UNIVERSIDADE ESTADUAL DE CAMPINAS
FACULDADE DE ENGENHARIA ELETRICA
FEVEREIRO DE 1989

FERRAMENTA PARA GERAÇÃO AUTOMATICA DE REDES DE PETRI A PARTIR DA ESPECIFICAÇÃO DE UM SISTEMA DE SOFTWARE COM CARACTERISTICAS TEMPO REAL

Este exemplar corresponde à redação final da tese defendida por Luiz Manuel Aguilera e aprovada pela Comissão Julgadora em 16/02/1989.

Autor : Luiz Manoel Aguilera

Orientadores: Prof. Dr. Márcio Luiz de Andrade Netto
Profa. Dra. Beatriz Mascia Daltrini

Tese apresentada à Faculdade de Engenharia Elétrica da Universidade Estadual de Campinas, como parte dos requisitos exigidos para obtenção do título de Mestre em Engenharia Elétrica.

DEDICADO À MEMÓRIA DE:

Meu pai, José Cosme Aguilera (Zezito) e meu irmão, Justo José Aguilera (Bill)

A G R A D E C I M E N T O S

Gostaria de expressar meus sinceros agradecimentos a todos aqueles que direta ou indiretamente contribuiram para a elaboração desta Tese:

- Aos Profs. Beatriz e Márcio (FEE/UNICAMP) pela orientação objetiva e atenção especial dedicadas a este trabalho;

- A Dra. Graça Bressan (IME/USP) pela cessão e direito de uso do Analisador Invarian.

- Ao Dr. Robert Valette (LAAS, Toulouse-França) pela cessão e direito de uso do Simulador P.S.I.

- Aos amigos do IA/CTI e ao CTI pelo apoio institucional.

R E S U M O

O conteúdo deste trabalho aborda a geração automática de Redes de Petri a partir da Especificação de um Sistema de Software com Características Tempo Real, utilizando ambiente de programação Prolog.

A especificação do sistema é feita através de duas interfaces: "Interface de Especificação de Sistemas" que reune um conjunto de conceitos cujo objetivo é especificar, decompor e inter-relacionar diferentes objetos de projeto e "Interface de Especificação da Dinâmica de Sistemas" cujo objetivo é especificar, interpretar e descrever as condições de simulação de algumas das características dinâmicas de um Sistema de Software Tempo Real.

O modelo em Rede de Petri gerado automaticamente é traduzido em arquivos atendendo aos padrões de entrada de dados de um *Analisador* de Redes de Petri desenvolvido na USP/SP e um *Simulador* de Redes de Petri desenvolvido no LAAS/Toulouse/França; cujo objetivo é validar nas etapas iniciais do ciclo de vida, as especificações do Sistema de Software com Características Tempo Real Projetado.

I N D I C E

CAPITULO I

INTRODUÇÃO

1.1. MOTIVAÇÃO

O desenvolvimento de um sistema de software passa por um ciclo onde várias *fases* se sucedem, iniciando com a concepção do sistema e terminando quando este deixar de existir [JINO 86], [LAUB 81].

Este ciclo recebe comumente a denominação de *ciclo de vida* de um sistema de software e podemos estabelecer como referência as seguintes fases:

- Especificação de Requisitos;
- Especificação Funcional;
- Projeto Lógico;
- Implementação;
- Testes de aceitação;
- Manutenção.

Especificação de Requisitos

Consiste em definir as especificações globais de um sistema de software que satisfaça uma necessidade existente.

Estas especificações são descritas na forma de textos verbais, atendendo a conceitos preliminares de estruturação e organização do sistema.

Especificação Funcional

Consiste em traduzir os requisitos da fase anterior, na forma de funções que o sistema deverá ter, para satisfazer a aplicação a que se destina.

Esta fase inclui a decomposição do sistema de forma a facilitar o trabalho de estruturação do mesmo.

Projeto Lógico

Consiste em descrever detalhadamente como a implementação do sistema deve ser feita.

Implementação

Tomando como referência as descrições da fase anterior, esta fase consiste na implementação propriamente dita dos programas que compõem o sistema de software em desenvolvimento.

Testes de Aceitação

Consiste em testar o sistema para verificar se ele atende às necessidades originais e às várias especificações e descrições geradas nas fases anteriores.

Manutenção

Consiste em manter o sistema em funcionamento, atendendo às finalidades de uso a que foi desenvolvido.

Existem atualmente diversos sistemas de apoio a desenvolvimento de software, que possuem metodologias específicas para cada fase do projeto de software.

A tendência evolutiva na área é a concepção de sistemas que integrem de forma automática diferentes fases do ciclo de vida de um projeto de software.

1.2. OBJETIVO

O presente trabalho de Tese de Mestrado tem por objetivo a *geração automática* de uma Rede de Petri, a partir da Especificação de um Sistema de Software com Características Tempo Real.

Esta *especificação* tem por finalidade servir de suporte na fase de *Especificação Funcional* de um Sistema de Software com características Tempo Real e a Rede de Petri gerada tem por finalidade a *validação* desta especificação, nos primeiros estágios, (fases iniciais), de desenvolvimento de um projeto de software.

A *especificação* do sistema é feita através de duas interfaces, "*Interface de Especificação de Sistemas*" e "*Interface de Descrição da Dinâmica de Sistemas*", cujo objetivo é gerar um conjunto de base de dados que traduzam de forma estruturada esta *descrição* do sistema.

Os conceitos elementares utilizados nestas interfaces estão baseados na "*Linguagem de Especificação SPOS-S*" que possui um conjunto sintático/semântico formalizado para descrever o projeto de um Sistema de Software Tempo Real [BIEW 79], [BIEW 80], [LAUB 81].

A estes conceitos foram adicionados outros conceitos que permitam a troca de mensagem entre elementos de projeto, baseados nos tópicos de "*Definição*" contidos na tese de doutorado "*Ambiente de Programação Distribuída. Definição, Análise e Síntese*" [BRES 85].

As interfaces desenvolvidas no trabalho têm duas estruturas; a primeira voltada para o usuário, permitindo uma entrada de dados interativa e utilizando vários níveis de menus; a segunda voltada para o gerenciamento das funções básicas de manipulação das bases de dados.

No caso específico de sistemas de apoio ao desenvolvimento de Software Tempo Real, além das especificações convencionais de projeto de um sistema de software é necessário adicionar ferramentas de análise e simulação que permitam validar estas especificações [BIEW 80].

Neste contexto a Rede de Petri gerada de forma automática tem por finalidade *validar* as especificações do sistema de software projetado.

Esta validação é feita através da tradução das Redes de Petri em *arquivos* atendendo os padrões de entrada de dados de um *analisador* de Redes de Petri desenvolvido na USP/SP e um *simulador* de Redes de Petri desenvolvido no LAAS-Toulouse/França, [BRES 85], [VALE 84b].

Os programas de análise e simulação possuem linguagens próprias com recursos de análise sintática e semântica, fato que a princípio poupa a necessidade de incluir estes recursos nas interfaces implementadas.

1.3. ORGANIZAÇÃO DO TRABALHO

O Capítulo 2 descreve a Interface de Especificação de Sistemas, seus módulos, os elementos e comandos que constituem cada módulo.

O Capítulo 3 descreve a Interface de Descrição da Dinâmica do Sistema, seus módulos e os comandos que constituem cada módulo.

O Capítulo 4 contém um estudo sobre Redes de Petri, incluindo um estudo teórico dos conceitos fundamentais, análise e simulação.

O Capítulo 5 descreve a conversão automática dos comandos das Interfaces de Descrição de Sistemas de Software Tempo Real em primitivas de Redes de Petri.

O Capítulo 6 contém um estudo sobre análise de Redes de Petri incluindo a apresentação do analisador desenvolvido na USP/SP-Brasil.

O Capítulo 7 contém um estudo sobre simulação de Redes de Petri incluindo a apresentação do simulador desenvolvido no LAAS/ Toulouse-França.

O Capítulo 8 contém a Estrutura do Sistema com os sucessivos refinamentos em módulos e as correspondentes Bases de dados.

O Capítulo 9 contém os resultados experimentais, através de três conjuntos de programas interligados através do interpretador Prolog, incluindo a geração automática das Redes de Petri, Análise e Simulação.

O Capítulo 10 apresenta as conclusões finais e sugestões de continuidade dos trabalhos iniciados nesta tese.

A tese possui um apêndice, com uma descrição sucinta da Linguagem Prolog utilizada na programação.

CAPITULO II

INTERFACE DE ESPECIFICAÇÃO DE SISTEMAS (LES)

2.1. INTRODUÇÃO

A Interface de Especificação de Sistemas (LES) reune um conjunto de conceitos objetivando especificar, decompor e inter-relacionar diferentes *objetos de projeto* de software de um sistema.

Os *objetos de projeto* estão reunidos em dois grupos que são os *elementos de projeto* e os *elementos operacionais*.

Os *elementos de projeto* são denominados de sistema, módulo e ação e estão descritos no item 2.2.

Os *elementos operacionais* são denominados de porta, condição, evento e dado e estão descritos no item 2.3.

A nível de especificação esta interface possui recursos adequados à tarefa de organizar uma ficha para cada objeto de projeto, possuindo também recursos que permitam mostrar, alterar e imprimir as informações constantes nestas fichas.

A nível de decomposição esta Interface possui recursos que permitam estabelecer as relações hierárquicas existentes entre os elementos de projeto em três níveis sucessivos de refinamento; Sistema, Módulo e Ação.

A nível de inter-relação esta Interface possui recursos adequados às tarefas de conectar os diferentes elementos operacionais através de fluxos de controle, comunicação e enlace.

A Interface LES compõe-se de seis módulos abaixo relacionados, que serão descritos nos itens a seguir:

- Módulo de Especificação de Elementos de Projeto (MEP)
- Módulo de Especificação de Elementos Operacionais (MEO)
- Módulo de Decomposições (MDE)
- Módulo de Fluxo de Controle (MFC)

- Módulo de Comunicação (MCO)

- Módulo de Enlace (MEN).

Esta divisão em módulos facilita a implementação do sistema e a estruturação da base de dados, cujo principal objetivo neste trabalho é conter as informações necessárias e suficientes para a *Geração Automática* das Redes de Petri e geração de arquivos de entrada do Analisador e Simulador de Redes de Petri.

2.2. MÓDULO DE ESPECIFICAÇÃO DE ELEMENTOS DE PROJETO (MEP)

2.2.1. Introdução

Os elementos de projeto são denominados de *Sistema*, *Módulo* e *Ação* e traduzem três níveis sucessivos de refinamento de um projeto de software de um sistema [BIEW 79], [BIEW 80].

O objetivo deste módulo é organizar uma ficha de especificação para cada *elemento de projeto*, possuindo recursos que permitam inserir, mostrar, alterar e imprimir estas especificações.

Os dados constantes destas fichas estão representados na figura a seguir:

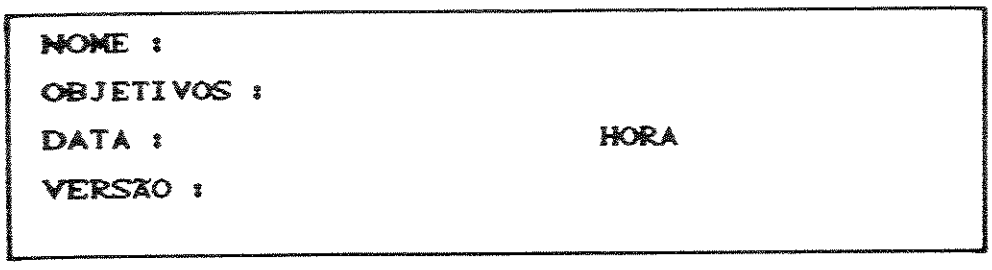


Figura 2.2.1 - Ficha de descrição de elementos de sistema

As informações são introduzidas através de um *Editor* que possui as funções básicas de um editor de texto.

O sistema atribui, de forma automática, um código de identificação para cada elemento à medida em que estes são definidos pelo projetista.

2.2.2. Elemento de Projeto Sistema

O elemento de projeto definido como *Sistema* é o elemento de nível mais alto, representando todo o sistema a ser projetado.

A representação gráfica e o símbolo do elemento *Sistema* estão descritos na figura a seguir:

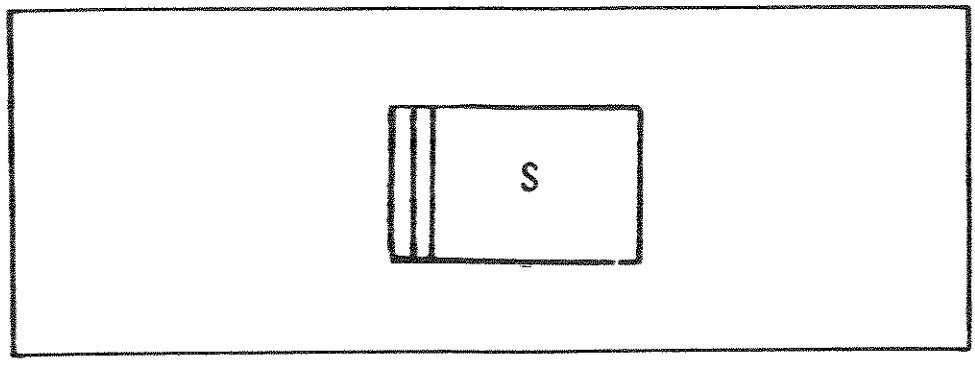


Figura 2.2.2 - Elemento Sistema
Representação gráfica e símbolo

2.2.3. Elemento de Projeto Módulo

O elemento de projeto definido como *Módulo* é o primeiro nível de decomposição do sistema, representando subsistemas e sem conter elementos operacionais.

A representação gráfica e o símbolo do elemento *Módulo* estão descritos na figura a seguir:

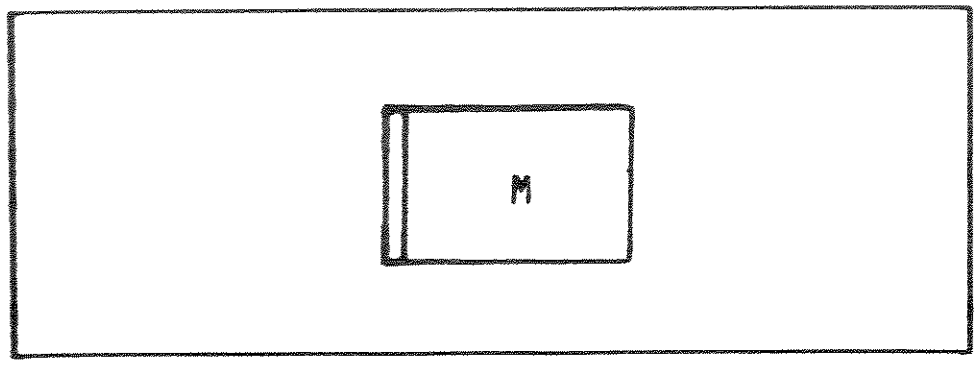


Figura 2.2.3 - Elemento Módulo
Representação gráfica e símbolo

2.2.4. Elemento de Projeto Ação

O elemento de projeto definido como *Ação* é o segundo nível de composição do sistema.

O elemento ação representa o processamento de informações e pode ser decomposto em outras ações, as quais podem ser independentes ou estar inter-relacionadas com elementos operacionais, através de Fluxos de Controle, Comunicação e Enlace.

A representação gráfica e o símbolo do elemento *Ação* estão descritos na figura a seguir:

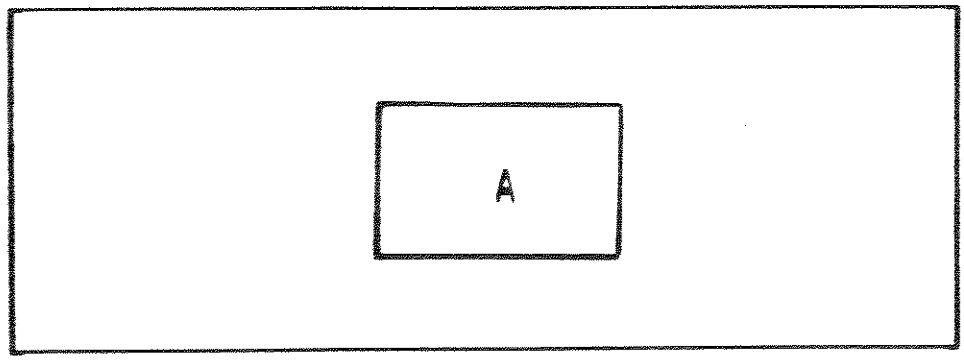


Figura 2.2.4 - Elemento Ação
Representação gráfica e símbolo

2.3. MÓDULO DE ESPECIFICAÇÃO DE ELEMENTOS OPERACIONAIS (MEO)

2.3.1. Introdução

Os elementos operacionais são denominados de *Condição*, *Porta*, *Evento* e *Dado* e traduzem as inter-relações de fluxos de controle, comunicação e enlace do sistema.

O objetivo deste módulo é organizar uma ficha de especificação para cada elemento de projeto, possuindo recursos que permitam inserir, mostrar, alterar e imprimir estas especificações.

As inter-relações entre ações e elementos operacionais através das estruturas de Fluxo de Controle, comunicação e enlace, permitem descrever as características do sistema incluindo características de tempo real [BIEW 79], [BIEW 80].

Os dados constantes destas fichas são os mesmos representados na Figura 2.2.1.

As informações são introduzidas através de um *Editor* que possui as funções básicas de um editor de texto.

O sistema atribui, de forma automática, um código de identificação para cada elemento à medida em que estes são definidos pelo projetista.

2.3.2. Elemento Operacional Condição

O elemento operacional *Condição* é uma entidade lógica possuindo duas saídas, *verdadeira* se a condição for satisfeita e *falsa* se a condição não for satisfeita.

A representação gráfica e o símbolo do elemento *Condição* estão descritos na figura a seguir:

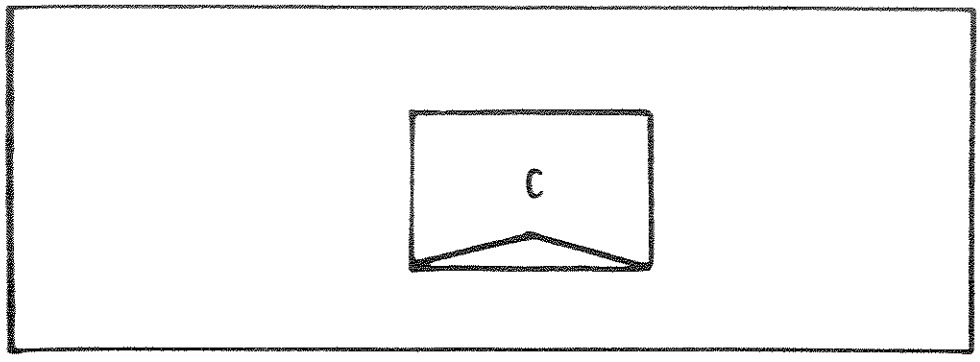


Figura 2.3.2 - Elemento Condição
Representação gráfica e símbolo

2.3.3. Elemento Operacional Porta

O elemento operacional *Porta* é responsável pela troca de mensagem entre as ações.

Associado ao conceito de *Porta* está um comando definido como *Enlace* que estabelece a ligação entre uma *Porta de Saída* da ação onde originou a mensagem e uma *Porta de Entrada* da ação destino da mensagem.

A representação gráfica e o símbolo do elemento *Porta* estão descritos na figura a seguir:

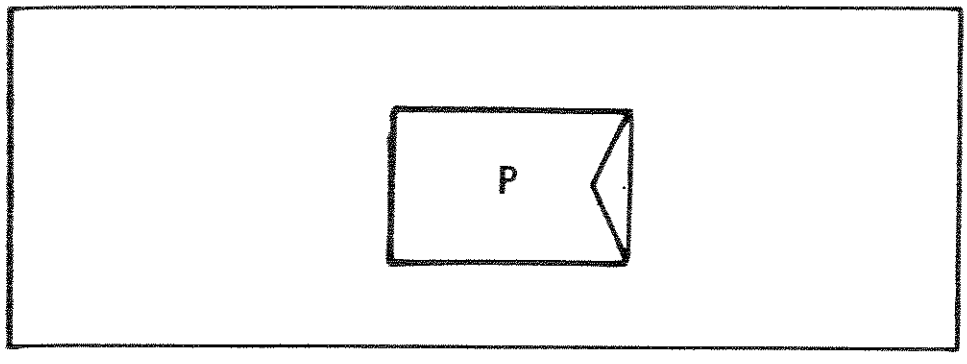


Figura 2.3.3 - Elemento Porta
Representação gráfica e símbolo

2.3.4. Elemento Operacional Evento

O elemento operacional *Evento* é responsável pela ativação de uma ação.

A representação gráfica e o símbolo de elemento *Evento* estão descritos na figura a seguir:

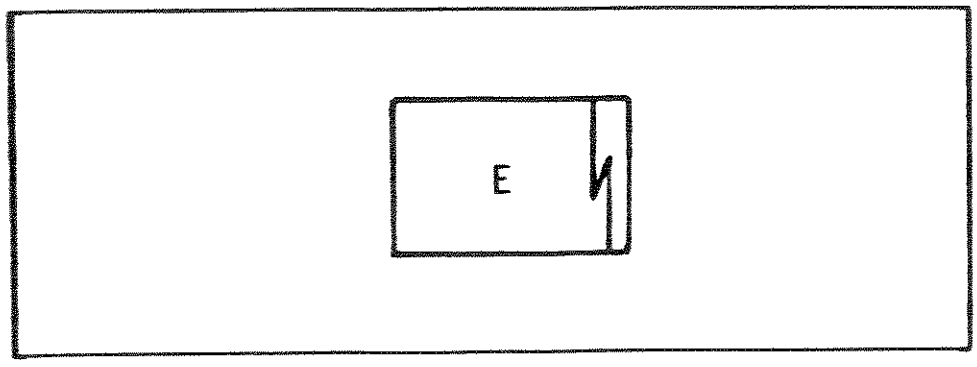


Figura 2.3.4 - Elemento Evento
Representação gráfica e símbolo

2.3.5. Elemento Operacional Dado

O elemento operacional *Dado* é utilizado para descrever os elementos de informação existentes no sistema projetado.

