que a detecção de anomalias através de uma análise dinâmica possibilita ao usuário concluir um projeto corretamente. Acrescenta-se que a obtenção de um projeto livre de erros e de falhas possibilita a diminuição do tempo e dos custos de implentação, pois a correção de erros durante a implementação do projeto é uma atividade bastante dispendiosa e, por vezes, excessivamente demorada.

3 - O MEIO DE SIMULAÇÃO

O simulador do GMB* desenvolvido neste trabalho é uma ferramenta de simulação de eventos discretos. Sua concepção está baseada em uma estrutura modular, implementada com a utilização da linguagem de programação Pascal.

A característica de modularidade, associada às técnicas utilizadas no projeto de cada módulo do simulador, cria um ambiente flexível de simulação. Tal ambiente é capaz de simular modelos de diversas dimensões, desde que o microcomputador possua memória suficiente para manipulá-los. O usuário, de posse do aplicativo modelado em GMB* /07/, insere interativamente os diversos domínios através dos módulos de definição de estrutura disponíveis no simulador.

3.1 Utilização do Simulador GMB*

Para a utilização do simulador, o usuário dispõe de um conjunto de variáveis que permite definir os parâmetros da simulação. Para a análise da simulação, o usuário utiliza os comandos disponíveis no simulador. A análise poderá ser efetuada a cada passo de simulação ou ao final de um número de passos predeterminados.

3.1.1 Variáveis de Simulação

O simulador do GMB* possui quatro variáveis de simulação, definidas pelo usuário. A utilização de tais variáveis visa oferecer a possibilidade de definir os passos iniciais e finais da simulação, definir o tipo de simulação e o dispositivo de armazenamento (drive) em que as estruturas serão armazenadas.

- Variável de definição do modo de simulação:

Com o objetivo de fornecer ao usuário uma maior flexibilidade na simulação de um determinado modelo, foram criados três modos distintos de simulação. No modo "A" ou "a", o simulador interrompe a simulação enviando mensagens ao usuário, oferecendo-lhe a possibilidade de retornar ao módulo de interface de operação toda vez que este encontrar uma inconsistência no grafo de dados ou no grafo de controle. No modo "B" ou "b", o simulador apenas emite mensagens ao usuário, continuando a execução da simulação. Já no modo "C" ou "c", o simulador continua a simulação sem emitir mensagens.

- Variável de definição do dispositivo de armazenamento:

Devido a utilização de um microcomputador para a

implementação do simulador do GMB*, associada ao fato de o simulador possibilitar a simulação de modelos de diversas dimensões, optou-se pela criação de uma variável de definição do dispositivo de armazenamento. O objetivo da criação dessa variável é o de permitir que as estruturas do modelo do aplicativo sejam armazenadas em dispositivos de armazenamento de massa (disketes) independentes do dispositivo em que o simulador se encontra.

- Variável de definição de início de simulação:

O objetivo dessa variável é fornecer ao simulador o passo inicial de simulação. Com base no conteúdo da variável, o simulador inicia a simulação considerando que nenhuma iteração foi executada (passo 0), ou inicia a simulação tendo em vista que iterações ocorreram anteriormente (passo diferente de 0) e, por consequencia, alguns nós do grafo de controle podem estar no estado habilitado ou mesmo no estado ativo, executando a interpretação do processador.

- Variável de definição de final de simulação.

O conteúdo dessa variável especifica o passo final de simulação, definindo o número de iterações que o simulador deve executar antes de retornar ao módulo de interface de operação.

3.1.2 Comandos do Simulador

Durante a execução da simulação, a estrutura do vetor de marcas e o conteúdo dos armazenadores são alterados. Associada a essas alterações está a criação ou a alteração das estruturas dos nós habilitados, nós ativos e máquina de "tokens". As alterações realizadas nessas estruturas retratam o comportamento do modelo no decorrer da simulação. Para permitir ao usuário analisar tal comportamento, o simulador é dotado de comandos que possibilitam a verificação das estruturas a seguir relacionadas, listando-as na tela do microcomputador ou imprimindo-as através da utilização de uma impressora.

- Grafo de controle;
- Vetor de marcas;
- Grafo de dados;
- Estrutura dos armazenadores;
- Interpretação;
- Máquina de tokens;
- Nós habilitados;
- Nós ativos;
- Variáveis de simulação.

Através da utilização desses comandos, o usuário analisa a evolução dinâmica do modelo simulado.

3.2 Funcionamento do simulador GMB*

O simulador GMB* começa o seu processamento observando os estados iniciais do grafo de controle e dos armazenadores, evoluindo em seu processamento, passo a passo, e efetuando as transições ditadas pelo modelo GMB*. O termo "passo" está relacionado com o processamento executado no intervalo entre a eliminação de uma marca no arco de entrada de um determinado nó e o surgimento de uma nova marca em um arco de saída desse nó.

A evolução da simulação é regida pela ativação e terminação de um nó de controle. Tal fato altera o fluxo de controle e/ou os estados do modelo, causando a ativação de novos nós.

3.2.1 Ativação e terminação de um nó de controle

Visando possibilitar ao simulador GMB* tratar eventos paralelos situados em estações distintas, definiu-se, ao nível de projeto, que apenas um nó, por estação, é ativado em um determinado passo de simulação. Para tanto, criou-se o estado habilitado que é um estado do grafo de controle intermediário ao estado passivo e ao estado ativo.

O simulador, por intermédio da execução do módulo responsável pelo tratamento do grafo de controle, ao detectar que determinados nós do grafo possuem suas entradas lógicas

satisfeitas, insere-os na estrutura de nós habilitados. A execução desse procedimento garante que todos os nós possuidores de entradas lógicas satisfeitas podem vir a ser executados.

Para a escolha do nó que passará para o estado ativo, adotou-se o critério de prioridades ditado pelo modelo GMB*. Com isso, apenas um nó passa para o estado ativo, permanecendo os outros no estado habilitado. Após a ativação de um nó, as marcas de maior prioridade são removidas dos arcos de entrada responsáveis pela ativação do nó.

A ativação de um nó do grafo de controle implica a execução da interpretação do processador associado no grafo de dados. A execução da interpretação pode realizar procedimentos tais como:

- Escrita e/ou leitura, destrutiva ou não, em armazenadores do tipo fixo ou variável;
- Atraso associado à execução de primitivas;
- Execução de primitivas condicionais;
- Marcação dos arcos de saída do nó de controle ao término da interpretação.

A marcação do arco de saída caracteriza a terminação do nó de controle, findando a interpretação do processador associado.

3.3 Estrutura do Simulador GMB*

O desenvolvimento do projeto do simulador GMB* teve como objetivo conceber uma ferramenta de simulação para ser processada em um sistema operacional CP/M, tendo como suporte um microcomputador de oito "bits". A escolha do sistema operacional e a escolha do microcomputador, baseou-se na perspectiva de se poder ter uma ferramenta de simulação que oferecesse um alto grau de utilização e uma alta portabilidade. Posteriormente, com a difusão dos computadores pessoais compatíveis com o IBM-PC, optou-se por transportar o simulador para um ambiente baseado no sistemas operacional MS-DOS. Devido a implementação altamente modularizada e o uso de construções da linguagem Pascal padrão o transporte foi bastante facilitado, tendo sido realizado rapidamente.

Analisando-se as características do modelo GMB* /07/, observou-se que ele permite o modelamento de aplicativos de diversas dimensões. Assim, procurou-se direcionar o projeto do simulador GMB* no intuito de se obter um ambiente de simulação que não oferecesse restrições quanto à dimensão dos modelos simulados.

Ao se projetar a estrutura dos programas do simulador GMB*, observou-se que esta ocuparia um grande volume de memória, pois a

manipulação das estruturas do modelo GMB* e o tratamento das características de projeto citadas acima, requerem uma quantidade relevante de programas. Tal fato se tornou conflitante com a especificação do microcomputador de oito 'bits', uma vez que o seu volume de memória disponível é insuficiente para o processamento de todos os programas juntos. Buscando contornar essas dificuldades, adotou-se uma estrutura modular através da divisão do simulador em módulos independentes. Com isso, reduziu-se consideravelmente o volume de memória necessário para o processamento de cada módulo, viabilizando a utilização de um microcomputador de oito "bits".

Para a implementação dos módulos, adotou-se a linguagem de programação Pascal, pois, uma vez que é uma linguagem estruturada, facilita a implementação dos diversos módulos. Cada módulo do simulador manipula estruturas de dados na forma de listas ligadas, permitindo, portanto, tamanhos variados para as estruturas de dados que definem o modelo. Sendo os módulos projetados para serem processados separadamente, de modo que apenas um módulo ocupará a memória principal do microcomputador em um determinado instante de tempo, definiu-se que o relacionamento entre os diversos módulos seria efetuado através de arquivos de dados residentes em memória de massa. Tal artifício, além de possibilitar a implementação da estrutura, facilita efetuar possíveis alterações em um módulo sem provocar repercussão em outros, uma vez que estes possuem funções

específicas e todos os parâmetros são passados explicitamente por meio da utilização dos arquivos de dados.

Para uma melhor visualização da estrutura do simulador, utilizou-se o próprio GMB* para obtenção do seu modelo. O modelo obtido será utilizado para demonstrar o funcionamento dos diversos módulos, bem como o relacionamento entre os mesmos.

A figura 3.1 representa o grafo de controle do modelo do simulador e a figura 3.2, o grafo de dados associado.

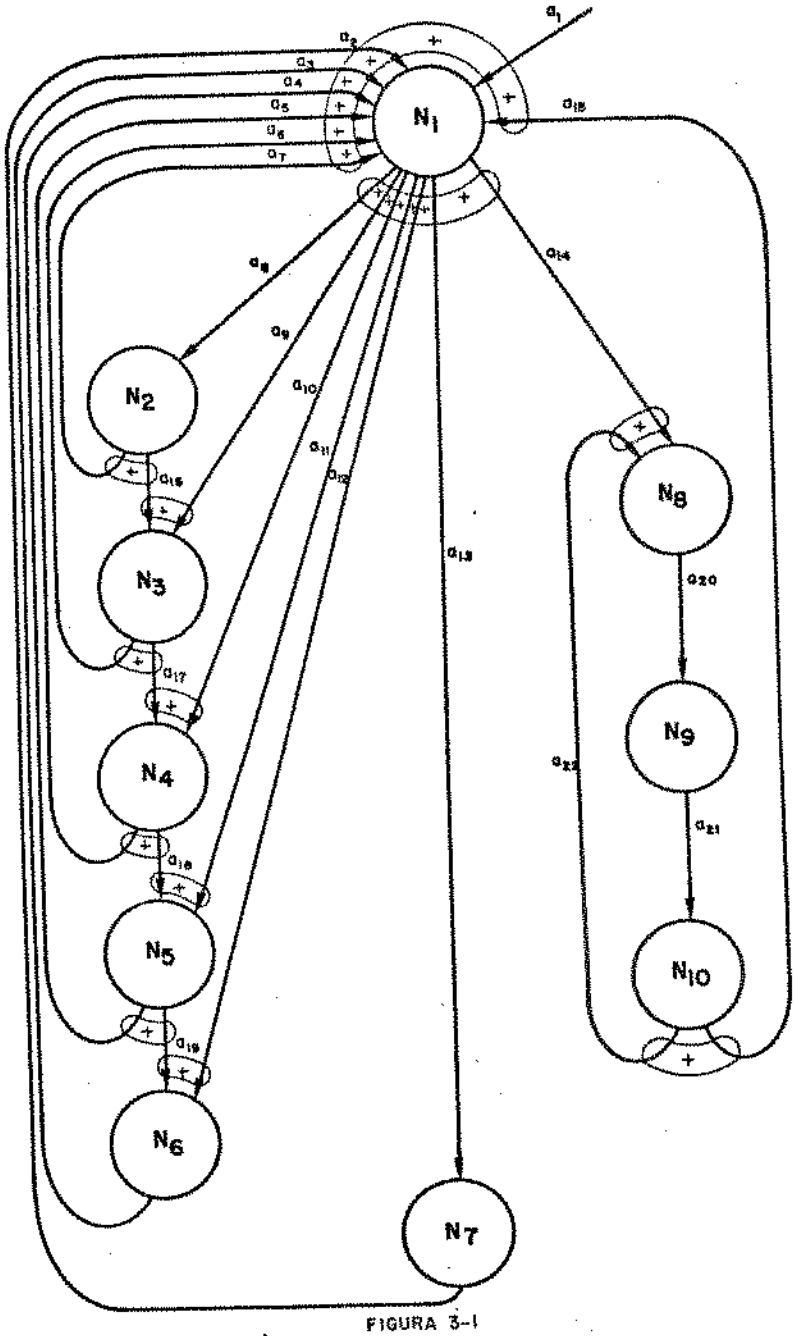


FIGURA 3-1
GRAFO DE CONTROLE

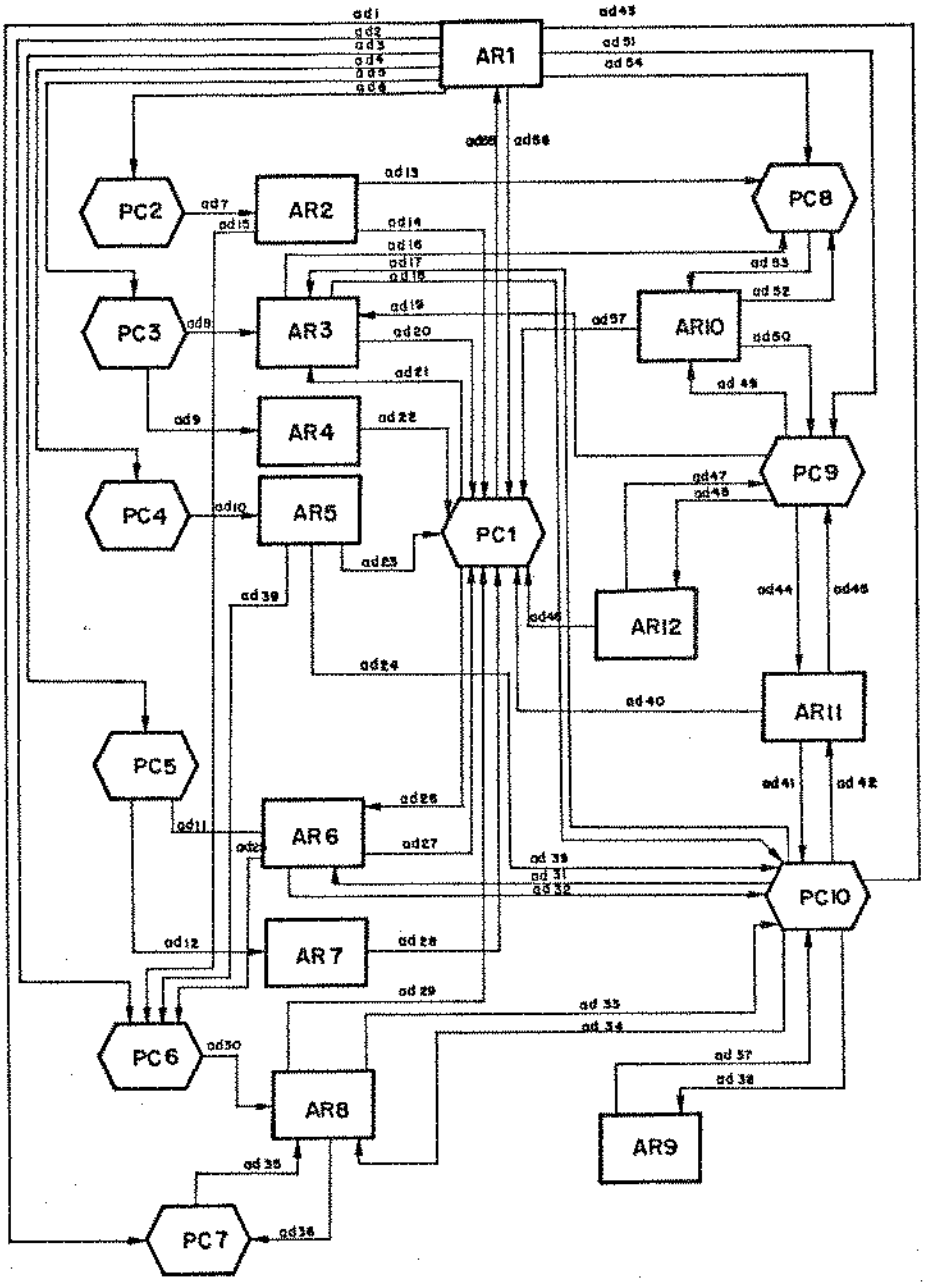


FIGURA 3-2
GRAFO DE DADOS

3.3.1 Domínio de Controle do Simulador GMB*

O grafo de controle apresentado na figura 3.1 representa a estrutura de controle que compõe o simulador GMB*. Observe-se que ele possui dez nós de controle e vinte e dois arcos responsáveis pela representação do relacionamento entre os nós.

3.3.1.1 Descrição dos Nós do Grafo de Controle

A cada nó do grafo de controle se encontra associado um módulo pertencente ao simulador GMB*. Os módulos estão implementados, conforme mencionado anteriormente, por programas utilizando a linguagem Pascal. Tem-se, a seguir, a descrição do relacionamento dos nós com os programas utilizados na implementação.

- nó N1 -- SIMGMB.COM -- módulo de interface de operação;
- nó N2 -- EST1.COM -- módulo de definição do grafo de controle;
- nó N3 -- POEMARCA.COM -- módulo de definição do vetor de marcas;
- nó N4 -- DDET1.COM -- módulo de definição do grafo de dados;
- nó N5 -- ARMAZ.COM -- módulo de definição da estrutura dos armazenadores;

- nó N8 -- INTERPRE.COM -- módulo de definição da interpretação;
- nó N7 -- REDEFINT.COM -- módulo de redefinição da estrutura da interpretação;
- nó N8 -- GCONTROL.COM -- módulo de tratamento da estrutura do grafo de controle;
- nó N9 -- TRAT.COM -- módulo de tratamento da estrutura dos nós habilitados;
- nó N10 -- TRATAINT.COM -- módulo de tratamento da estrutura da interpretação.

Analisando-se o grafo de controle observa-se que o fluxo de controle pode assumir diversos caminhos. A definição do caminho a ser seguido é efetuada pelo usuário através da interação direta com o simulador, ou através das variavéis que determinam o passo inicial e final de simulação.

3.3.2 Domínio dos Dados do Modelo

No grafo de dados apresentado na figura 3.2, os módulos que compõem o simulador são representados pelos processadores. Observa-se que os arquivos de dados que definem o relacionamento entre os módulos são representados no grafo pelos armazenadores aí existentes. Os arcos de dados são utilizados para representarem os tipos de acessos que os diversos módulos efetuam

nos arquivos de dados, isto é, no grafo de dados há módulos que efetuam leituras e escritas nos arquivos e há módulos que somente efetuam leituras ou escritas.

O grafo de dados é composto de vinte e quatro processadores, doze armazenadores e cinquenta e sete arcos de dados. Cada processador se encontra associado a um nó do grafo de controle, de onde se tem as seguintes correspondências:

- PC1 -- N1;
- PC2 -- N2;
- PC3 -- N3;
- PC4 -- N4;
- PC5 -- N5;
- PC6 -- N6;
- PC7 -- N7;
- PC8 -- N8;
- PC9 -- N9;
- PC10 -- N10.

3.3.2.1 Especificação dos Armazenadores

Como já mencionado no item anterior, a utilização dos diversos arquivos de dados visa possibilitar a implementação modular do simulador do GMB*. As estruturas de dados definidas e geradas durante a simulação de um determinado aplicativo estão

representadas no grafo de dados pelos armazenadores. A seguir, encontram-se listados os armazenadores e as estruturas de dados por eles representadas.

- AR1 -- Estrutura da variáveis de simulação;
- AR2 -- Estrutura do grafo de controle;
- AR3 -- Estrutura do vetor de marcas;
- AR4 -- Estrututra de reserva do vetor de marcas;
- AR5 -- Estrutura do grafo de dados;
- AR6 -- Estrutura dos armazenadores;
- AR7 -- Estrutura de reserva dos armazenadores;
- AR8 -- Estrutura da interpretação;
- AR9 -- Estrutura das primitivas modificada;
- AR10 -- Estrutura dos nós habilitados;
- AR11 -- Estrutura dos nós ativos;
- AR12 -- Estrutura da máquina de tokens.

A implementação das estruturas é obtida através da utilização dos arquivos de dados, ou seja, uma dada estrutura pode se valer de um ou mais arquivos de dados para a sua implementação. A seguir encontram-se listados os armazenadores relacionados com os arquivos de dados que compõem as estruturas.

AR1	-	COND.SIM	--	Variáveis de simulação;
	-	EST1.GCT	--	Estrutura dos nós do grafo de controle;
AR2	-	ARCEXS.OUT	--	Estrutura dos arcos de saída dos nós;
	-	ARCEXE.IN	--	Estrutura dos arcos de entrada dos nós;
	-	MARK.OUT	--	Estrutura dos arcos do vetor de
AR3				marcas;
	-	PRI.MAR	--	Estrutura de marcas dos arcos do vetor;
	-	MARK.BAK	--	Estrutura de reserva dos arcos do
AR4				vetor;
	-	PRI.BAK	--	Estrutura de reserva das marcas;
	-	PROCONT.GDD	--	Estrutura dos processadores;
AR5	-	ADENT.GDD	--	Estrutura dos arcos de entrada dos processadores;
	-	ADSAI.GDD	--	Estrutura dos arcos de saída dos processadores;
	-	ARMAZEM.GDD	--	Estrutura dos armazenadores;
AR6	-	MENSAG.GDD	--	Estrutura das mensagens dos

armazenadores do tipo variável;

- ARMAZEM.BAK -- Estrutura de reserva dos armazenadores;

AR7

- MENSAG.BAK -- Estrutura de reserva das mensagens dos armazenadores do tipo variável;

- TERP.CAO -- Estrutura dos processadores da

AR8

interpretação;

- TIVA.CAO -- Estrutura das primitivas da interpretação;

AR9 - CUNHO.CAO -- Estrutura das primitivas da interpretação modificada;

- NOATIVO.ATV -- Estrutura dos nós habilitados;
- ARCATIVO.ATV -- Estrutura dos arcos de entrada dos

AR10

nós habilitados;

- MACATIVO.ATV -- Estrutura das marcas dos nós habilitados;

- NOEX.EXE -- Estrutura dos nós ativos;

AR11

- AREX.EXE -- Estrutura dos arcos de saída dos nós ativos;

- MAQUINA.TOK -- Estrutura dos arcos que compem a máquina de "tokens";

AR12

- PRIO.TOK -- Estrutura das marcas dos arcos da máquina de "tokens".

3.3.3 Interpretação dos Módulos que Constituem o Simulador GMB*

Através da interpretação associada a cada processador representado pelo grafo de dados da figura 3.2, descreve-se o funcionamento dos módulos do simulador GMB*.

PC1 - Interface de Operação

A criação da interface de operação objetivou dotar o simulador GMB* de uma interface "homem-máquina" eficiente e amigável. Através dessa interface, o usuário interage com o simulador definindo e manipulando as variáveis de simulação e as estruturas do modelo.

Para a obtenção da interação entre o usuário e o simulador, implementou-se a interface de operação, utilizando-se procedimentos de telas. Uma vez carregado o simulador no microcomputador, através da chamada do programa "SIMGMB.COM", a tela do microcomputador exibirá as seguintes opções:

0 - RETORNAR AO SISTEMA OPERACIONAL;

1 - DEFINIR AS VARIAVEIS DE SIMULAÇÃO;

2 - DEFINIR ESTRUTURAS;

3 - LISTAR ESTRUTURAS;

4 - EXECUTAR A SIMULAÇÃO.

1 - Definir as Variáveis de Simulação

Escolhendo-se a opção '1', é oferecido ao usuário a possibilidade de definir as variáveis de simulação do sistema. Após a definição das variáveis de simulação, a estrutura obtida é armazenada no arquivo de dados "COND.SIM". Findado o armazenamento dessa estrutura, é oferecido ao usuário a possibilidade de retornar ao procedimento de tela anterior.

2 - Definir Estruturas

Ao ser escolhida essa opção, o simulador executa a chamada de um novo procedimento de tela, passando esta a exibir as opções abaixo:

1 - ESTRUTURA DE CONTROLE;

2 - ESTRUTURA DE DADOS;

3 - ESTRUTURA DOS ARMAZENADORES;

4 - VETOR DE MARCAS;

5 - ESTRUTURA DA INTERPRETAÇÃO;

6 - CONDIÇÕES INICIAIS;

7 - REDEFINIR A ESTRUTURA DA INTERPRETAÇÃO;

8 - RETORNAR A TELA ANTERIOR.

As opções "1", "2", "3", "4", "5" e "7" terminam a execução do módulo de interface de operação, chamando os respectivos módulos para a definição das estruturas.

Ao ser utilizada a opção "6", o conteúdo da estrutura do vetor de marcas e o conteúdo da estrutura dos armazenadores são atualizados. Tal atualização objetiva o retorno ao estado inicial em ambas as estruturas. Esse recurso foi projetado visando possibilitar ao usuário o retorno às condições iniciais do modelo, sem a necessidade de redefinir as estruturas do vetor de marcas e do grafo de dados.

3 - Listar Estruturas

A opção de listar estruturas constitui o meio de acesso aos comandos de listagem do simulador GMB*. O usuário utiliza essa opção para listar as diversas estruturas do modelo simulado, o que permite a realização da análise do mesmo.