A representação gráfica e o símbolo do elemento *Dado* estão descritos na figura a seguir:

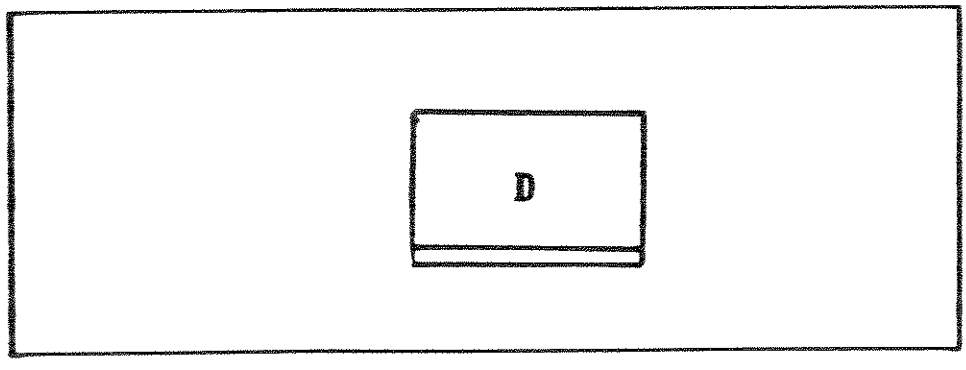


Figura 2.3.5 - Elemento Dado
Representação gráfica e símbolo

2.4. MÓDULO DE DECOMPOSIÇÃO (MDE)

2.4.1. Introdução

Este módulo descreve a decomposição dos elementos de projeto, sistema, módulo e ação, que são os três níveis do refinamento do sistema no âmbito de análise estruturada.

Esta decomposição consiste em estabelecer relações entre um elemento de nível mais alto e os correspondentes elementos nos quais ele é decomposto.

2.4.2. Decomposição do Elemento Sistema

O elemento *Sistema* pode ser decomposto em diferentes módulos atendendo às diferentes funções a que se destina o sistema em projeto.

A figura a seguir ilustra a decomposição de um sistema em módulos.

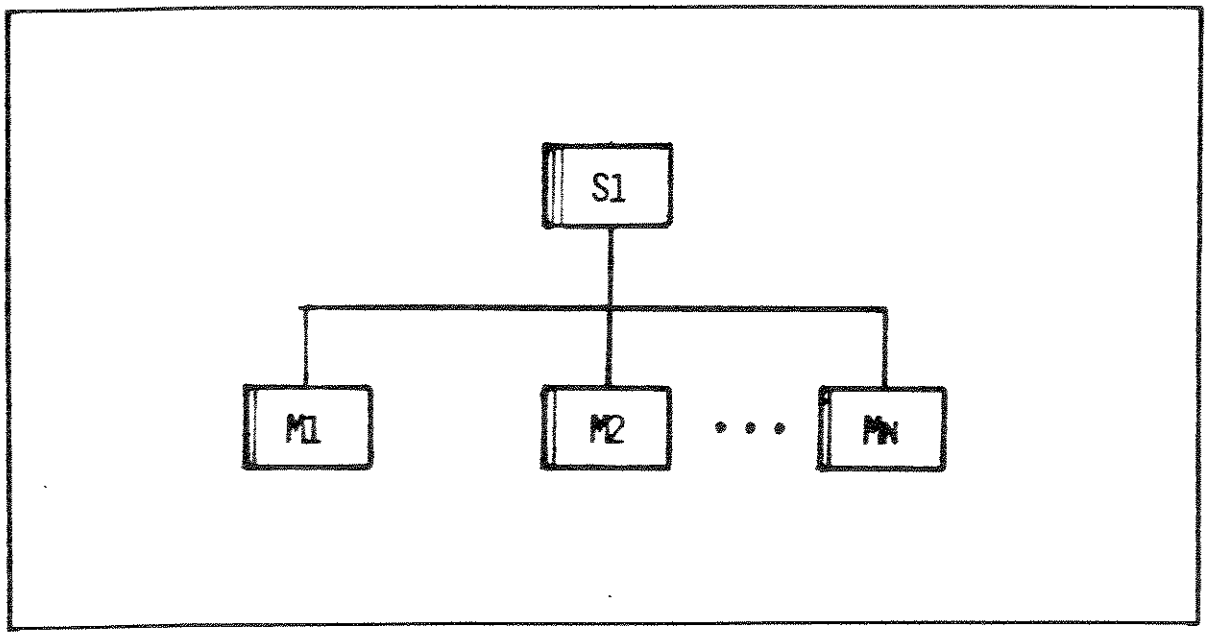


Figura 2.4.2 - Decomposição do Elemento Sistema

2.4.3. Decomposição do Elemento Módulo

O elemento *Módulo* pode ser decomposto em diferentes ações que traduzem um contexto mais ligado ao projeto lógico do sistema.

A figura a seguir ilustra a decomposição de um módulo em ações.

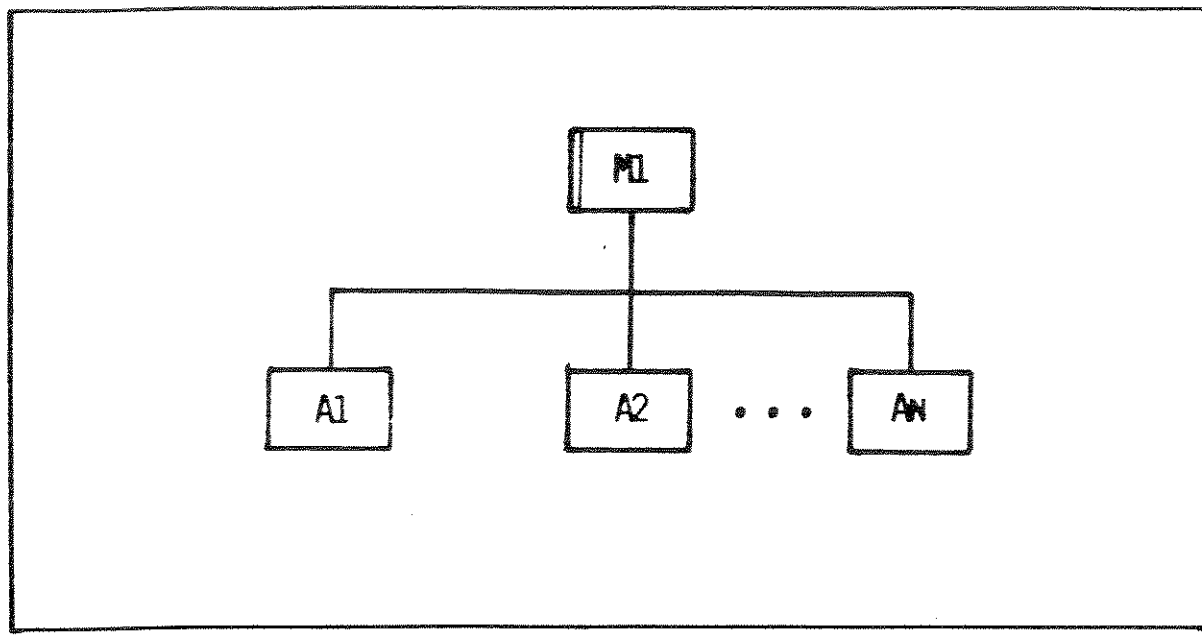


Figura 2.4.3 - Decomposição do elemento Módulo

2.4.4. Decomposição do Elemento Ação

O elemento *Ação* é o nível mais baixo de estruturação do sistema e será decomposto em ações que serão inter-relacionadas com os elementos operacionais através das estruturas de Fluxo de Controle, Comunicação e Enlace.

2.5. MÓDULO DE FLUXO DE CONTROLE (MFC)

2.5.1. Introdução

Este módulo possui um conjunto de comandos cuja função é estabelecer o Fluxo de Controle entre os elementos operacionais de um sistema.

Estas conexões são definidas em grupos estabelecendo relações de sequenciamento, seleção, iteração e paralelismo, representando as várias construções para a especificação do Fluxo de Controle do sistema.

2.5.2. Comando Sequencial

Este comando descreve as ações que são executadas de forma sequencial.

A representação gráfica do *Comando Sequencial* está descrita na figura a seguir:

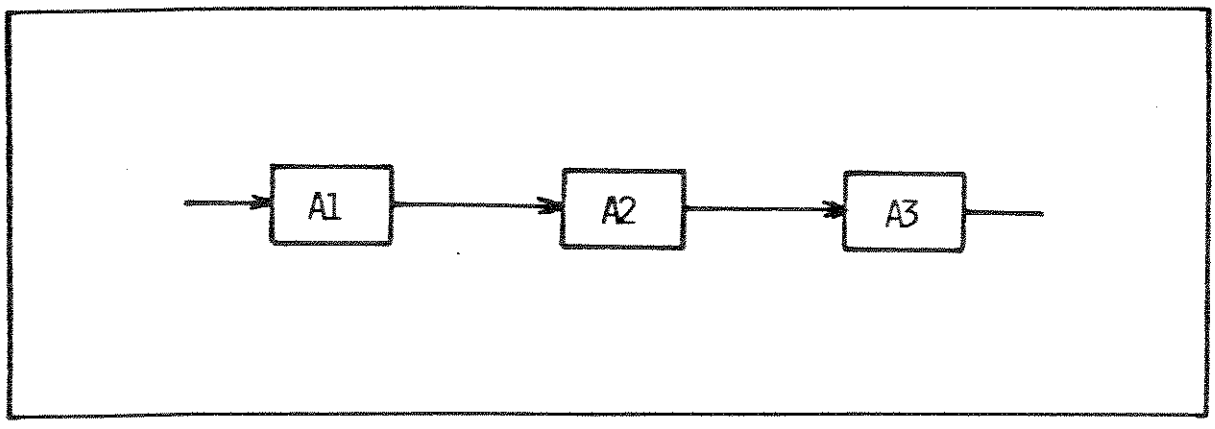


Figura 2.5.2 - Comando Sequencial
Representação gráfica

2.5.3. Comando Cíclico (WHILE / REPEAT UNTIL)

Este comando descreve a repetição de uma ação, enquanto uma condição estiver sendo satisfeita.

O teste da condição é feito antes da execução da ação.

A representação gráfica do *Comando Cíclico* está descrita na figura a seguir.

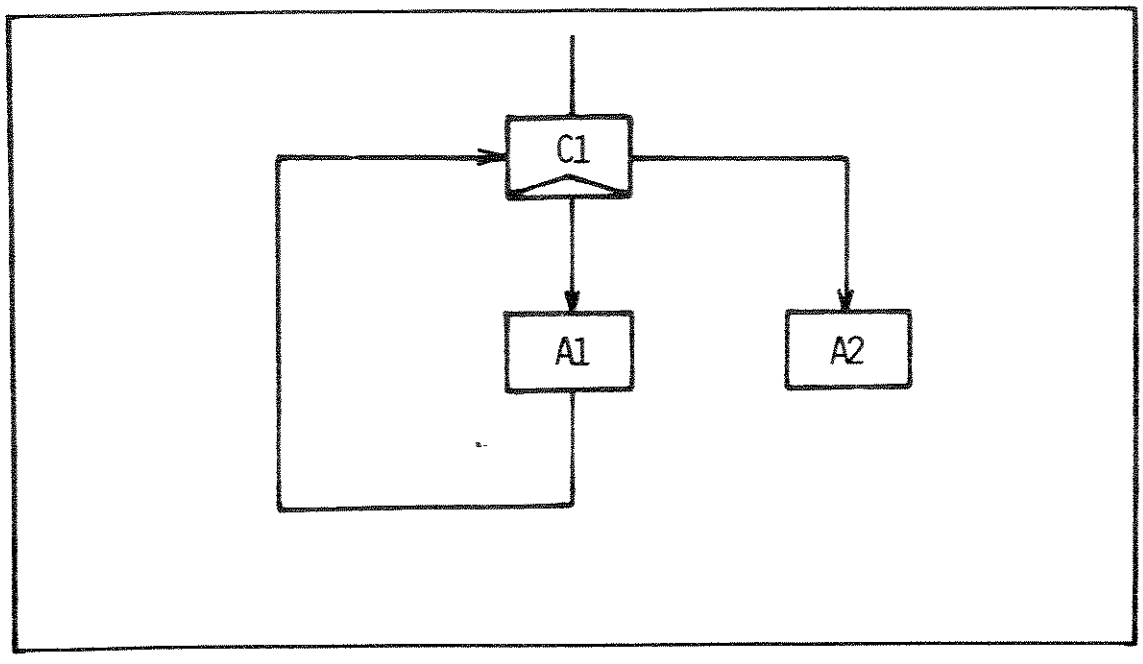


Figura 2.5.3 - Comando Cíclico
Representação gráfica

2.5.4. Comando Desvio Condicional (IF-THEN-ELSE)

Este comando descreve a divisão do fluxo de controle em dois caminhos alternativos, dependendo do valor booleano assumido pelo elemento condição.

Se a condição for satisfeita, o caminho será definido pela saída *verdadeira* e em caso contrário o caminho será definido pela saída *falsa*.

A representação gráfica do *Comando Desvio Condicional* está descrita na figura a seguir.

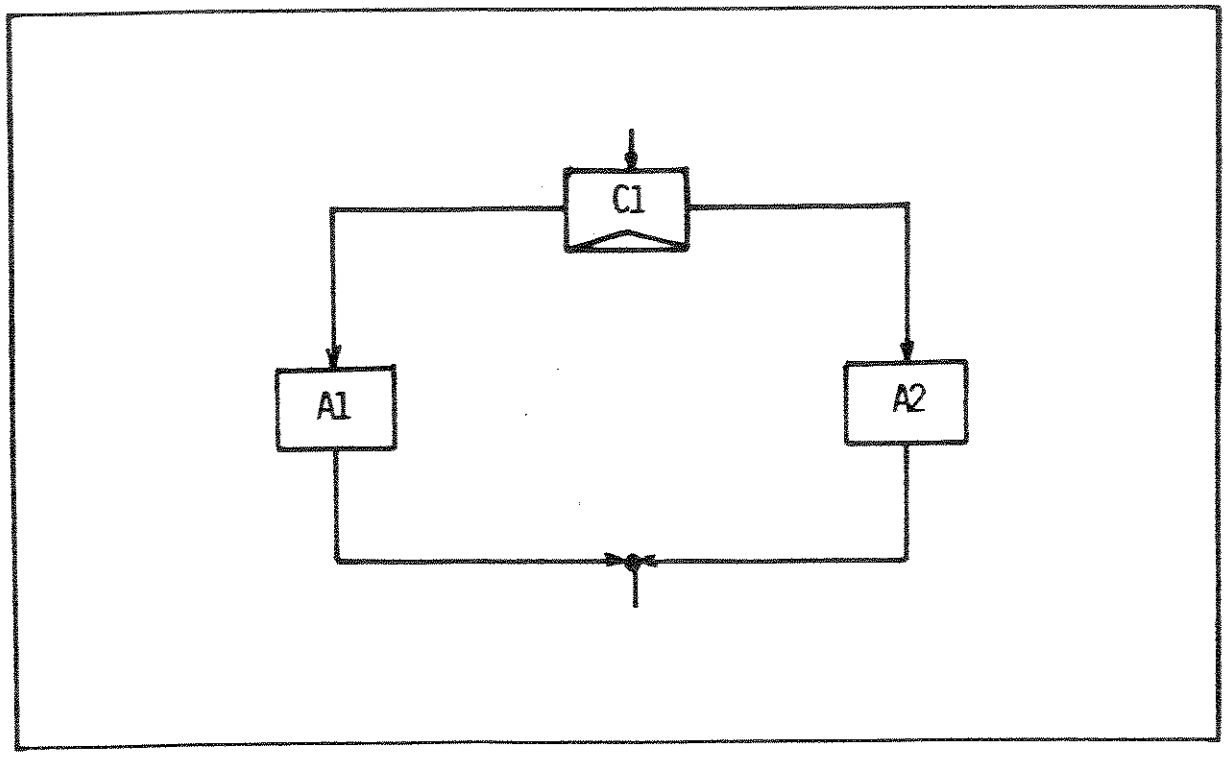


Figura 2.5.4 - Comando Desvio Condicional
Representação gráfica

2.5.5. Comando Desvio Múltiplo (CASE)

Este comando descreve a divisão do fluxo de controle em diversos caminhos alternativos dependendo do valor assumido pelo elemento condição.

A representação gráfica do *Comando Desvio Múltiplo* está descrita na figura a seguir.

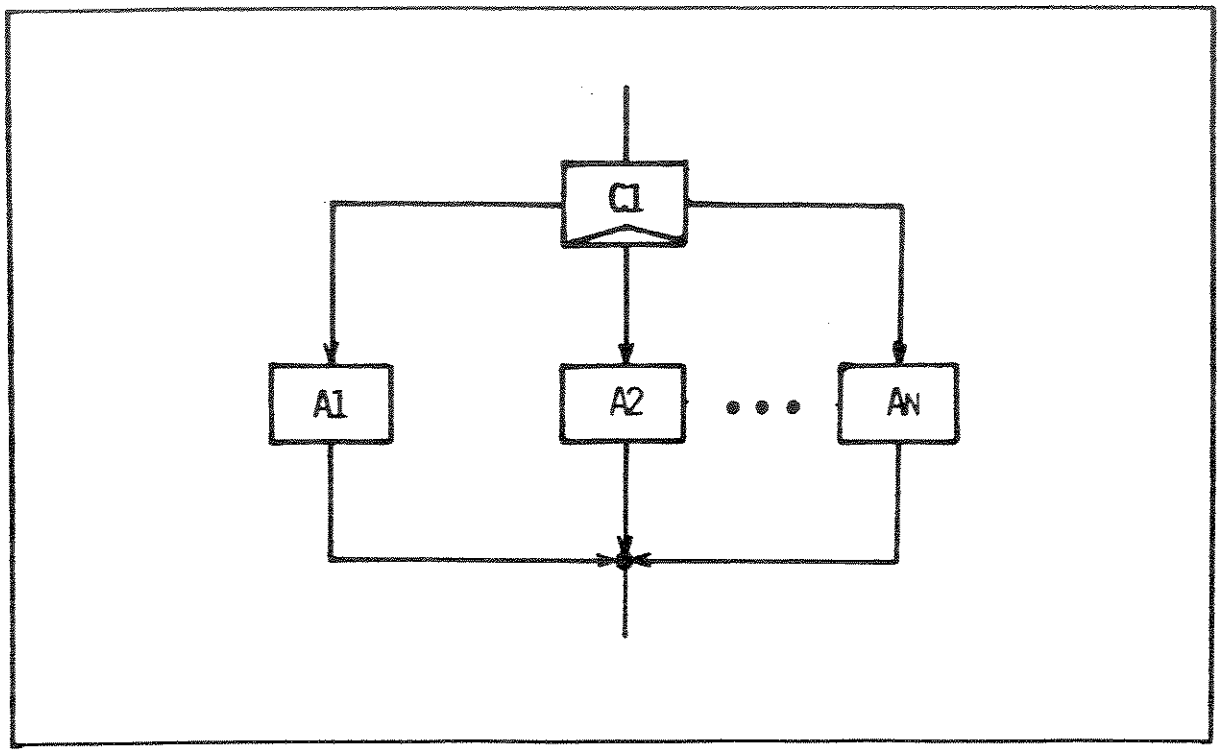


Figura 2.5.5 - Comando Desvio Múltiplo
Representação gráfica

2.5.6. Comando Paralelo

Este comando descreve um conjunto de ações que são executadas simultaneamente.

O fluxo de controle só será retomado após o término de execução de todas as ações.

A representação gráfica do *Comando Paralelo* está descrita na figura a seguir.

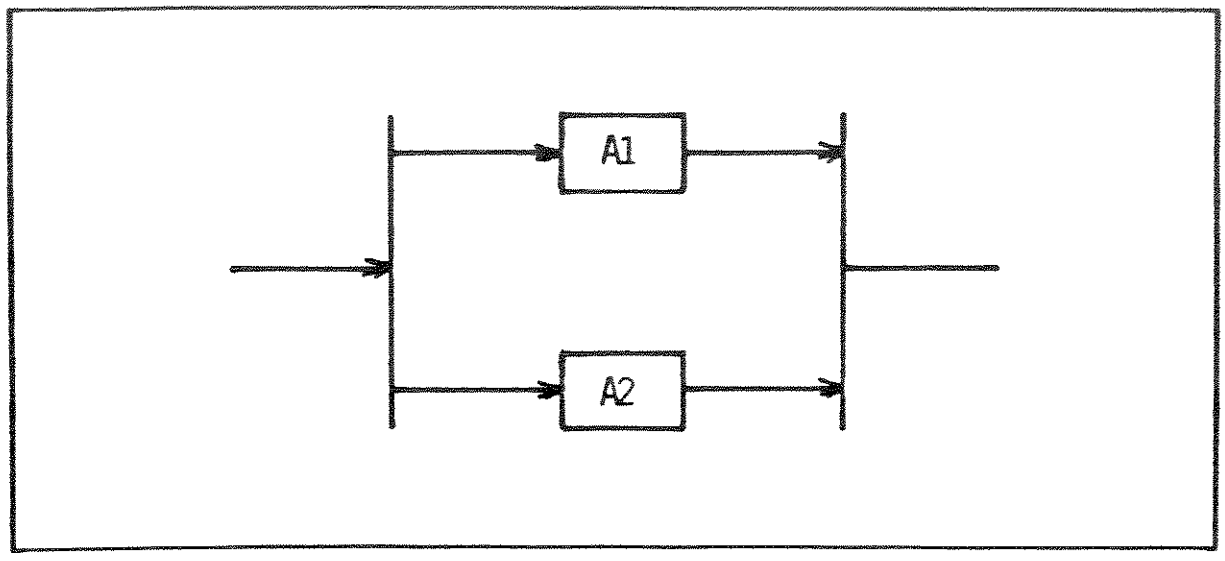


Figura 2.5.6 - Comando Paralelo
Representação gráfica

2.6. MÓDULO DE COMUNICAÇÃO (MCO)

2.6.1. Introdução

Este módulo possui um conjunto de comandos cuja função é estabelecer *a troca de mensagens* entre as ações de um módulo ou sistema [BRES 85].

A mensagem é a unidade elementar para transporte de informação, não existindo nenhuma outra forma de comunicação tal como memória compartilhada.

O sistema possui primitivas *Envia* e *Recebe* para gerenciar o envio e o recebimento de mensagens.

Para que uma ação A1 de um modelo M1 envie uma mensagem M para uma ação A2 do próprio módulo M1 ou outro módulo qualquer, seria suficiente utilizar uma operação primitiva *Envia* da forma *Envia M para A2* enquanto que A2 para ter acesso à mensagem enviada executaria numa operação *Recebe* da forma *Recebe M de A1*.

Em termos de aplicações práticas este modelo simplificado não é suficiente para gerenciar todos os aspectos envolvidos na troca de mensagens entre duas ações (processos) e para se resolver este problema introduz-se o elemento operacional *porta* associado às ações.

Uma ação passa portanto a possuir *portas de entrada* e *portas de saída* e os comandos de *Envia* deve especificar o nome de uma porta de saída da ação em lugar do nome da ação que deve receber a mensagem. O mesmo deve ocorrer com o comando *Recebe*.

Os comandos *Envia* e *Recebe* podem ser desdobrados em um conjunto de opções que atendam a diversos requisitos ou restrições do processo de comunicação.

- Opção ENVIA/RECEBE COM RESPOSTA (REPLY TO)

Esta opção consiste em acrescentar aos comandos *Envia* e *Recebe* originais uma porta de envio/recebimento de resposta.

- Opção FALHA (WHEN FAILURE DO)

Esta opção pode ser especificada no comando *Envia* para detectar falhas na transmissão de mensagens.

- Opção ESPERA (WHEN TIMEOUT)

Esta opção pode ser especificada no comando *Recebe*, sendo útil na detecção de falhas. Neste caso um parâmetro define o tempo máximo que a ação que executa o comando *Recebe* ficará bloqueada à espera de uma mensagem. Decorrido este tempo e se não ocorrer a recepção de alguma mensagem a execução da ação será retomada.

Além das opções descritas, pode haver outras que atendam a algum tipo de aplicação prática.

A implementação dos comandos de comunicação, neste trabalho, ficarão restritas aos comandos *Envia*, *Envia com Resposta*, *Recebe* e *Recebe com Resposta* descritos a seguir.

2.6.2. Comando Envia

Este comando descreve a forma mais simples de se enviar uma mensagem de uma ação origem para outra ação destino.

O envio da mensagem é feito de forma *assíncrona* através de uma porta de saída.

O termo *assíncrona* está associado ao fato do Fluxo de Controle não ficar interrompido em decorrência do envio da mensagem.

A representação gráfica do *Comando Envia* está descrita na figura a seguir.

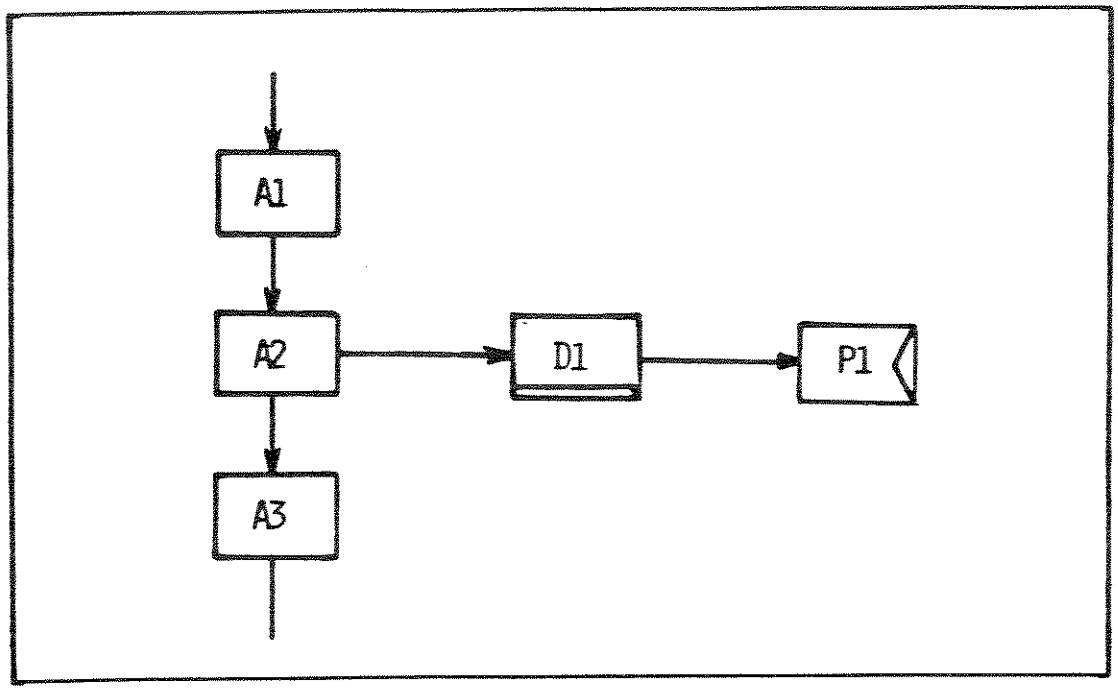


Figura 2.6.2 - Comando Envia
Representação gráfica

2.6.3. Comando Envia com Resposta

Este comando descreve o envio de mensagem de uma ação origem para outra destino, de forma *síncrona* através de uma porta de saída.

Neste caso o termo *síncrona* significa que o fluxo de controle fica bloqueado até que a confirmação do envio da mensagem seja recebida através de uma porta de entrada para recebimento de resposta.

A representação gráfica do *Comando Envia com Resposta* está descrita a seguir.

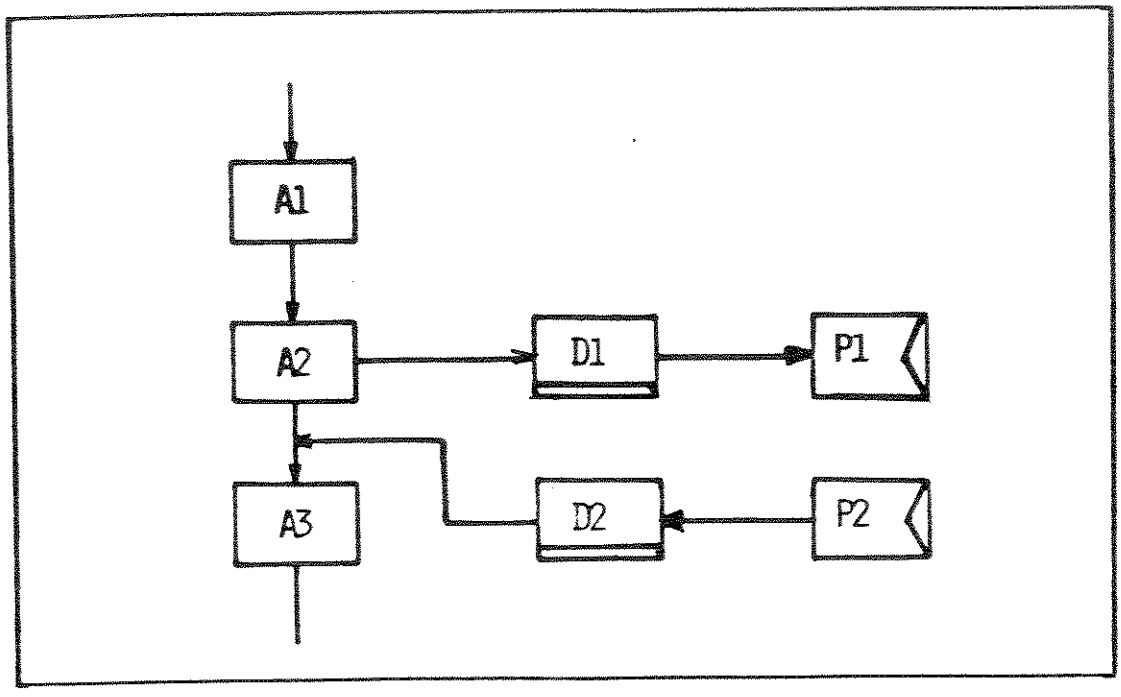


Figura 2.6.3 - Comando Envia com Resposta
Representação gráfica

2.6.4. Comando Recebe (Seletivo e Síncrono)

Este comando descreve a forma mais simples de uma ação destino receber uma mensagem enviada por uma ação origem.

O comando possui diversas portas de entrada, entre as quais uma é selecionada para receber a mensagem.

A representação gráfica do *Comando Recebe* está descrita na figura a seguir:

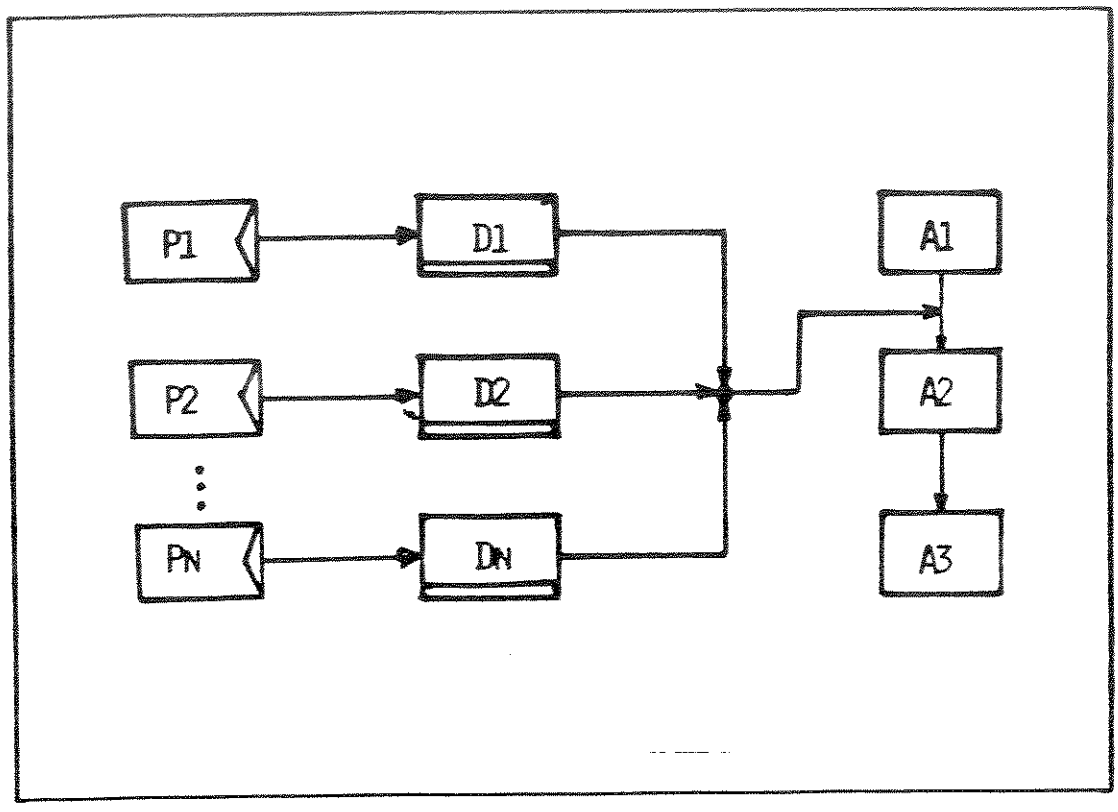


Figura 2.6.4 - Comando Recebe
Representação gráfica

2.6.5. Comando Recebe com Resposta (Seletivo e Síncrono)

Este comando descreve o recebimento de mensagem de uma ação destino com envio de uma resposta à ação de origem da mensagem, confirmando o recebimento da mensagem.

O envio da confirmação será feito através de uma porta de saída para envio de resposta.

A representação gráfica do *Comando Recebe com Resposta* está descrita na figura a seguir.

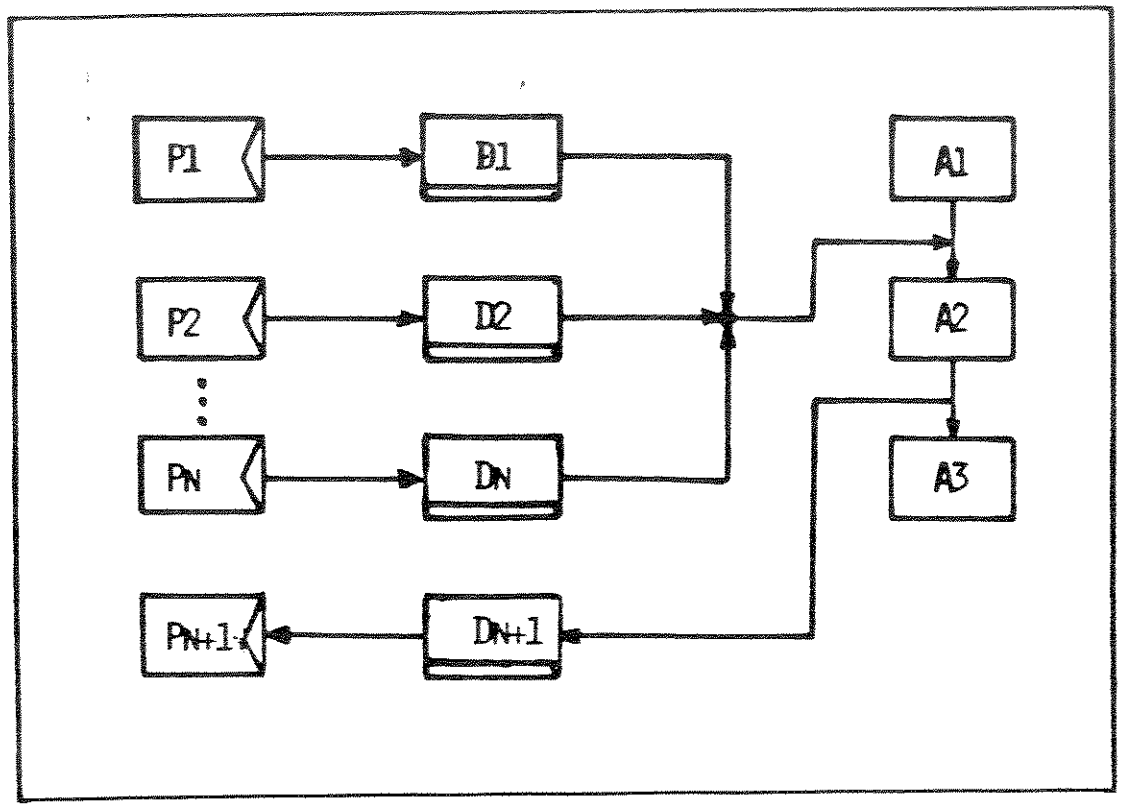


Figura 2.6.5 - Comando Recebe com Resposta
Representação gráfica

2.7. MÓDULO DE ENLACE (MEN)

2.7.1. Introdução

Os mecanismos apresentados no item anterior não são suficientes para a determinação dos destinos das mensagens uma vez que estas são enviadas a uma porta de saída da própria ação que executa os comandos de comunicação *envia* [BRES 85].

A introdução da entidade *enlace* permite que o destino da mensagem seja determninado. Uma mensagem pode chegar a um processo somente se existir um *enlace* entre uma porta de saída da ação origem e uma porta de entrada da ação destino.

Os *enlaces* são definidos através de comandos podendo ser de dois tipos:

2.7.2. Enlace Sistema

O *enlace* neste nível permite estabelecer as ligações entre ações de módulos distintos em um único sistema.

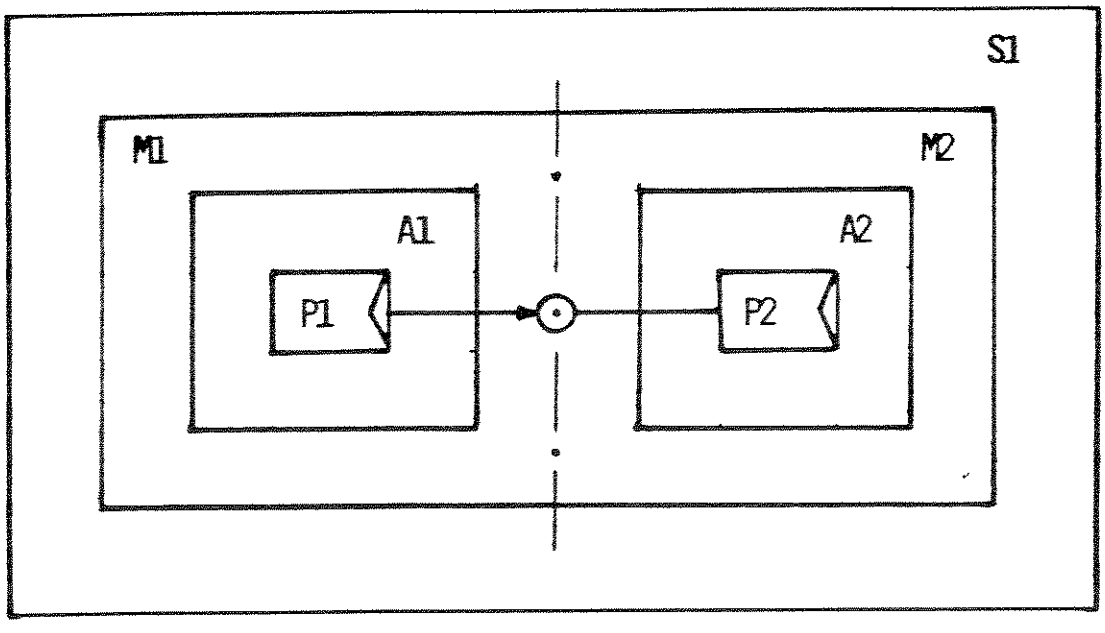


Figura 2.7.2 - Enlace Sistema
Representação Gráfica

2.7.3. Enlace Módulo

O *Enlace* neste nível permite estabelecer as interligações entre as ações de um único módulo.

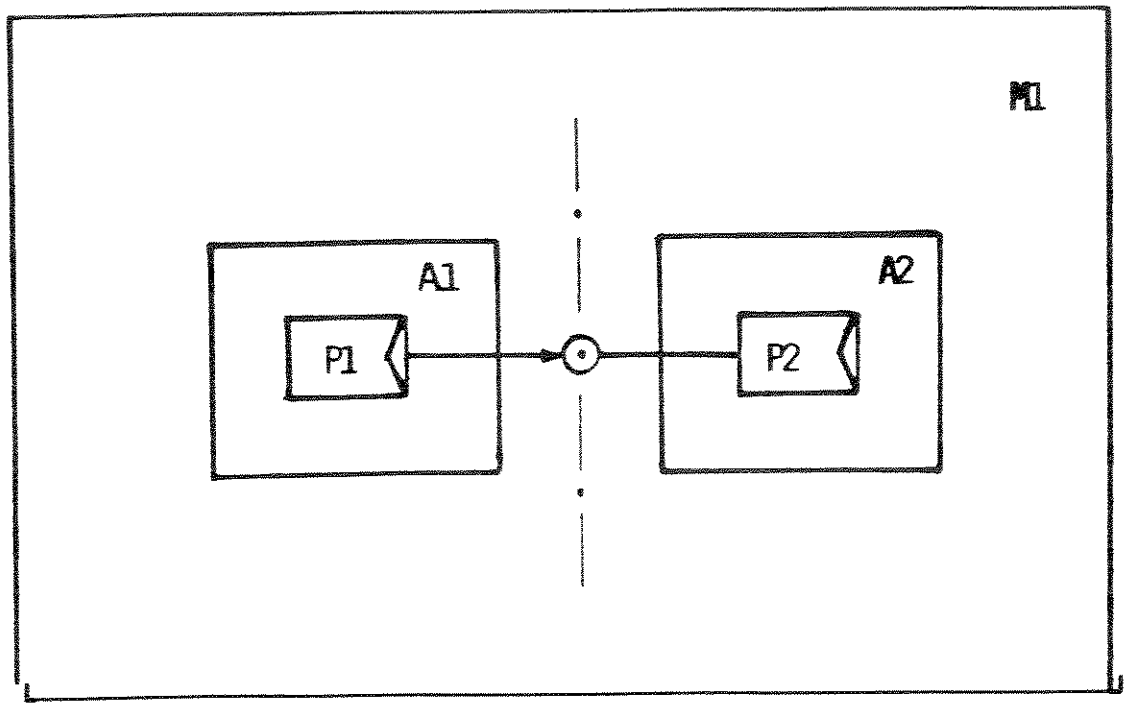


Figura 2.7.3 - Enlace Módulo
Representação Gráfica

Uma evolução natural seria dotar as estruturas de comunicação de elementos de enlace *inter-sistemas*, atribuindo portanto características de computação distribuída.