Ao ser selecionada a opção de listagem de estruturas, o conteúdo da tela do microcomputador é renovado, passando a exibir as seguintes opções:

0 - HABILITAR IMPRESSORA;

1 - GRAFO DE CONTROLE;

2 - GRAFO DE DADOS;

3 - CONTEÚDO DOS ARMAZENADORES;

4 - INTERPRETAÇÃO;

5 - MÁQUINA DE TOKENS;

6 - NÓS HABILITADOS;

7 - NÓS ATIVOS;

8 - VARIÁVEIS DE SIMULAÇÃO;

9 - RETORNAR À TELA ANTERIOR.

4 - Executar a Simulação

Por intermédio da escolha da opção "4", o usuário termina a execução do módulo de interface de operação, chamando o módulo de tratamento do grafo de controle e, consequentemente, dando início a um passo de simulação.

PG2 - Definição do Grafo de Controle

Esse módulo permite ao usuário criar o grafo de controle definindo o nome de cada nó, a estação onde será processado, os arcos de entrada e de saída e suas respectivas expressões lógicas. Acrescente-se que, havendo a necessidade de se definir uma expressão lógica do tipo **a1 * (a2 + a3)**, esta deverá ser expressa da seguinte forma: **a1 * a2 + a1 * a3**. Tal fato é

decorrente da necessidade de expressar de forma distributiva a expressão lógica "AND", quando esta estiver associada a arcos relacionados por expressões lógicas "OR".

Após a inserção de toda a estrutura faz-se a sua impressão na tela do microcomputador, sendo posteriormente armazenada nos arquivos de dados "EST1.GCT", "ARCEXS.OUT" e "ARCEXE.IN". Ao término da execução do módulo, é oferecida ao usuário a possibilidade de retornar ao módulo de interface de operação ou de chamar o módulo de definição do vetor de marcas. As listas ligadas existentes na figura 3.3 ilustram a forma de implementação da estrutura.

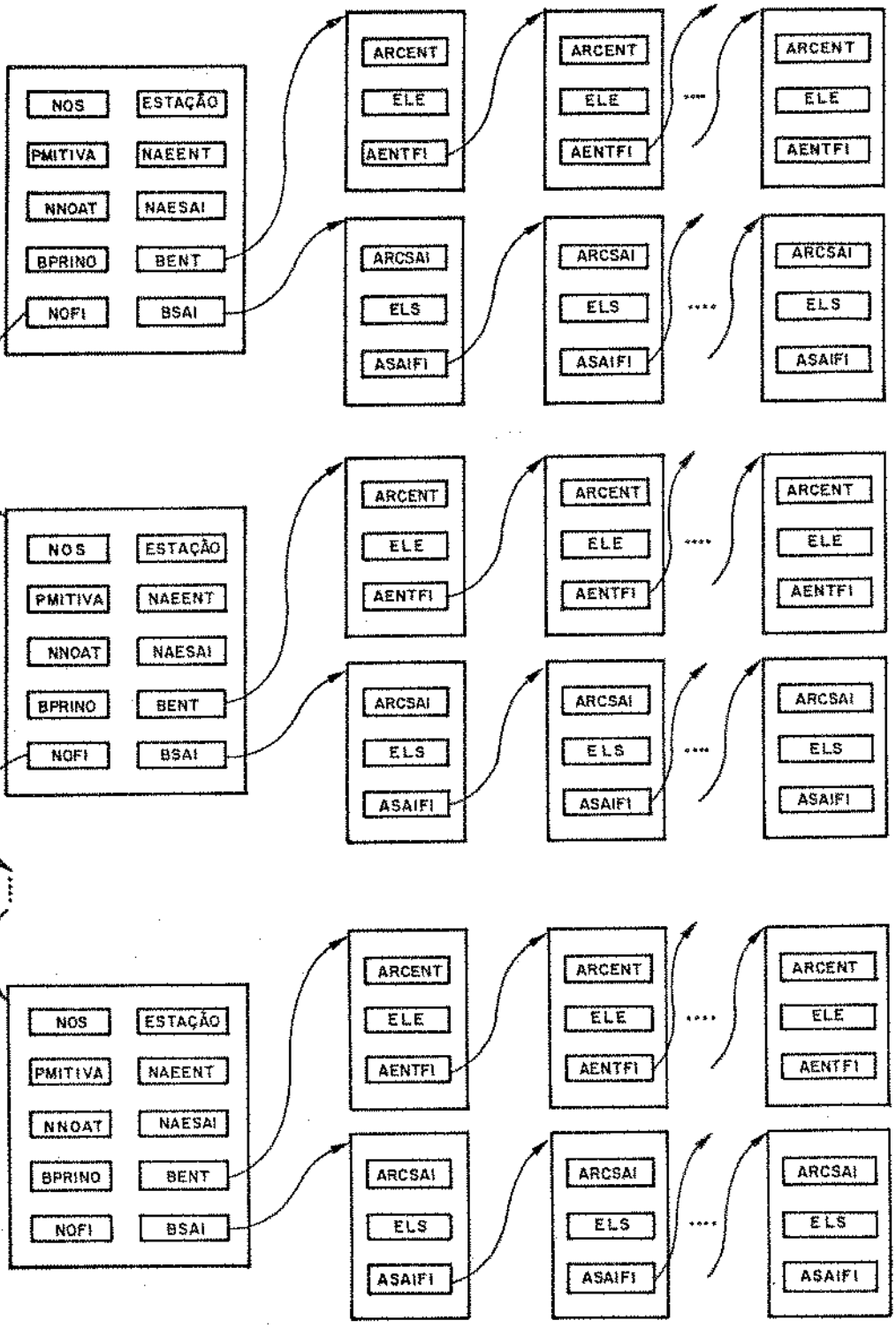


FIGURA 3.3
ESTRUTURA DO GRAFO DE CONTROLE

A figura 3.3 mostra as três listas ligadas que compõem a estrutura do grafo de controle. As listas são constituídas de 'records', sendo que a primeira armazena os nós e as outras armazenam, respectivamente, os arcos de entrada e os seus arcos de saída.

O 'record' da lista de nós é composto por dez campos que se encontram especificados abaixo:

Campo		Conteúdo
NOS	--	nome do nó;
ESTAÇÃO	--	nome da estação em que o nó se encontra;
NAEENT	--	número de arcos de entrada do nó;
NAESAI	--	número de arcos de saída do nó;
BENT	--	apontador para a lista de arcos de entrada;
BSAI	--	apontador para a lista de arcos de saída;
PMITIVA	--	primitiva da interpretação em execução;
NNOAT	--	número de passos de permanência do nó no estado habilitado;
BPRINO	--	apontador para a lista de marcas;
NOFI	--	apontador para o próximo nó da fila de nós.

Observe-se que os campos PMITIVA, NNOAT e BPRINO não estão sendo utilizados para a representação da estrutura do grafo de controle. A existência destes campos é necessária uma vez que a

lista de nós é utilizada para a obtenção da estrutura de nós habilitados e para obtenção da estrutura de nós ativos.

O 'record' da lista de arcos de entrada é composto por três campos, a saber:

Campo		Conteúdo
ARCENT	--	nome do arco de entrada;
ELE	--	expressão lógica de entrada associada ao arco;
AENTFI	--	apontador para o próximo 'record' da fila de arcos.

O 'record' da lista de arcos de saída é composto por três campos:

Campo		Conteúdo
ARCSAI	--	nome do arco de saída;
ELS	--	expressão lógica de saída associada ao arco;
ASAIFI	--	apontador para o próximo 'record' da fila de arcos.

PC3 - Definição do Vetor de Marcas

Esse módulo possibilita ao usuário definir os nomes dos arcos existentes no grafo de controle e suas respectivas marcas iniciais. Isto feito, a estrutura é impressa na tela do microcomputador, sendo posteriormente armazenada nos arquivos de dados "MARK.OUT" e "PRI.MAR". Objetivando a criação de uma estrutura de reserva, o simulador faz uma cópia da estrutura do vetor de marcas e a armazena nos arquivos de dados "MARK.BAK" e "PRI.BAK". Essa estrutura de reserva será utilizada pelo simulador para retornar a estrutura do vetor de marcas às condições iniciais, caso o usuário tenha feito a solicitação através do módulo de interface de operação.

Ao término da execução do módulo, é oferecida ao usuário a possibilidade de retornar ao módulo de interface de operação ou chamar o módulo de definição do grafo de dados. As listas ligadas existentes na figura 3.4 ilustram a forma de implementação da estrutura.

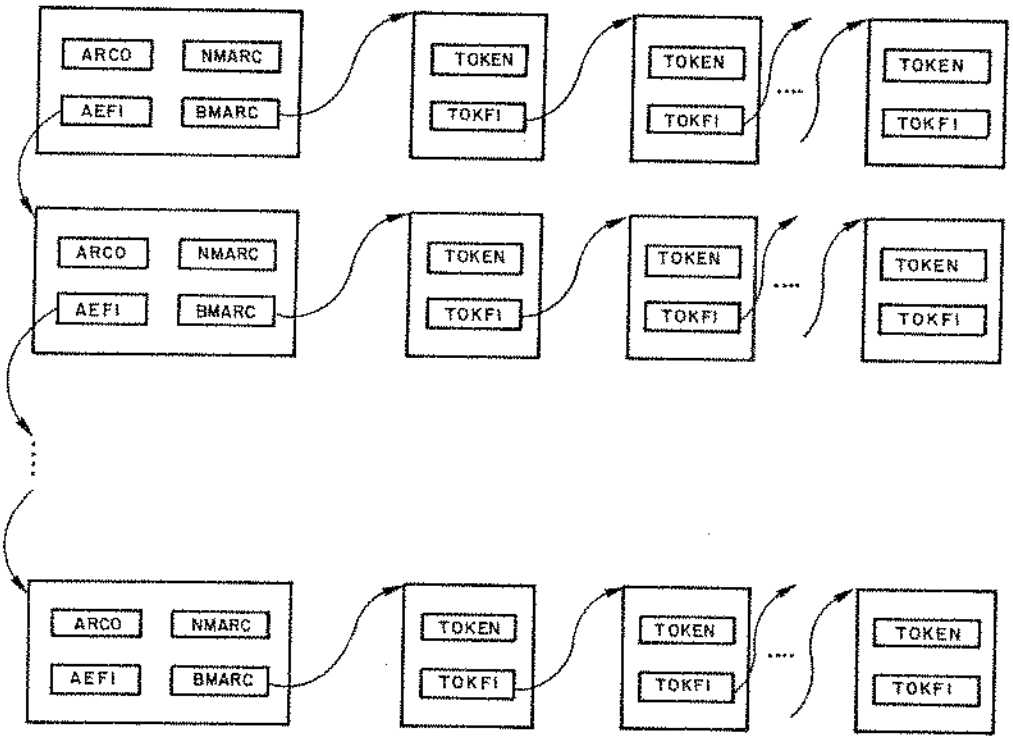


FIGURA 3.4
ESTRUTURA DO VETOR DE MARCAS

A figura 3.4 mostra as duas lista ligadas que compõem a estrutura do vetor de marcas. As listas são compostas por 'records', sendo que a primeira armazena o nome do arco e a segunda armazena as marcas neles existentes.

O 'record' da lista de arcos é composto por quatro campos, conforme segue:

Campo		Conteúdo
ARCO	--	nome do arco;
NMARC	--	número de marcas;
BMARC	--	apontador para a lista de marcas;
AEFI	--	apontador para o próximo 'record' da lista de arcos.

O 'record' da lista de marcas é composto por dois campos, a saber:

Campo		Conteúdo
TOKEN	--	prioridade da marca;
TOKFI	--	apontador para o próximo 'record' da fila de marcas.

PC4 - Definição do Grafo de Dados

Para a definição do domínio dos dados, é necessário fornecer ao simulador o relacionamento entre os processadores e os armazenadores. Na definição da estrutura dos processadores, o usuário deve fornecer, para cada processador, um nome, o nó do grafo de controle associado e os arcos de entrada e de saída. Para cada arco, deve-se definir o armazenador que a ele se encontra vinculado.

Após a inserção da estrutura, faz-se a sua impressão na tela do microcomputador, armazenando-a, posteriormente, nos arquivos de dados "PROCONT.GDD", "ADENT.GDD" e "ADSAI.GDD". Ao término da execução do módulo, é oferecido ao usuário a possibilidade de retornar ao módulo de interface de operação ou de chamar o módulo de definição da estrutura dos armazenadores. As listas ligadas existentes na figura 3.5 ilustram a forma de implementação da estrutura.

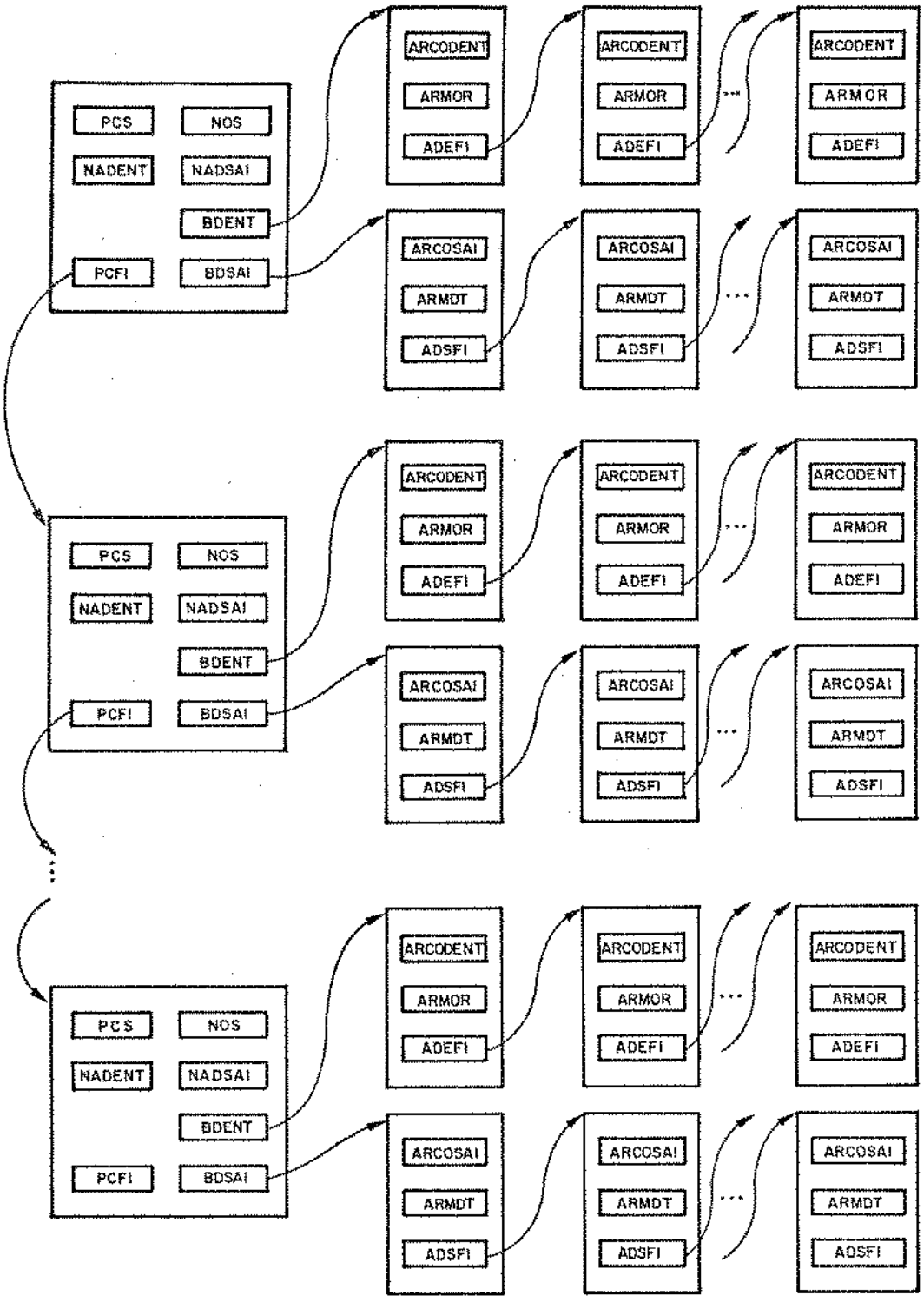


FIGURA 3.5

ESTRUTURA DO GRAFO DE DADOS

A figura 3.5 mostra as três listas ligadas que compõem a estrutura do grafo de dados. As listas são compostas por 'records', sendo que a primeira armazena os processadores e as outras armazenam, respectivamente, os seus arcos de entrada e os arcos de saída.

O 'record' da lista de processadores é composto por sete campos, assim especificados:

Campo		Conteúdo
PCS	--	nome do processador;
NOS	--	nome do nó cujo processador está associado;
NADENT	--	número de arcos de entrada;
NADSAI	--	número de arcos de saída;
BDENT	--	apontador para a lista de arcos de entrada;
BDSAI	--	apontador para a lista de arcos de saída;
PCFI	--	apontador para o próximo 'record' da lista de processadores.

O 'record' da lista de arcos de entrada é composto por três campos, conforme segue:

Campo		Conteúdo
ARCODENT	--	nome do arco;
ARMOR	--	nome do armazenador associado ao arco;

ADEFI -- apontador para o próximo 'record' da lista de arcos de entrada.

O 'record' da lista de arcos de saída é composto por três campos, a saber:

Campo		Conteúdo
ARCOSAI	--	nome do arco;
ARMDT	--	nome do armazenador associado ao arco;
ADSFI	--	apontador para o próximo 'record' da lista de arcos de saída.

PC5 - Definição da Estrutura dos Armazenadores

Esse módulo obtém, por intermédio do usuário, a estrutura dos armazenadores e efetua um teste de consistência, verificando se os armazenadores definidos pertencem à estrutura do grafo de dados. Para efetuar o teste, o simulador obtém o nome do armazenador e, em seguida, verifica se esse se encontra relacionado com algum processador do grafo de dados. Se o armazenador não estiver definido na estrutura de dados, o simulador emite mensagens requisitando a redefinição do armazenador em questão.

Visando oferecer ao usuário uma maior flexibilidade para a manipulação dos dados do modelo, criaram-se dois tipos de armazenadores: armazenadores do tipo fixo e armazenadores do tipo variável.

O armazenador do tipo fixo tem como característica a capacidade de armazenar valores numéricos de tamanho fixo. Esse valor fixo constitui a unidade de armazenamento do armazenador. Através da utilização desse tipo de armazenador, o usuário pode simular a escrita e/ou a leitura de mensagens de tamanho fixo.

O armazenador do tipo variável tem a possibilidade de armazenar valores numéricos de tamanhos variáveis. Para tanto, utiliza-se uma estrutura de lista ligada que possibilita o armazenamento de diversas mensagens de tamanhos distintos. Através da utilização desse tipo de armazenador, o usuário pode simular a escrita e/ou a leitura de mensagens de tamanhos variáveis.

Na definição da estrutura dos armazenadores, o usuário deve definir para cada armazenador um nome, o tipo, o tamanho máximo, a unidade de armazenamento (somente para armazenadores do tipo fixo) e a estação. Após a criação de cada armazenador, é oferecida ao usuário a possibilidade de iniciá-los. A iniciação consiste em inserir as mensagens iniciais em cada armazenador. Ao término de cada inserção o simulador checa a consistência do armazenador, observando se esse não foi iniciado de forma que sua

capacidade de armazenamento tenha sido excedida. Ocorrendo tal fato, é oferecida ao usuário a possibilidade de uma nova iniciação desse mesmo armazenador, uma vez que a anterior não foi considerada válida. É importante ressaltar que as mensagens manipuladas pelo simulador se traduzem somente no tamanho de uma informação e não no seu conteúdo.

Ao término da execução do módulo, é oferecida ao usuário a possibilidade de retornar ao módulo de interface de operação ou de chamar o módulo de definição da interpretação. As listas ligadas existentes na figura 3.6 ilustram a forma de implementação da estrutura.

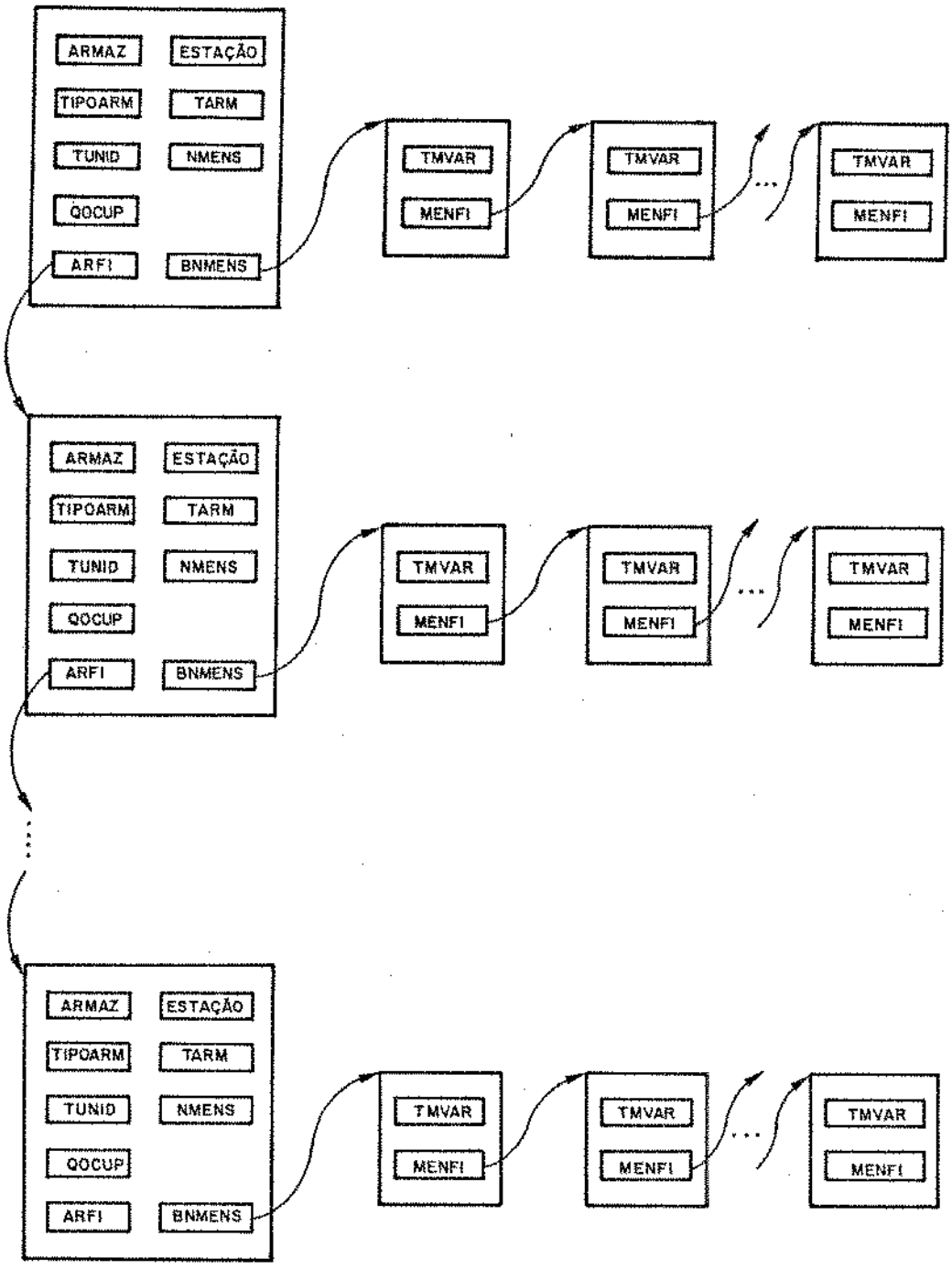


FIGURA 3.6
ESTRUTURA DE ARMAZENADORES

A figura 3.6 mostra as duas listas ligadas que compõem a estrutura dos armazenadores. As listas são compostas por 'records', sendo que a primeira contém os armazenadores e a segunda contém as mensagens dos armazenadores do tipo variável.

O 'record' da lista de armazenadores é composto por nove campos, a seguir:

Campo		**Conteúdo**
ARMAZ	--	nome do armazenador;
ESTAÇÃO	--	nome da estação em que o armazenador se encontra;
TIPOARM	--	tipo do armazenador;
TARM	--	tamanho do armazenador;
TUNID	--	tamanho da unidade de armazenamento;
NMENS	--	número de mensagens;
QOCUP	--	espaço ocupado no armazenador;
BNMENS	--	apontador para a lista de mensagens;
ARFI	--	apontador para o próximo record da lista de armazenadores.

O 'record' da lista de mensagens é composto pelos dois campos descritos a seguir:

Campo		Conteúdo
TMVAR	--	tamanho da mensagem;
MENFI	--	apontador para o próximo 'record' da lista de mensagens.

PCB - Definição da interpretação

O domínio da interpretação do simulador GMB* dispõe de um conjunto de primitivas de alto nível. Através da utilização dessas primitivas, o usuário determina o fluxo de controle e a disposição dos dados do modelo simulado.

Para a definição da interpretação, o usuário define o nome do processador associado à interpretação e, em seguida, insere as primitivas vinculadas a esse processador. Ao ser inserido o nome de um processador, o simulador verifica se o mesmo se encontra especificado no grafo de dados. Caso o processador não esteja definido na estrutura de dados, o simulador emite mensagem solicitando a redefinição do mesmo. Caso contrário, o simulador solicita a definição das primitivas.

Durante a inserção de cada primitiva, o simulador efetua

testes com o intuito de detectar erros léxicos, sintáticos e semânticos. Ao ser detectado qualquer erro, o simulador emite mensagem solicitando a redefinição da primitiva.

Essa característica de verificação e testes torna o módulo de definição da interpretação bastante confiável, uma vez que dá ao usuário uma segurança para a criação de uma dada interpretação.

Após a inserção da estrutura, faz-se a sua impressão na tela do microcomputador, sendo posteriormente armazenada nos arquivos de dados "TERP.CAO" e "TIVA.CAO". Isso feito, a execução do módulo de definição da interpretação termina chamando o módulo de interface de operação. As listas ligadas existentes na figura 3.7 ilustram a forma de implementação da estrutura.