CAPÍTULO III

INTERFACE DE DESCRIÇÃO DA DINÂMICA DE SISTEMAS (LDS)

3.1. INTRODUÇÃO

Os sistemas computadorizados em geral trabalham com dois tipos de informações que são: programas e dados [GERT 75], [MAGA 86]:

Os programas são decompostos em três subtipos; Tarefa, Rotinas e Subrotinas, dependendo da aplicação a que se destinam.

Os dados são de dois tipos, *dedicados* quando associados a uma tarefa específica ou *comuns* quando servirem a diferentes tarefas. Os dados comuns são utilizados na comunicação entre diferentes tarefas.

As características *dinâmicas* de um sistema computadorizado podem ser analisadas através dos conceitos de *tempo* e *evento*.

Baseado no primeiro conceito (tempo) os programas que constituem este sistema podem ser executados em um instante específico de tempo, após um certo atraso ou ciclicamente em certos intervalos.

Quanto ao segundo conceito (evento) os programas podem ser inicializados através de eventos externos originados pelo sistema ou pelo operador, ou internamente dentro dos Fluxos de Controle.

As atividades *dinâmicas* de um sistema computadorizado são em geral organizadas por *Executivos de Tempo Real*, cujas funções básicas são as seguintes:

- Inicialização
- Gerenciamento de processos
- Alocação de recursos
- Tratamento de interrupções.

- Inicialização

A inicialização consiste em fazer a alocação dos recursos iniciais necessários a um processo, tais como:

- Inicializar estruturas de dados
- Criar descritores de processos
- Inicializar a interface de periféricos
- Inicializar o relógio de tempo real
- Alocar memória e pilhas necessárias.

- Gerenciamento de Processos

O Gerenciamento de Processos suporta um conjunto de funções associadas aos processos, dentre as quais destacam-se as seguintes:

- Criação de processos
- Eliminação de processos
- Suspensão de processos
- Reativação de processos
- Mudança de prioridade de processos
- Bloqueio de processos
- Comunicação e sincronização de processos
- Escalonamento de processos
- Retardo de processos.

- Alocação de Recursos

Os Sistemas de Controle Tempo Real têm como característica principal a execução de processos de forma concorrente, fato que requer a alocação de recursos tais como: CPU, Memória e Dispositivos Periféricos.

- Tratamento de Interrupções

Interrupções constituem uma ferramenta fundamental em um ambiente de Sistemas de Tempo real, onde o sistema tem que reagir rapidamente a eventos externos. Devido à sua natureza assíncrona estes eventos interferem de uma maneira *não determinística* com o fluxo de execução e podem causar muitos efeitos no sistema.

O Executivo Tempo Real deve permitir suficiente flexibilidade para o programador da aplicação no que se refere ao uso de interrupções, bem como proteger o sistema de interferências indesejáveis.

A existência de uma tarefa para o Executivo Tempo Real ocorre através do *Bloco de Controle da Tarefa* (BCT) que é uma estrutura de dados contendo informações importantes sobre a tarefa. Em geral estas informações são as seguintes:

- Identificação da tarefa
- Prioridade da tarefa
- Apontador para a área de memória da tarefa
- Apontadores para recursos alocados
- Área de salvaguarda de registradores
- Campo de tempo de retardo.

O bloco de controle da tarefa centraliza um conjunto de informações que permite ao núcleo tempo-real localizar todas as informações chaves relativas a uma tarefa.

O comportamento dinâmico de um sistema computadorizado controlado por um *Executivo Tempo Real*, pode ser analisado pelo estudo de quatro *estados* possíveis das tarefas que compõem este sistema:

- Estado Ativo:
  Uma tarefa está no estado ativo quando seu processamento está sendo efetuado;

- Estado Parado:
  Uma tarefa está no estado parado quando seu processamento estiver desativado;

- Estado Habilitado:
  Uma tarefa está no estado habilitado quando estiver pronta para ser processada, aguardando apenas o momento de início de processamento;

- Estado Bloqueado:
  Uma tarefa está no estado bloqueado quando sua execução foi suspensa pela ativação de uma tarefa com maior prioridade. Neste caso os registros da tarefa são salvos e retomam quando esta volta a ser executada.

A figura a seguir ilustra os estados das tarefas de um sistema computadorizado controlado por um *Executivo Tempo Real*.

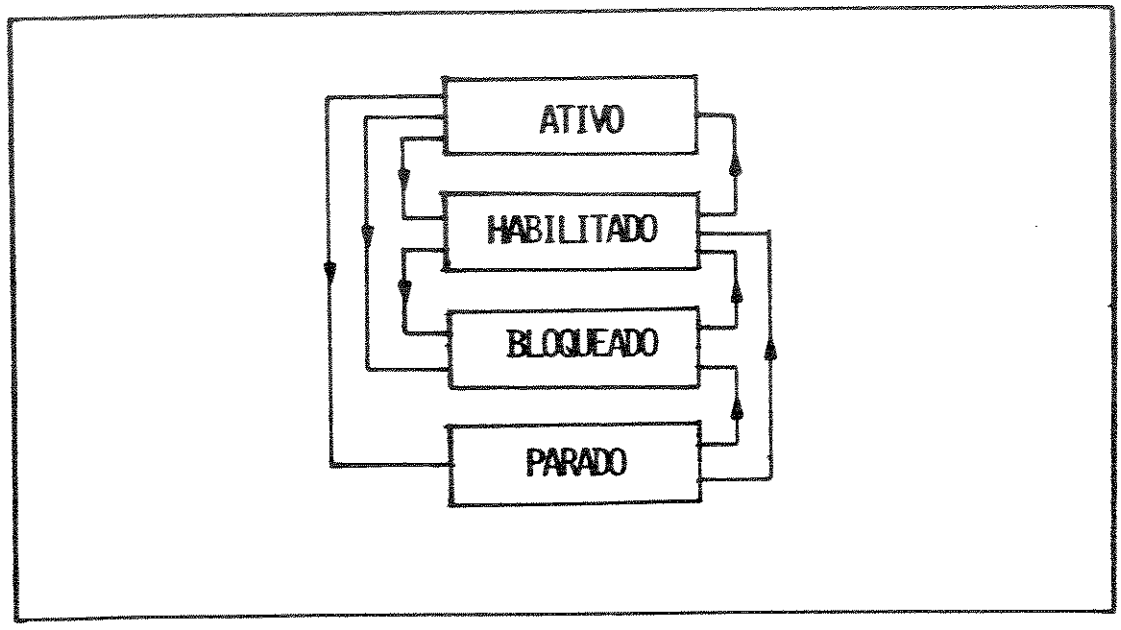


Figura 3.1 - Estados das tarefas

A Interface de Descrição da Dinâmica do Sistema (LDS) reune um conjunto de conceitos objetivando especificar, interpretar e simular algumas das características dinâmicas de um Sistema de Software Tempo Real.

A nível de especificação a Interface possui recursos adequados à tarefa de descrição da dinâmica do sistema, permitindo ainda mostrar, alterar e imprimir estas especificações.

A nível de interpretação a Interface permite descrever um conjunto de operadores e variáveis lógicas e algébricas que auxiliam na solução de conflitos e sincronização entre ações do sistema.

A nível de simulação a Interface possui recursos adequados ao estabelecimento de tempos de simulação e estatística, bem como recursos que permitem uma análise dos resultados obtidos.

A Interface LDS tem também como objetivo, servir de interface de alto nível para utilização de um simulador de Redes de Petri, desenvolvido no LAAS-Toulouse/França a nível de protótipo, com uma versão disponível no CTI-Campinas/Brasil [VALE 84a] [VALE 84b].

A Interface LDS compõe-se de três módulos abaixo relacionados, que serão descritos nos itens a seguir:

- Módulo de Especificação da Dinâmica (MED)
- Módulo de Interpretação da Dinâmica (MID)
- Módulo de Condições de Simulação (MCS).

Esta divisão em módulos facilita a implementação do sistema e a estruturação da base de dados, cujo principal objetivo neste trabalho é conter as informações necessárias e suficientes para a *Geração Automática* das Redes de Petri e geração dos arquivos de entrada do Analisador e Simulador de Redes de Petri.

3.2. MÓDULO DE ESPECIFICAÇÃO DA DINAMICA (MED)

3.2.1. Introdução

Este módulo tem por objetivo descrever o comportamento dinâmico do sistema.

Esta descrição é feita através de comandos que estabelecem critérios de *alocação de tempo* e definem relações entre eventos e tarefas.

3.2.2. Comando Ativação Cíclica

Este comando define um intervalo cíclico de ativação de uma determinada tarefa.

O dado D1 contém os valores Tempo da Primeira Ativação (Tempo_ativ) e Tempo de Repetição Cíclica de Ativação (Int_ativ).

A representação gráfica do *Comando Ativação Cíclica* está descrita na figura a seguir:

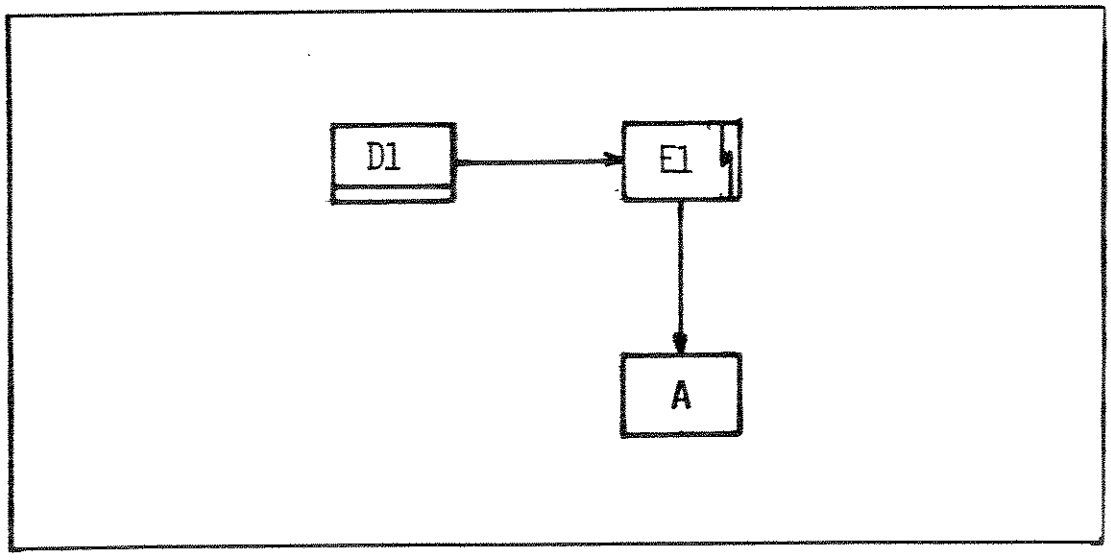


Figura 3.2.2 - Comando Ativação Cíclica
Representação gráfica

3.2.3. Comando Ativação por Interrupção

Este comando define a ativação de uma tarefa através da ocorrência de uma interrupção.

O dado D1 contém os valores Tempo da Primeira Ativação (Tempo_ativ) e Intervalo de Tempo (Int_ativ) onde poderá aleatoriamente ocorrer uma ativação da ação.

A representação gráfica do *Comando Ativação por Interrupção* está descrita na figura a seguir.

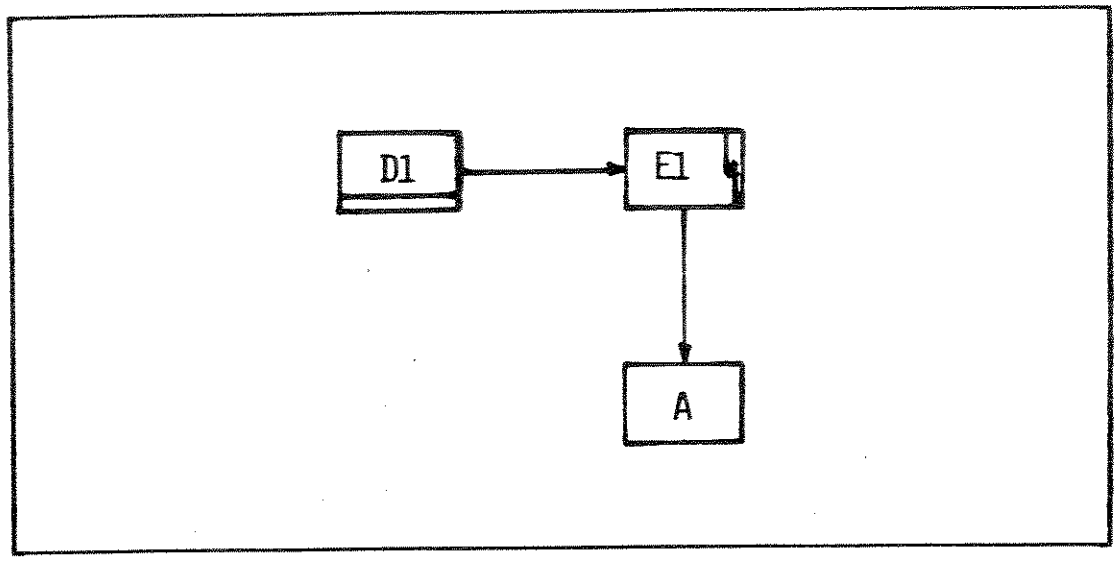


Figura 3.2.3 - Comando Ativação por Interrupção
Representação gráfica

3.2.4. Comando Atraso de Ativação

Este comando define um tempo (D1) de Atraso na Ativação de uma ação.

A representação gráfica do *Comando Atraso de Ativação* está descrita na figura a seguir.

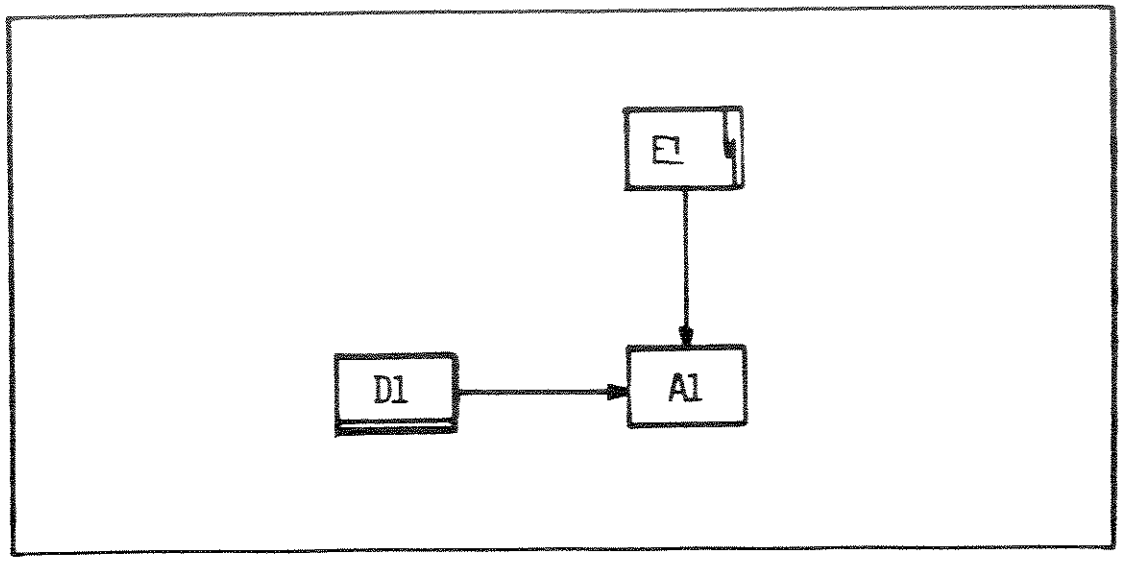


Figura 3.2.4 - Comando Atraso de Ativação
Representação gráfica

3.3. MÓDULO DE INTERPRETAÇÃO DA DINÂMICA (MID)

3.3.1. Introdução

Este módulo tem por objetivo interpretar *algumas* das características dinâmicas do sistema através de operadores e variáveis lógicas e algébricas.

Estas funções estão implementadas na forma de comandos interativos que estabelecem critérios para solução de conflitos, e sincronização na ativação de duas ações.

3.3.2. Comando Solução de Conflito

Este item define uma solução para o conflito de ativação de duas ou mais ações que sucedem o elemento condição.

Através de variáveis lógicas/algébricas e operadores relacionais, pode-se estabelecer critérios de escolha para os diferentes caminhos do fluxo de controle.

A reunião de variáveis lógicas/algébricas e dos operadores relacionais permite definir expressões lógicas/algébricas de dois tipos [VALE 84a], [VALE 84b]:

- Ação

  A = expressão algébrica e/ou
  A = expressão lógica

- Condição

  C = expressão algébrica e/ou
  C = expressão lógica

A implementação destas expressões (lógicas/algébricas) é feita através do editor, sendo que a *Análise Sintática* e *Semântica* será feita no ambiente do simulador.

A figura a seguir ilustra a utilização destes operadores, no caso do conflito existente no Comando Desvio Condicional.

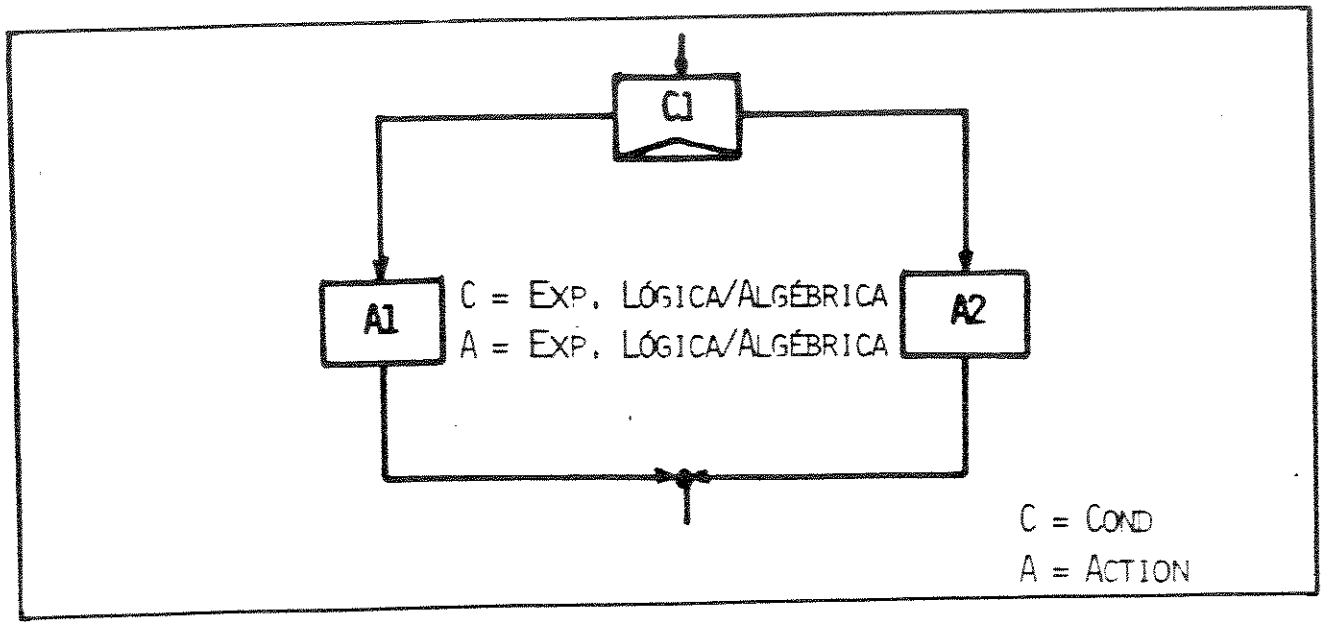


Figura 3.3.2 - Solução de Conflito

3.3.3. Comando Sincronização de Ações

Este item define uma solução para a ativação por *exclusão mútua* de duas ou mais ações.

Através de operadores e variáveis lógicas/inteiras, pode-se estabelecer os critérios de sincronização.

As expressões lógicas/algébricas neste caso são idênticas às definidas no item anterior.

A figura a seguir ilustra a utilização destes operadores, com duas ações enviando mensagens através de suas respectivas *portas de saída* a uma ação que irá receber uma das duas mensagens (exclusão) através de uma única *porta de entrada*.

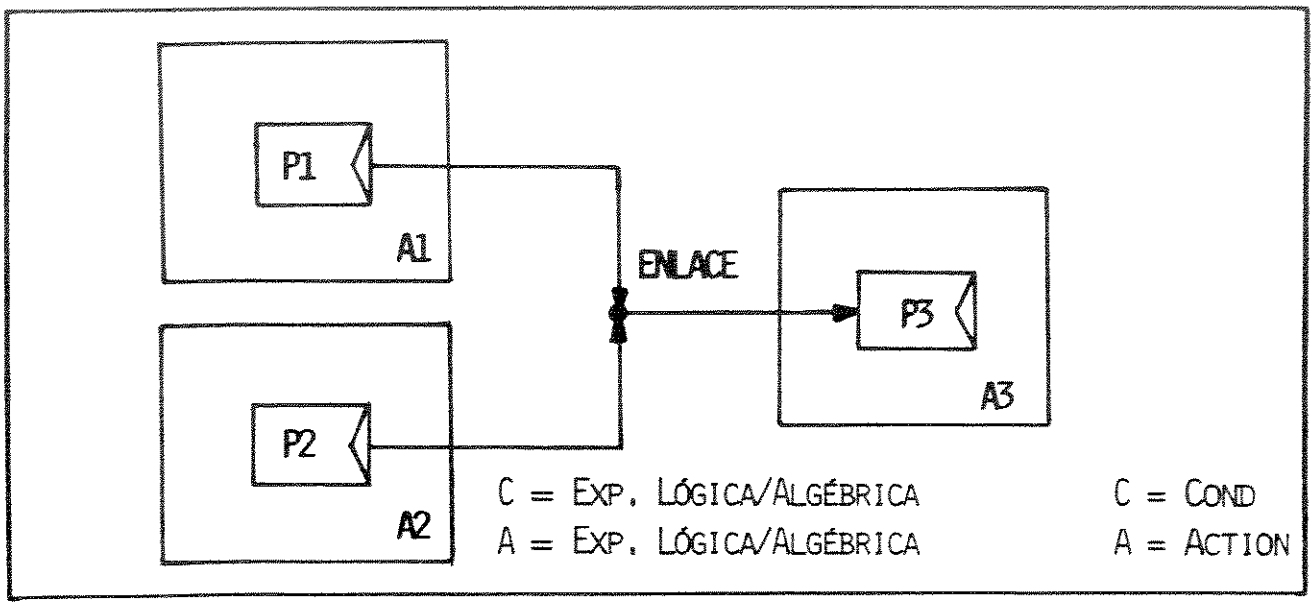


Figura 3.3.3 - Sincronização

3.4. MÓDULO DE CONDIÇÕES DE SIMULAÇÃO (MCS)

3.4.1. Introdução

Este módulo tem por objetivo descrever as condições de simulação do sistema projetado.

Estas condições são traduzidas pela determinação dos tempos de início e fim da simulação, tempos de observação/amostragem estatística, bem como do estabelecimento de condições iniciais de simulação.

3.4.2. Comando Tempo de Simulação

Este comando tem por objetivo definir os tempos inicial e final da simulação do sistema.

3.4.3. Comando Tempo de Observação

Este comando define o tempo inicial e a periodicidade de observação da simulação, ou seja, os tempos nos quais serão feitos relatórios estatísticos.

3.4.4. Comando Inicializa Execução

Este comando define as ações que estão em execução no início da simulação.

CAPITULO IV

ESTUDO SOBRE REDES DE PETRI

4.1. INTRODUÇÃO

Um modelo, é uma representação, frequentemente matemática, das características de um sistema real que esteja sendo estudado.

Através da manipulação desta representação, espera-se realizar diversos estudos, sem os custos e riscos advindos da manipulação do sistema real.

Dependendo do grau de complexidade de um sistema, a tarefa de se criar um modelo para o mesmo pode se tornar muito difícil, senão impossível. A busca de solução deste problema levou ao desenvolvimento de diversas metodologias e ferramentas, que facilitassem a tarefa de representação de um sistema.

Neste contexto, surgiu no ano de 1962 a teoria de *Redes de Petri* através da tese de doutorado *Kommunikation mit Automaten* de Carl Adam Petri, formulada como base para a comunicação entre componentes assíncronos de um sistema computadorizado [PETE 81].

As Redes de Petri são geralmente utilizadas para modelar sistemas onde eventos possam ocorrer concomitantemente, ou seja, eventos paralelos.

4.2. ESTRUTURA DE UMA REDE DE PETRI

As Redes de Petri são compostas de dois componentes básicos:

- Um conjunto de lugares P
- Um conjunto de transições T

Um elemento arbitrário de P que possui notação P_i, $i = 1,2,\ldots,n$, e um elemento arbitrário de T, que possui notação T_j, $j = 1,2,\ldots,m$.

O relacionamento entre os *lugares* e *transições* é feito através de duas funções:

- Uma função de entrada I
- Uma função de saída O

A função de entrada $I(t_j)$, define o conjunto de *lugares de entrada* correspondentes a uma transição t_j.

A função de saída $O(t_j)$ define o conjunto de *lugares de saída* correspondentes a uma transição t_j.

Estes quatro itens definem a *estrutura* de uma Rede de Petri que formalmente é definida por uma quádrupla $C = (P,T,I,O)$.

4.3. GRAFICO DE UMA REDE DE PETRI

O *gráfico* de uma Rede de Petri é uma representação alternativa da estrutura correspondente.

As correspondências entre os componentes da *estrutura* e do *gráfico* estão definidas a seguir:

- Um *lugar* geralmente é representado por um *círculo*
- Uma *transição* geralmente é representada por uma *barra*
- As *entradas* são representadas por arcos orientados no sentido lugares/transições
- As *saídas* são representadas por arcos orientados no sentido transições/lugares
- Os lugares são ligados às transições e vice-versa.

Os gráficos de Redes de Petri são multigrafos pois podem existir diversos arcos (entradas/saídas), interligando os círculos (lugares) e as barras (transições).

Em geral, a representação gráfica é a mais utilizada para ilustrar os conceitos da *Teoria de Redes de Petri*.

4.4. MARCAS DE UMA REDE DE PETRI

Uma *marca* é um conceito primitivo de Redes de Petri, assim como *lugares* e *transições*.

As marcas são representadas por pontos e são colocadas nos lugares de uma Rede de Petri, através de uma atividade denominada *Marcação de uma Rede de Petri*.

A *Marcação de uma Rede de Petri* pode ser definida como um vetor M(P) que determina para cada *lugar* (P_i) da Rede de Petri, o número de marcas correspondente.

O número e posição das marcas podem sofrer alterações durante a execução de uma Rede de Petri.

A notação M = (C,M) significa que uma *Estrutura de Rede de Petri* C = (P,T,I,O) possui uma marcação M. Esta notação pode ser às vezes substituída por C = (P,T,I,O,M).

4.5. REGRAS DE EXECUÇÃO DE REDES DE PETRI

A *Execução de uma Rede de Petri* é controlada pelo número e distribuição de suas marcas.

Uma Rede de Petri é executada através do *disparo* de transições. Uma transição dispara, removendo marcas dos lugares de entrada da transição e criando novas marcas que são distribuídas nos lugares de saída da transição.

O disparo de uma transição representa uma mudança no estado da Rede de Petri, passando de uma marcação M para M'.

A distribuição de uma marca é feita através dos arcos de saída de uma transição.

Uma transição só pode ser disparada se estiver *habilitada*.

Uma transição está *habilitada* se cada um de seus lugares de entrada I(tj), tiver no mínimo um número de marcas correspondente ao número de arcos que interligam estes lugares de entrada à transição.

As marcas dos lugares de entrada que habilitam uma transição são suas marcas de habilitação.

A execução de uma Rede de Petri pode ser representada por duas sequências:

- Sequência de marcas (M,M',M'',...)

- Sequência de transições disparadas (t_j0, t_j1, t_j2,...)

O relacionamento entre estas duas sequências fornece uma representação da execução da Rede de Petri.

Os disparos de uma transição podem continuar, enquanto houver pelo menos uma transição habilitada. Quando não houver mais transições habilitadas a execução pára.

4.6. REDES DE PETRI TEMPORIZADAS

A seguir apresenta-se um resumo de três modelos de Redes de Petri temporizadas, apresentadas por Ramchandani, Merlin e Sikafis [MENA 85].

Ramchandani propôs um modelo denominado *"Timed Petri Nets"* (TdPNs) no qual um tempo de disparo é associado a cada transição do modelo clássico. O tempo neste caso é um número real e não negativo associado a cada transição.

Neste modelo a rede passa a ter dois tipos de estados:

- Estado ativo:
  Estado onde existem transições habilitadas, porém incompletas, pois o tempo de disparo não foi concluído;

- Estado inativo:
  Estado onde não existem transições ativas.

Merlin propôs um modelo denominado *"Time Petri Nets"* (TPNs), no qual um intervalo de tempo de disparo é associado a cada transição do modelo clássico. O tempo neste caso é um intervalo de tempo fechado $[a_i, b_i]$ associado a cada transição.

As transições são habilitadas da mesma forma que nas Redes de Petri clássicas, porém o *disparo* ocorre apenas dentro dos limites de tempo definido pelo intervalo de tempo associado à transição.

Sikafis propôs um modelo denominado *"Timed Place-Transition Nets"* (TdPTNs), no qual um *tempo de espera* é associado a cada lugar.

Neste modelo a rede passa a possuir dois tipos de marcação:

- Marcação viável:
  Marcação de um lugar cujo tempo de espera esteja esgotado;

- Marcação não viável:
  Marcação de um lugar cujo tempo de espera não esteja esgotado.

Transições neste caso são habilitadas da mesma forma que em Redes de Petri clássicas, porém o disparo de uma transição só ocorre quando a marcação de seus lugares de entrada for viável.

4.7. SUBCLASSES DE REDES DE PETRI

As subclasses de Redes de Petri são variações de Redes de Petri utilizadas para modelar sistemas com características específicas.

A seguir são descritos os dois tipos mais utilizados de subclasses de Redes de Petri:

- Máquinas de Estado

Uma máquina de estado é uma Rede de Petri na qual cada transição é restrita a ter apenas uma entrada e uma saída.

Entre as propriedades de uma máquina de estado, pode-se destacar:

- As máquinas de estado são estritamente conservativas;
- A árvore de alcançabilidade de uma máquina de estados é finita;
- As questões de análise para máquinas de estado são passíveis de solução;
- As máquinas de estado são apropriadas para modelar conflito.

- Grafos Marcados

Um grafo marcado é uma Rede de Petri na qual cada lugar é restrito a ter apenas uma entrada e uma saída.

As propriedades que têm sido pesquisadas para grafos marcados são vivacidade, segurança e alcançabilidade. Estas propriedades em geral são estudadas através dos *ciclos* do grafo.

Um *ciclo* é um caminho fechado do grafo, onde a partir de uma transição há o retorno após uma sequência de disparos, à mesma transição.

Os grafos marcados são apropriados para modelar concorrência.

4.8. MODELAMENTO DE UM SISTEMA DE SOFTWARE COM CARACTERÍSTICAS TEMPO REAL UTILIZANDO REDES DE PETRI

Os capítulos anteriores apresentaram duas Interfaces cujos objetivos eram fazer a descrição de um sistema de software e de suas características dinâmicas, dentro do enfoque de desenvolvimento de Software com características Tempo Real.

O presente item tem por objetivo associar os elementos descritos com uma estrutura de Rede de Petri de forma a permitir o modelamento do sistema, bem como simular algumas de suas características dinâmicas.

Neste contexto temos:

- Os elementos *Ações*, *Condições* e *Portas* são modelados como lugares;

- O elemento operacional *Evento Interno* é modelado como transição;

- O elemento operacional *Evento Externo* é modelado como transição fonte;

- Os *Conflitos* da rede decorrentes dos elementos condição (comandos *Desvio Condicional* e *Desvio Múltiplo*) são resolvidos através de operadores e variáveis lógicas e algébricas.

- As *Sincronizações* de duas ou mais ações são resolvidas também através de operadores e variáveis lógicas e algébricas.

- O *estado ativo* de um elemento (ações, condições, portas) é representado por um lugar com uma marca e com o tempo de ativação esgotado.

- O *estado parado* de um elemento (ações, condições, portas) é representado por um lugar sem marca;

- O *estado habilitado* de um elemento (ações, condições, portas) é representado por um lugar com uma marca e com o tempo de ativação não esgotado.

O modelo proposto não atende a todos os requisitos de um *Sistema de Software com Características Tempo Real*, tratando-se de um ensaio preliminar sobre a potencialidade de se modelar através de Redes de Petri algumas das características dinâmicas típicas destes sistemas, a nível de especificação funcional.

CAPITULO V

CONVERSÃO AUTOMÁTICA DOS COMANDOS DESCRITOS PARA PRIMITIVAS DE REDES DE PETRI (GRP)

5.1. INTRODUÇÃO

O presente capitulo tem por objetivo descrever a conversão automática de um Sistema de Software com Características Tempo Real descrito através das Interfaces de Especificação (LES) e Descrição da Dinâmica (LDS) (apresentadas nos Capítulos 2 e 3) em modelos de Rede de Petri.

Esta conversão resulta em duas Redes de Petri, uma modelando o Fluxo de Controle e outra, sobreposta à primeira, modelando a Comunicação entre os elementos do sistema.

Os eventos externos que definem as Ativações Cíclicas, Ativação por Interrupção e Atraso na Ativação são modelados como transições fonte.

As situações que envolvem conflito ou exclusão mútua são modelados utilizando-se variáveis lógicas/inteiras associadas a operadores relacionais.

5.2. CONVERSÃO DOS COMANDOS DO MÓDULO FLUXO DE CONTROLE EM PRIMITIVAS DE REDES DE PETRI (PETMFC)

5.2.1. Introdução

Os itens a seguir apresentam a conversão dos comandos descritos no item 2.5, referentes ao Fluxo de Controle em *primitivas* de Redes de Petri [BRES 85].

A representação de um modelo de Rede de Petri (RP) do Fluxo de Controle de uma ação será o resultado da interligação do conjunto de primitivas que compõem esta ação.

5.2.2. Primitiva em Rede de Petri do Comando Sequencial

A conversão deste comando em RP resulta no encadeamento de lugares modelando as *ações*, intercaladas por transições representando os *eventos internos* que ativam estas *ações*.

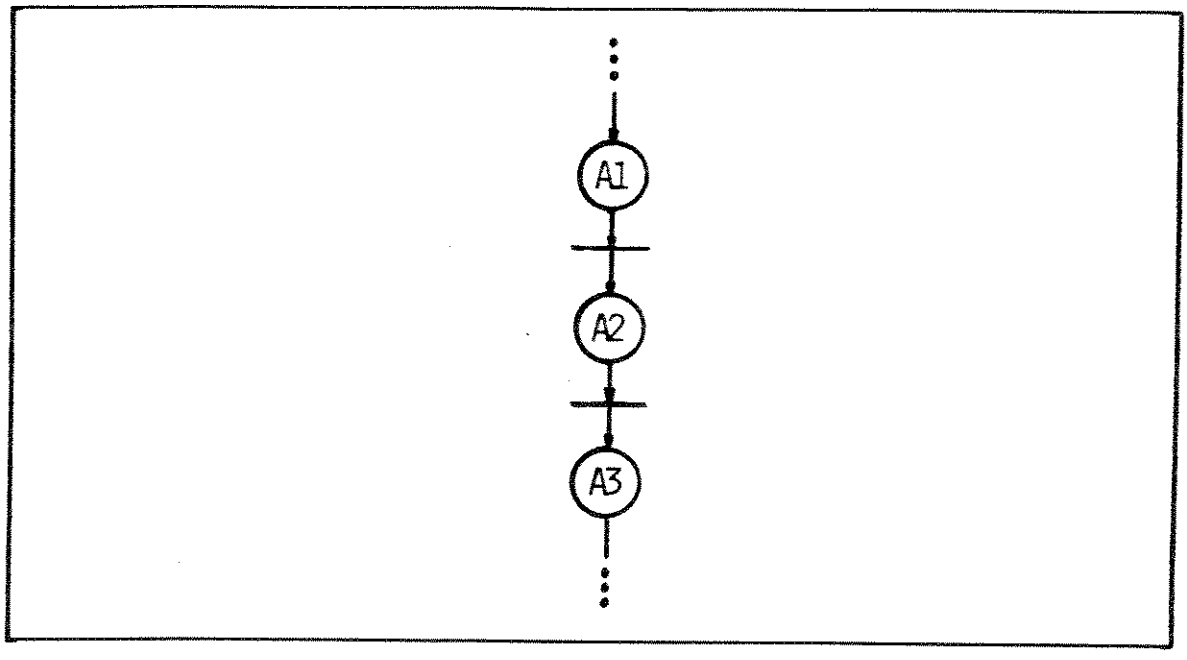


Figura 5.2.2 - Comando Sequencial
Primitiva em RP

5.2.3. Primitiva em Rede de Petri do Comando Cíclico (WHILE / REPEAT UNTIL)

A conversão deste comando em RP resulta em um lugar representando o elemento *Condição*, possuindo duas transições de saída.

Dependendo do valor assumido pela *Condição*, haverá uma decisão sobre qual das transições será disparada.

Esta decisão consiste em resolver um conflito da rede e será feito através do módulo de interpretação utilizando variáveis lógicas e/ou algébricas.

As transições de saída representam *Eventos Internos* que por sua vez irão ativar *Ações* distintas, uma dando sequência ao Fluxo de Controle e outra efetuando uma volta à *Condição Inicial* caracterizando um ciclo repetitivo.

O modelo da Rede de Petri neste caso exige uma transição interna para completar as especificações do Fluxo de Controle.

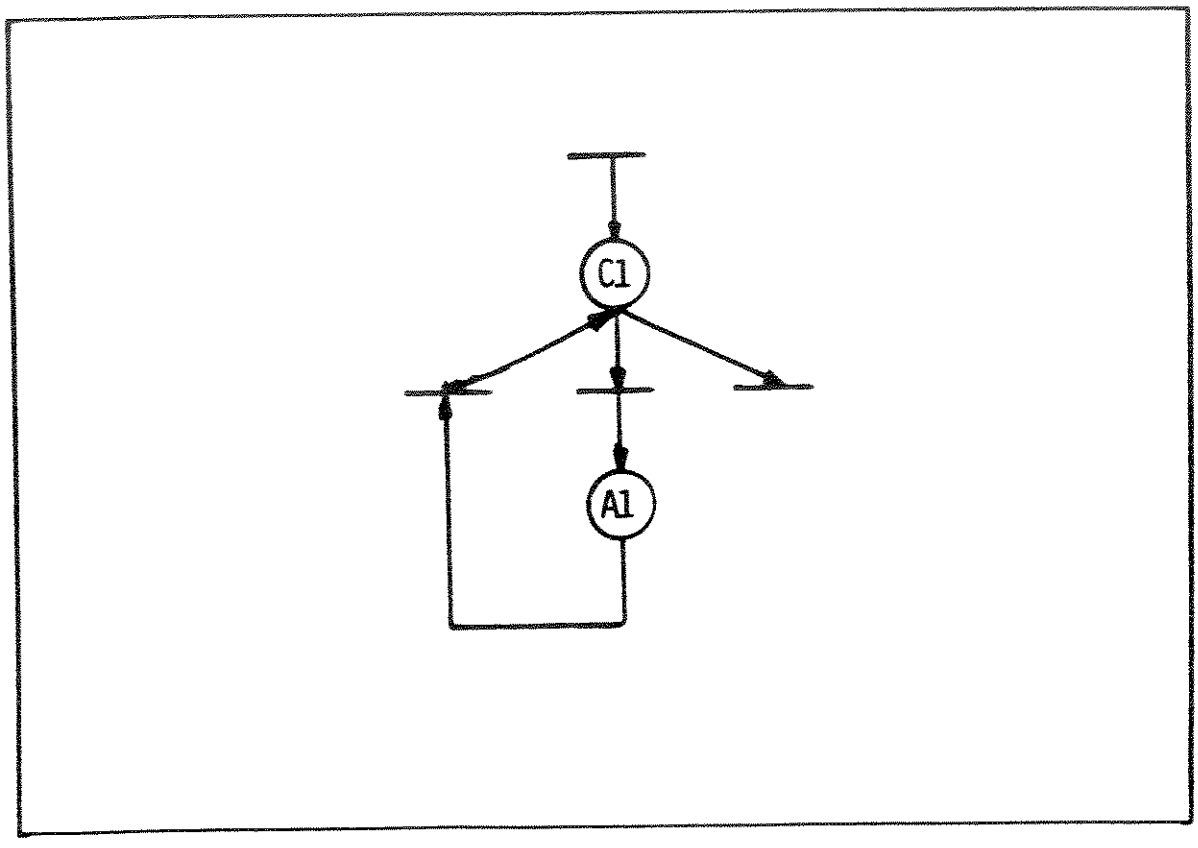


Figura 5.2.3 - Comando Cíclico
Primitiva em RP

5.2.4. Primitiva em Rede de Petri do Comando Desvio Condicional (IF-THEN-ELSE)

A conversão deste comando em RP resulta em um lugar representando um elemento *Condição*, possuindo duas transições de saída. Dependendo do valor assumido pela *Condição*, haverá uma decisão sobre qual das transições será disparada.

As transições de saída representam *Eventos Internos* que por sua vez irão ativar *Ações* distintas.

O modelo da RP neste caso exige três entidades internas (duas transições e um lugar) para completar as especificações do Fluxo de Controle.