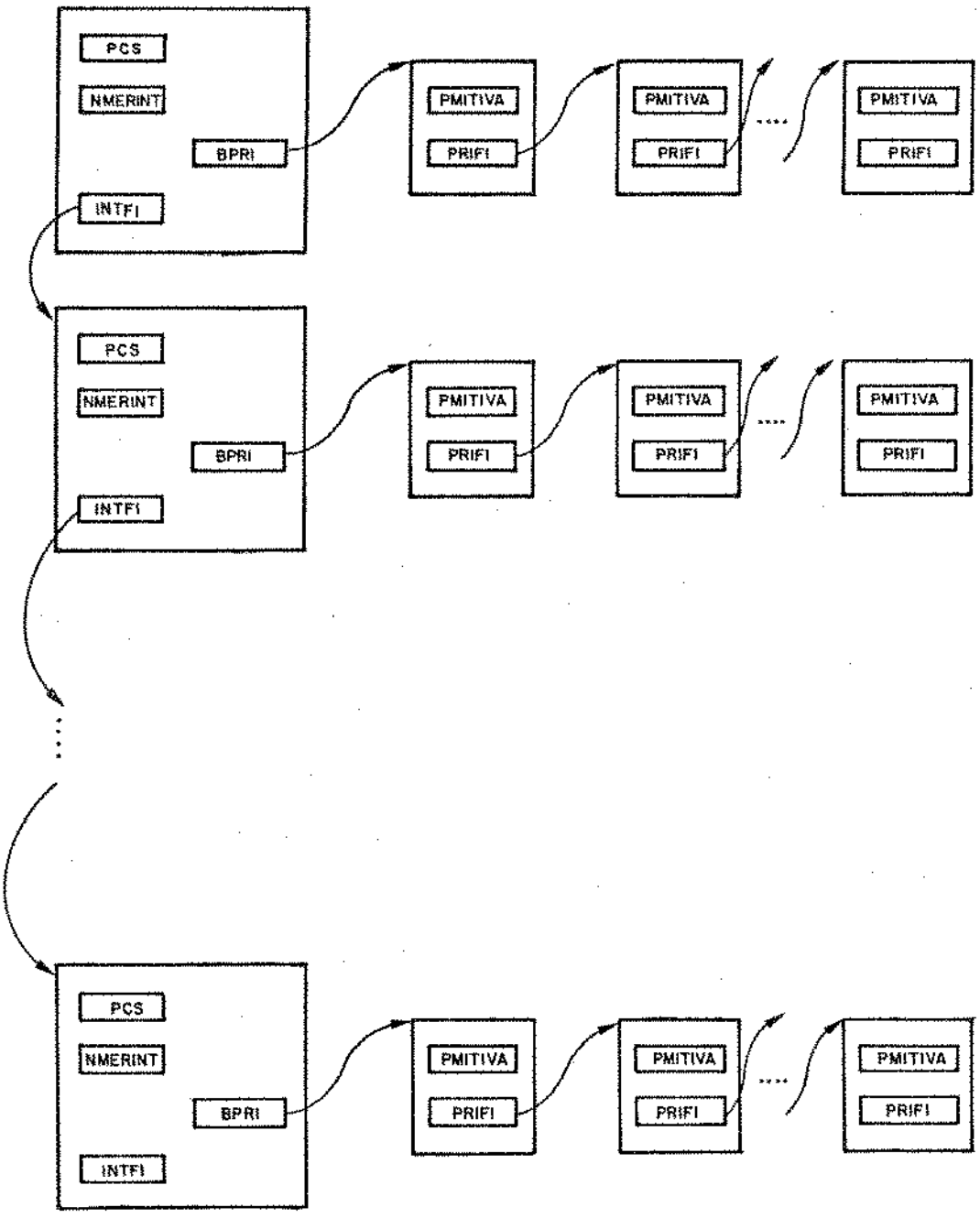


FIGURA 3.7
ESTRUTURA DE INTERPRETAÇÃO

A figura 3.7 mostra as duas listas ligadas que compõem a estrutura da interpretação. As listas são constituídas de 'records', sendo que a primeira armazena os nomes dos processadores e a segunda armazena as primitivas associadas ao processador.

O 'record' da lista de processadores é composto por quatro campos, assim especificados:

Campo		**Conteúdo**
PCS	--	nome do processador;
NMERINT	--	número de primitivas;
BPRI	--	apontador para a lista de mensagens;
INTFI	--	apontador para o próximo 'record' da lista de processadores.

O 'record' da lista de primitivas é composto por dois campos:

Campo		**Conteúdo**
PMITIVA	--	nome da primitiva;
PRIFI	--	apontador para o próximo 'record' da lista de mensagens.

Primitivas Disponíveis no domínio de interpretação

O simulador GMB* dispõe de um conjunto de sete primitivas, através das quais o usuário cria a interpretação de um dado modelo.

As primitivas que acessam aos armazenadores podem ser de dois tipos: primitiva de acesso destrutivo e primitiva de acesso não-destrutivo.

No acesso destrutivo, a primitiva efetua a operação de leitura eliminando do armazenador a mensagem mais antiga. Na operação de escrita somente é eliminada uma quantidade necessária para a inserção da nova mensagem. Essa quantidade é constituida pelas mensagens mais antigas.

No acesso não-destrutivo, o conteúdo do armazenador não decresce, isto é, não há eliminação de mensagens, há apenas um acréscimo no caso do acesso de escrita.

Como mencionado no item anterior, após a inserção de cada primitiva o simulador verifica se essa foi escrita corretamente. Em caso afirmativo, o simulador testa se a primitiva inserida faz parte do conjunto de primitivas disponíveis no simulador. Ocorrendo alguma anomalia, o simulador emite mensagem requisitando a redefinição da primitiva.

1 - Leitura de Armazenadores do Tipo Fixo

Sintaxe : **LERF (ARCO,D)**

Sendo : **ARCO** = string de até quatro caracteres;

D = string de um caractere.

Esta primitiva possui como argumentos o arco de dados associado ao armazenador no qual a leitura será efetuada, e o caractere de definição do tipo de leitura (leitura destrutiva ou leitura não-destrutiva).

Através da utilização dessa primitiva, o usuário simula a leitura, que pode ser destrutiva ou não-destrutiva, em armazenadores do tipo fixo.

Após a inserção da primitiva, o simulador checa se o processador associado tem a possibilidade de fazer acessos de leitura em armazenadores, isto é, o simulador checa se o arco especificado é um arco de entrada do processador em questão. Em caso afirmativo, o simulador testa se o armazenador relacionado é do tipo fixo. Caso ocorra alguma anomalia, o simulador emite mensagem solicitando a redefinição da primitiva.

2 - Leitura de Armazenadores do Tipo Variável

Sintaxe : **LERV (ARCO,D)**

Os argumentos da primitiva são semelhantes aos da anterior. Através da utilização desta primitiva o usuário simula a leitura, que pode ser destrutiva ou não-destrutiva, em armazenadores do tipo variável.

Após a inserção da primitiva, o simulador checa se o processador associado tem possibilidade de fazer acessos de leitura a armazenadores, isto é, o simulador checa se o arco especificado na primitiva é um arco de entrada do processador em questão. Em caso afirmativo, o simulador testa se o armazenador associado é do tipo variável. Ocorrendo alguma anomalia o simulador emite mensagem solicitando a redefinição da primitiva.

3 - Escrita em Armazenadores do Tipo Fixo

Sintaxe : **ESCREVERF (ARCO,D)**

Os argumentos dessa primitiva são semelhantes aos argumentos das primitivas anteriores. Por intermédio da utilização dessa primitiva, o usuário simula a escrita, que pode ser destrutiva ou não-destrutiva, em armazenadores do tipo fixo.

Após a inserção da primitiva, o simulador testa se o processador associado tem a possibilidade de fazer acessos de escrita em armazenadores, isto é, o simulador testa se o arco especificado na primitiva é um arco de saída do processador em

questão. Em caso afirmativo, o simulador checa se o armazenador vinculado é do tipo fixo. A ocorrência de qualquer anomalia fará com que o simulador emita mensagem solicitando a redefinição da primitiva.

4 - Escrita em Armazenadores do Tipo Variável

Sintaxe: **ESCREVERV (ARCO,TAM,D)**

Sendo: **ARCO** = string de até quatro caracteres;

TAM = inteiro de valor máximo igual a novecentos e noventa e nove;

D = string de um caractere.

Essa primitiva possui como argumentos o arco de dados associado ao armazenador, no qual a escrita será efetuada, o tamanho da mensagem e o caractere de definição de tipo de escrita.

O usuário pode se utilizar desta primitiva para simular a escrita, que pode ser destrutiva ou não-destrutiva, em armazenadores do tipo variável.

Após a inserção da primitiva, o simulador checa se o arco especificado na primitiva é um arco de saída do processador associado à interpretação. Em caso afirmativo, o simulador testa se o armazenador é do tipo variável. Ocorrendo alguma anomalia, o simulador emite uma mensagem solicitando a redefinição da primitiva.

5 - Marcação dos Arcos de Saída

Sintaxe: **SAIDA (ARCO,PRIO)**

Onde: **ARCO** = string de até quatro caracteres;

PRIO = inteiro de valor máximo igual a novecentos e noventa e nove.

Os argumentos da primitiva são o arco de saída do nó de controle e a prioridade da marca que será nele colocada.

Através da utilização dessa primitiva, o usuário efetua a marcação dos arcos de saída do nó de controle associado à interpretação.

Após a inserção da primitiva, o simulador checa se o arco definido é um arco de saída do nó de controle associado. Se o arco não pertencer ao conjunto de arcos de saída do nó de controle, o simulador emite mensagem solicitando a redefinição da primitiva. Estando o arco corretamente definido, o simulador verifica se ele está relacionado com outros arcos, por intermédio de expressões lógicas do tipo "and". Existindo o relacionamento envolvendo tais expressões lógicas, o simulador aguarda que o usuário forneça outras primitivas de marcação a fim de marcar os outros arcos relacionados. Não ocorrendo tal fato, o simulador emite mensagem ao usuário solicitando a inserção das primitivas para a marcação dos arcos relacionados.

6 - Tempo de Processamento de Tarefas

Sintaxe: **ATRASO (TEMPO)**

Sendo : **TEMPO** = inteiro de valor máximo igual a novecentos e noventa e nove.

O único argumento da primitiva é um valor que retrata uma quantidade de tempo, através da qual o usuário determina o tempo associado à execução de uma primitiva ou o tempo associado à execução de um conjunto de primitivas. O usuário pode utilizar diversas primitivas de atraso em uma mesma interpretação.

7 - Primitivas Condicionais

Sintaxe: **IF (ARCO,OL,QOCUP) THEN**
(conjunto de primitivas)
ELSE
(conjunto de primitivas)
ENDIF

Sendo: **ARCO** = string de até quatro caracteres;
OL = string de um caractere;
QOCUP = inteiro de valor máximo igual a noventa e nove.

A primitiva condicional é composta de duas primitivas de decisão **THEN** e **ELSE** e uma de término de condição **ENDIF**. A primitiva **IF** possui como argumentos o arco de dados associado ao armazenador, cuja quantidade ocupada deseja-se testar, o operador lógico (> = <) e a quantidade de referência para o teste.

Através da utilização desta primitiva o usuário determina o fluxo de processamento do modelo simulado. A condição analisada pela primitiva é a quantidade de espaço ocupada no armazenador. O usuário, através da utilização da primitiva, testa se um determinado armazenador satisfaz a uma dada condição. Em caso afirmativo, as primitivas associadas à primitiva **THEN** serão executadas; caso contrário, serão executadas as primitivas associadas à primitiva **ELSE**.

Após a inserção da primitiva **IF**, o simulador testa se o arco de dados especificado pertence ao processador vinculado à interpretação. Ocorrendo algum erro, o simulador requisita a redefinição da primitiva. Acrescente-se que é permitida a utilização de primitivas condicionais concatenadas, e que cada primitiva **IF** deve possuir uma primitiva **ENDIF** correspondente. Se a primitiva **ENDIF** não for inserida, o simulador envia uma mensagem solicitando a sua inserção.

A possibilidade de se utilizar primitivas condicionais concatenadas permite ao usuário efetuar testes em diversos níveis, criando dessa forma uma interpretação com um grau de

complexidade bastante elevado, pois os conteúdos dos armazenadores podem ser alterados por mais de um processador, quando se simula o processamento paralelo.

Acrescente-se que para a definição das primitivas disponíveis no simulador, pode-se utilizar tanto letras maiúsculas, como a notação utilizada neste texto, quanto letras minúsculas. É importante salientar também que todos os módulos de definição de estruturas possuem mecanismos que possibilitam ao usuário realizar correções durante a etapa de inserção das estruturas.

PC7 - Módulo de Redefinição da Estrutura da Interpretação

O módulo de redefinição da estrutura da interpretação foi implementado visando oferecer ao usuário a possibilidade de redefinir as primitivas que se encontram associadas a um dado processador, sem ter de redefinir toda a interpretação do modelo. A utilização do módulo de redefinição facilita o uso do simulador, pois um determinado modelo pode vir a ter diversas interpretações. O usuário, ao começar a simulação de um dado modelo, pode, inicialmente, fazê-lo utilizando-se de uma interpretação com poucas primitivas aumentando-as gradualmente, obtendo, dessa forma, uma interpretação mais complexa.

Para a obtenção da correção da estrutura, o módulo efetua a leitura dos arquivos de dados "TERP.CAO" e "TIVA.CAO",

transferindo para a memória do microcomputador a estrutura da interpretação. Após a obtenção da estrutura, o simulador emite mensagem solicitando a inserção do nome do processador cujas primitivas serão alteradas. De posse do nome do processador, o simulador efetua uma busca na estrutura e, uma vez encontrado o processador, é oferecida ao usuário a possibilidade de definição das novas primitivas a ele associadas. Caso o nome atribuído pelo usuário não corresponda a um processador definido na estrutura, o simulador emite mensagem solicitando o nome correto do processador que terá sua interpretação alterada.

Após a redefinição das primitivas associadas a um processador, é oferecida ao usuário a possibilidade de redefinir as primitivas de outro. Esse processo repetir-se-á até que o usuário termine todas as redefinições.

Acrescente-se que os testes de detecção de erros sintáticos, semânticos e léxicos existentes no módulo de definição da estrutura da interpretação são também efetuados por esse módulo. Assim, garante-se que as primitivas definidas no módulo de redefinição são consistentes, já que estas passam pelos mesmos testes existentes no módulo de definição da estrutura da interpretação.

Ao término da correção da estrutura, faz-se a sua impressão na tela do microcomputador. Assim o usuário terá a possibilidade de efetuar uma nova correção ou armazená-la nos arquivos de dados

"TERP.CAO" e "TIVA.CAO". Ao término da execução do módulo, é oferecida ao usuário a possibilidade de retornar ao módulo de interface de operação.

PC8 - Tratamento do Grafo de Controle

Ao iniciar a execução desse módulo, a estrutura do grafo de controle, composta pelo arquivo dos nós e pelo arquivo dos arcos de entrada é transferida para a memória interna do microcomputador. A estrutura do vetor de marcas é também transferida para a memória interna do microcomputador. Isto feito, inicia-se uma pesquisa no vetor de marcas, objetivando detectar os arcos que estão marcados. De posse desses arcos, o simulador analisa a estrutura do grafo de controle, buscando os nós que possuam expressões lógicas de entrada satisfeitas.

O algoritmo que efetua a busca analisa os arcos de entrada de cada nó, verificando se pertencem ao grupo de arcos marcados derivados da estrutura do vetor de marcas. Havendo uma correspondência, isto é, estando o arco marcado, o algoritmo passa a analisar a expressão lógica que relaciona o arco com o seu adjacente.

Sendo a expressão lógica um caractere branco, tal fato implica que existe apenas um arco de entrada no nó ou implica que o arco analisado é o último que compõe o conjunto de arcos de entrada do nó. No caso de haver um único arco de entrada no nó, o

simulador seleciona esse nó, colocando-o em uma estrutura de rascunho, pois a sua expressão lógica se encontra satisfeita.

Encontrando-se uma expressão lógica "OR", o simulador seleciona o nó, uma vez que o arco marcado fornece condições para a habilitação do mesmo. Isto feito, o simulador passa a analisar os outros arcos que compõem o conjunto de arcos de entrada do nó, objetivando detectar novos conjuntos de arcos que permitam a habilitação do nó. Ao término da análise de todos os arcos de entrada, o simulador armazena o nó e os conjuntos de arcos de entrada cujas expressões lógicas estão satisfeitas na estrutura de rascunho.

Encontrando-se uma expressão lógica "AND", o simulador passa a analisar o arco de entrada adjacente. Estando esse marcado, o simulador analisa a expressão lógica a ele relacionada. Esse procedimento repetir-se-á até que encontre um arco marcado cuja expressão lógica é um "OR" ou um caractere branco. Ao ser satisfeita essa condição, o simulador seleciona o nó e passa a analisar os outros arcos de entrada, objetivando encontrar novos grupos de arcos que possibilitem habilitar o nó. Terminada a pesquisa nos arcos de entrada, a estrutura do nó e os grupos de arcos que possibilitam habilitar o nó são armazenados na estrutura de rascunho.

Não sendo satisfeita a condição, ou seja, estando o arco relacionado com outros arcos que não possuam marcas, o simulador despreza o arco e passa a procurar entre os arcos de entrada do

nó um novo conjunto de arcos que satisfaça a condição para a habilitação. Não havendo nenhum conjunto que permita a habilitação do nó, o simulador passa a analisar os outros nós da estrutura.

O algoritmo de busca foi implementado de maneira que para cada arco marcado se faça uma busca na estrutura de nós, objetivando detectar os possíveis nós que possam vir a ser habilitados pelo arco. Acrescente-se que, devido a existência de arcos complexos, há possibilidade de um mesmo arco ser arco de entrada de distintos nós. Assim, um arco pode habilitar mais que um nó.

De posse da estrutura de rascunho, passa-se a verificar para cada nó o grupo de arcos de entrada que possua maior prioridade; o grupo de maior prioridade dita a prioridade global do nó.

Para a obtenção da prioridade dos grupos de arcos, utilizou-se o seguinte critério: nos grupos de arcos relacionados por expressões lógicas "AND", a prioridade global será igual a soma das marcas de maior prioridade de cada arco. Esse critério se baseia no fato de que a existência de uma expressão lógica do tipo "AND" requer que todos os arcos relacionados estejam marcados; assim a prioridade do grupo será a soma da maior prioridade associada a cada arco.

Após a escolha dos grupos de arcos de maior prioridade, o simulador testa o conteúdo da variável de definição do passo de

simulação. Sendo o valor da variável igual ao passo inicial de simulação, isto é, passo "0", o simulador armazena a estrutura de rascunho nos arquivos de dados "NOATIVO.ATV", "ARCATIVO.ATV" e "MACATIVO.ATV". Esse conjunto de arquivos de dados constitui a estrutura dos nós habilitados que contém, respectivamente, a estrutura dos nós de controle, os conjuntos de arcos que possibilitam a habilitação dos nós e a prioridade global dos nós.

Sendo o valor da variável diferente do passo inicial, o simulador transfere para a memória do microcomputador a estrutura dos nós habilitados criada nos passos anteriores. Terminada essa operação, o simulador analisa a estrutura dos nós habilitados, verificando se os nós da estrutura de rascunho já não se encontram ali inseridos. Havendo uma correspondência, observa-se a prioridade que se encontra associada ao nó, pois um mesmo nó que está na estrutura de nós habilitados, com uma certa prioridade, pode estar definido na estrutura de rascunho com outra prioridade. Esse procedimento visa atualizar a prioridade do nó pertencente a estrutura dos nós habilitados. Ao término desse procedimento, os nós que não se encontravam na estrutura de nós habilitados são a ela associados, sendo a estrutura armazenada em memória de massa (disketes).

Terminado o armazenamento da estrutura, o módulo termina sua execução chamando o módulo de tratamento da estrutura dos nós habilitados. As listas ligadas existentes na figura 3.8 ilustram a forma de implementação da estrutura.

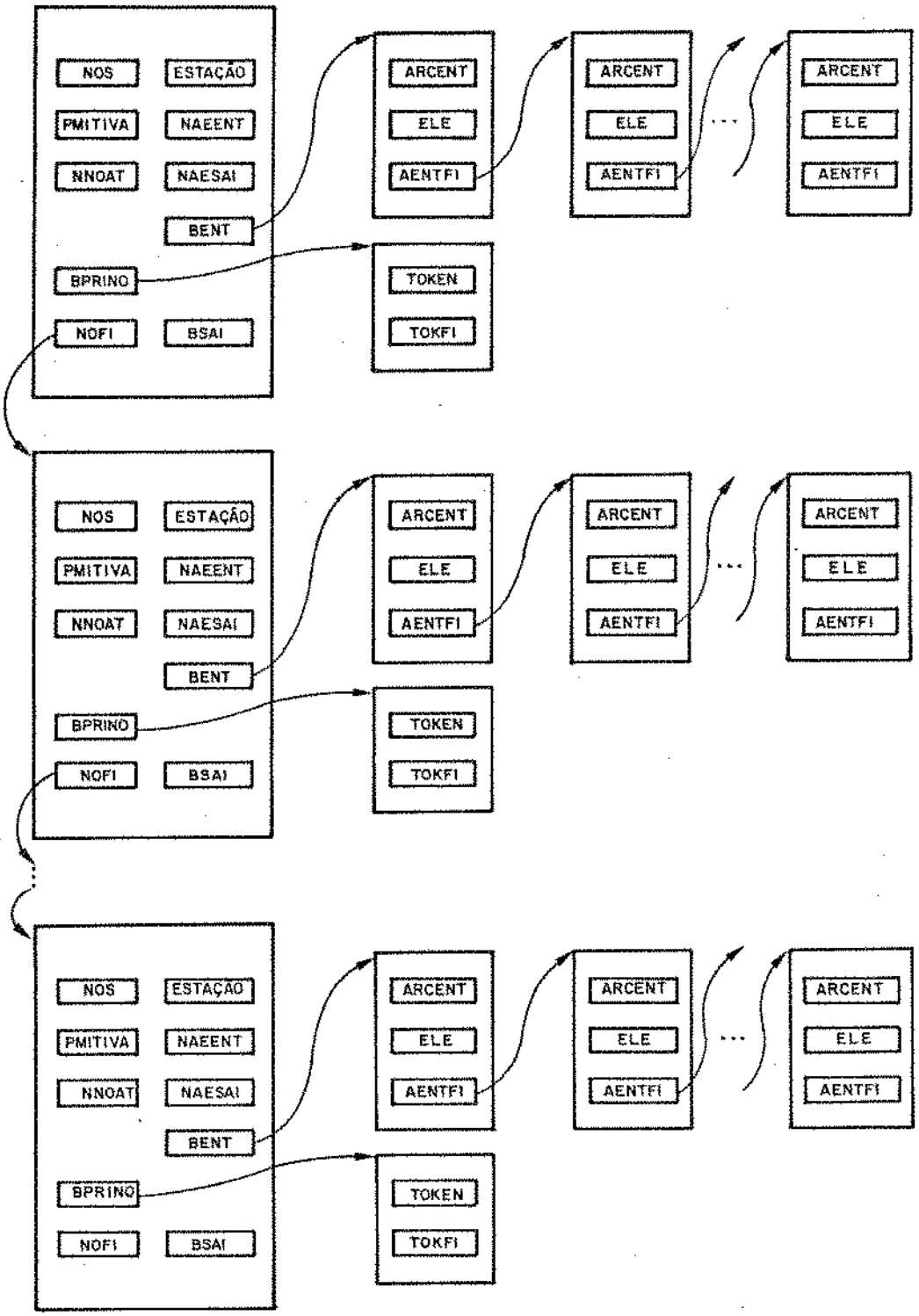


FIGURA 3.8
ESTRUTURA DE NÓS HABILITADOS

A figura 3.8 ilustra as três listas ligadas que compõem a estrutura dos nós habilitados. A primeira e a segunda lista são similares à mostrada na figura 3.3 e a terceira é composta de um 'record' cujo campo contém a prioridade do nó.

PC9 - Tratamento da Estrutura dos Nós Habilitados

Esse módulo ao ser executado irá selecionar os nós habilitados, que passarão para o estado ativo. Ao iniciar sua execução, a estrutura dos nós habilitados e a estrutura do vetor de marcas são transferidas para a memória do microcomputador. Encontrando-se a estrutura dos nós habilitados na memória do microcomputador, o simulador passa a executar o algoritmo para a seleção dos nós.

O algoritmo está baseado no critério de que apenas um nó por estação pode estar no estado ativo durante um determinado passo de simulação. A utilização desse algoritmo permite que apenas o nó de maior prioridade de uma estação passe para o estado ativo, permanecendo os demais no estado habilitado. No caso de empate, existe uma variável associada a cada nó habilitado cujo conteúdo exprime o número de passos em que o nó se encontra no estado habilitado. Assim sendo, o nó que possuir um maior número de passos no estado de espera é ativado. Para o caso em que a condição de empate persiste também na análise do conteúdo dessa

variável, o simulador utiliza um procedimento pseudoaleatório e, através desse, sorteia o nó que passará para o estado ativo.

Os nós escolhidos para a ativação são armazenados em uma estrutura de rascunho. Acrescente-se que estando a estrutura de nós habilitados vazia, isto é, não havendo nós que possam ser ativados, o simulador analisa o conteúdo das variáveis de simulação. Posteriormente atua informando o usuário da possibilidade da ocorrência de erros no grafo de controle, uma vez que não há nenhum nó habilitado. Essa anomalia pode ser causada por um "deadlock" ou por uma má terminação no grafo de controle.