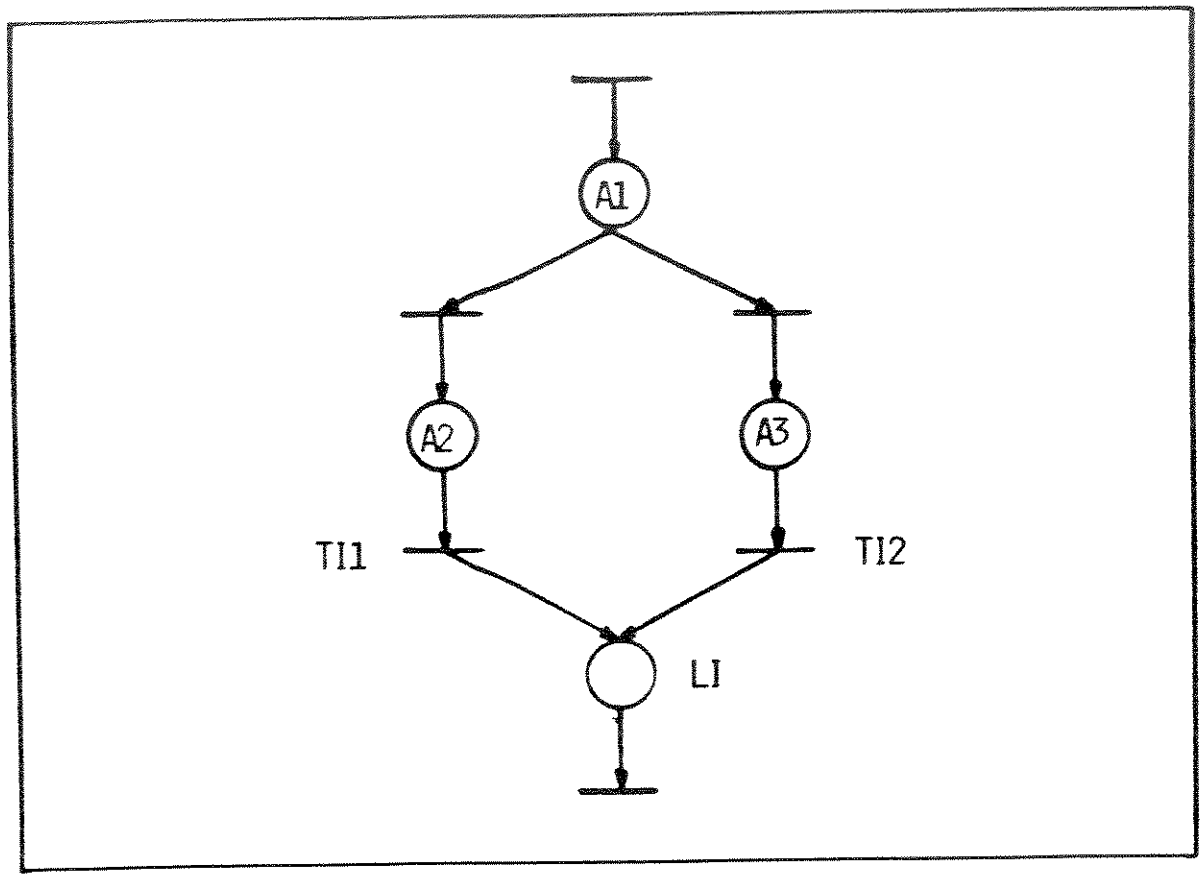


Figura 5.2.4 - Comando Desvio Condicional
Primitiva em RP

5.2.5. Primitiva em Rede de Petri do Comando Desvio Múltiplo (CASE)

A conversão deste comando em RP resulta em um lugar representando o elemento *Condição*, possuindo n transições de saída. Dependendo do valor assumido pela *Condição*, haverá uma decisão sobre qual das transições será disparada.

Esta decisão consiste em resolver um conflito da rede que será feito através do módulo de interpretação utilizando variáveis lógicas e/ou algébricas.

As transições de saída representam *Eventos Internos* que por sua vez irão ativar *Ações* distintas.

O modelo da RP neste caso exige um conjunto de entidades internas (um lugar e N transições) para completar as especificações do Fluxo de Controle.

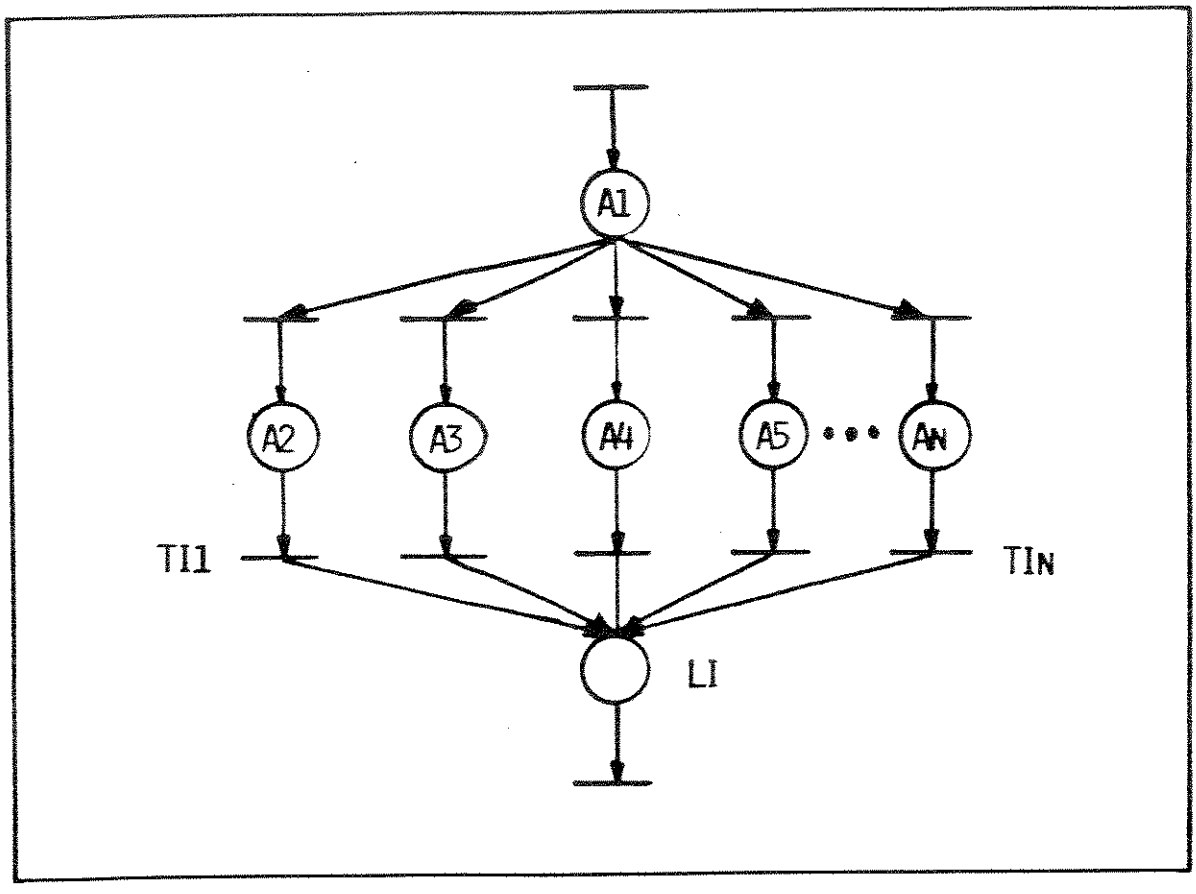


Figura 5.2.5 - Comando Desvio Múltiplo
Primitiva em RP

5.2.6. Primitiva em Rede de Petri do Comando Paralelo

A conversão deste comando em RP resulta em um conjunto de lugares representando ações, cuja característica é possuirem uma única transição de entrada e uma única transição de saída.

A transição de entrada representa um único *evento interno* que ativa simultaneamente todas as ações.

A transição de saída representa um único *evento interno* que para ser disparado depende do término de execução de todas as ações.

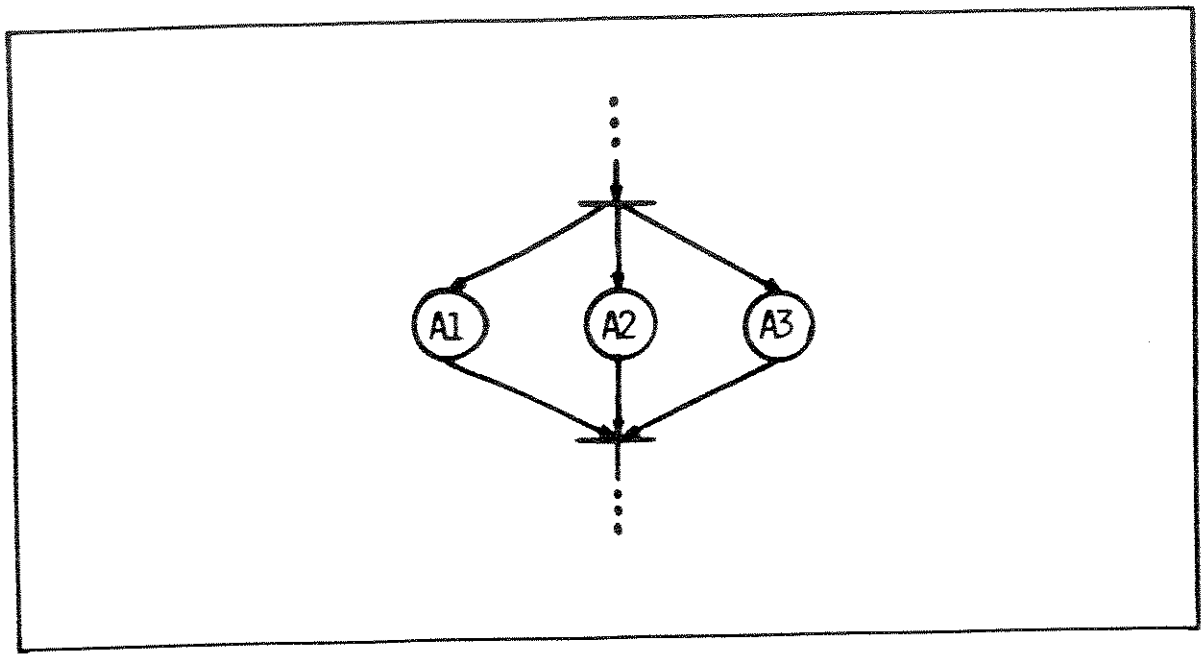


Figura 5.2.6 - Comando Paralelo
Primitiva em RP

5.3. CONVERSÃO DOS COMANDOS DO MÓDULO DE COMUNICAÇÃO EM PRIMITIVAS DE REDES DE PETRI (PETMCO)

5.3.1. Introdução

Os itens a seguir apresentam a conversão dos comandos descritos no item 2.6, referentes a Comunicação de ações em *primitivas* de Redes de Petri [BRES 85].

A representação de um modelo de RP de diversas ações interagindo entre si, através de trocas de mensagens, será o resultado da superposição da Rede de Petri correspondente ao Fluxo de Controle acrescida da Rede de Petri correspondente à Comunicação cujas primitivas estão descritas a seguir.

5.3.2. Primitiva em Rede de Petri do Comando Envia

A conversão deste comando em RP resulta em um lugar representando uma *Porta de Saída*, por onde serão enviadas *Mensagens* a outras RP representando outras *Ações*.

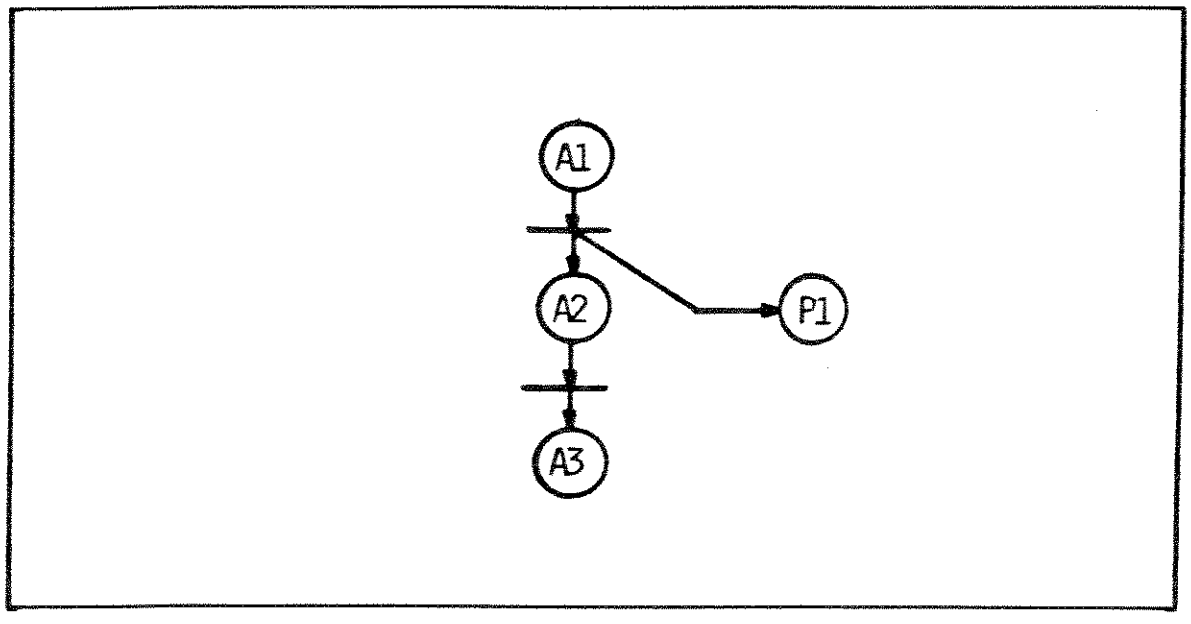


Figura 5.3.2 - Comando Envia
Primitiva em RP

5.3.3. Primitiva em Rede de Petri do Comando Envia com Resposta

A conversão deste comando em RP resulta em um lugar representando uma *Porta de Saída* por onde serão enviadas as *Mensagens* a outras redes representando outras *Ações*.

O fluxo da rede fica bloqueado até que o lugar representando uma *Porta de Entrada* receba a confirmação de envio da *Mensagem*.

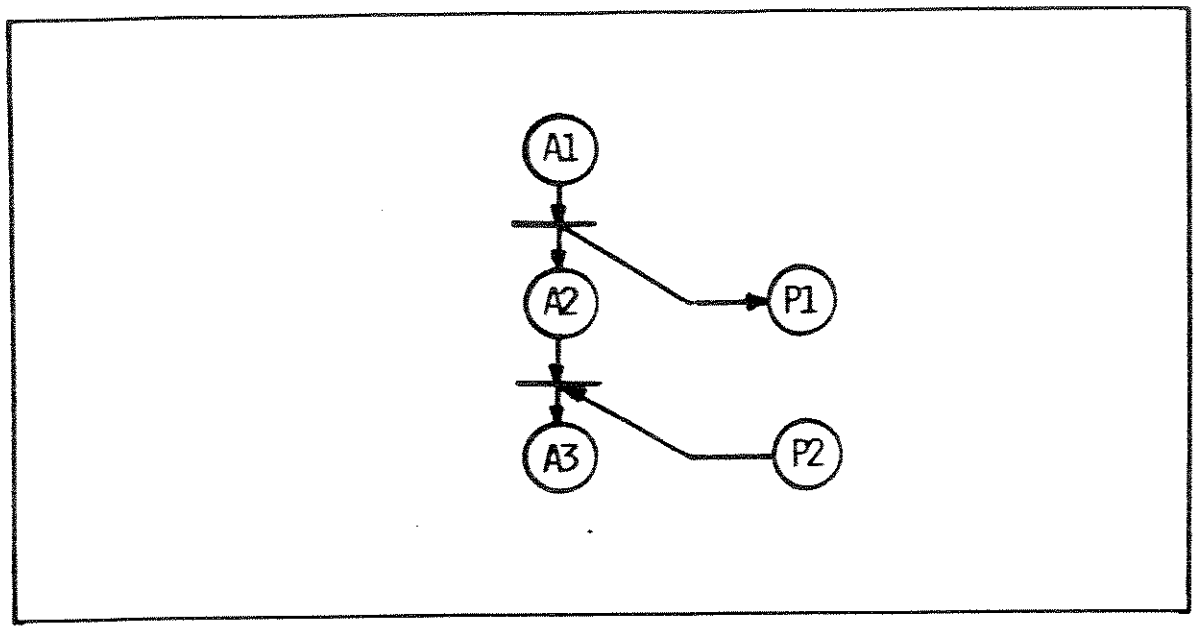


Figura 5.3.3 - Comando Envia com Resposta
Primitiva em RP

5.3.4. Primitiva em Rede de Petri do Comando Recebe

A conversão deste comando em RP resulta em diversos lugares representando *Portas de Entrada* seletivas por onde será estabelecida a comunicação com outras redes representando outras *Ações*.

O primeiro lugar a receber uma *Mensagem* bloqueia o recebimento das mensagens dos outros lugares até que a mensagem recebida seja utilizada pela ação destino.

O modelo da RP neste caso exige um conjunto de entidades internas (dois lugares e N transições) para completar as especificações do Fluxo de Controle.

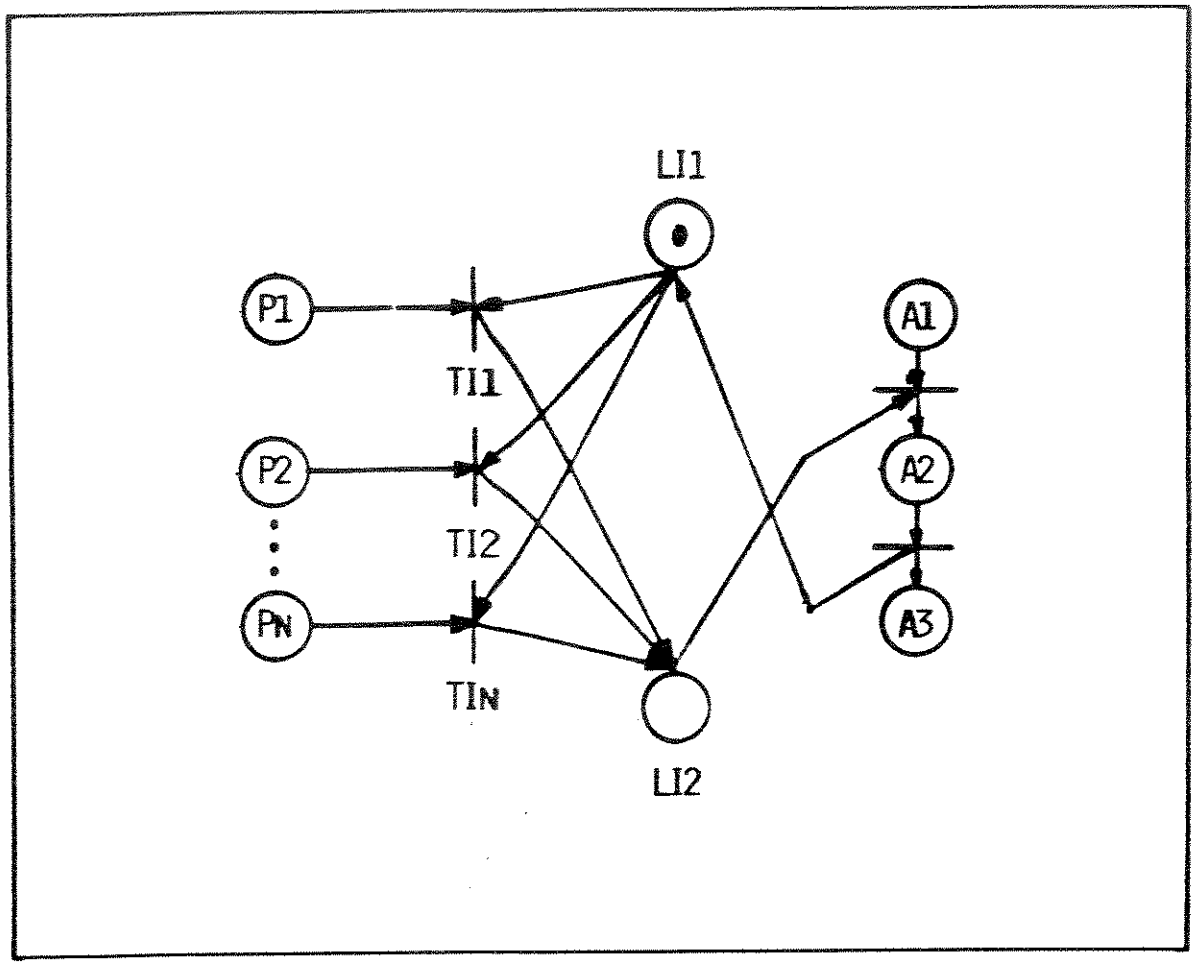


Figura 5.3.4 - Comando Recebe
Primitiva em RP

5.3.5. Primitiva em Rede de Petri do Comando Recebe com Resposta

Basicamente a conversão deste comando é igual à conversão do comando anterior, com a adição de um lugar que representa uma *Porta de Saída*, por onde serão enviadas as *Mensagens* às *Ações* de origem, confirmando o recebimento da *Mensagem*.

O modelo de RP neste caso exige um conjunto de entidades internas (dois lugares e N transições) para completar as especificações do Fluxo de Controle.

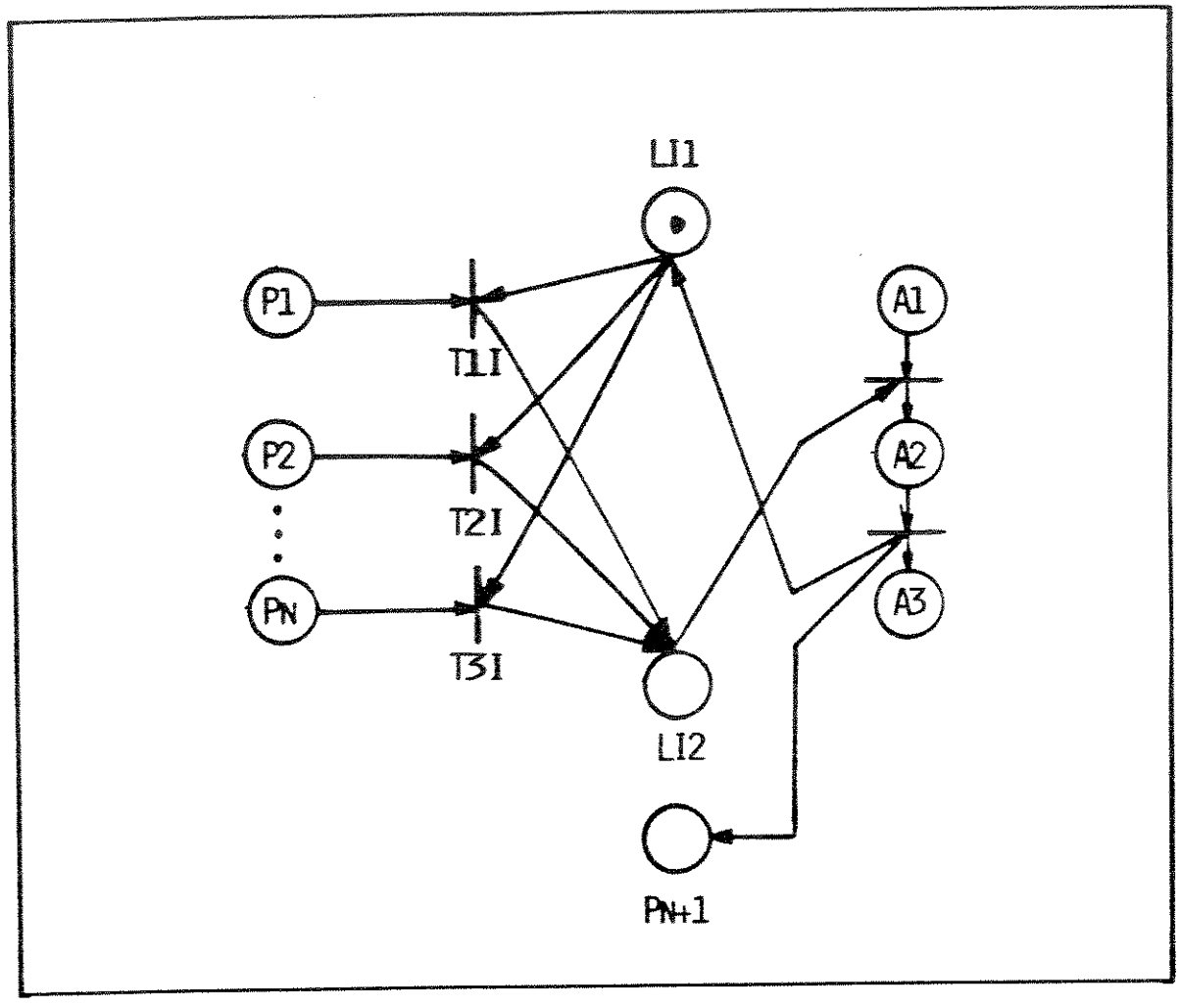


Figura 5.3.5 - Comando Recebe com Resposta
Primitiva em RP

5.4. CONVERSÃO DOS COMANDOS DO MÓDULO DE ENLACE EM PRIMITIVAS DE REDES DE PETRI

5.4.1. Introdução

Os itens a seguir apresentam a conversão dos comandos descritos no item 2.7, referentes ao enlace de ações.

A representação em RP de um módulo será o resultado da superposição das transições correspondente ao *enlace* módulo (Item 5.4.3) interligando os conjuntos de RP correspondentes ao Fluxo de Controle e Comunicação das ações do módulo [BRES 85].

A representação em RP de todo o sistema é uma extensão da representação em RP de cada módulo.

5.4.2. Primitiva em Rede de Petri do Enlace Sistema

A conversão deste comando em Rede de Petri resulta em um *evento interno* (transição de enlace) cuja entrada é a *porta de saída* (lugar) da *ação* que *envia* a mensagem, localizada em um primeiro *módulo* e a saída é a porta de entrada (lugar) da ação que *recebe* a mensagem localizada em um segundo módulo.

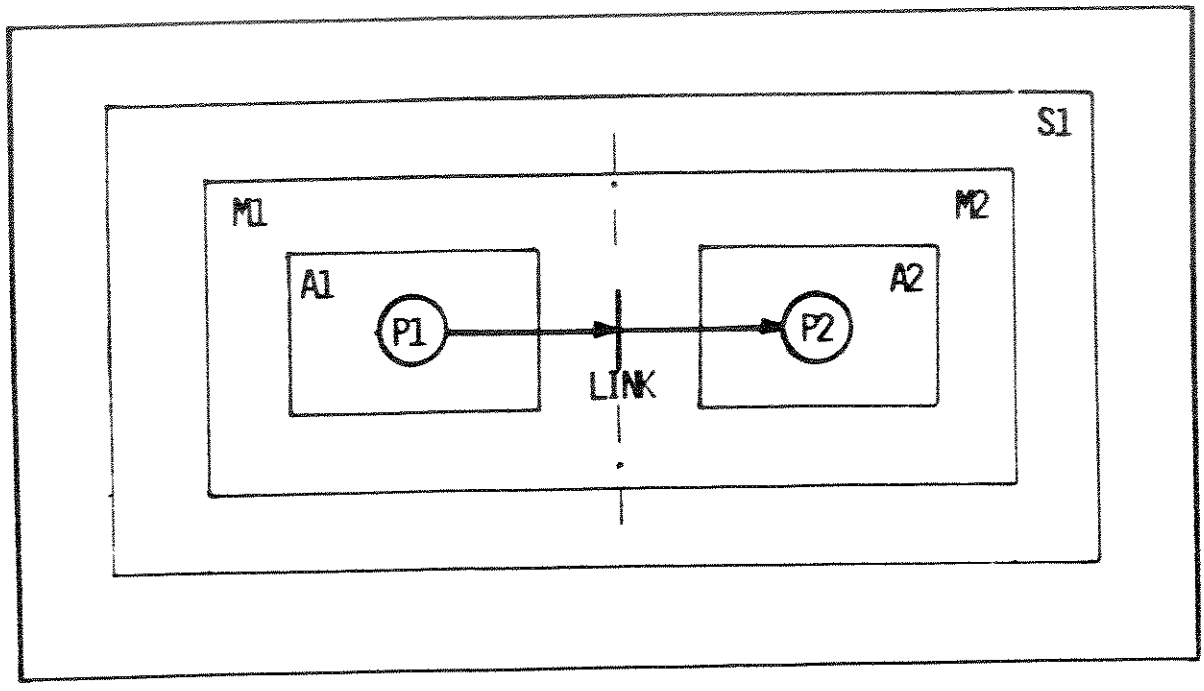


Figura 5.4.2 - Comando Enlace Sistema
Primitiva em RP

5.4.3. Primitiva em Rede de Petri do Enlace Módulo

A conversão deste comando em RP resulta em um *evento interno* (transição de enlace) cuja entrada é a *porta de saída* (lugar) da ação que *envia* a mensagem e a saída é a *porta de entrada* (lugar) da ação que *recebe* a mensagem, estando as duas ações localizadas no mesmo módulo.

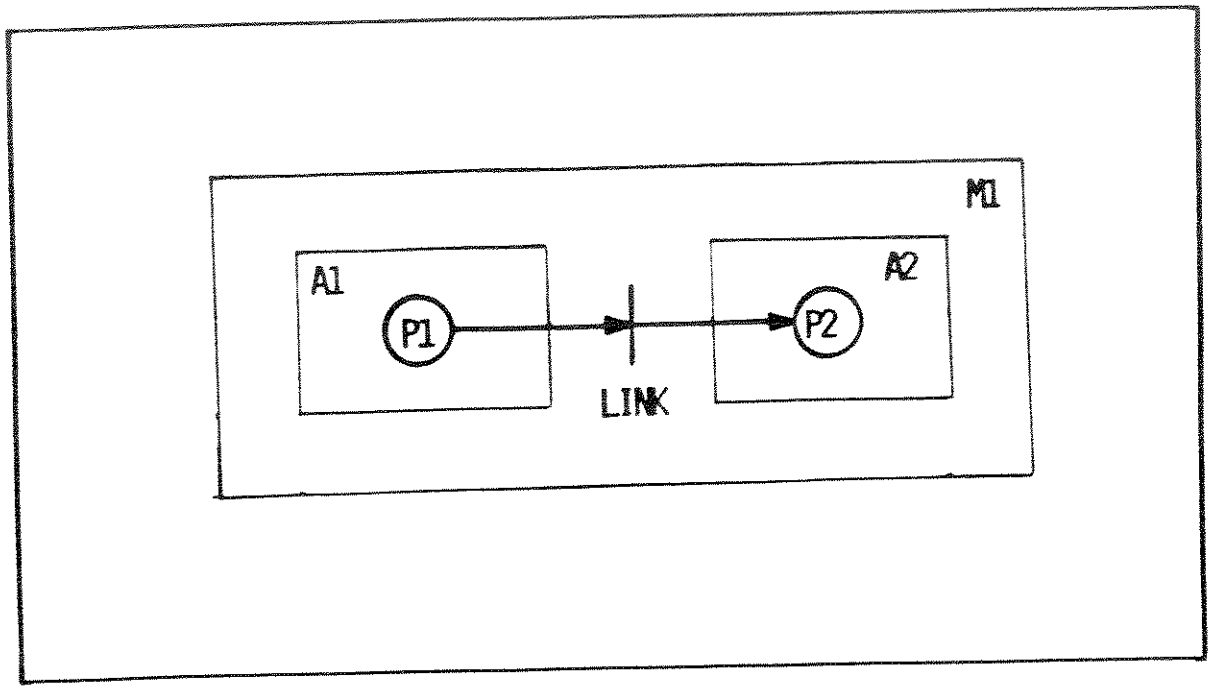


Figura 5.4.3 - Comando Enlace Módulo
Primitiva em RP

5.5. CONVERSÃO DOS COMANDOS DO MÓDULO DE ESPECIFICAÇÃO DA DINÂMICA (MED) EM PRIMITIVAS DE REDES DE PETRI

5.5.1. Introdução

O item a seguir apresenta a conversão para RP dos comandos descritos no item 3.2, referente a especificação da dinâmica do sistema.

5.5.2. Primitivas em Rede de Petri dos Comandos Ativação Cíclica, Ativação por Interrupção e Atraso de Ativação

A conversão destes comandos em RP resulta em uma transição fonte que não possui lugar de entrada, e o lugar de saída corresponde à ação que se pretende ativar através de um *evento externo*.

5.6. CONVERSÃO DOS COMANDOS DO MÓDULO DE INTERPRETAÇÃO DA DINAMICA EM VARIÁVEIS LÓGICAS/ALGÉBRICAS E OPERADORES RELACIONAIS

5.6.1. Introdução

Os itens a seguir apresentam a conversão dos comandos descritos no item 3.3 em variáveis lógicas/algébricas e operadores relacionais.

5.6.2. Conversão para RP do Comando Solução de Conflito

A conversão para RP deste comando ocorre em todos os comandos que possuam um elemento *Condição*, com a adição de expressões lógicas/algébricas *associadas* às *Transições de Saída* do lugar que representa o elemento *Condição*.

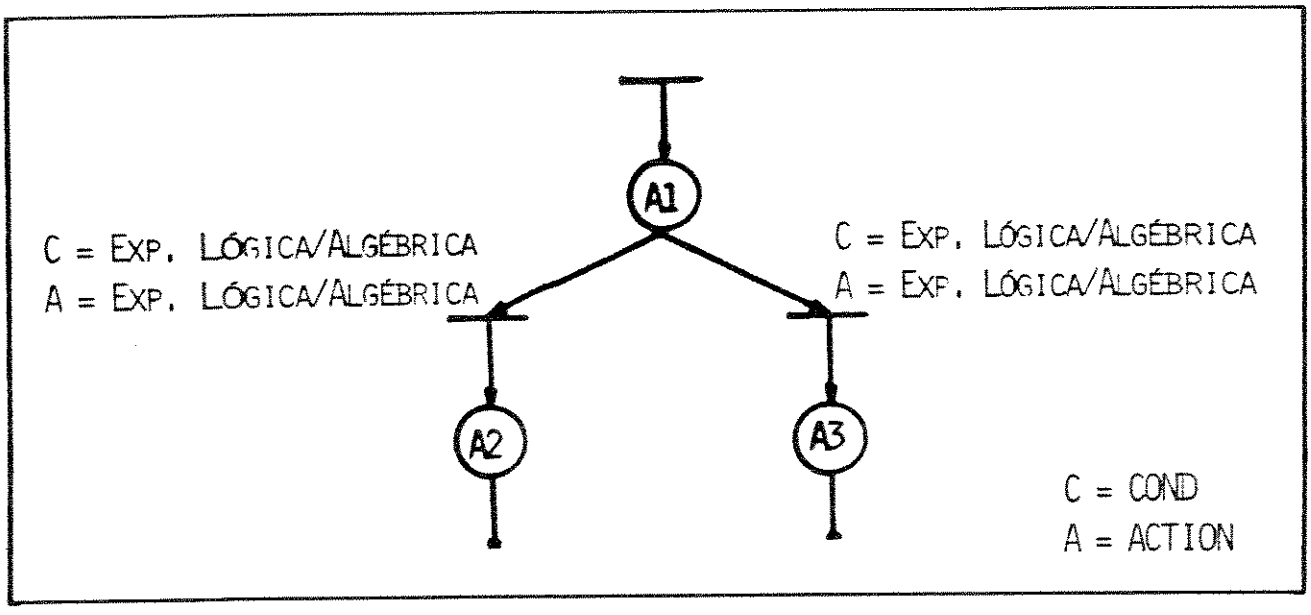


Figura 5.6.2 - Comando Solução de Conflito
Conversão para Rede de Petri

5.6.3. Conversão para RP do Comando Sincronização de Ações

A conversão para RP deste comando resulta em associar expressões lógicas/algébricas às *Transições de Enlace* que ligam duas *Portas de Saída* de duas ações distintas que enviam mensagens a uma única *Porta de Entrada* de uma ação que recebe a mensagem.

A função destas expressões lógicas/algébricas é a sincronização (exclusão mútua) na troca de mensagens.

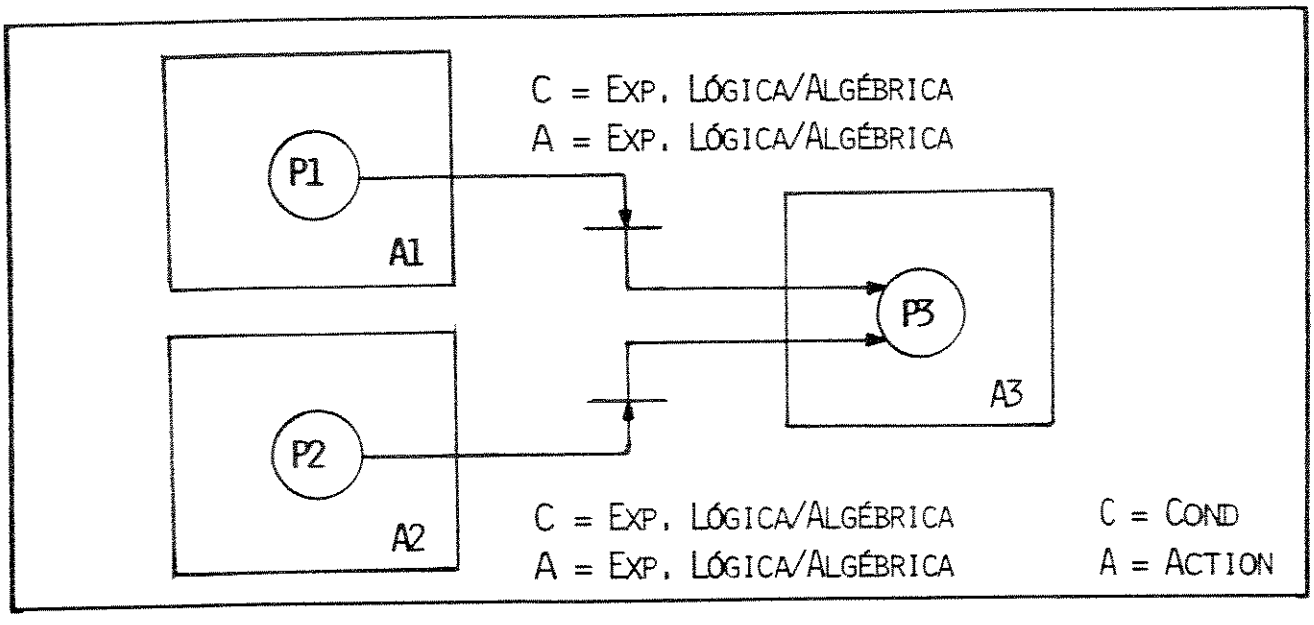


Figura 5.6.3 - Comando Sincronização de Ações
Conversão para Rede de Petri

CAPITULO VI

ANÁLISE DE REDES DE PETRI

6.1. INTRODUÇÃO

O presente capítulo tem por objetivo apresentar um estudo sobre as características ou propriedades de uma Rede de Petri.

Em seguida são apresentados três diferentes métodos de análise de uma Rede de Petri baseados em suas propriedades.

Dentre os métodos apresentados, destaca-se o método de cálculo de invariantes com a utilização de um *Programa de Análise pelo Método de Invariantes* (INVARIAN) desenvolvido no âmbito de uma tese de doutoramento da USP/SP [BRES 85].

6.2. PROPRIEDADES DE UMA REDE DE PETRI

Entre as características de propriedades de uma Rede de Petri, podemos destacar as seguintes [PETE 81], [PETE 77]:

- Segurança;
- Limitação;
- Conservação;
- Vivacidade;
- Alcançabilidade;
- Cobertura.

6.2.1. Segurança

Uma Rede de Petri é segura se todos os seus lugares forem seguros.

Um lugar em uma Rede de Petri é seguro se o seu número de marcas nunca for superior a um.

As características acima podem ser garantidas através das seguintes restrições:

- Uma transição não pode disparar, a menos que todos os seus lugares de saída estejam vazios;
- Uma transição pode possuir múltiplos arcos de entrada e múltiplos arcos de saída, porém cada lugar deverá possuir um único arco de entrada e um único arco de saída.

6.2.2. Limitação

Uma Rede de Petri é k-limitada ou k-segura, se cada um dos lugares desta rede for k-limitado ou k-seguro.

Um lugar em uma Rede de Petri é k-limitado ou k-seguro, se o número de marcas neste lugar não exceder um inteiro k.

Esta propriedade é um caso mais geral de Redes de Petri seguras.

6.2.3. Conservação

Uma Rede de Petri é conservativa, se o número de marcas da rede permanece constante.

Uma transição em geral não pode possuir diversos arcos de saída, com apenas um arco de entrada, pois isto implica em uma multiplicação de marcas.

Uma transição em geral não pode possuir diversos arcos de entrada, com apenas um arco de saída, pois isto implica em uma redução de marcas.

Uma Rede de Petri é estritamente conservativa se não possui multiplicação ou redução de suas marcas.

6.2.4. Vivacidade

O conceito de vivacidade de uma Rede de Petri abrange 4 níveis:

- Nível 0:
  Uma transição t_j nunca pode ser disparada

- Nível 1:
  Uma transição t_j pode ser disparada uma única vez

- Nível 2:
  Uma transição t_j pode ser disparada um número fixo de vezes

- Nível 3:
  Uma transição t_j pode ser disparada infinitas vezes

- Nível 4:
  Uma transição t_j é potencialmente disparável

Uma Rede de Petri atinge um estado de *impasse* (deadlock) se nenhuma de suas transições puder ser disparada neste estado.

Uma Rede de Petri está em um *impasse total*, se nenhuma de suas transições puder ser disparada.

A análise da *vivacidade* de uma Rede de Petri é a mais importante, pois através desta é possível definir as limitações do sistema estudado.

6.2.5. Alcançabilidade

Dada uma Rede de Petri com marcação inicial M, após o disparo de uma transição, a marcação (estado) da rede passa a ser M', que será portanto um estado *alcançável* da rede.

O conjunto *alcançável* de uma Rede de Petri é o conjunto de todas as marcações (estados) que uma Rede de Petri pode possuir, a partir de uma sequência de disparos.

6.2.6. Cobertura

Cobertura é uma extensão de alcançabilidade e consiste em verificar se existe um estado M'' *alcançável* a partir de um estado M'. Neste caso, o estado M'' será uma cobertura de M'.

6.3. MÉTODOS DE ANÁLISE DE REDES DE PETRI

As propriedades acima descrita podem ser analisadas através de diferentes métodos, com ênfase às propriedades de *vivacidade*, a seguir:

- Árvore de alcançabilidade;
- Método algébrico;
- Cálculo de invariantes.

6.3.1. Árvore de Alcançabilidade

A *Árvore de Alcançabilidade* apresenta as sucessivas marcações (estados) de uma Rede de Petri, resultante do disparo de suas transições.

- Os nós da árvore representam a marcação (estado) da Rede de Petri;
- Os arcos da árvore representam as transições da rede.

A árvore de estado pode ser infinita, dependendo do fato de suas transições poderem disparar infinitas vezes.

Para ser utilizada como ferramenta de análise, uma árvore de alcançabilidade infinita deve ser reduzida, utilizando os seguintes conceitos:

- Nós terminais:
  Representam um estado da Rede de Petri onde nenhuma transição pode ser disparada (marcação morta);

- Nós duplicados:
  Correspondem a um estado da Rede de Petri já ocorrido anteriormente, não justificando portanto a existência de estados decorrentes deste.

Outra forma de redução de uma Rede de Petri é através da utilização de um símbolo especial w que representa um número de marcas muito grande.

Através do exame de sua árvore de alcançabilidade, em sua representação finita, algumas das propriedades de uma Rede de Petri poderão ser analisadas.

A seguir tem-se um resumo das potencialidades e limitações de uma análise efetuada através da utilização de árvore de alcançabilidade [BRES 85]:

- Permite localizar as transições *mortas* e as transições *potencialmente disparáveis* de uma Rede de Petri;

- Permite localizar *algumas situações de impasse* da rede caracterizadas pelos nós terminais da árvore de alcançabilidade;

- Permite localizar transições *vivas de nível* 2 através da detecção de ocorrências de transições em caminhos da árvore de alcançabilidade, que apresentem duas ocorrências de uma mesma marcação M' sendo que as transições vivas em nível 2 estarão entre essas duas ocorrências;

- Não permite determinar transições *vivas em nível* 3 *ou* 4;

- Não permite determinar todos os possíveis impasses envolvendo todas as transições da Rede de Petri;

- Não permite determinar *impasses* envolvendo *subconjuntos de transições* diferentes de T.