Após a criação da estrutura de rascunho, o simulador carrega a estrutura dos nós do grafo de controle e a estrutura dos arcos de saída do mesmo. O simulador necessita utilizar estas estruturas, pois a estrutura dos nós ativos é composta pela estrutura de nós de controle e pela estrutura dos arcos de saída. Isto feito, o simulador analisa o conteúdo da variável que dita o passo de simulação. Sendo esse igual ao passo inicial, o simulador efetua uma pesquisa na estrutura do grafo de controle, objetivando encontrar os arcos de saída que estarão associados os nós que passarão para o estado ativo. Terminado esse procedimento, os nós que se encontram na estrutura de rascunho são armazenados no arquivo de dados "NOEX.EXE" e a estrutura dos arcos de saída desses nós é armazenada no arquivo de dados "AREX.EXE". Esses dois arquivos de dados constituem, dessa forma,

a estrutura de nós ativos.

Sendo o conteúdo da variável que dita o passo de simulação diferente do passo inicial, o simulador carrega a estrutura dos nós ativos na memória do microcomputador. De posse da estrutura de nós ativos, o simulador passa a analisá-la, juntamente com a estrutura dos nós de rascunho. Essa análise tem como objetivo encontrar e retirar da estrutura de rascunho todos os nós cuja estação associada já possua um nó em execução. Tal procedimento se baseia no critério que assegura que apenas um nó por estação pode estar no estado ativo em um determinado passo de simulação. Terminado esse procedimento, o simulador associa à estrutura de nós ativos o conteúdo ainda existente na estrutura de rascunho. Utilizando ainda o conteúdo da estrutura de rascunho, o simulador pesquisa a estrutura de nós habilitados retirando daí os nós que passarão para o estado ativo e incrementando o conteúdo das variáveis de número de passos de espera dos nós restantes na estrutura.

Terminado o procedimento de atualização da estrutura dos nós habilitados, o simulador armazena em memória de massa as estruturas dos nós ativos.

Associado à manipulação da estrutura de nós habilitados e à criação da estrutura de nós ativos, são atribuídas a esse módulo a criação e a manipulação da estrutura da máquina de "tokens". A máquina de "tokens" é constituída pelos arcos marcados existentes

no grafo de controle. Assim, a cada passo de simulação, um conjunto de arcos de controle é associado à estrutura da máquina de "tokens". Através da análise da estrutura, o usuário obtém a informação do comportamento dinâmico do fluxo de controle, já que a cada passo de simulação há uma alteração na disposição das marcas nos arcos, registrada pela estrutura da máquina de "tokens".

Para a obtenção da máquina de "tokens", o simulador pesquisa o vetor de marcas, copiando em uma estrutura de rascunho os arcos marcados. Isto feito, o simulador verifica o conteúdo da variável que dita o passo de simulação. Sendo esse igual ao passo inicial, o simulador armazena a estrutura de rascunho nos arquivos de dados "MAQUINA.TOK" e "PRIO.TOK", dando origem à estrutura da máquina de "tokens".

Sendo o passo de simulação diferente do passo inicial, o simulador insere na estrutura da máquina de "tokens" um campo de controle para a delimitação de um passo de simulação e, em seguida, associa a ela a estrutura de rascunho. Terminado o procedimento, a execução do módulo de tratamento da estrutura de nós habilitados chega ao fim chamando o módulo de tratamento dos nós ativos. As listas ligadas existentes na figura 3.9 ilustram a forma de implementação da estrutura.

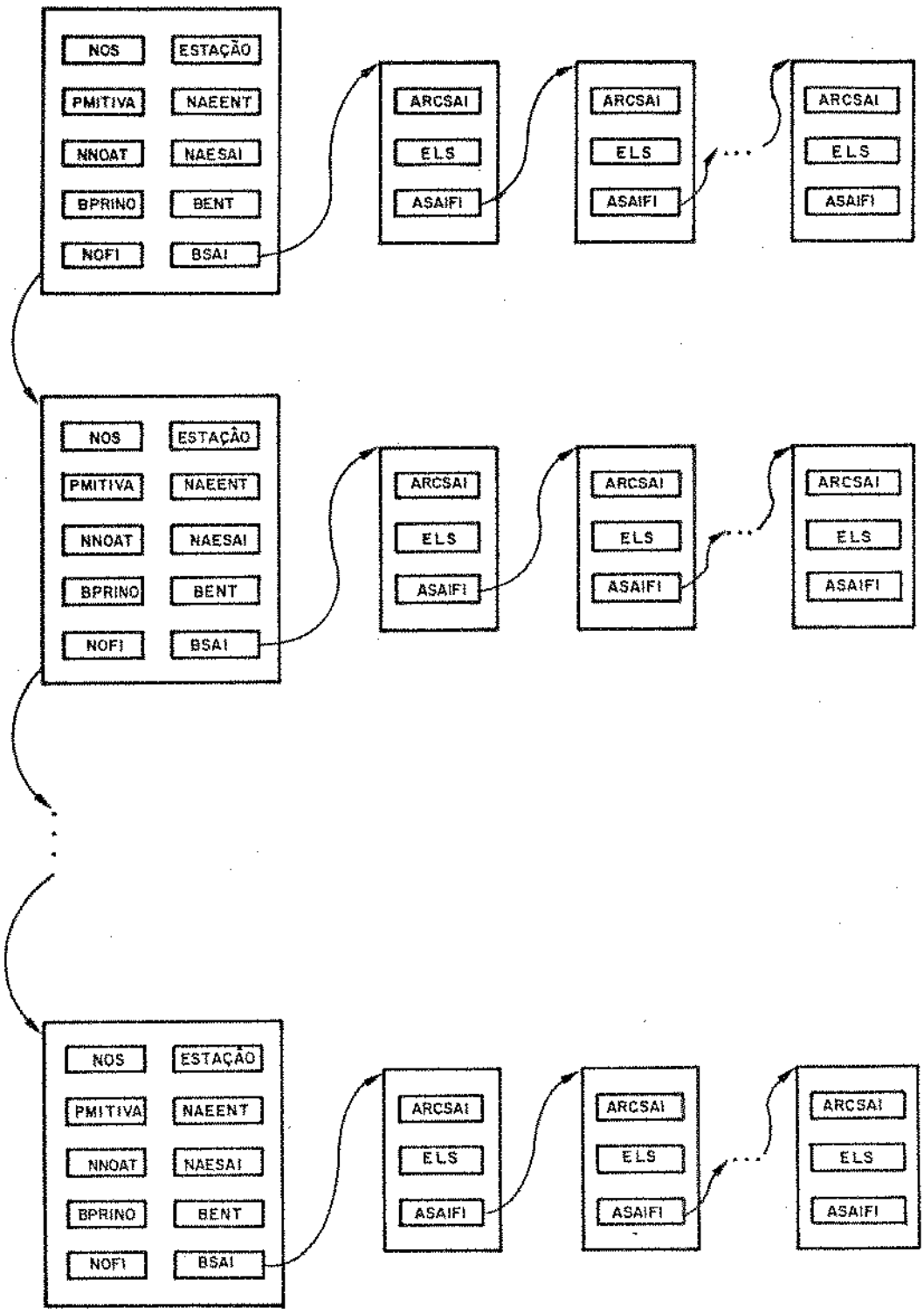


FIGURA 3.9

ESTRUTURA DE NÓS ATIVOS

A figura 3.9 mostra as duas listas ligadas que compõem a estrutura dos nós ativos. Observe-se que as listas são similares às da figura 3.3, e que, nessa estrutura, o campo 'PMITIVA' é utilizado para armazenar a primitiva que se encontra em execução caso ocorra o término de um passo de simulação. Esse campo é necessário, pois na simulação envolvendo nós pertencentes a estações distintas, alguns processadores podem estar executando uma primitiva quando ocorre o término do passo de simulação.

As listas ligadas existentes na figura 3.10 ilustram a forma de implementação da estrutura da máquina de 'tokens'.

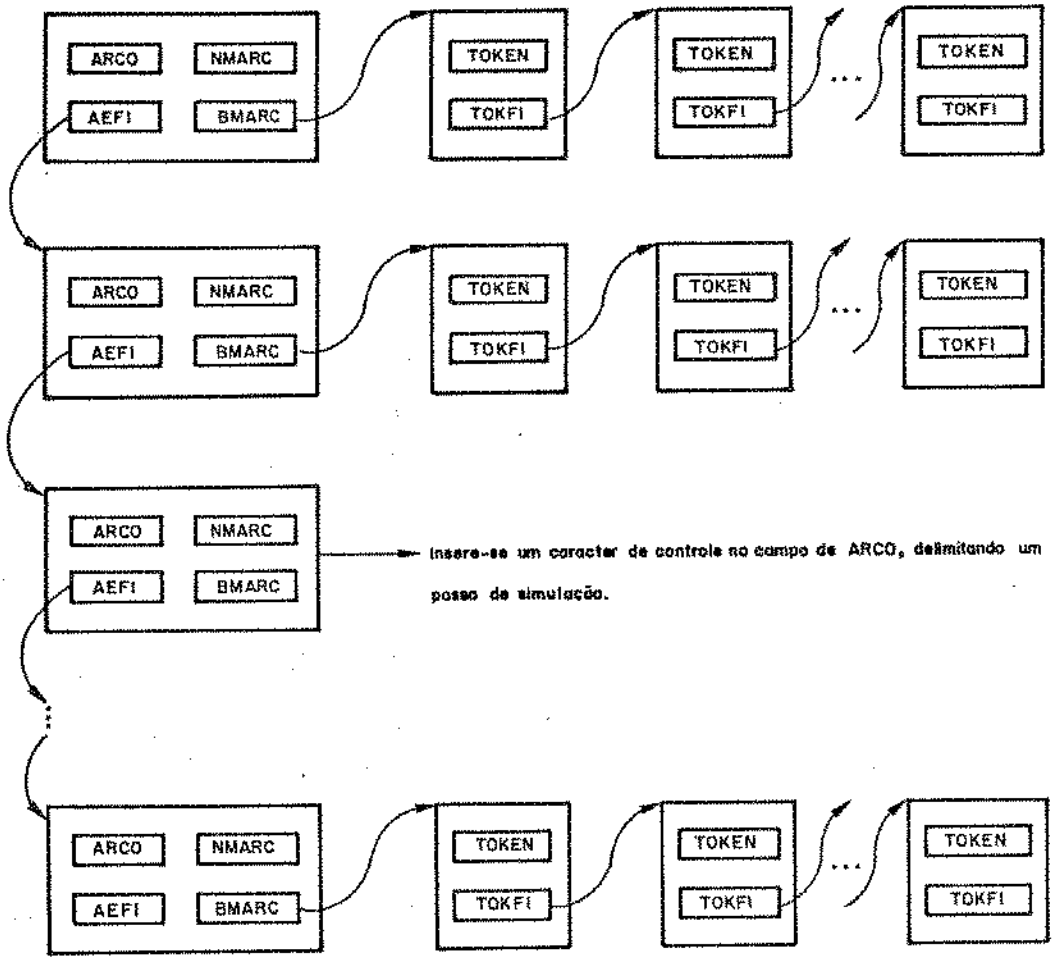


FIGURA 3.10
ESTRUTURA DA MÁQUINA DE TOKENS

A estrutura da máquina de 'tokens' é similar à estrutura do vetor de marcas ilustrado na figura 3.4. Observe-se que a diferença existente entre as estruturas é que a estrutura da máquina de 'tokens' utiliza um 'record' da lista de arcos para inserir o delimitador de passo de simulação.

PC1D - Tratamento da Estrutura dos Nós Ativos

A tarefa executada por esse módulo se baseia em obter os nós que se encontram na estrutura de nós ativos e em executar, emulando um paralelismo, as interpretações dos processadores associados aos nós. Durante a execução das interpretações, são efetuados testes de consistência nos armazenadores da estrutura de dados.

Ao iniciar a execução do módulo a estrutura de nós ativos, a estrutura do grafo de dados, a estrutura dos armazenadores e a estrutura da interpretação original, definida pelo usuário, são transferidas para a memória do microcomputador. Terminado esse procedimento, o simulador analisa o conteúdo da variável que dita o passo de simulação. Sendo esse diferente ao passo inicial, o simulador transfere para a memória do microcomputador a estrutura da interpretação modificada que foi criada no passo anterior. Nessa estrutura os argumentos das primitivas "ATRASO" podem estar alterados, devido a possibilidade de ter ocorrido o término do passo de simulação durante a execução delas. Ao

término desse procedimento, o simulador inicia uma pesquisa na estrutura do grafo de dados e na estrutura de nós ativos. Tal pesquisa, objetiva encontrar na estrutura do grafo de dados o processador cujo nó se encontra no estado ativo.

Encontrando-se o processador, o simulador o verifica na estrutura de nós ativos se no 'record' do nó em questão existe alguma primitiva no campo 'PMITIVA'. Em caso afirmativo, o simulador de posse do nome processador, inicia uma busca na estrutura da interpretação modificada com o intuito de encontrar a interpretação associada ao nó. Encontrando-se a interpretação, o simulador obtém o nome da primitiva existente no campo 'PMITIVA' e em seguida pesquisa a interpretação até que a primitiva correspondente seja encontrada. Caso contrário, o simulador de posse do nome do processador, efetua a busca na estrutura da interpretação originalmente definida pelo usuário objetivando encontrar a primeira primitiva da interpretação associada ao nó.

Encontrando-se a primitiva, o simulador armazena o apontador da primitiva em uma estrutura de apontadores de rascunho. Esse procedimento é repetido até que todos os apontadores das interpertações associadas aos nós ativos estejam armazenados na estrutura de rascunho.

Testes de consistência são efetuados durante as buscas, pois, apesar de terem sido efetuados testes de correspondência entre as estruturas, durante a fase de definição, o simulador repete-os

durante a execução deste módulo, garantindo, com isso, uma máxima segurança para o desenvolvimento da simulação.

De posse da estrutura contendo os apontadores das interpretações, o simulador inicia a execução das primitivas. O algoritmo que foi implementado executa uma primitiva de cada interpretação por vez.

A execução das diversas primitivas pode implicar a execução de primitivas que efetuam acessos a armazenadores. Para a obtenção desses acessos, o simulador pega o arco definido na primitiva e busca na estrutura do grafo de dados o nome do armazenador a ele associado. De posse desse nome, pesquisa a estrutura dos armazenadores até que o mesmo seja encontrado.

Visando garantir a consistência dos armazenadores, o simulador efetua, a cada acesso, testes nesses armazenadores. Ocorrendo uma tentativa de escrita, não-destrutiva em armazenadores cheios ou uma leitura em armazenadores vazios, o simulador, com base nas variáveis de simulação, informa o usuário do ocorrido.

Para a execução das primitivas "IF", o simulador obtém o nome do arco explicitado na primitiva. De posse desse, o simulador pesquisa a estrutura do grafo de dados, objetivando encontrar o armazenador associado. Encontrando-o, pesquisa a estrutura dos armazenadores e, ao atingir o armazenador desejado, verifica de acordo com o operador lógico existente na primitiva, se a

quantidade ocupada é igual, maior ou menor ao argumento explicitado na primitiva. Estando o teste correto, o simulador atribui um valor a uma variável de controle, permitindo, dessa forma, a execução do grupo de primitivas associadas à primitiva condicional. Caso contrário, um outro valor será armazenado na variável, fazendo com que as primitivas associadas à primitiva "IF" não sejam executadas, possibilitando a execução das primitivas associadas à primitiva "ELSE", caso existam.

Ao encontrar uma primitiva "ATRASO", o simulador interrompe a execução da interpretação associada, passando a executar as primitivas pertencentes as outras interpretações. Caso em um determinado instante todas as interpretações estejam executando primitivas "ATRASO"; o simulador analisa os argumentos das primitivas "ATRASO" e escolhe a primitiva cujo valor do argumento é o menor. De posse do menor valor, decrementa todos os argumentos das diversas primitivas. Após a execução dessa tarefa, a primitiva que possuir o menor argumento passa a possuir um argumento de valor zero e as outras primitivas passam a possuir um argumento decrescido desta dimensão. Assim, o simulador continua a execução da interpretação, cujo argumento da primitiva "ATRASO" é zero. O simulador executa tal interpretação até que outra primitiva "ATRASO" seja encontrada ou executa a interpretação até o final, caso não haja outra primitiva "ATRASO".

Encontrando-se uma nova primitiva "ATRASO", o procedimento

de comparação repetir-se-á, possibilitando a execução dos outros processadores ou a execução do mesmo, caso esse ainda possua um menor argumento na primitiva "ATRASO".

No fim da execução de uma dada interpretação, o simulador irá executar as primitivas "SAIDA", objetivando a marcação dos arcos de saída do nó de controle associado. A execução da primitiva "SAIDA" implica uma prévia transferência da estrutura do vetor de marcas para a memória do microcomputador. Assim, o simulador, ao executar a primitiva, obtém o nome do arco de controle descrito na primitiva. De posse desse nome, o simulador efetua uma pesquisa na estrutura do vetor de marcas, buscando o arco correspondente. Encontrando o arco, efetua sua marcação com a prioridade contida na primitiva.

Ao término da execução de uma interpretação, o simulador verifica se há processadores, pertencentes a outras estações, executando suas interpretações. Em caso afirmativo, o simulador obtém, para cada processador, o nome da primitiva que está sendo executada, guardando-o no campo 'PMITIVA' da estrutura de nós ativos. Isto feito, a estrutura da interpretação é armazenada no arquivo de dados "CUNHO.CAO", criando com isso a estrutura da interpretação modificada. Esse procedimento é executado devido ao fato de que, durante a execução das primitivas, há a possibilidade se de alterar os argumentos das primitivas "ATRASO". Dessa forma, esse procedimento de criar uma

interpretação alterada se torna relevante, pois, para a execução do próximo passo de simulação, o simulador pode utilizar essa interpretação para dar proceguimento à execução da interpretação dos processadores.

Após a criação da estrutura da interpretação modificada, o simulador elimina da estrutura de nós ativos o nó associado ao processador que tenha terminado a execução da interpretação. Estando atualizada a estrutura de nós ativos, esta é armazenada em memória de massa juntamente com a estrutura dos armazenadores e com a estrutura do vetor de marcas.

Após o armazenamento das estruturas, o simulador incrementa o conteúdo da variável que dita o passo corrente, armazenando em seguida as variáveis de simulação no arquivo de dados "COND.SIM". Terminado esse procedimento, o simulador verifica o conteúdo da variável que dita o passo final de simulação. Sendo esse igual ao passo corrente, o simulador chama o módulo de interface de operação. Caso contrário, o módulo de tratamento do grafo de controle passará a ser executado, dando início a outro passo de simulação.

4 - EXEMPLO DE UTILIZAÇÃO DO SIMULADOR GMB* PARA A VALIDAÇÃO DE PROJETOS

A validação de um projeto de aplicação, que tenha sido modelado através do GMB*, é efetuada gradualmente. Inicialmente, simula-se o modelo utilizando-se uma interpretação contendo apenas primitivas de marcação dos arcos de saída, com o intuito de analisar o comportamento do grafo de controle. Nessa simulação o usuário além de verificar o comportamento do grafo em relação à ocorrência de anomalias como, má terminação e 'deadlock', pode observar a dinâmica de funcionamento do grafo através dos comandos disponíveis no simulador.

Um segundo nível de simulação visa analisar a estrutura de dados utilizando-se as primitivas da interpretação disponíveis no simulador GMB*. Nessa simulação o usuário pode detectar a ocorrência de eventos, tais como: leitura em armazenadores vazios e/ou escritas não-destrutiva em armazenadores cheios. Esses eventos correspondem, portanto, a verificação do comportamento da taxa de produção de informação para os armazenadores do grafo de dados em relação a taxa de consumo dessas, ou vice-versa.

Um terceiro nível de análise é efetuado distribuindo os nós do grafo de controle e os armazenadores do grafo de dados em estações de processamento distintas. Essa distribuição implica em

uma nova validação do modelo, pois o paralelismo existente em um ambiente distribuído pode provocar evoluções diferentes das ocorridas na simulação em um ambiente centralizado.

Este capítulo ilustra a validação de um modelo simplificado através do simulador GMB*. Todavia, deve ser ressaltado que o simulador permite a simulação de situações complexas e importantes, as quais estão associadas à programação de aplicativos com características de concorrência e/ou paralelismo. A alteração dinâmica do modelo, por exemplo, permite a simulação de situações de falhas como a pane em uma (ou mais) estações em um ambiente distribuído.

4.1 Simulação Visando a Validação do Grafo de Controle.

Com base nessa metodologia, primeiramente simula-se o grafo de controle ilustrado na figura 4-1. Para tanto, são definidos os domínios de controle, dados e interpretação. O domínio de controle é composto por nove nós, sendo que a princípio todos os nós se encontram em uma mesma estação, definida como sendo a estação "E1".

O domínio dos dados, ilustrado pela figura 4-2, contém apenas os processadores, pois para esta simulação não há necessidade da utilização de armazenadores.

O domínio da interpretação é composto de primitivas de marcação de saída uma vez que essa simulação objetiva analisar

somente o comportamento do grafo de controle.

Domínio da interpretação

PC1 - SAIDA (A3,1)

PC2 - SAIDA (A5,1)

PC3 - SAIDA (A6,1)

SAIDA (A7,1)

SAIDA (A8,1)

PC4 - SAIDA (A9,1)

PC5 - SAIDA (A10,1)

PC6 - SAIDA (A11,1)

PC7 - SAIDA (A12,1)

PC8 - SAIDA (A13,1)

PC9 - SAIDA (A1,1)

SAIDA (A14,1)

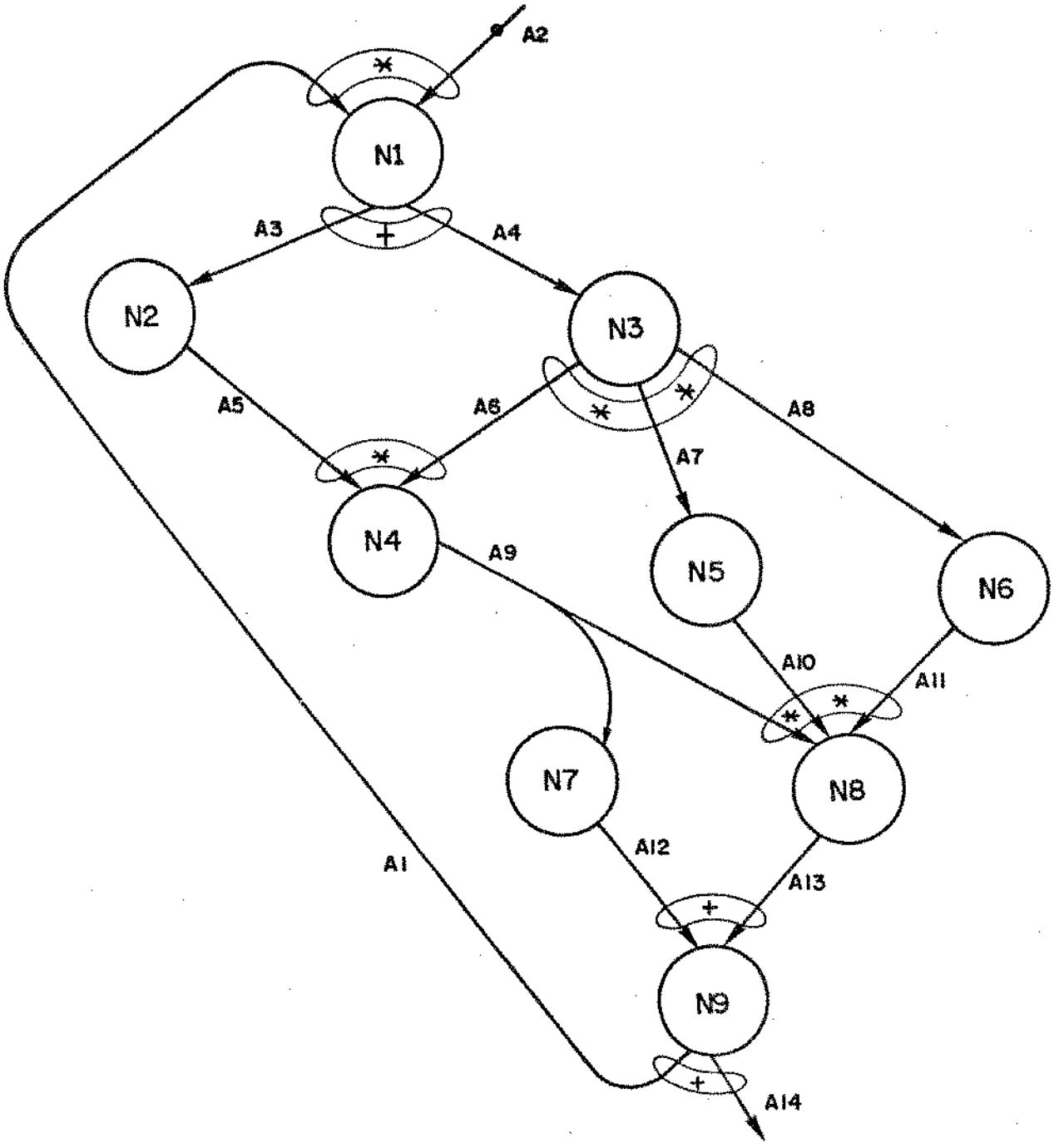


FIGURA 4.1
GRAFO DE CONTROLE

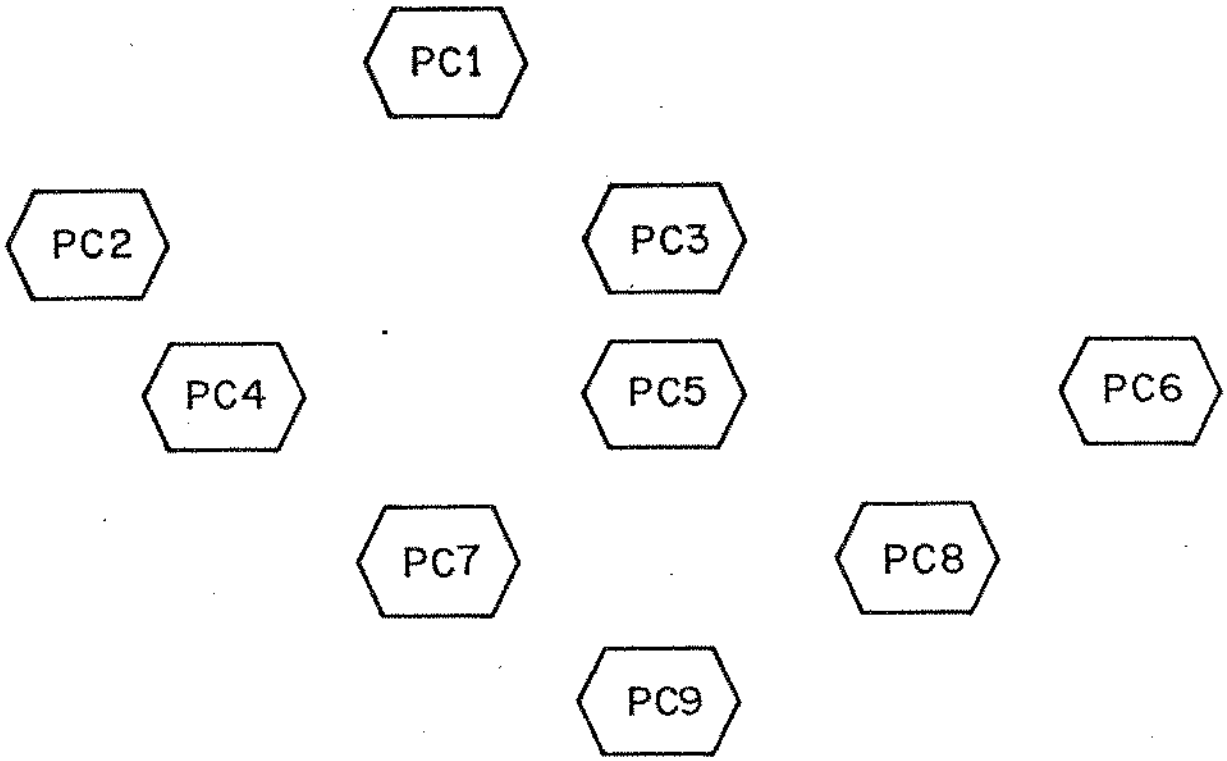


FIGURA 4.2

GRAFO DE DADOS

Associado à definição dos três domínios do modelo GMB*, é necessário criar o vetor de marcas iniciais e definir as variáveis de simulação. Essas variáveis devem ser criadas antes de qualquer outra definição, pois, há necessidade de se especificar o "drive" de armazenamento das estruturas de dados antes da sua definição.

Vetor de Marcas Iniciais.

Constituído pelos quatorze arcos de controle, sendo que inicialmente apenas o arco "A2" possui uma marca de prioridade unitária.

Variáveis de Simulação.

Inicialmente, são definidas as seguintes variáveis de simulação:

- PASSO INICIAL - 0
- PASSO FINAL - 9
- TIPO DE SIMULAÇÃO - A
- DRIVE DESTINO - F

Execução da Simulação

O simulador inicia a execução da simulação ativando o nó "N1"

e eliminando a marca do arco "A2". Ao ser executada a primitiva "SAIDA" existente na interpretação associada ao processador "PC1", o arco "A3" recebe uma marca de prioridade unitária. Findada a execução dessa interpretação, o simulador verifica o conteúdo da variável que determina o passo final de simulação. Sendo o passo de simulação atual menor que o conteúdo da variável analisada, o simulador executa o segundo passo.

Nessa iteração, o nó "N2" é ativado eliminando a marca do arco "A3" e, ao ser executada a primitiva "SAIDA" existente na interpretação associada ao processador "PC2", o arco "A5" recebe uma marca de prioridade unitária.

Após a análise do conteúdo da variável de fim de simulação, o simulador inicia a execução da terceira iteração e detecta um "deadlock", pois não há nenhum nó no grafo de controle em condições de ser ativado, uma vez que apenas o arco "A5" está marcado. Ao detectar o "deadlock", o simulador interrompe a execução da simulação emitindo mensagens que informam ao usuário do ocorrido.

Para eliminar o "deadlock" existente no grafo de controle, é necessário que o grafo de controle e a interpretação do nó "N1" sejam alterados, marcando, por exemplo, o arco "A4" na interpretação associada ao processador "PC1". A figura 4-3 ilustra o grafo de controle alterado.

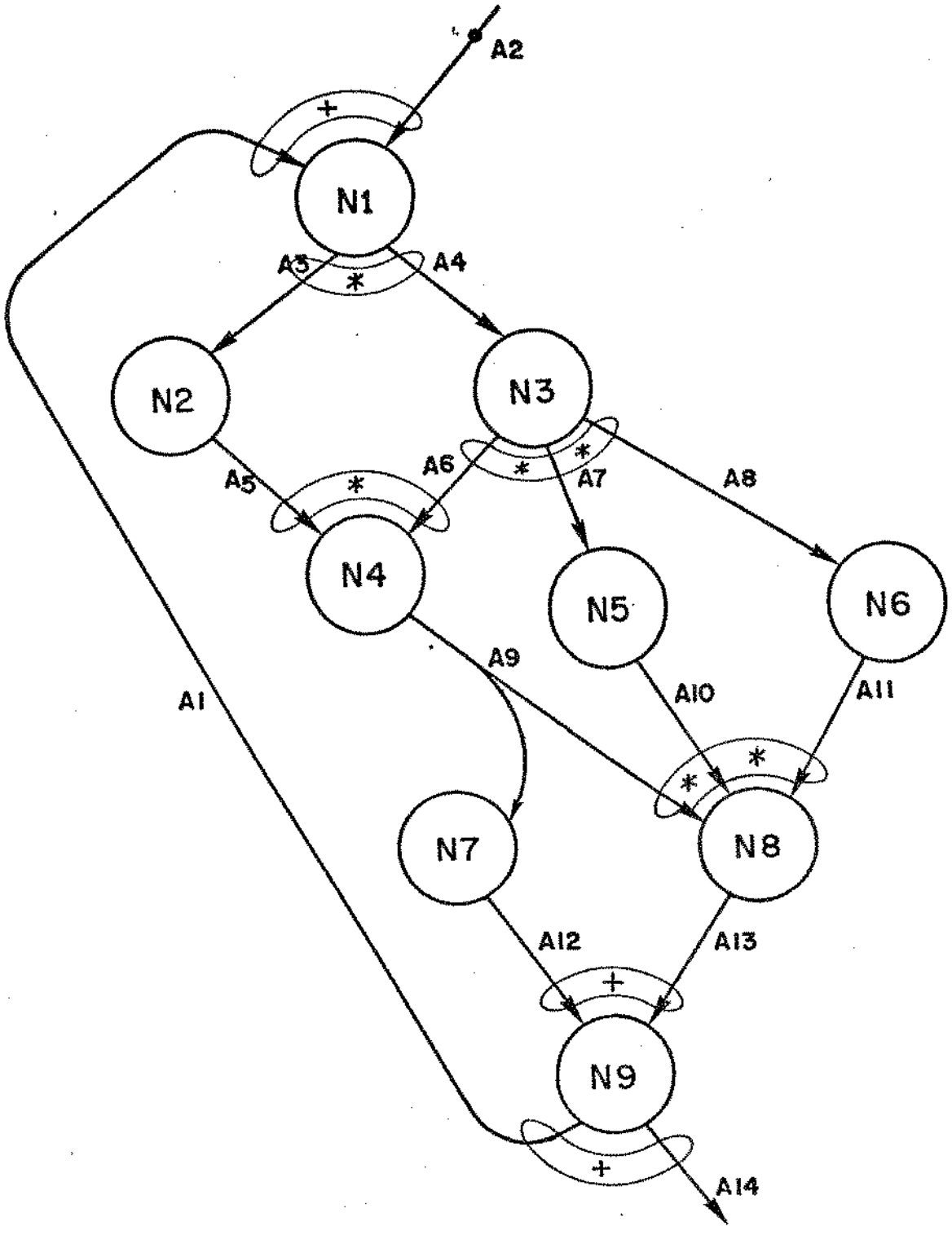


FIGURA 4.3

GRAFO DE CONTROLE

Domínio da Interpretação

PC1 - SAIDA (A3,1)
SAIDA (A4,1)
PC2 - SAIDA (A5,1)
PC3 - SAIDA (A6,1)
SAIDA (A7,1)
SAIDA (A8,1)
PC4 - SAIDA (A9,1)
PC5 - SAIDA (A10,1)
PC6 - SAIDA (A11,1)
PC7 - SAIDA (A12,1)
PC8 - SAIDA (A13,1)
PC9 - SAIDA (A1,1)
SAIDA (A14,1)

Execução da Simulação

Para iniciar a execução da simulação, é necessário que as variáveis de simulação sejam redefinidas, obtendo-se dessa maneira valores iguais ao estado inicial da primeira simulação.

De acordo com a variável de fim de simulação, o simulador executa nove passos de simulação e retorna ao módulo de tela, oferecendo ao usuário a possibilidade de listar as estruturas obtidas ou efetuar alterações nessas. Assim, após a execução dos passos de simulação, listam-se as estruturas da máquina de

'tokens' e vetor de marcas com o intuito de analisar o comportamento do grafo de controle. A tabela 4-1 ilustra as estruturas obtidas.

	ARCOS DE CONTROLE													
	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14
Condição inicial prioridades		1												
Iteração 01 prioridades			1	1										
Iteração 02 prioridades			1			1	1							
Iteração 03 prioridades					1	1	1	1						
Iteração 04 prioridades							1	1	1					
Iteração 05 prioridades								1	1	1				
Iteração 06 prioridades									1	1		1		
Iteração 07 prioridades													1	
Iteração 08 prioridades	1													1
Iteração 09 prioridades			1	1										1

tabela 4-1

Análise das Estruturas Obtidas

Para a obtenção de um estudo do comportamento dinâmico do grafo de controle, efetua-se a análise de cada iteração. Dessa maneira, obtém-se:

Na primeira iteração, o nó "N1" é ativado e, ao término da execução do processador "PC1", os arcos "A3" e "A4" são marcados.

Na segunda iteração, o nó "N3" é ativado. Observa-se que tanto o nó "N2" quanto o nó "N3" são habilitados, entretanto, devido ao fato dos nós possuirem prioridade iguais e o modelo GMB* permitir somente que um nó, por estação, seja ativado em um determinado intervalo de tempo, apenas o nó "N3" é ativado, ficando o nó "N2" no estado habilitado. Ao término da execução do processador "PC3", os arcos "A6", "A7" e "A8" são marcados.

Na terceira iteração, o nó "N2" é ativado e ao término da execução do processador "PC2", o arco "A5" é marcado.

Na quarta iteração os nós "N4", "N5" e "N6" são habilitados, todavia, apenas o nó "N4" é ativado, uma vez que possui uma expressão lógica de entrada cuja prioridade é igual à soma das prioridades dos arcos "A5" e "A6". Ao término da execução do processador "PC4", o arco "A9" é marcado.

Na quinta iteração, o nó "N5" é ativado, ficando os nós "N6" e "N7" no estado habilitado, pois todos os nós possuem prioridades unitárias e os nós "N5" e "N6" já se encontravam habilitados no passo anterior. Ao término da execução do

processador "PC5", o arco "A10" é marcado.

Na sexta iteração, o simulador ativa o nó "N6" e o nó "N7" permanece no estado habilitado, pois o nó "N6" foi habilitado antes do nó "N7". Ao término da execução do processador "PC6", o arco "A11" é marcado.

Na sétima iteração, o nó "N8" é ativado, pois a soma das prioridades dos arcos de entrada do nó "N8" é maior que a prioridade do nó "N7". Com a ativação do nó "N8", o nó "N7" sai do estado habilitado, uma vez que o arco "A9" é um arco complexo e sua única marca é utilizada para a ativação do nó "N8". Ao término da execução do processador "PC8", o arco "A13" recebe uma marca.

Na oitava iteração, o nó "N9" é ativado e, ao término da execução do processador "PC9", os arcos "A1" e "A14" são marcados. A marcação do arco "A1" habilita o nó "N1", reiniciando dessa maneira o fluxo da simulação.

Findado o estudo do comportamento dinâmico do grafo de controle, observa-se que ele não apresenta anomalias. Entretanto, a existência de um arco complexo (arco "A9"), efetuando a exclusão mútua entre os nós "N7" e "N8", associado ao fato de que todas as marcas possuem prioridades unitárias, faz com que o nó "N7" não venha a ser executado em nenhuma das iterações da simulação.

Objetivando-se a ativação do nó "N7" altera-se o domínio da interpretação associada ao processador "PC4", aumentando-se a

prioridade da primitiva de marcação "SAIDA".

Domínio da interpretação.

PC1 - SAIDA (A3,1)
SAIDA (A4,1)
PC2 - SAIDA (A5,1)
PC3 - SAIDA (A6,1)
SAIDA (A7,1)
SAIDA (A8,1)
PC4 - SAIDA (A9,2)
PC5 - SAIDA (A10,1)
PC6 - SAIDA (A11,1)
PC7 - SAIDA (A12,1)
PC8 - SAIDA (A13,1)
PC9 - SAIDA (A1,1)
SAIDA (A14,1)

Uma vez efetuada a alteração no domínio da interpretação, redefinem-se as variáveis de simulação de forma similar à simulação anterior, alterando-se apenas a variável de final de simulação, pois, nesse caso, o simulador executa dezessete iterações.

Execução da Simulação

De forma similar ao caso anterior, o simulador executa os passos de simulação e, ao término da execução, retorna ao módulo de tela oferecendo ao usuário a possibilidade de examinar as estruturas obtidas. A tabela 4-2 ilustra as estruturas obtidas.

ARCOS DE CONTROLE

	'A1'	'A2'	'A3'	'A4'	'A5'	'A6'	'A7'	'A8'	'A9'	'A10'	'A11'	'A12'	'A13'	'A14'
Condição inicial prioridades		1												
Iteração 01 prioridades			1	1										
Iteração 02 prioridades			1			1	1	1						
Iteração 03 prioridades					1	1	1	1						
Iteração 04 prioridades							1	1	2					
Iteração 05 prioridades							1	1				1		
Iteração 06 prioridades								1		1		1		
Iteração 07 prioridades										1	1	1		
Iteração 08 prioridades	1									1	1			1
Iteração 09 prioridades			1	1						1	1			1
Iteração 10 prioridades			1			1	1	1		1	1			1
Iteração 11 prioridades					1	1	1	1		1	1			1
Iteração 12 prioridades							1	1	2	1	1			1
Iteração 13 prioridades							1	1					1	1
Iteração 14 prioridades							1				1		1	1

Iteração 15							
prioridades				1	1	1	1
Iteração 16							
prioridades	1			1	1		1,1
Iteração 17							
prioridades		1	1	1	1		1,1

tabela 4-2

Análise das Estruturas Obtidas

Através da análise do conteúdo da tabela 4-2 observa-se que, na quinta iteração, o nó "N7" é ativado, ficando os nós "N5" e "N6" no estado habilitado. Todavia, no decorrer da simulação do modelo, observa-se que na décima terceira iteração os nós "N5", "N6", "N7" e "N8" estão habilitados. Entretanto somente o nó "N8" é ativado, permanecendo os nós "N5" e "N6" no estado habilitado. O nó "N7" é desabilitado, pois a marca existente no arco complexo "A9" é consumida na ativação do nó "N8".

Com a execução dessa simulação, conclui-se que apesar da prioridade da marca associada ao arco complexo "A9", o no "N7" somente será ativado uma vez no decorrer da simulação, pois a sua evolução faz com que o nó "N8" seja ativado, desabilitando o nó "N7". Uma possilidade de tornar o nó "N7" apto para a execução mantendo a exclusão mútua com o nó "N8" pode ser obtida por intermédio do grafo mostrado na figura 4-4. Nesse grafo, a exclusão entre os nós é obtida através do arco "A13".

4.2 Simulação Visando à Validação do Grafo de Dados

Estando validado o grafo de controle, passa-se para a análise do comportamento do grafo de dados. Tal estudo é efetuado objetivando-se a análise do comportamento dos

armazenadores existentes no domínio dos dados. Para a realização dessa simulação, é necessário que o grafo de dados, utilizado nas simulações anteriores, seja substituído pelo grafo ilustrado pela figura 4-5. Acrescente-se que o domínio da interpretação é também alterado e que o grafo de controle não sofre modificações, mantendo a mesma estrutura validada anteriormente.

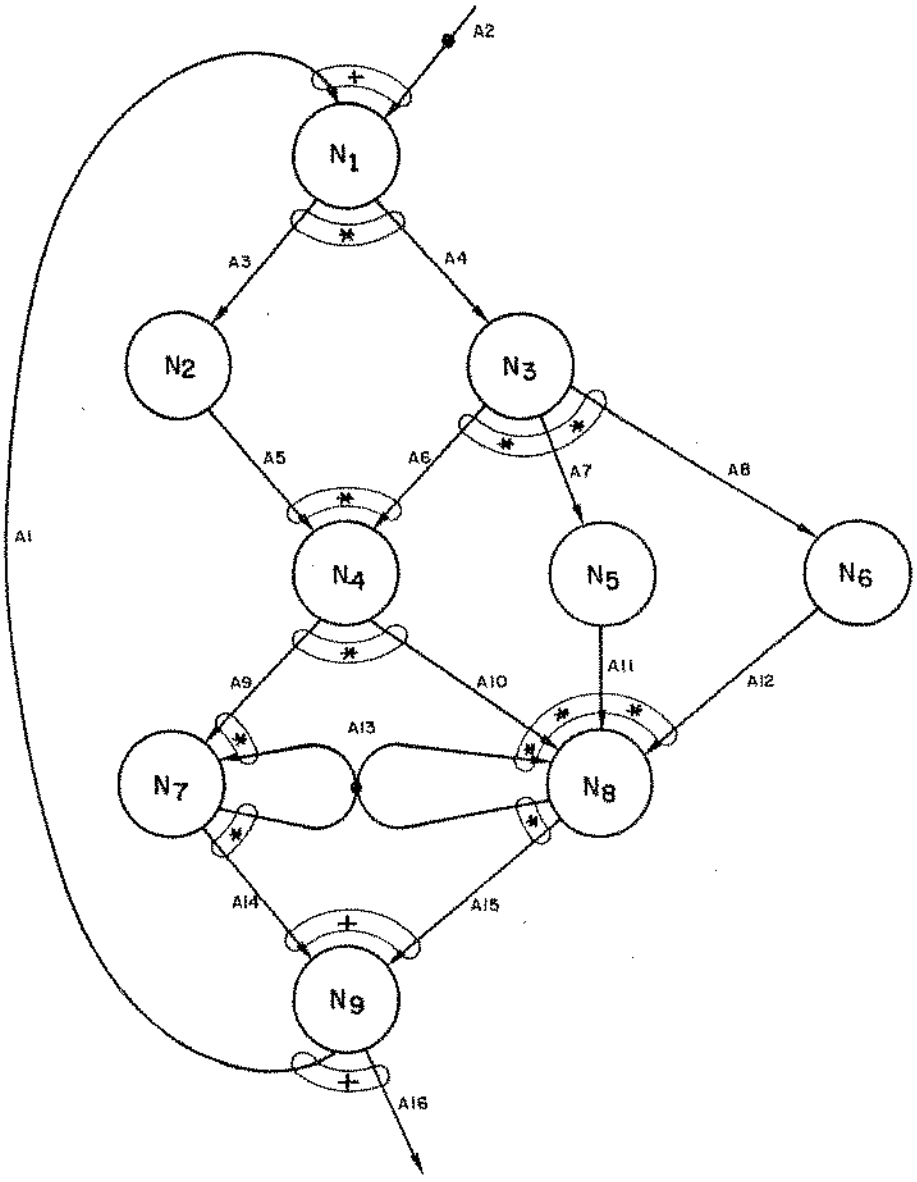


FIGURA 4.4
GRAFO DE CONTROLE

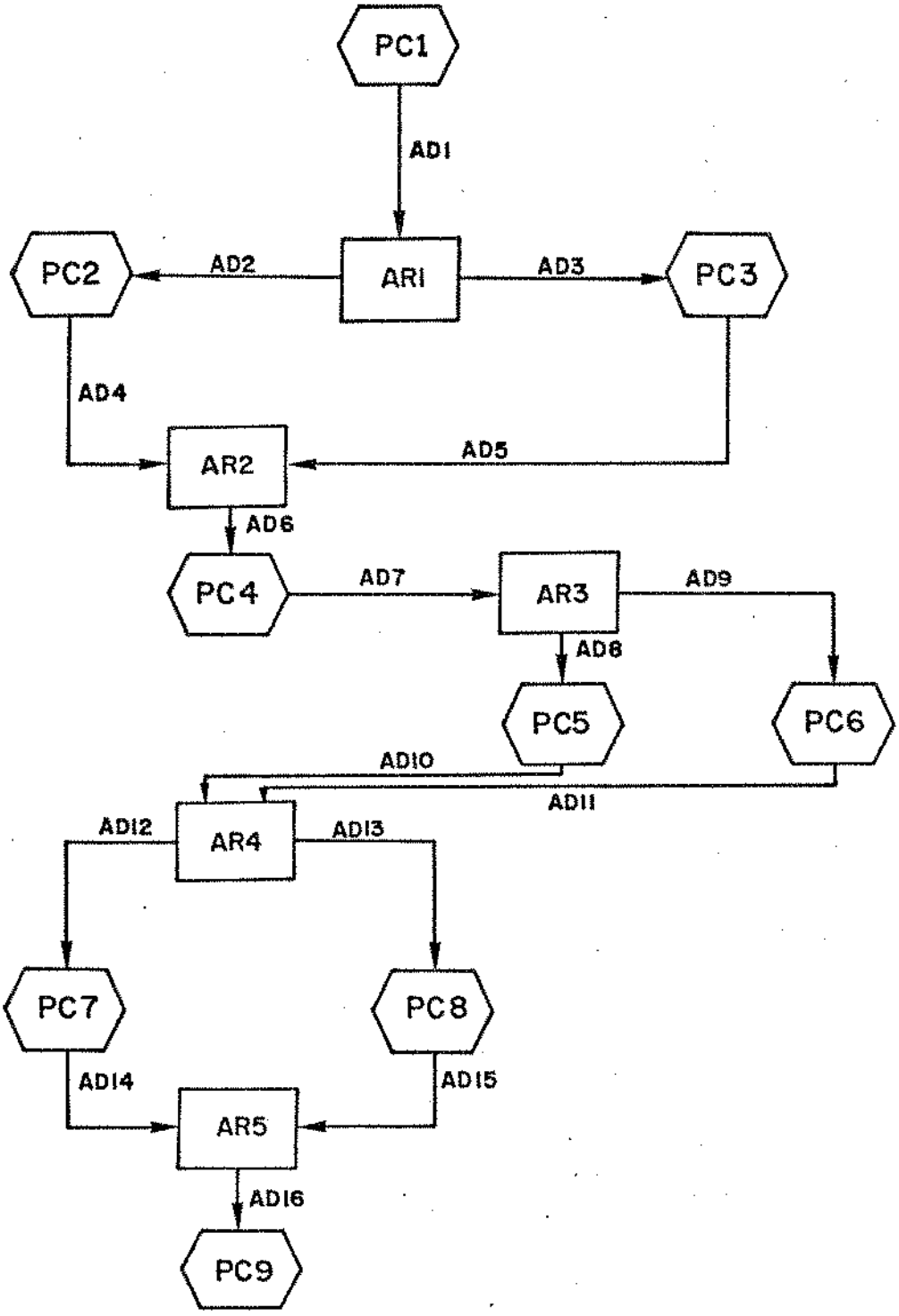


FIGURA 4.5
GRAFO DE DADOS

Associado à definição do grafo de dados, ilustrado pela figura 4-5, faz-se necessário especificar os armazenadores existentes.

Especificação dos Armazenadores

ARMAZENADOR	AR1
ESTAÇÃO	E1
TIPO	F
TAMANHO DO ARMAZENADOR	10
TAMANHO DA UNIDADE DE ARMAZENAMENTO	5
ARMAZENADOR	AR2
ESTAÇÃO	E1
TIPO	V
TAMANHO DO ARMAZENADOR	20
NUMERO DE MENSAGENS	1
TOTAL OCUPADO	10
TAMANHO DA MENSAGEM 1	10
ARMAZENADOR	AR3
ESTAÇÃO	E1
TIPO	F
TAMANHO DO ARMAZENADOR	15
TAMANHO DA UNIDADE DE ARMAZENAMENTO	5

NUMERO DE MENSAGENS	1
TOTAL OCUPADO	5
ARMAZENADOR	AR4
ESTAÇÃO	E1
TIPO	V
TAMANHO DO ARMAZENADOR	50
NUMERO DE MENSAGENS	1
TOTAL OCUPADO	5
TAMANHO DA MENSAGEM 1	5
ARMAZENADOR	AR5
ESTAÇÃO	E1
TIPO	F
TAMANHO DO ARMAZENADOR	5
TAMANHO DA UNIDADE DE ARMAZENAMENTO	1

Domínio da Interpretação

PC1 - ESCREVERF (AD1,D)
SAIDA (A3,1)
SAIDA (A4,1)

PC2 - LERF (AD2,D)
ESCREVERV (AD4,5,D)
SAIDA (A5,1)

```
PC3 - LERF (AD3,D)
      ESCREVERV (AD5,10,D)
      SAIDA (A6,1)
      SAIDA (A7,1)
      SAIDA (A8,1)
PC4 - LERV (AD6,D)
      ESCREVERF (AD7,D)
      SAIDA (A9,2)
PC5 - LERF (AD8,N)
      ESCREVERV (AD10,2,D)
      SAIDA (A10,1)
PC6 - LERF (AD9,D)
      ESCREVERV (AD11,4,D)
      SAIDA (A11,1)
PC7 - LERV (AD12,D)
      ESCREVERF (AD14,N)
      SAIDA (A12,1)
PC8 - LERV (AD13,D)
      ESCREVERF (AD15,D)
      SAIDA (A13,1)
PC9 - LERF (AD16,D)
      SAIDA (A1,1)
      SAIDA (A14,1)
```

De forma análoga à simulação anterior são definidas as variáveis de simulação, obtendo-se:

- PASSO INICIAL - 0
- PASSO FINAL - 9
- TIPO DE SIMULAÇÃO - A
- DRIVE DESTINO - F

Execução da Simulação

O simulador inicia a primeira iteração ativando o nó "N1", o que acarreta a execução do processador "PC1". Ao ser executada a primitiva "ESCREVERF", o conteúdo do armazenador "AR1" é alterado.

Na segunda iteração, o nó "N3" é ativado, sendo executada a interpretação associada ao processador "PC3". A leitura destrutiva efetuada pela primitiva "LERF" deixa o armazenador "AR1" vazio.

Na terceira iteração, o nó "N2" que estava habilitado é ativado e, ao ser executada a primitiva "LERF", o simulador detecta uma inconsistência no grafo de dados, pois o armazenador "AR1" está vazio, não havendo a possibilidade de se efetuar leituras.

Ao detectar tal inconsistência, o simulador interrompe a simulação e envia a seguinte mensagem ao usuário:

ERRO NO ACESSO DE LEITURA, A PRIMITIVA LERF (AD2,D), PERTENCENTE AO PROCESSADOR PC2, ESTA ACESSANDO O ARMAZENADOR AR1 QUE ENCONTRA-SE VAZIO.

Para eliminar a inconsistência ocorrida no armazenador "AR1", altera-se a interpretação associada ao processador "PC2" e a interpretação associada ao "PC3" . Tal alteração consiste na substituição da primitiva de leitura fixa por uma que faça o acesso de maneira não destrutiva. É necessário alterar as duas interpretações, pois os nós associados possuem a mesma prioridade e se encontram na mesma estação indicando, portanto, a possibilidade de execução de qualquer um dos nós através da escolha pseudoaleatória.

Domínio da interpretação

```
PC1 - ESCREVERF (AD1,D)
      SAIDA (A3,1)
      SAIDA (A4,1)
PC2 - LERF (AD2,N)
      ESCREVERV (AD4,5,D)
      SAIDA (A5,1)
PC3 - LERF (AD3,N)
      ESCREVERV (AD5,10,D)
      SAIDA (A6,1)
```

```
      SAIDA (A7,1)
      SAIDA (A8,1)
PC4 - LERV (AD6,D)
      ESCREVERF (AD7,D)
      SAIDA (A9,2)
PC5 - LERF (AD8,N)
      ESCREVERV (AD10,2,D)
      SAIDA (A10,1)
PC6 - LERF (AD9,D)
      ESCREVERV (AD11,4,D)
      SAIDA (A11,1)
PC7 - LERV (AD12,D)
      ESCREVERF (AD14,N)
      SAIDA (A12,1)
PC8 - LERV (AD13,D)
      ESCREVERF (AD15,D)
      SAIDA (A13,1)
PC9 - LERF (AD16,D)
      SAIDA (A1,1)
      SAIDA (A14,1)
```

Estando alterado o domínio da interpretação, redefinem-se as variáveis de simulação e, partindo-se das condições iniciais, executa-se novamente a simulação.

Execução da Simulação

Nessa simulação são executados cento e setenta e cinco passos de simulação, sendo que nos nove primeiros faz-se a simulação passo a passo. E, em seguida, executa-se as simulações até cinquenta, cem, cento e cinquenta e cento e setenta e cinco. Ao término de cada simulação, examina-se a estrutura dos armazenadores para a análise do comportamento dos mesmos. A tabela 4-3 ilustra as informações obtidas através da estrutura dos armazenadores.

ARMAZENADORES

	AR1	AR2	AR3	AR4	AR5
Estação	E1	E1	E1	E1	E1
Tipo	F	V	F	V	F
Tamanho	10	20	15	50	5
Tamanho da Unidade de Armazenamento	5	-	5	-	1
Condições Iniciais número de mensagens	0	1	1	1	0
total ocupado	0	10	5	5	0
tamanho das mensagens	-	10	-	5	-
Iteração 01 número de mensagens	1	1	1	1	0
total ocupado	5	10	5	5	0
tamanho das mensagens	-	10	-	5	-
Iteração 02 número de mensagens	1	2	1	1	0
total ocupado	5	15	5	5	0
tamanho das mensagens	-	10, 5	-	5	-
Iteração 03 número de mensagens	1	2	1	1	0
total ocupado	5	15	5	5	0
tamanho das mensagens	-	5, 10	-	5	-
Iteração 04 número de mensagens	1	1	2	1	0
total ocupado	5	10	10	5	0
tamanho das mensagens	-	10	-	5	-

Iteração 05					
número de mensagens	1	1	2	0	1
total ocupado	5	10	10	0	1
tamanho das mensagens	-	10	-	-	-
Iteração 06					
número de mensagens	1	1	2	1	1
total ocupado	5	10	10	2	1
tamanho das mensagens	-	10	-	2	-
Iteração 07					
número de mensagens	1	1	1	2	1
total ocupado	5	10	5	6	1
tamanho das mensagens	-	10	-	2, 4	-
Iteração 08					
número de mensagens	1	1	1	2	0
total ocupado	5	10	5	6	0
tamanho das mensagens	-	10	-	2, 4	-
Iteração 09					
número de mensagens	2	1	1	2	0
total ocupado	10	10	5	6	0
tamanho das mensagens	-	10	-	2, 4	-

.
.

Iteração 50					
número de mensagens	2	2	1	7	0
total ocupado	10	10	5	20	0
tamanho das mensagens	-	5,5	-	2,4,2, 2,4,2,4	-

.
.

Iteração 100					
número de mensagens	2	1	2	13	0
total ocupado	10	5	10	40	0
tamanho das mensagens	-	5	-	4,4,2, 4,2,2,4 4,2,4,2 2,4	-

.
.
.

Iteração 150					
número de mensagens	2	1	2	17	1
total ocupado	10	10	10	50	1
tamanho das mensagens	-	10	-	4,2,2, 4,2,4,4 2,4,2,2 4,2,4,4 2,2	-

.

.
.
.
.

iteração 175 número de mensagens	2	1	2	17	0
total ocupado	10	10	10	50	0
tamanho das mensagens	-	10	-	2,4,2, 2,4,2,4 4,2,2,4 2,4,4,2 4,2	-

tabela 4-3

Análise da Estrutura Obtida

Ao analisar a tabela 4-3, observa-se que no instante inicial da simulação o armazenador "AR2" possui uma mensagem de tamanho "10" e o armazenador "AR4" possui uma mensagem de tamanho "5".

Na primeira iteração, o nó "N1" é ativado, implicando na execução do processador "PC1", que através de sua interpretação altera o conteúdo do armazenador "AR1".

Na segunda iteração o nó "N2" é ativado e, através da execução da interpretação associada ao processador "PC2", o conteúdo do armazenador "AR2" é alterado.

Na terceira iteração, o nó "N3" é ativado, sendo executada a interpretação associada ao processador "PC3", alterando novamente o conteúdo do armazenador "AR2".

Na quarta iteração, o nó "N4" é ativado e, através da execução da interpretação associada ao processador "PC4", os conteúdos dos armazenadores "AR2" e "AR3" são alterados.

Na quinta iteração, o nó "N7" é ativado, sendo executada a interpretação associada ao processador "PC7", alterando dessa forma os conteúdos dos armazenadores "AR4" e "AR5".

Na sexta iteração o nó "N5" é ativado, sendo executada a interpretação associada ao processador "PC5", alterando dessa maneira o conteúdo do armazenador "AR4".

Na sétima iteração, o nó "N6" é ativado e, através da execução da interpretação associada ao processador "PC6", os

conteúdos dos armazenadores "AR3" e "AR4" são alterados.

Na oitava iteração, o nó "N9" é ativado sendo executada a interpretação associada ao processador "PC9", alterando dessa maneira o conteúdo do armazenador "AR5".

Na nona iteração, o nó "N1" é reativado, repetindo-se a alteração ocorrida na primeira iteração.

Posteriormete com a execução dos cento e setenta e cinco passos de simulação, observou-se que o grafo de dados entra em regime não apresentando inconsistências nos armazenadores. Todavia, acrescente-se que o armazenador 'AR4' possui uma taxa de entrada de mensagens maior do que a taxa de saída e não houve problema de 'estouro' do mesmo, porque os acessos de escrita neles são do tipo destrutivo.

4.3 Simulação Visando a Validação do Modelo em um Ambiente Distribuído

Para a execução dessa simulação, altera-se a disposição dos nós do grafo de controle, distribuindo-os em três estações distintas. Dessa forma, os nós "N1", "N2", "N3" e "N4" são alocados na estação "E1", os nós "N5" e "N6" na estação "E2" e os nós "N7", "N8" e "N9" na estação "E3". Altera-se também a disposição dos armazenadores e dos processadores na estrutura de dados e o domínio da interpretação, uma vez que nessa simulação são utilizadas todas as primitivas disponíveis no simulador.

Domínio da interpretação.

```
PC1 - ATRASO (5)
      ESCREVERF (AD1,D)
      SAIDA (A3,1)
      SAIDA (A4,1)
PC2 - IF (AD2,=,5) THEN
      ATRASO (5)
      LERF (AD2,N)
      ESCREVERV (AD4,5,D)
      ELSE
      ATRASO (7)
      ESCREVERV (AD4,2,D)
      ENDIF
      SAIDA (A5,1)
PC3 - ATRASO (10)
      LERF (AD3,N)
      ESCREVERV (AD5,10,D)
      SAIDA (A6,1)
      SAIDA (A7,1)
      SAIDA (A8,1)
PC4 - IF (AD6,=,5) THEN
      ATRASO (5)
```

```
        ESCREVERF (AD7,N)
        LERV (AD6,N)
        ELSE
        ATRASO (5)
        LERV (AD6,N)
        ATRASO (5)
        ESCREVERF (AD7,D)
        ENDIF
        SAIDA (A9,2)
PC5 - ATRASO (15)
        LERF (AD8,N)
        ATRASO (5)
        ESCREVERV (AD10,2,N)
        SAIDA (A10,1)
PC6 - ATRASO (5)
        LERF (AD9,D)
        ESCREVERV (AD11,4,N)
        SAIDA (A11,1)
PC7 - ATRASO (5)
        IF (AD12,=,5) THEN
        ATRASO (10)
        LERV (AD12,D)
        IF (AD14,=,2) THEN
        ATRASO (15)
```

```
        ESCREVERF (AD14,N)
        ESCREVERF (AD14,N)
        ELSE
        ATRASO (5)
        ESCREVERF (AD14,N)
        ENDIF
        ENDIF
        SAIDA (A12,1)
PC8 -   ATRASO (5)
        LERV (AD13,D)
        ESCREVERF (AD15,D)
        SAIDA (A13,1)
PC9 -   IF (AD16,=,2) THEN
        LERF (AD16,D)
        ATRASO (10)
        SAIDA (A14,1)
        ELSE
        LERF (AD16,N)
        ATRASO (5)
        ENDIF
        SAIDA (A1,1)
```

Especificação dos Armazenadores

ARMAZENADOR	AR1
ESTAÇÃO	E1
TIPO	F
TAMANHO DO ARMAZENADOR	10
TAMANHO DA UNIDADE DE ARMAZENAMENTO	5
ARMAZENADOR	AR2
ESTAÇÃO	E1
TIPO	V
TAMANHO DO ARMAZENADOR	20
NUMERO DE MENSAGENS	1
TOTAL OCUPADO	10
TAMANHO DA MENSAGEM 1	10
ARMAZENADOR	AR3
ESTAÇÃO	E2
TIPO	F
TAMANHO DO ARMAZENADOR	15
TAMANHO DA UNIDADE DE ARMAZENAMENTO	5
NUMERO DE MENSAGENS	1
TOTAL OCUPADO	5
ARMAZENADOR	AR4

ESTAÇÃO	E3
TIPO	V
TAMANHO DO ARMAZENADOR	50
NUMERO DE MENSAGENS	1
TOTAL OCUPADO	5
TAMANHO DA MENSAGEM 1	5
ARMAZENADOR	AR5
ESTAÇÃO	E3
TIPO	F
TAMANHO DO ARMAZENADOR	5
TAMANHO DA UNIDADE DE ARMAZENAMENTO	1

A utilização das primitivas "ATRASO" na interpretação, possibilita ao usuário estimar o tempo de execução de cada primitiva ou de um conjunto de primitivas existente na interpretação de um determinado processador.

Execução da Simulação

Nessa simulação são executados cento e vinte e três passos, sendo que nos quatorze primeiros faz-se a simulação passo a passo. E, em seguida, executa-se as simulações até o passo cinquenta, setenta e cinco, cem e cento e vinte e três. Ao

término de cada simulação são utilizados os comandos de listagem do simulador para a obtenção da análise das estruturas do vetor de marcas, máquina de tokens, conteúdo dos armazenadores, nós habilitados e nós ativos.

A tabela 4-4 ilustra a variação no conteúdo dos armazenadores e a tabela 4-5 ilustra a evolução dinâmica das marcas dos arcos do grafo de controle, após cada passo de simulação.

ARMAZENADORES

	AR1	AR2	AR3	AR4	AR5
Estação	E1	E1	E2	E3	E3
Tipo	F	V	F	V	F
Tamanho	10	20	15	50	5
Tamanho da Unidade de Armazenamento	5	-	5	-	1
Condições Iniciais número de mensagens	0	1	1	1	0
total ocupado	0	10	5	5	0
tamanho das mensagens	-	10	-	5	-
Iteração 01 número de mensagens	1	1	1	1	0
total ocupado	5	10	5	5	0
tamanho das mensagens	-	10	-	5	-
Iteração 02 número de mensagens	1	2	1	1	0
total ocupado	5	15	5	5	0
tamanho das mensagens	-	10, 5	-	5	-
Iteração 03 número de mensagens	1	2	1	1	0
total ocupado	5	15	5	5	0
tamanho das mensagens	-	5, 10	-	5	-
Iteração 04 número de mensagens	1	2	2	1	0
total ocupado	5	15	10	5	0
tamanho das mensagens	-	5, 10	-	5	-

Iteração 05					
número de mensagens	1	2	2	2	0
total ocupado	5	15	10	7	0
tamanho das mensagens	-	5, 10	-	5, 2	-
Iteração 06					
número de mensagens	1	2	1	2	0
total ocupado	5	15	5	6	0
tamanho das mensagens	-	5, 10	-	2, 4	-
Iteração 07					
número de mensagens	1	2	1	2	1
total ocupado	5	15	5	6	1
tamanho das mensagens	-	5, 10	-	2, 4	-
Iteração 08					
número de mensagens	1	2	1	2	1
total ocupado	5	15	5	6	1
tamanho das mensagens	-	5, 10	-	2, 4	-
Iteração 09					
número de mensagens	2	2	1	2	1
total ocupado	10	15	5	6	1
tamanho das mensagens	-	5, 10	-	2, 4	-
Iteração 10					
número de mensagens	2	3	1	2	1
total ocupado	10	17	5	6	1
tamanho das mensagens	-	5,10,2	-	2, 4	-
Iteração 11					
número de mensagens	2	2	1	2	1
total ocupado	10	12	5	6	1
tamanho das mensagens	-	2, 10	-	2, 4	-

Iteração 12					
número de mensagens	2	2	2	2	1
total ocupado	10	12	10	6	1
tamanho das mensagens	-	2, 10	-	2, 4	-
Iteração 13					
número de mensagens	2	2	2	1	2
total ocupado	10	12	10	4	2
tamanho das mensagens	-	2, 10	-	4	-
Iteração 14					
número de mensagens	2	2	2	2	1
total ocupado	10	12	10	6	1
tamanho das mensagens	-	2, 10	-	4, 2	-

.
.
.
.
.

Iteração 50					
número de mensagens	2	3	1	7	1
total ocupado	10	14	5	20	1
tamanho das mensagens	-	10,2,2	-	2,2,4,4 2,4,2	-

.
.
.
.
.

.
.
.
.

Iteração 75 número de mensagens	2	2	1	11	1
total ocupado	10	12	5	34	1
tamanho das mensagens	-	10,2	-	4,2,4,2 2,4,2,4 4,2,4	-

.
.
.
.

Iteração 100 número de mensagens	2	2	1	14	1
total ocupado	10	12	5	42	1
tamanho das mensagens	-	10,2	-	2,4,2,4 4,2,4,2 2,4,2,4 4,2	-

tabela 4-4

ARCOS DE CONTROLE

	'A1	'A2	'A3	'A4	'A5	'A6	'A7	'A8	'A9	'A10	'A11	'A12	'A13	'A14
Condição Inicial prioridades		1												
Iteração 01 prioridades			1	1										
Iteração 02 prioridades				1	1									
Iteração 03 prioridades					1	1	1	1						
Iteração 04 prioridades								1	2					
Iteração 05 prioridades								1		1				
Iteração 06 prioridades										1	1			
Iteração 07 prioridades										1	1	1		
Iteração 08 prioridades	1									1	1			
Iteração 09 prioridades			1	1						1	1			
Iteração 10 prioridades				1	1					1	1			
Iteração 11 prioridades					1	1	1	1		1	1			
Iteração 12 prioridades								1	2	1	1			
Iteração 13 prioridades								1					1	
Iteração 14 prioridades								1		1				

.
.
.
.
.

Iteração 50								
prioridades		1	1			1	1	1,1,1
								1,1

.
.
.
.

Iteração 75								
prioridades					2	1	1,1	1,1,1
								1,1,1
								1,1

.
.
.
.

Iteração 100								
prioridades					1	1		1,1,1
								1,1,1
								1,1,1
								1,1,1

tabela 4-5

Análise das Estruturas Obtidas

Na primeira iteração, o nó "N1" é ativado e, através da execução da interpretação, associada ao processador "PC1", o conteúdo do armazenador "AR1" é alterado.

Na segunda iteração, o nó "N2" é ativado e ao ser executada a primitiva condicional pertencente à interpretação do processador "PC2", a quantidade de espaço ocupado no armazenador "AR1" é analisada. Sendo igual a cinco unidades, o teste condicional é satisfeito e, por conseguinte, as primitivas existentes no intervalo relacionado com a primitiva condicional "THEN" são executadas. A execução de tais primitivas alteram o conteúdo do armazenador "AR2".

Na terceira iteração de simulação, o nó "N3", que estava no estado habilitado passa para o estado ativo e, através da execução das primitivas, associadas ao processador "PC3", o conteúdo do armazenador "AR2" é alterado.

Na quarta iteração, os nós "N4", "N5" e "N6" são habilitados, sendo ativados os nós "N5" e "N4", uma vez que se encontram em estações de processamento distintas. Acrescenta-se que o nó "N6" permanece no estado habilitado, uma vez que esse possui a mesma prioridade do nó "N5" e ambos pertencem a mesma estação de processamento. A ativação dos nós "N4" e "N5" acarreta na execução dos processadores "PC4" e "PC5", respectivamente. O simulador executa as interpretações intercaladamente até que uma

primitiva de atraso seja encontrada. Dessa forma, ao ser executada a primitiva condicional pertencente à interpretação do processador "PC4", a quantidade de espaço ocupado no armazenador "AR2" é verificada e comparada com o valor cinco; sendo diferentes, o simulador obtém a informação de que as primitivas existentes no intervalo associado à primitiva "ELSE" deverão ser executadas. Tendo executado uma primitiva associada ao processador "PC4", o simulador passa para a execução da interpretação associada ao processador "PC5". Sendo a primeira primitiva um "ATRASO", o simulador retorna à execução da interpretação associada ao processador "PC4. Ao iniciar a execução das primitivas pertencentes ao intervalo vinculado à primitiva "ELSE", o simulador encontra uma primitiva de atraso. Com isso, o simulador busca, dentre as duas primitivas de atraso, qual a que possui o menor tempo associado, zera-o e subtrai esse valor da que possui o tempo maior. Dessa forma a primitiva "ATRASO", associada à interpretação do processador "PC4", tem seu tempo zerado, e a primitiva, associada à interpretação do processador "PC5", passa a possuir um atraso de dez unidades de tempo. Tendo executada essa operação, o simulador retorna à execução da interpretação, associada ao processador "PC4", até que outra primitiva de atraso seja encontrada. Ao encontrar uma outra primitiva "ATRASO", o simulador repete a operação anterior, zerando o tempo associado à primitiva pertencente à interpretação do processador "PC4" e decrementando de cinco unidades o tempo

associado à primitiva pertencente ao processador "PC5". Após esse processamento, o simulador termina a execução da interpretação do processador "PC4", alterando o conteúdo do armazenador "AR3". Através da análise da estrutura de nós ativos, observa-se que o processador "PC5" permanece em execução, processando a primitiva "ATRASO (5)".

Na quinta iteração, o nó "N7" é ativado. Com isso, os processadores "PC5" e "PC7" têm suas interpretações executadas. Devido ao fato de o processador "PC5" estar executando a primitiva "ATRASO (5)", o simulador passa para a execução da primeira primitiva pertencente à interpretação do processador "PC7". Sendo essa uma primitiva "ATRASO (5)", o simulador zera o tempo associado às primitivas, passando a executar uma primitiva de cada interpretação até que uma nova primitiva de atraso seja encontrada. Ao executar a primitiva condicional existente na interpretação associada ao processador "PC7", o simulador testa a quantidade de espaço ocupado no armazenador "AR4", sendo esse igual a cinco unidades o simulador obtém a informação de que as primitivas contidas no intervalo vinculado à primitiva condicional "THEN" devem ser executadas. Após esse processamento, o simulador retorna à execução da interpretação associada ao processador "PC5", encontrando uma primitiva de atraso. Com isso, o simulador retorna à execução da interpretação associada ao processador "PC7" e, ao detectar a existência de uma primitiva de

atraso, o simulador zera a unidade de tempo da primitiva pertencente à interpretação do processador "PC5" e decrementa de cinco unidades a primitiva associada à interpretação do processador "PC6". Após, o simulador termina a execução da interpretação do processador "PC5", alterando o conteúdo do armazenador "AR4". Acrescente-se que o processador "PC7" permanece em execução, processando a primitiva atraso (5).

Na sexta iteração, o nó "N6" passa para o estado ativo. Com isso, a interpretação do processador "PC6" é executada juntamente com a interpretação do processador "PC7". Sendo a primeira primitiva existente na interpretação do processador "PC6" uma primitiva "ATRASO (5)", o simulador zera as unidades de tempo de ambas primitivas e passa para a execução das interpretações. O simulador, ao executar a primitiva "IF", existente na interpretação do processador "PC7", testa o conteúdo do armazenador "AR5". Sendo este diferente de duas unidades, o simulador executa as primitivas vinculadas ao intervalo associado à primitiva "ELSE", encontrando uma primitiva "ATRASO", o simulador passa a executar a interpretação associada ao processador "PC6". Ao término da execução do processador "PC6", os conteúdos dos armazenadores "AR3" e "AR4" são alterados, sendo que o processador "PC7" permanece no estado de execução, processando a primitiva "ATRASO (5)".

Na sétima iteração, o simulador detecta que não há nós em condições de serem ativados. Ao detectar tal fato, o simulador

interrompe a simulação e envia mensagens ao usuário informando-o do ocorrido e oferece condições para que a simulação tenha prosseguimento, uma vez que o nó "N7" ainda se encontra em execução. Dessa forma, retoma-se a simulação, terminando a execução do processador "PC7", o que acarreta a alteração do conteúdo do armazenador "AR5".

Na oitava iteração, o nó "N9"é ativado e ao ser executada a primitiva condicional pertencente ao processador "PC9", o conteúdo do armazenador "AR5" é analisado. Sendo esse diferente de duas unidades, as primitivas pertencentes ao intervalo associado à primitiva "ELSE" são executadas.

Na nona iteração, o nó "N1" é novamente ativado e consequentemente o processador "PC1" é executado, sendo repetido o procedimento ocorrido na primeira iteração.

Na décima iteração, o nó "N2" é ativado, repetindo-se o procedimento ocorrido na segunda interação.

Na décima primeira iteração, o nó "N3" é ativado, sendo repetido o procedimento ocorrido na terceira iteração.

Na décima segunda iteração, os nós "N4", "N5" e "N6" são habilitados, repetindo-se o processamento ocorrido na terceira iteração.

Na décima terceira iteração, o nó "N8" passa para o estado ativo. Com isso, a interpretação do processador "PC8" é executada juntamente com a interpretação do processador "PC5" que já se

encontrava em execução processando a primitiva "ATRASO (5)". Sendo a primeira primitiva do processador "PC8" uma primitiva "ATRASO (5)", o simulador zera ambos os tempos, passando a executar as interpretações até que outra primitiva de atraso seja encontrada. Ao encontrar uma primitiva "ATRASO (5)" na interpretação associada ao processador "PC5", o simulador passa a executar a interpretação do processador "PC4". Ao término dessa execução, os conteúdos dos armazenadores "AR4" e "AR5" são alterados, sendo que o processador "PC5" permanece em execução, processando a primitiva "ATRASO (5)".

Na décima quarta iteração, o nó "N9" passa para o estado ativo e, consequentemente, a interpretação associada ao processador "PC9" é executada juntamente com a interpretação do processador "PC5". Ao ser executada a primitiva condicional, o simulador testa a quantidade de espaço ocupado no armazenador "AR5". Sendo essa igual a duas unidades, o simulador executa as primitivas contidas no intervalo associado à primitiva condicional "THEN". Ao ser detectada a primitiva "ATRASO (10)", as unidades de tempo das primitivas são decrescidas de cinco unidades o simulador passa a executar as outras primitivas da interpretação do processador PC5. Ao término do processamento da interpretação, os conteúdos dos armazenadores "AR4" e "AR5" são alterados. E, através da análise da estrutura dos nós habilitados, observa-se que o nó "N6" permanece no estado habilitado. Observa-se por intermédio da estrutura de nós ativos

que o nó "N9" permanece no estado ativo executando a primitiva ATRASO (5).

Na quinquagésima iteração, os arcos 'A4', 'A5', 'A10', 'A11' e 'A14' estão marcados e os cinco armazenadores possuem mensagens, sendo que o armazenador 'AR4' possui sete mensagens.

Na setuagésima quinta iteração, os arcos 'A9', 'A10', 'A11' e 'A14' estão marcados, permanecendo os cinco armazenadores com mensagens.

Na centésima iteração, os arcos 'A10', 'A11' e 'A14' estão marcados. O grafo de controle evoluiu nesses cem passos de simulação sem apresentar falhas, por outro lado, a análise do grafo de dados mostra que o mesmo apesar de não ter apresentado nenhuma inconsistência em seus armazenadores até essa iteração, possui uma distribuição crescente de mensagens no armazenador 'AR4'. Essa distribuição mostra que está sendo gerado nesse armazenador um número de mensagens superior ao que é consumido. Observando-se as estruturas do grafo de controle e da interpretação, verifica-se que os processadores 'PC5' e 'PC8' escrevem no armazenador 'AR4' de forma não-destrutiva. Esse procedimento de escritas não-destrutivas, associado ao fato da taxa de entrada de mensagens ser maior do que a taxa de saída, implica o 'estouro' do armazenador durante a execução do passo cento e vinte e três. Neste passo, o simulador detecta o 'estouro' do armazenador e em seguida interrompe a simulação

informando o usuário que o processador 'PC6' está acessando o armazenador 'AR4' por intermédio da'ESCREVERV', 'estourando-o'.

Com a execução dessas simulações tornou-se possivel visualizar o comportamento dinâmico do modelo em um ambiente distribuido. Como já mencionado, o grafo de controle não apresentou nenhuma falha durante as diversas simulações, onde conclui-se que essa distribuição não gera inconsistências no mesmo. Todavia, o grafo de dados deve ser alterado, uma vez que ocorreu o 'estouro' do armazenador 'AR4'. Essa alteração poderá ser obtida aumentando-se a taxa de consumo de mensagens em relação a taxa de escrita no armazenador. Outra possibilidade é tornar destrutivo o acesso efetuado pelas primitivas de escrita.

5 - CONCLUSÃO

O simulador GMB* desenvolvido neste trabalho constitui uma ferramenta para testes e validação de projetos de sistemas. A utilização de uma estrutura modular para a sua implementação possibilita o seu uso em microcomputadores que possuam uma pequena quantidade de memória de trabalho. A utilização de listas ligadas para o tratamento dos dados que constituem o modelo, permite que modelos de qualquer dimensão sejam definidos pelo usuário. Dessa forma, a única limitação quanto à dimensão dos modelos está associada à capacidade de memória de trabalho do microcomputador.

O simulador possibilita que o usuário simule o modelo de um aplicativo, de forma que, em uma primeira simulação, analise-se somente o fluxo de controle do modelo e, posteriormente, efetuem-se outras simulações, visando à análise do comportamento dos armazenadores do grafo de dados e à análise do projeto em um ambiente distribuído.

Acredita-se que o simulador GMB* pode se tornar uma ferramenta bastante útil no desenvolvimento de projetos de sistemas, principalmente no desenvolvimento de projetos de sistemas de controle de processos em tempo real, pois as suas características de concorrência e paralelismo são facilmente simuladas.

Dando continuidade ao trabalho, propõe-se a criação de uma

interface gráfica para o tratamento dos domínios de controle e de dados do modelo. Através dessa interface, o usuário poderá editar e analisar esses domínios utilizando diretamente os grafos, contribuindo dessa maneira, para uma melhor interação homem-máquina. Outra proposta importante para o uso adequado do simulador é o desenvolvimento de ferramentas automáticas de validação que através das estruturas geradas pelo simulador, forneçam ao usuário uma análise do modelo simulado.

APÊNDICE A - EXEMPLO DE UTILIZAÇÃO E APLICAÇÃO DO SIMULADOR GMB*

Esse apêndice visa demonstrar basicamente, a forma de utilização da interface 'homem-máquina' do simulador GMB* através da execução de três simulações de uma aplicação. As figuras A.1.1 e A.1.2 representam a aplicação modelada através do GMB*. Com base na metodologia proposta no capítulo anterior, efetua-se as seguintes simulações: a primeira é uma simulação fraca, objetivando a detecção de anomalias no grafo de controle; a segunda é efetuada visando o comportamento dos armazenadores; e a terceira é obtida utilizando-se completamente os três domínios do GMB* a fim de analisar o modelo em um ambiente distribuído.

Com o intuito de exemplificar a interação do usuário com o simulador é demostrado, na primeira simulação, o potencial de edição do mesmo.

A.1 - Simulação do grafo de controle

Para a obtenção de uma simulação visando apenas ao grafo de controle, é necessário que sejam definidos os domínios de controle, dados (figuras A.1.1 e A.1.2) e interpretação. Os domínios de dados e interpretação somente serão utilizados devido a necessidade da existência de primitivas de marcação para a evolução do grafo de controle.

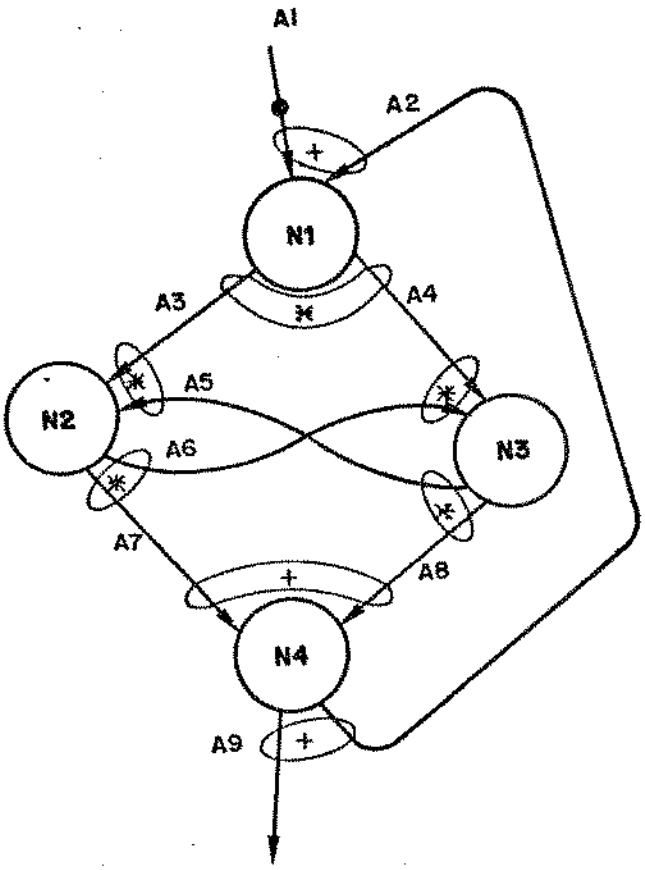


FIGURA A.1.1
GRAFO DE CONTROLE

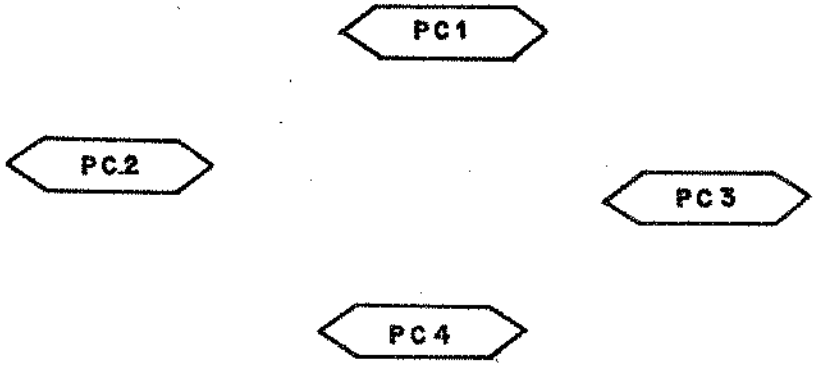


FIGURA A.1.2
GRAFO DE DADOS

A.1.1 - Definição dos domínios no simulador

Ao ser carregado o simulador no microcomputador, as seguintes opções são apresentadas na tela:

1: DEFINIR VARIAVEIS DE SIMULACAO;

2: DEFINIR ESTRUTURAS;

3: LISTAR ESTRUTURAS;

4: EXECUTAR A SIMULAÇÃO.

Inicialmente, escolhe-se a opção '1' com o intuito de definir as variáveis de simulação. Interativamente, atribui-se o valor '0' ao passo inicial, '3' ao passo final, 'A' ao tipo de simulação e 'B' ao 'drive' destino dos arquivos de dados. Isto feito, retorna-se à tela anterior e, através da escolha opção '2', o simulador exibe as seguintes opções:

1: ESTRUTURA DE CONTROLE;

2: VETOR DE MARCAS;

3: ESTRUTURA DE DADOS;

4: ESTRUTURA DOS ARMAZENADORES;

5: ESTRUTURA DA INTERPRETACAO;

6: RETORNAR A TELA ANTERIOR.

Através da escolha da opção '1', o simulador chama o módulo responsável pela definição da estrutura do grafo de controle, o

qual apresenta na tela mensagens para a criação da estrutura interativamente. Dessa maneira, tem-se:

```
DEFINICAO DO GRAFO DE CONTROLE:

ENTRE COM O NUMERO DE NOS
4
ENTRE COM O NO 1
N1
ENTRE COM A ESTACAO DO NO N1
E1
ENTRE COM O NUMERO DE ARCO DE ENTRADA DO NO
2
ENTRE COM O ARCO DE ENTRADA 1
A1
ENTRE COM A E.L.E. DO ARCO A1
+
ENTRE COM O ARCO DE ENTRADA 2
A2
ENTRE COM O NUMERO DE ARCOS DE SAIDA
2
ENTRE COM O ARCO DE SAIDA 1
A3
ENTRE COM A E.L.S DO ARCO A3
*
ENTRE COM O ARCO DE SAIDA 2
A4

ENTRE COM O NO 2
N2
ENTRE COM A ESTACAO DO NO N2
E2
ENTRE COM O NUMERO DE ARCO DE ENTRADA DO NO N2
2
ENTRE COM O ARCO DE ENTRADA 1
A3
ENTRE COM A E.L.E. DO ARCO  A3
*
ENTRE COM O ARCO DE ENTRADA 2
A5
ENTRE COM O NUMERO DE ARCOS DE SAIDA DO NO N2
2
ENTRE COM O ARCO DE SAIDA 1
A7
ENTRE COM A E.L.S. DO ARCO A7
```

```
*
ENTRE COM O ARCO DE SAIDA 2
A6

ENTRE COM O NO 3
N3
ENTRE COM A ESTACAO DO NO N3
E2
ENTRE COM O NUMERO DE ARCO DE ENTRADA DO NO N3
0
OCORREU UM ERRO DE DIGITACAO, REDIGITE OS PARAMETROS DO NO


ENTRE COM O NO 3
N3
ENTRE COM A ESTACAO DO NO N3
E2
ENTRE COM O NUMERO DE ARCOS DE ENTRADA DO NO N3
2
ENTRE COM O ARCO DE ENTRADA 1
A2
ENTRE COM A E.L.E. DO ARCO A2

OCORREU UM ERRO DE DIGITACAO, REDIGITE OS PARAMETROS DO NO

ENTRE COM O NO 3
N3
ENTRE COM A ESTACAO DO NO N3
E2
ENTRE COM O NUMERO DE ARCOS DE ENTRADA DO NO N3
2
ENTRE COM O ARCO DE ENTRADA 1
A8
ENTRE COM E.L.E. DO ARCO A8
*
ENTRE COM O ARCO DE ENTRADA 2
A4
ENTRE COM O NUMERO DE ARCOS DE SAIDA DO NO N3
2
ENTRE COM O ARCO DE SAIDA 1
A5
ENTRE COM E.L.S. DO ARCO A5
*
ENTRE COM O ARCO DE SAIDA 2
A8

ENTRE COM O NO 4
N4
ENTRE COM A ESTACAO DO NO N4
```

```
E3
ENTRE COM O NUMERO DE ARCOS DE ENTRADA DO NO N4
2
ENTRE COM O ARCO DE ENTRADA 1
A7
ENTRE COM A E.L.E. DO ARCO A7
+
ENTRE COM O ARCO DE ENTRADA 2
A8
ENTRE COM O NUMERO DE ARCOS DE SAIDA DO NO N4
2
ENTRE COM O ARCO DE SAIDA 1
A9
ENTRE COM A E.L.S. DO ARCO A8
+
ENTRE COM O ARCO DE SAIDA 2
A2
```

Na definição do nó 'N3', demonstra-se a versatilidade do modulo de edição do simulador, visto que é permitido ao usuário efetuar correções em sua definição sem ter de redefinir toda a estrutura.

Terminada a inserção da estrutura do grafo de controle, essa é impressa na tela do computador, oferecendo ao usuário a possibilidade de redefinição da mesma ou o seu armazenamento no arquivo de dados. Assim, o conteúdo da tela é renovado, apresentando o seguinte:

```
TERMINO DA DEFINICAO DO GRAFO DE CONTROLE.
GRAFO DE CONTROLE OBTIDO:
PARA O NO N1 OBTEMOS:
ESTACAO E1
```

```
ARCOS E EXPRESSOES LOGICAS DE ENTRADA

A1 + A2

ARCOS E EXPRESSOES LOGICAS DE SAIDA

A3 * A4

PARA O NO N2 OBTEMOS:

ESTACAO E2

ARCOS E EXPRESSOES LOGICAS DE ENTRADA

A3 * A5

ARCOS E EXPRESSOES LOGICAS DE SAIDA

A7 * A6

PARA O NO N3 OBTEMOS:

ESTACAO E2

ARCOS E EXPRESSOES LOGICAS DE ENTRADA

A6 * A4

ARCOS E EXPRESSOES LOGICAS DE SAIDA

A5 * A8

PARA O NO N4 OBTEMOS:

ESTACAO E3

ARCOS E EXPRESSOES LOGICAS DE ENTRADA

A7 + A8

ARCOS E EXPRESSOES LOGICAS DE SAIDA

A9 + A2

O GRAFO APRESENTADO CONFERE COM A DESCRICAO ?

S
```

Deve-se observar que a impressão da estrutura obtida é efetuada de forma que, ao ser preenchida a tela do computador, o simulador fica paralisado, aguardando um comando do usuário para o prosseguimento da impressão. Uma vez armazenada a estrutura no arquivo de dados, é criada a estrutura do vetor de marcas através do programa de definição, o qual renova a tela apresentando o seguinte:

```
DEFINICAO DA ESTRUTURA DE MARCAS INICIAIS.

ENTRE COM O NUMERO DE ARCOS DE ENTRADA.
9
ENTRE COM O ARCO 1
A1
ENTRE COM O NUMERO DE MARCAS DO ARCO A1
1
ENTRE COM A PRIORIDADE DA MARCA 1
3
ENTRE COM O ARCO 2
A2
ENTRE COM O NUMERO DE MARCAS DO ARCO A2
0
ENTRE COM O ARCO 3
A3
ENTRE COM O NUMERO DE MARCAS DO ARCO A3
0
ENTRE COM O ARCO 4
A4
ENTRE COM O NUMERO DE MARCAS DO ARCO A4
0
ENTRE COM O ARCO 5
A5
ENTRE COM O NUMERO DE MARCAS DO ARCO A5
0
ENTRE COM O ARCO 6
A6
ENTRE COM O NUMERO DE MARCAS DO ARCO A6
0
ENTRE COM O ARCO 7
A7
```

```
ENTRE COM O NUMERO DE MARCAS DO ARCO A7
0
ENTRE COM O ARCO 8
A8
ENTRE COM O NUMERO DE MARCAS DO ARCO A8
0
ENTRE COM O ARCO 9
A9
ENTRE COM O NUMERO DE MARCAS DO ARCO A9
0
```

De forma similar à definição anterior, é oferecida ao usuário a possibilidade de corrigir trechos da estrutura sem ter de redefini-la totalmente. Findada a inserção da estrutura, é também impressa na tela, oferecendo a possibilidade de redefinição. Encontrado-se correta, a estrutura é armazenada no arquivo de dados, sendo oferecida ao usuário a possibilidade de criação da estrutura de dados.

Como já mencionado anteriormente, mesmo na simulação apenas do grafo de controle, faz-se necessário definir a estrutura de dados, pois as primitivas de marcação dos arcos de saída devem estar associadas a processadores do grafo de dados. Tal fato implica a definição de um grafo de dados onde os processadores não possuem arcos de dados e, consequentemente, não existem armazenadores.

```
DEFINICAO DO GRAFO DE DADOS

ENTRE COM O NUMERO DE P.Cs.
4
ENTRE COM O NUMERO DE ARMAZENADORES
0
```

```
ENTRE COM O P.C. 1
PC1
ENTRE COM O NO DE CONTROLE ASSOCIADO AO P.C. PC1
N1
ENTRE COM O NUMERO DE ARCOS DE DADOS DE ENTRADA DO P.C. PC1
0
ENTRE COM O NUMERO DE ARCOS DE DADOS DE SAIDA DO P.C. PC1
0

ENTRE COM O P.C. 2
PC2
ENTRE COM O NO DE CONTROLE ASSOCIADO AO P.C. PC2
N2
ENTRE COM O NUMERO DE ARCOS DE DADOS DE ENTRADA DO P.C. PC2
0
ENTRE COM O NUMERO DE ARCOS DE DADOS DE SAIDA DO P.C. PC2
0

ENTRE COM O P.C. 3
PC3
ENTRE COM O NO DE CONTROLE ASSOCIADO AO P.C. PC3
N3
ENTRE COM O NUMERO DE ARCOS DE DADOS DE ENTRADA DO P.C. PC3
0
ENTRE COM O NUMERO DE ARCOS DE DADOS DE SAIDA DO P.C. PC3
0

ENTRE COM O P.C. 4
PC4
ENTRE COM O NO DE CONTROLE ASSOCIADO AO P.C. PC4
N4
ENTRE COM O NUMERO DE ARCOS DE DADOS DE ENTRADA DO P.C. PC4
0
ENTRE COM O NUMERO DE ARCOS DE DADOS DE SAIDA DO P.C. PC4
0
```

Como nas definições, anteriores é oferecida ao usuário a possibilidade de correção da estrutura e, ao término da sua definição, essa é impressa na tela, oferecendo ao usuário a possibilidade de redefinição. Isto feito, passa-se à definição da estrutura da interpretação, uma vez que a estrutura dos armazenadores não é criada.

Na definição da interpretação, observa-se que o simulador

oferece uma importante ferramenta de apoio à edição, uma vez que efetua testes léxicos, sintáticos e semânticos em todas as primitivas. Tal fato pode ser observado na definição abaixo:

```
DEFINICAO DA INTERPRETACAO

TERMINE A INSERCAO DO ARQUIVO DEGITANDO O CARACTERE ].

ENTRE COM O P.C. ASSOCIADO A INTERPRETACAO 1
PC1

ENTRE COM AS PRIMITIVAS E TERMINE A INSERCAO COM O CARACTERE ].

SAIDA (A3,3)
SAIDA (A4,2)
]

ENTRE COM O P.C. ASSOCIADO A INTERPRETACAO 2
PC2D

ERRO NA DEFINICAO DO PROCESSADOR ASSOCIADO, O P.C.
PC2D NAO ENCONTRA-SE NA ESTRUTURA DE DADOS.

ENTRE COM O P.C. ASSOCIADO A INTERPRETACAO 2
PC2

ENTRE COM AS PRIMITIVAS, TERMINE A INSERCAO COM O CARACTERE ].

SAIDA (A7,2
ERRO DE SINTAXE, FALTA ")" NA PRIMITIVA SAIDA
SAIDA (A7,2)
]
ERRO, OS ARCOS ABAIXO DEVEM SER MARCADOS POIS ESTES
ENCONTRAM-SE RELACIONADOS POR EXPRESSOES "*"
ARCO A6

SAIDA (A6,2)
]

ENTRE COM O P.C. ASSOCIADO A INTERPRETACAO 3
PC3

ENTRE COM AS PRIMITIVAS E TERMINE A INSERCAO COM O CARACTERE ].
```

```
SAIDA (A5,2)
SAIDA (A8,3)
]

ENTRE COM O P.C. ASSOCIADO A INTERPRETACAO 4
PC4

ENTRE COM AS PRIMITIVAS E TERMINE A INSERCAO COM O CARACTERE ).

SAIDA (A9,1)
]

ENTRE COM P.C. ASSOCIADO A INTERPRETACAO 5
]
```

Terminada a inserção, o simulador lista a estrutura na tela, obtendo-se o seguinte:

```
TERMINO DA DEFINICAO DA INTERPRETACAO

INTERPRETACAO OBTIDA:

PARA O P.C. PC1 OBTEMOS 2 PRIMITIVAS:

SAIDA (A3,3)
SAIDA (A4,2)

PARA O P.C. PC2 OBTEMOS 2 PRIMITIVAS:

SAIDA (A7,2)
SAIDA (A6,2)

PARA O P.C. PC3 OBTEMOS 2 PRIMITIVAS:

SAIDA (A5,2)
SAIDA (A8,3)

PARA O PC4 OBTEMOS 1 PRIMITIVAS:

SAIDA (A9,1)
```

Estando definidas as estruturas, retorna-se ao módulo de operação e, escolhendo-se o item '4', passa-se para a fase de execução da simulação.

A.1.2 - Execução da simulação

Regido pelas regras do GMB* /07/ e pelo conteúdo variáveis de simulação, o simulador executa o primeiro passo da simulação ativando o nó 'N1', retirando a marca do arco 'A1' (marca de prioridade '3') e marcando os arcos 'A3' e 'A4'. Ao iniciar o segundo passo de simulação, o simulador detecta a existência de um 'deadlock' entre os nós 'N2' e 'N3', pois os arcos 'A5' e 'A6' estão dispostos de forma tal que não permitem a ativação de nenhum dos dois nós. A detecção de um 'deadlock' implica o envio de mensagens ao usuário e a paralisação da simulação.

Para a eliminação do 'deadlock' existente no modelo basta substituir os arcos 'A5' e 'A6' por um arco complexo, colocando nesse uma marca inicial de forma a manter a exclusão mútua entre os nós 'N2' e 'N3'. A figura A.2.1 ilustra o grafo de controle modificado.

A.2 - Simulação do modelo observando o comportamento dos armazenadores

Nesse exemplo, simula-se o modelo modificado (figuras A.2.1 e A.2.2) com o intuito de observar o comportamento dos armazenadores. Para tanto, é necessário que os três domínios do GMB* sejam definidos.

A.2.1 - Definição dos domínios no simulador

De forma similar ao caso anterior, inserem-se interativamente as variáveis de simulação e os domínios do modelo, obtendo-se as seguintes descrições:

```
VARIAVEIS DA SIMULACAO:
PASSO INICIAL                                   0
PASSO FINAL                                     3
TIPO DE SIMULACAO                               A
DRIVE DESTINO DOS ARQUIVOS DE DADOS             B

GRAFO DE CONTROLE OBTIDO:
PARA O NO N1 OBTEMOS:
ESTACAO E1
ARCOS E EXPRESSOES LOGICAS DE ENTRADA
A1 + A2
ARCOS E EXPRESSOES LOGICAS DE SAIDA
```

A3 * A4

PARA O NO N2 OBTEMOS:

ESTACAO E2

ARCOS E EXPRESSOES LOGICAS DE ENTRADA

A3 * A5

ARCOS E EXPRESSOES LOGICAS DE SAIDA

A6 * A5

PARA O NO N3 OBTEMOS:

ESTACAO E2

ARCOS E EXPRESSOES LOGICAS DE ENTRADA

A5 * A4

ARCOS E EXPRESSOES LOGICAS DE SAIDA

A5 * A7

PARA O NO N4 OBTEMOS:

ESTACAO E3

ARCOS E EXPRESSOES LOGICAS DE ENTRADA

A6 + A7

ARCOS E EXPRESSOES LOGICAS DE SAIDA

A8 + A2

ESTRUTURA DE MARCAS OBTIDA:

PARA O ARCO A1 OBTEMOS:

PRIORIDADE DAS MARCAS

3 1

PARA O ARCO A2 OBTEMOS:

0

PARA O ARCO A3 OBTEMOS:

0

PARA O ARCO A4 OBTEMOS:

0

PARA O ARCO A5 OBTEMOS:

PRIORIDADE DAS MARCAS

1

PARA O ARCO A6 OBTEMOS:

0

PARA O ARCO A7 OBTEMOS:

0

PARA O ARCO A8 OBTEMOS:

0

GRAFO DE DADOS OBTIDO:

PARA O PC1 OBTEMOS:

NO DE CONTROLE N1

PARA O PC2 OBTEMOS:

NO DE CONTROLE N2

ARCO DE DADOS DE SAIDA SC2

ARMAZENADOR DESTINO ARM1

PARA O PC3 OBTEMOS:

NO DE CONTROLE N3

ARCO DE DADOS DE ENTRADA	EC3
ARMAZENADOR ORIGEM	ARM1

PARA O PC4 OBTEMOS:

NO DE CONTROLE	N4

ESPECIFICACOES DOS ARMAZENADORES:

ARMAZENADOR	ARM1
ESTACAO	E1
TIPO	F
TAMANHO	10
TAMANHO DA UNIDADE DE ARMAZENAMENTO	3
NUMERO DE MENSAGENS	3
TOTAL OCUPADO	9

INTERPRETACAO OBTIDA:

PARA O P.C. PC1 OBTEMOS 2 PRIMITIVAS:

SAIDA (A3,3)
SAIDA (A4,2)

PARA O P.C. PC2 OBTEMOS 3 PRIMITIVAS:

ESCREVERF (SC2,N)
SAIDA (A5,2)
SAIDA (A6,2)

PARA O P.C. PC3 OBTEMOS 3 PRIMITIVAS:

LERF (EC3,N)
SAIDA (A5,2)
SAIDA (A7,3)

PARA O PC4 OBTEMOS 1 PRIMITIVAS:

SAIDA (A8,1)

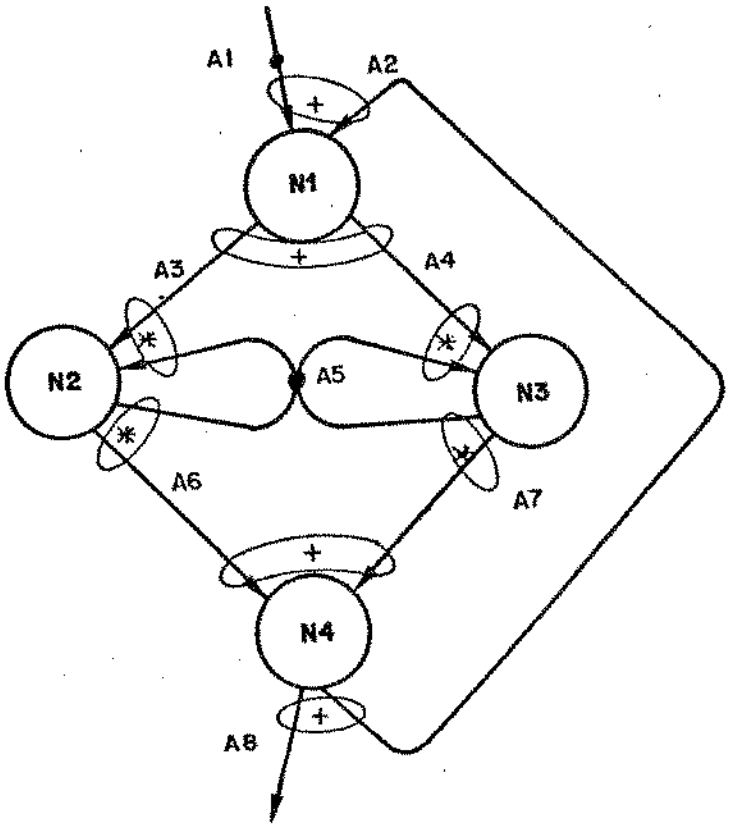


FIGURA A.2.1

GRAFO DE CONTROLE

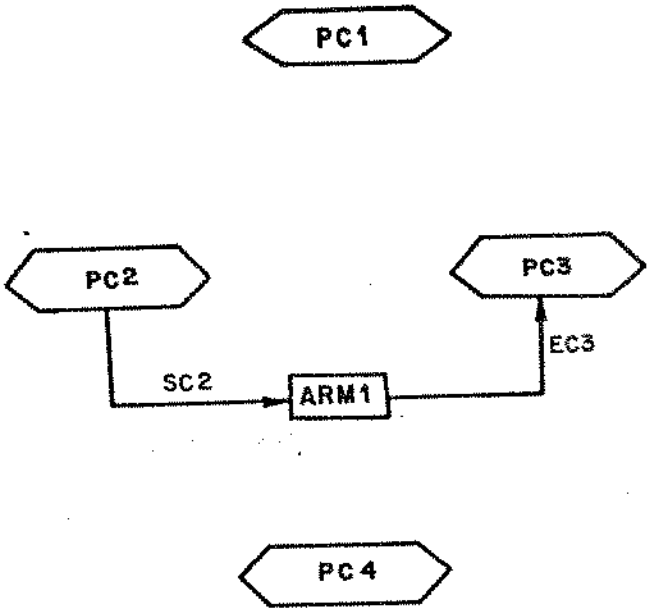


FIGURA A.2.2

GRAFO DE DADOS

Observe-se que nesse exemplo é necessário definir a estrutura dos armazenadores e que, além das primitivas de marcação, são também utilizadas na interpretação as primitivas de escrita e leitura de armazenadores.

A.2.2 - Execução da simulação

Terminada a definição dos diversos domínios, passa-se à fase de execução da simulação, em que,no primeiro passo, o nó 'N1' é ativado. A ativação do nó 'N1' implica a retirada da marca de maior prioridade do arco de entrada 'A1' (prioridade 3) e a respectiva marcação dos arcos de saída 'A3' e 'A4' ao término da ativação do nó. No segundo passo de simulação, os nós 'N1' e 'N2' são ativos, uma vez que se encontram em estações diferentes, possuindo expressões logicas de entrada satisfeitas. Observa-se que tanto o nó 'N2' quanto o 'N3' poderiam ser ativados; entretanto, devido ao fato de ambos estarem relacionados por um arco complexo, o qual efetua a exclusão mútua entre eles, apenas o de maior prioridade passa para o estado ativo ('N2') ficando o outro ('N3') desabilitado.

Durante a execução da interpretação associada ao nó 'N2' a primitiva 'ESCREVERF' é acessada, e tenta escrever no armazenador 'ARM1'. Sendo esta escrita não destrutiva, ocorre um 'estouro' do armazenador, uma vez que ele não comporta mais mensagens sem que haja uma atualização. O simulador, ao detectar tal anomalia,

interrompe a simulação e emite mensagens ao usuário informando-o do ocorrido.

A.3 - Simulação visando a validação do modelo em um ambiente distribuido

Para essa simulação, utilizou-se o exemplo anterior alterando os domínios de dados e interpretação. Os grafos apresentados nas figuras A.2.1 e A.3.2 aparentemnte não possuem erros de projeto. Entretanto, durante a evolução da simulação é possivel que sejam detectadas anomalias, visto que o modelo possui nós pertencentes a estações distintas, e a sua execução paralela, ou não, pode alterar o fluxo de controle e/ou a estrutura dos dados.

A utilização de todos os recursos do simulador o torna uma ferramenta muito eficaz para a validação de projetos, pois é oferecida ao usuário a possilidade de definir armazenadores de tipo fixo e variável e criar interpretações utilizando-se de todo o conjunto de primitivas. Com isso, é possível formar um ambiente de simulação onde diversos nós estejam sendo executados paralelamente, efetuando alterações nas estruturas de dados e na evolução do grafo de controle. Observa-se que a utilização das primitivas de atraso possui um papel importante na simulacão de um determinado nó já que é responsável pela simulação do tempo de execução de uma determinada interpretação.

Neste exemplo são executados três passos de simulação e, após o término de cada passo, os comandos de listagem do simulador são utilizados para a observação do comportamento do sistema.

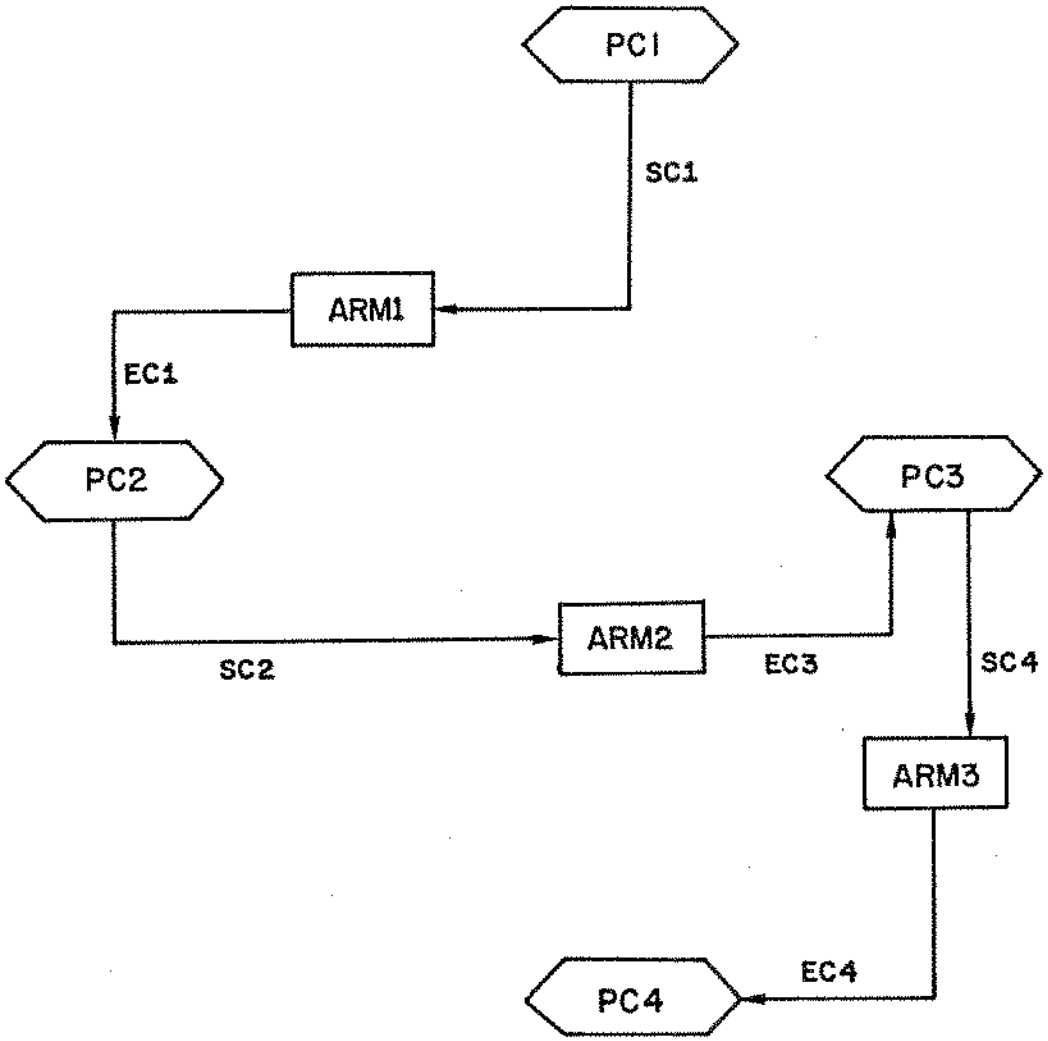


FIGURA A.3.2

GRAFO DE DADOS

A.3.1 - Definição dos domínios no simulador

Para as definições dos domínios do modelo, procede-se de forma análoga aos exemplos anteriores. As diversas estruturas obtidas estão dispostas abaixo:

```
VARIAVEIS DA SIMULACAO:
PASSO INICIAL                                    0
PASSO FINAL                                      1
TIPO DE SIMULACAO                                A
DRIVE DESTINO DOS ARQUIVOS DE DADOS              B

GRAFO DE CONTROLE OBTIDO:
PARA O NO N1 OBTEMOS:
ESTACAO  E1
ARCOS E EXPRESSOES LOGICAS DE ENTRADA
A1 + A2
ARCOS E EXPRESSOES LOGICAS DE SAIDA
A3 * A4
PARA O NO N2 OBTEMOS:
ESTACAO E2
ARCOS E EXPRESSOES LOGICAS DE ENTRADA
A3 * A5
ARCOS E EXPRESSOES LOGICAS DE SAIDA
A6 * A5
PARA O NO N3 OBTEMOS:
```

ESTACAO E2

ARCOS E EXPRESSOES LOGICAS DE ENTRADA

A5 * A4

ARCOS E EXPRESSOES LOGICAS DE SAIDA

A5 * A7

PARA O NO N4 OBTEMOS:

ESTACAO E3

ARCOS E EXPRESSOES LOGICAS DE ENTRADA

A6 + A7

ARCOS E EXPRESSOES LOGICAS DE SAIDA

A8 + A2

ESTRUTURA DE MARCAS OBTIDA:

PARA O ARCO A1 OBTEMOS:

PRIORIDADE DAS MARCAS

2 1

PARA O ARCO A2 OBTEMOS:

0

PARA O ARCO A3 OBTEMOS:

0

PARA O ARCO A4 OBTEMOS:

0

PARA O ARCO A5 OBTEMOS:

PRIORIDADE DAS MARCAS

1

PARA O ARCO A6 OBTEMOS:

0

PARA O ARCO A7 OBTEMOS:

0

PARA O ARCO A8 OBTEMOS:

0

GRAFO DE DADOS OBTIDO:

PARA O PC1 OBTEMOS:

NO DE CONTROLE	N1
ARCOS DE DADOS DE SAIDA	SC1
ARMAZENADOR DESTINO	ARM1

PARA O PC2 OBTEMOS:

NO DE CONTROLE	N2
ARCOS DE DADOS DE ENTRDA	EC2
ARMAZENADOR ORIGEM	ARM1
ARCO DE DADOS DE SAIDA	SC2
ARMAZENADOR DESTINO	ARM2

PARA O PC3 OBTEMOS:

NO DE CONTROLE	N3
ARCO DE DADOS DE ENTRADA	EC3
ARMAZENADOR ORIGEM	ARM2

ARCO DE DADOS DE SAIDA	SC3
ARMAZENADOR DESTINO	ARM3

PARA O PC4 OBTEMOS:

NO DE CONTROLE	N4
ARCO DE DADOS DE ENTRADA	EC4
ARMAZENADOR ORIGEM	ARM3

ESPECIFICACOES DOS ARMAZENADORES:

ARMAZENADOR	ARM1
ESTACAO	E2
TIPO	F
TAMANHO	10
TAMANHO DA UNIDADE DE ARMAZENAMENTO	2
NUMERO DE MENSAGENS	3
TOTAL OCUPADO	6
ARMAZENADOR	ARM2
ESTACAO	E2
TIPO	V
TAMANHO DO ARMAZENADOR	100
NUMERO DE MENSAGENS	2
TOTAL OCUPADO	35
TAMANHO DA MENSAGEM 1	15
TAMANHO DA MENSAGEM 2	20
ARMAZENADOR	ARM3
ESTACAO	E3
TIPO	F
TAMANHO DO ARMAZENADOR	20
TAMANHO DA UNIDADE DE ARMAZENAMENTO	5
NUMERO DE MENSAGENS	1
TOTAL OCUPADO	5

INTERPRETACAO OBTIDA:

PARA O P.C. PC1 OBTEMOS 8 PRIMITIVAS:

```
ESCREVERF (SC1,D)
IF (SC2,=,6) THEN
ATRASO (10)
ELSE
ATRASO (15)
ENDIF
SAIDA (A3,2)
SAIDA (A4,2)
```

PARA O P.C. PC2 OBTEMOS 9 PRIMITIVAS:

```
LERF (EC2,N)
IF (SC2,=,20) THEN
ESCREVERV (SC2,30,D)
ATRASO (10)
ELSE
ATRASO (5)
ENDIF
SAIDA (A5,2)
SAIDA (A6,2)
```

PARA O P.C. PC3 OBTEMOS 8 PRIMITIVAS:

```
IF (EC3,=,50) THEN
LERV (EC3,D)
LERV (EC3,D)
ENDIF
ATRASO (5)
ESCREVERF (SC3,N)
SAIDA (A5,2)
SAIDA (A7,3)
```

PARA O PC4 OBTEMOS 8 PRIMITIVAS:

```
LERF (EC4,N)
ATRASO (15)
IF (EC4,=,5) THEN
SAIDA (A8,1)
SAIDA (A2,1)
ELSE
SAIDA (A2,2)
ENDIF
```

A.3.2 - Execução da simulação

A execução passo a passo da simulação possibilita obter uma análise bastante detalhada do comportamento do modelo. A seguir apresentam-se os conteúdos das estruturas, os quais são obtidos por intermédio dos comandos do simulador ao término de cada passo.

PASSO 1 : Observando-se o conteúdo da estrutura da máquina de "tokens" e da estrutura do vetor de marcas, nota-se que nesse passo apenas o nó 'N1' foi ativado e, por conseguinte, sua interpretação executada. A execução dessa interpretação não causou alterações na estrutura de dados, pois a primitiva 'ESCREVERF' acessa o armazenador não alterando o número de posições ocupadas, já que a escrita é destrutiva.

Estruturas que sofreram alterações:

VETOR DE MARCAS

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A1

1

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A2

0

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A3

2

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A4

```
2
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A5
0
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A6
0
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A7
0
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A8
0

MAQUINA DE TOKENS
NA ITERACAO 1 OBTEMOS:
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A1
2  1
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A5
1
```

Observa-se que ao término da execução do nó 'N1' tanto a estrutura de nós habilitados quanto a estrutura de nós ativos se encontram vazias.

PASSO 2: Nesse passo os nós 'N1' e 'N2' foram ativados. A existência da exclusão mútua entre os nós da estação '2' força a ativação de apenas um dos nós. Dessa forma, o simulador escolhe o

nó para a ativação, obedecendo às prioridades das expressões lógicas. No caso atual, os nós 'N2' e 'N3' possuem as mesmas prioridades. Assim, o simulador executa um procedimento pseudoaleatório, sorteando o nó 'N2' e ativando-o.

Através da análise das interpretações dos nós que se encontram ativos, constata-se que na interpretação do nó 'N1' a primitiva condicinal 'THEN' é satisfeita e, por conseguinte, a primitiva 'ATRASO (10)' é executada. Já, na interpretação do nó 'N2' a condicão não é satisfeita o que implica a execução da primitiva 'ATRASO (5)'. A diferença entre os argumentos das primitivas faz com que o nó 'N1' continue em execução mesmo após o término da execução do nó 'N2'(término do passo 2). Tal fato é constatado observando-se a estrutura de nós ativos.

Estruturas que sofreram alterações:

VETOR DE MARCAS

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A1

0

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A2

0

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A3

0

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A4

2

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A5

```
2
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A6
2
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A7
0
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A8
0

MAQUINA DE TOKENS
NA ITERACAO 1 OBTEMOS:
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A1
2  1
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A5
1
NA ITERACAO 2 OBTEMOS:
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A1
1
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A3
2
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A4
2
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A5
1
```

ESTRUTURA DOS NOS ATIVOS

O NO N1 ENCONTRA-SE NA ESTACAO E1

A INTERPRETACAO ASSOCIADA AO NO N1 ESTA EXECUTANDO A PRIMITIVA ATRASO (005)

PASSO 3: Nesse passo, os nós 'N1', 'N3' e 'N4' se encontram ativos. O nó 'N1' prossegue a execução da interpretação a partir da primitiva 'ATRASO (005)', pois o argumento foi decrementado de cinco unidades devido ao fato desse tempo já ter sido utilizado no passo anterior. Ao ser observada a interpretação do nó 'N3', verifica-se que a primitiva condicional não foi satisfeita e que o argumento da primitiva 'ATRASO' é igual ao da interpretação associada ao nó 'N1'. Tal fato implica o término simultâneo de ambas as interpretações. Entretanto, na interpretação do nó 'N4', o argumento da primitiva 'ATRASO' é maior do que os outros, o que faz com que a interpretação do nó 'N4' permaneça em execução mesmo após o término desse passo de simulação. Deve-se observar também que ao ser executada a interpretação do nó 'N3', a primitiva 'ESCREVERF' altera o conteúdo do armazenador 'ARM3'.

Estruturas que sofreram alterações:

ESTRUTURA DOS ARMAZENADORES:

ARMAZENADOR	ARM1
ESTACAO	E2
TIPO	F

TAMANHO	10
TAMANHO DA UNIDADE DE ARMAZENAMENTO	2
NUMERO DE MENSAGENS	3
TOTAL OCUPADO	8
ARMAZENADOR	ARM2
ESTACAO	E2
TIPO	V
TAMANHO DO ARMAZENADOR	100
NUMERO DE MENSAGENS	2
TOTAL OCUPADO	35
TAMANHO DA MENSAGEM 1	15
TAMANHO DA MENSAGEM 2	20
ARMAZENADOR	ARM3
ESTACAO	E3
TIPO	F
TAMANHO DO ARMAZENADOR	20
TAMANHO DA UNIDADE DE ARMAZENAMENTO	5
NUMERO DE MENSAGENS	2
TOTAL OCUPADO	10

VETOR DE MARCAS

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A1

0

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A2

0

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A3

2

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A4

2

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A5

2

PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A6

0

```
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A7
3
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A8
0

MAQUINA DE TOKENS
NA ITERACAO 1 OBTEMOS:
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A1
2  1
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A5
1
NA ITERACAO 2 OBTEMOS:
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A1
1
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A3
2
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A4
2
PRIORIDADE DAS MARCAS DO ARCO DE CONTROLE A5
1
NA ITERACAO 3 OBTEMOS:
PRIORIDADES DAS MARCAS DO ARCO A4
2
PRIORIDADES DAS MARCAS DO ARCO A5
2
```

PRIORIDADES DAS MARCAS DO ARCO A6

2

ESTRUTURA DOS NOS ATIVOS

O NO N4 ENCONTRA-SE NA ESTACAO E3

A INTERPRETACAO ASSOCIADA AO NO N4 ESTA EXECUTANDO A PRIMITIVA

ATRASO (D1D)

A execução dos três passos de simulação possibilitou visualizar a potencialidade do simulador, uma vez que, o usuário pode se utilizar de inúmeros recursos para a validação de seu projeto. A simulação passo a passo é apenas uma das opções, pois não há limitação alguma em se executar os três passos e, ao final desses, observar o comportamento das estruturas.

REFERÊNCIAS BIBLIOGRÁFICAS

/01/ - Allworth S.T., "Introduction to Real-time Software Design", Macmillam Computer Science Series, 1981.

/02/ - Bacon J., "An Approach to Distributed Software Systems", ACM Oper. Syst. Review, Vol.15, n 4, october,1978.

/03/ - Bigon Jean M., "Un Automate Programmable Par Reseaux De Petri", mai 1983, Note L.A.A.S 83.026.

/04/ - Cheriton D.R., Malcon M.A., Melen L.S., Sager G.R., "Thoth, a Protable Real-time", Operaring System Comm. ACM.,Vol.22, february 1979.

/05/ - Coffman E.G. et al, "System Deadlocks" Comp.Surveys, Vol.3 n 2, June 1971.

/06/ - De Martino J. M., "Relatório Técnico - Setor de Computação e Automação Industrial", outubro 1983, DEE, FEC, UNICAMP.

/07/ - De Martino J. M., "Um Ambiente GMB* para o Desenvolvimento de Sistemas Distribuidos de Controle Digital - A Máquina GMB*", Tese de Mestrado JULHO 1986, DEE, FEC, UNICAMP.

/08/ - Dijkstra E.W., "Complexity Controlled by Hierarchical Ordering of Function and Variabitity", Report on a Conference on Sftware Engineering, Grasmich, oct. 1968.

/09/ - Gentleman W.N., "Message Passing Betweem Sequential Processes" The Replay Primitivas and Administratior Concept. Soft. Pract. and Exp, vol 11, 435 - 466; 1981.

/10/ - Gordon Geoffrey, "System Simulation" Prentice-Hall, Inc.,

Engle Wood Cliffs, New Jersey 07632, 1978.

/11/ - Hansen P.B., "Concurrent Programming Concepts", Computing Surveys, Vol.5, n 4, December 1973.

/12/ - Hoare C.A.R., "Communicating Sequential Processes", Comm. ACM,Vol.21,n 8, August 1978.

/13/ - Hoare C.A.R., "Monitors: An Operating System Structuring Concept", Communications of the ACM, October 1974, Vol 17, n 10.

/14/ - Kramer J., Magee J., Sloam M., "Intertask Communication Primitives for Distributed Computer Control System", 2 Int. Conf. Distributed Computing Systems, Paris, April 1981.

/15/ - Liskov B., "Primitives for Distributed Computing", Proc. VII Symp. on Operating Systems Principeles, December 1979.

/16/ - Lister A., Magee J., Sloman M., Kramer J., "Distributed Process Control Systems", Programming and Configuration. Research Report n 80/12. Departament of Computing and Control, Imperial Colleg, May 1980.

/17/ - Manning E., Livesey N.J., Tokuda H., "Interprocess Communication In Distributed Systems", One view. Proc. IFIP, 1980.

/18/ - Myers W., "Toward a local Network Standard", IEE Micro 1982.

/19/ - Penedo M. Heloisa, "The Use of a Module Interface Description In Then Synthesis of Reliable Software Systems", January, 1981, UCLA - ENG - 8091 (DOI: UCLA - 34P214-108).

/20/ - Peterson J. L., "Petri Net Theory and The Modeling of Systems", Prentice-Hall, Inc., Englewood Cliffs, N.J. 07632 1981.

/21/ - Razouk Rami R., "Computer-Aided Design And Evaluation of Digital Computer Systems", february 1981, UCLA - ENG - 8091 (DDI: UCLA - 34P214 - 105).

/22/ - Razouk Rami R. and Estrin Gerald, "Then Graph Model of Behavior Simulator", Proc. of Symp. on Design Automation and Microprocessos, Palo Alto CA, february 1977.

/23/ - Razouk Rami R., Vernon Mary and Estrin Gerald, "Evaluation Methods in SARA the Graph Model Simulator", Proceedings of the 1979 conference on Simulation, Measurement,and Modeling of Computer Systems.

/24/ - Ruggiero W., " A Distributed Data and Control Driven Machine : Programming and Architecture", november, 1978, UCLA - ENG - 7878 (DDI: UCLA - 34P214-77).