6.3.2. ANÁLISE ALGÉBRICA

Esta técnica de análise consiste em definir duas matrizes Ci e Co para representar as funções de entrada e saída da rede.

- As linhas da matriz (n) correspondem às *transições* da Rede de Petri;

- As colunas da matriz (m) correspondem aos *lugares* da Rede de Petri.

A maior vantagem da utilização desta técnica é que a análise é feita utilizando teoria de matrizes, que é uma área da álgebra linear bastante explorada.

A seguir é feita uma descrição algébrica deste método, tomando como referência uma Rede de Petri representada por uma matriz de ordem nxm.

- Matriz de Entrada

$$Ci = \begin{bmatrix} Ci_{11} & Ci_{12} & Ci_{13} & \dots & Ci_{1m} \\ Ci_{21} & Ci_{22} & Ci_{23} & \dots & Ci_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ Ci_{n1} & Ci_{n2} & Ci_{n3} & \dots & Ci_{nm} \end{bmatrix}$$

C_i = número de arcos de entrada.

- Matriz de Saída

$$Co = \begin{bmatrix} Co_{11} & Co_{12} & Co_{13} & \dots & Co_{1m} \\ Co_{21} & Co_{22} & Co_{23} & \dots & Co_{2m} \\ \vdots & \vdots & \vdots & & \vdots \\ Co_{n1} & Co_{n2} & Co_{n3} & \dots & Co_{nm} \end{bmatrix}$$

C_o = número de arcos de saída.

- Matriz de Incidência Genérica

$$C = [Co] - [Ci]$$

$$C = \begin{bmatrix} (Co_{11}-Ci_{11}) & (Co_{12}-Ci_{12}) & (Co_{1m}-Ci_{1m}) \\ (Co_{21}-Ci_{21}) & (Co_{22}-Ci_{22}) & (Co_{2m}-Ci_{2m}) \\ \vdots & \vdots & \vdots \\ (Co_{n1}-Ci_{n1}) & (Co_{n2}-Ci_{n2}) & (Co_{nm}-Ci_{nm}) \end{bmatrix}$$

Neste caso o valor dos arcos de entrada e saída (Ci,Co) de uma transição será um número inteiro.

Dada uma marcação inicial M = [M1,M2,M3,...Mm] e um vetor de disparo t = [t1,t2,t3,...,tn], obtém-se a seguinte marcação:

$$M' = M + tC$$

$$M' = [M1, M2, M3 \ldots Mm] + [t1, t2, t3 \ldots tn] \times$$

$$\begin{bmatrix} (Co_{11}-Ci_{11}) & (Co_{12}-Ci_{12}) & (Co_{1m}-Ci_{1m}) \\ (Co_{21}-Ci_{21}) & (Co_{22}-Ci_{22}) & (Co_{2m}-Ci_{2m}) \\ \vdots & \vdots & \vdots \\ (Co_{n1}-Ci_{n1}) & (Co_{n2}-Ci_{n2}) & (Co_{nm}-Ci_{nm}) \end{bmatrix}$$

$$M' = [M1', M2', M3' \ldots Mm']$$

Seguindo a mesma linha de raciocínio, pode-se definir se uma dada marcação $M'' = (M1'', M2'', M3'' \ldots M4'')$ *é alcançável* a partir de M'.

$$M'' = M' + t \times C$$

$$[M1''+M2''+M3'' \ldots Mm''] = [M1', M2', M3' \ldots Mm'] + t \times$$

$$\begin{bmatrix} (Co_{11}-Ci_{11}) & (Co_{12}-Ci_{12}) & (Co_{1m}-Ci_{1m}) \\ (Co_{21}-Ci_{21}) & (Co_{22}-Ci_{22}) & (Co_{2m}-Ci_{2m}) \\ \vdots & \vdots & \vdots \\ (Co_{n1}-Ci_{n1}) & (Co_{n2}-Ci_{n2}) & (Co_{nm}-Ci_{nm}) \end{bmatrix}$$

onde $t = [t_1, t_2, \ldots t_n]$ define a sequência de disparos para atingir marcação M''.

Caso a equação acima não tivesse solução, a conclusão seria de que a marcação M" dada não é *alcançável* a partir de M'.

A análise de Redes de Petri utilizando método algébrico possui algumas restrições:

- O vetor de transições resultante define o número de vezes que cada transição dispara, porém não define a *ordem* de ocorrência dos disparos;

- O vetor de transições resultante é uma condição *necessária* para a definição da alcançabilidade, porém não é *suficiente* pois não verifica se as transições estão efetivamente habilitadas.

6.3.3. METODO DOS INVARIANTES

O método dos Invariantes consiste em calcular os impasses baseado nas propriedades algébricas de uma Rede de Petri [LAUT 74]. A seguir são apresentadas as definições básicas deste método:

Definição 1

- Matriz de Incidência para uma rede pura

$$C(p,t) = \begin{cases} -1 & \text{se } p \in Ci(t) \\ +1 & \text{se } p \in Co(t) \\ 0 & \text{outros casos} \end{cases}$$

Na rede pura os conjuntos Ci(t) e Co(t) são disjuntos.

Definição 2

Uma transição está *habilitada* para uma marcação M se todos os lugares de entrada desta transição estiverem marcados.

Definição 3

O disparo de uma transição ativa $t \in T$ é definida como uma transformação de uma marcação M em outra marcação M' onde:

$$M' = M + C(\_, t)$$

Isto significa que o disparo de uma transição ativa t, remove uma marca dos lugares de entrada e adiciona uma marca nos lugares de saída.

Definição 4

Dado duas marcações M e M', existe um sistema de equações lineares com um vetor t, tal que:

$$M' = M + tC$$

t = vetor de disparo de transições.;

Se M' for *alcançável* a partir de M, diz-se que o vetor é *realizável*.

O conjunto de marcações atingíveis a partir da marcação M, recebe a denominação $[M]$.

Definição 5

Uma marcação é chamada *viva* se:

$$\forall\ M' \in [M] : |[M']| > 1$$

Isto significa que cada marcação atingível a partir de uma marcação M, pode por sua vez ser transformada em outras marcações.

Esta definição corresponde à noção de sistema livre de impasse (deadlockfreeness).

Definição 6

Uma Rede de Petri será conservativa se o número de marcas no conjunto [M] for constante.

Seja Z um vetor que define uma marcação (vetor peso), logo

$$M\ Z = M'\ Z$$

Definição 7

Cálculo de Invariantes

Tomando como base as def. 6 e def. 4 tem-se:

$$M\ Z = M'\ Z$$

$$M\ Z = (M + tC)\ Z$$

$$M\ Z = M\ Z + tC\ Z$$

portanto

$$t\ C\ Z = 0$$

$t \neq 0$ pois é o vetor de disparo de transições, logo:

$$C\ Z = 0$$

As soluções do sistema de equações $C\ Z = 0$, são chamadas de *invariantes*, e por sua vez são vetores característicos de um conjunto de lugares se

$$\forall\ p \in P : Z(p) \in \{0,1\}$$

O cálculo dos invariantes portanto consiste em determinar as soluções de um sistema de equações lineares homogêneas do tipo $Ay = 0$.

6.4. APRESENTAÇÃO DO ANALISADOR INVARIAN

O objetivo deste item é apresentar um programa de cálculo de Invariantes elaborado no âmbito da tese de doutorado "*Ambiente de Programação Distribuída: Definição, Análise e Síntese*" da USP/SP [BRES 85].

O programa está dividido em dois subprogramas, sendo um para realizar o *cálculo dos invariantes* e outro para realizar o *cálculo de possíveis impasses* de uma Rede de Petri.

6.4.1. Programa de Cálculo dos Invariantes

O programa consiste em calcular os invariantes de uma Rede de Petri a partir do conjunto de definições apresentado no item anterior (6.3.3).

Este conjunto de definições estabelece que o conjunto de invariantes inteiras positivas é o conjunto de soluções linearmente independentes do sistema de equações homogêneas C Z = 0.

O cálculo destas soluções é feito através de métodos de álgebra linear.

6.4.2. Programa de Cálculo de Impasses

Este programa é dividido em duas partes, a primeira consiste em gerar um conjunto de marcações que garantam que uma determinada transição não dispare.

Em seguida verifica se estas marcações são alcançáveis.

Para todas as marcações alcançáveis, é feita uma análise se ela provoca o disparo de alguma transição, não consistindo portanto em um impasse.

A marcação que for alcançável e não provocar o disparo de nenhuma transição será um possível impasse.

6.4.3. Exemplo de Utilização do Analisador

A Figura 6.4.3.a a seguir ilustra uma Rede de Petri a ser analisada.

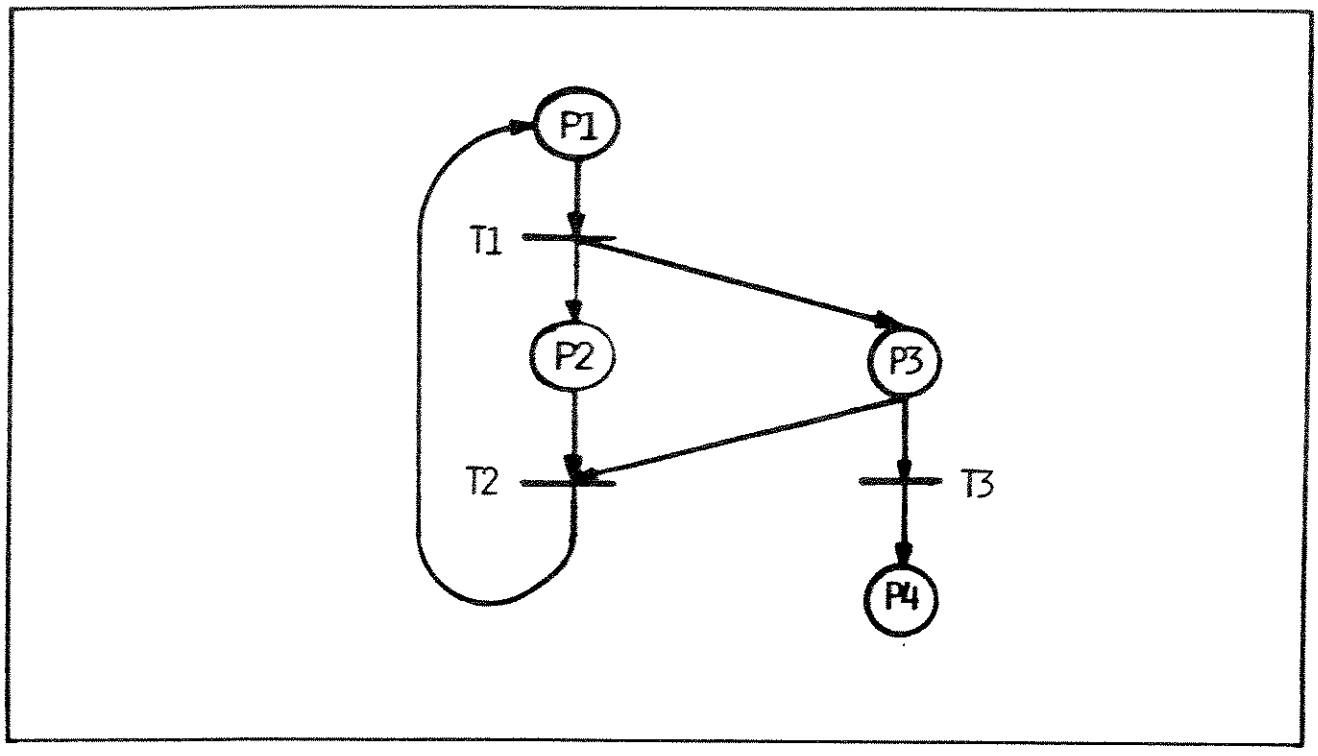


Figura 6.4.3.a - Rede de Petri
Exemplo de Cálculo de Invariantes

As matrizes da rede são:

$$Ci = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad Co = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$C = \begin{bmatrix} -1 & 1 & 1 & 0 \\ 1 & -1 & -1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

Cálculo dos Invariantes

A solução do sistema de equações homogêneas Cy = 0, através do método de escalonamento da matriz C, fornece os seguintes vetores linearmente independentes:

$$V1 = (1\ 1\ 0\ 0) \quad e \quad V2 = (1\ 0\ 1\ 1)$$

Os elementos destes vetores estão contidos no conjunto (0,1) satisfazendo portanto a condição de que sejam *invariantes* da Rede de Petri (definição 7).

$$Z = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{bmatrix}$$

Cálculo dos Possíveis Impasses

Considerando a matriz de entrada da Rede de Petri:

$$Ci = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

A condição que garante a transição t1 (linha 1) não dispare é de que $Ci_{11} = 0$.

Para $Ci_{11} = 0$ tem-se 2**(m-1) combinações possíveis de *vetores de marcação* [M], sendo m as colunas da matriz E.

$$[M] = \begin{matrix} (0\ 0\ 0\ 0), & (0\ 0\ 0\ 1), & (0\ 0\ 1\ 0), & (0\ 0\ 1\ 1) \\ (0\ 1\ 0\ 0), & (0\ 1\ 0\ 1), & (0\ 1\ 1\ 0), & (0\ 1\ 1\ 1) \end{matrix}$$

Em seguida é verificado se estas marcações são alcançáveis, ou seja, se Z.M = CTE (definição 6).

Efetuando os cálculos, verifica-se que ZM é constante para as marcações (0 1 0 1) e (0 1 1 0).

A marcação (0 1 1 0) provoca o disparo das transições t2 ou t3, logo não se trata de um impasse.

A marcação (0 1 0 1) não provoca o disparo de nenhuma transição, logo é um *possível impasse* da Rede de Petri.

Os cálculos acima são repetidos para Ci_{22}, Ci_{28} e Ci_{88}, para as quais também, a marcação (0 1 0 1) é a única que constitui impasse.

A execução do programa para este exemplo está listada na Figura 6.4.3.b a seguir:

```
ARQUIVO DE ENTRADA:  petri 1.res
3 4
1 1 0
1 1 0 2 3 0
2 2 3 0 1 0
3 3 0 4 0

ARQUIVO RESULTANTE:  petri 1.inv

DETERMINAÇÃO DE INVARIANTES E IMPASSES DA REDE
DE PETRI: petri1.res

MARCAÇÃO INICIAL: P 1=1

LUGARES DE ENTRADA E DE SAÍDA DAS TRANSIÇÕES
T 1 - ENTRADAS: P 1
      SAÍDAS  : P 2 P 3
T 2 - ENTRADAS: P 2 P 3
      SAÍDAS  : P 1
T 3 - ENTRADAS: P 3
      SAÍDAS  : P 4

INVARIANTES DA REDE DE PETRI:
I 1 : < P 1  P 2>
I 2 : < P 1  P 3  P 4 >

POSSÍVEIS IMPASSES DA REDE DE PETRI:
IMPASSE 1:
M[ 1] = 0 M[ 2] = 1   M[ 3] = 0  M[ 4] = 1
```

Figura 6.4.3.b - Execução do Programa de Análise de Redes de Petri

Comentário

O arquivo de entrada é descrito de forma numérica.

Os números da primeira linha representam respectivamente o número de lugares e o número de transições da Rede de Petri.

Os números da segunda linha representam a marcação inicial dos lugares.

Os lugares das linhas seguintes representam na primeira coluna as transições seguida dos lugares de entrada e saída.

O número 0 é utilizado como delimitador.

6.5. TRADUÇÃO REDE DE PETRI/ARQUIVO DE ENTRADA DO ANALISADOR DE REDES DE PETRI (ANA)

Esse módulo tem por objetivo traduzir uma Rede de Petri para um arquivo de dados de entrada do analisador de Redes de Petri descrito no item 6.4 [BRES 85].

Esta Rede de Petri representa um *modelo* de um Sistema de Software Tempo Real gerado a partir das Interfaces de Especificação e Descrição da Dinâmica de um Sistema de Software Tempo Real, e é *gerada automaticamente* segundo os critérios definidos no Capítulo 5.

O primeiro programa deste módulo cria uma base de dados específica para o analisador, traduzindo a base de dados de descrição da Rede de Petri.

O segundo programa cria um arquivo de entrada no padrão de um analisador de Redes de Petri descrito no item 6.4.

Os dois últimos programas permitem deletar a base de dados do analisador e imprimir uma Rede de Petri nos moldes do analisador.

CAPITULO VII

SIMULAÇÃO DE REDES DE PETRI

7.1. INTRODUÇÃO

O programa de análise pelo método dos invariantes apresentado no capítulo anterior tem como objetivo fazer um estudo das características e *comportamento estático* de uma Rede de Petri.

O presente capítulo tem por objetivo apresentar um programa de simulação desenvolvido no LAAS/Toulouse-França, cujo objetivo é fazer um estudo das características e *comportamento dinâmico* de uma Rede de Petri.

7.2. APRESENTAÇÃO DO SIMULADOR P.S.I.

O simulador PSI de Redes de Petri é constituído dos seguintes módulos [VALE 84a], [VALE 84b]:

- Módulo de Descrição da Rede de Petri (DESY/2)
- Módulo de Descrição das Condições de Simulação (DESAL)
- Módulo de Simulação (SIM).

7.2.1. Módulo de Descrição da Rede de Petri

O módulo de descrição da Rede de Petri é constituído de uma Linguagem de Descrição de Redes de Petri (DESY/2) e de um Programa de Análise Sintática e Semântica da Descrição (DES2).

A Linguagem de Descrição de Redes de Petri (DESY/2) é baseada na estrutura do Pascal e possui cinco instruções básicas:

- *Nós* - declarações de lugares e transições;

- *Arcos* - declaração dos lugares de entrada e saída das transições ou descrição dos caminhos orientados da rede;

- *Var* - declaração das variáveis booleanas ou inteiras utilizadas para a interpretação da rede;

- *Interpretação* - declaração de expressões lógicas e algébricas associadas às transições. Estas expressões são implementadas utilizando um conjunto de operadores relacionais associados a variáveis lógicas e algébricas.

- *Fim* - termina a descrição de uma rede interpretada.

O Programa de Análise Sintática e Semântica (DES2) compila os comandos e gera uma tabela que é uma estrutura de dados a ser utilizada pelo módulo de simulação.

7.2.2. Módulo de Descrição das Condições de Simulação

O módulo de descrição das condições de simulação é constituído de uma Linguagem de Descrição das Condições de Simulação (DESAL) e de um Programa de Análise Sintática e Semântica (DESE).

A linguagem de descrição das condições de simulação possui quatro instruções básicas:

- *Lugar* - agrupa a marcação inicial e uma temporização de cada um dos lugares da rede;

Lugar

Marcação P_{ii} = m (m = número marcas)

Tempos P_{ii} = i (t = tempo)

- *Inteiro* - inicialização das variáveis inteiras (DEFAULT = 0);

- *Booleana* - inicialização de variáveis booleanas (DEFAULT = FALSE);

- *Calendário* - responsável pela execução da simulação da rede, compreendendo os seguintes comandos:

  . *Estatística*: define o período de tempo em que se deseja efetuar a simulação;

  . *Observador*: define o intervalo de tempo em que se deseja obter estatísticas do estado do sistema;

  . *Ação*: define os instantes de disparo das transições fonte, bem como os instantes de chaveamento da lógica (verdadeiro/falso) das variáveis booleanas;

  . *Aleatório*: igual ao comando ação porém com tempos aleatórios (dentro de um intervalo definido).

O Programa de Análise Sintática e Semântica (DESE) compila os comandos e gera uma tabela que é uma estrutura de dados a ser utilizada pelo módulo de simulação.

7.2.3. Módulo de Simulação

O módulo de simulação é constituído por um programa (SIM) que executa as seguintes tarefas:

- Ler as tabelas de *Descrição* da Rede de Petri;
- Ler as tabelas de *Condições de Simulação* e o *Calendário Inicial*;
- Pesquisar o estado estável da Rede de Petri;
- Incrementar o tempo, executar o primeiro evento do calendário e atualizar o calendário;

- Efetuar as estatísticas:
  - Tempo de início da simulação;
  - Tempo de duração de simulação;
  - Número de disparos de cada transição;
  - Tempo associado a cada lugar;
  - Número máximo de marcas em cada lugar;
  - Número médio de marcas em cada lugar;
  - Número de vezes que um lugar foi marcado.

O núcleo do simulador é um *jogador de Rede de Petri* que coordena a execução da rede, através de uma cadeia de eventos ordenada por tempos crescentes.

Um *tempo de duração* é associado aos lugares de forma que quando uma marca é colocada em um lugar temporizado, ela permanece não-disponível durante este *tempo de duração* após o qual a marca passa a ser disponível.

Expressões booleanas e inteiras são associadas às transições permitindo estabelecer *condições* e *ações* em seus disparos. Estas expressões implicam na interpretação e permitem a resolução de conflitos no modelamento de Redes de Petri.

O simulador pode ser utilizado tanto em modo interativo como autônomo.

- No modo autônomo todos os comandos devem ser previstos no início da simulação e incluídos no calendário;

- No modo interativo, os comandos são introduzidos no calendário através do comando *observação*. A cada ponto de observação o programa de simulação passa o controle ao operador que através de um *menu* de comandos opera a continuação da execução da simulação.

7.2.4. Exemplo de Utilização do Simulador

A Figura 7.2.4a mostra uma Rede de Petri a ser simulada, utilizando temporização e expressões booleanas [PASS 86].

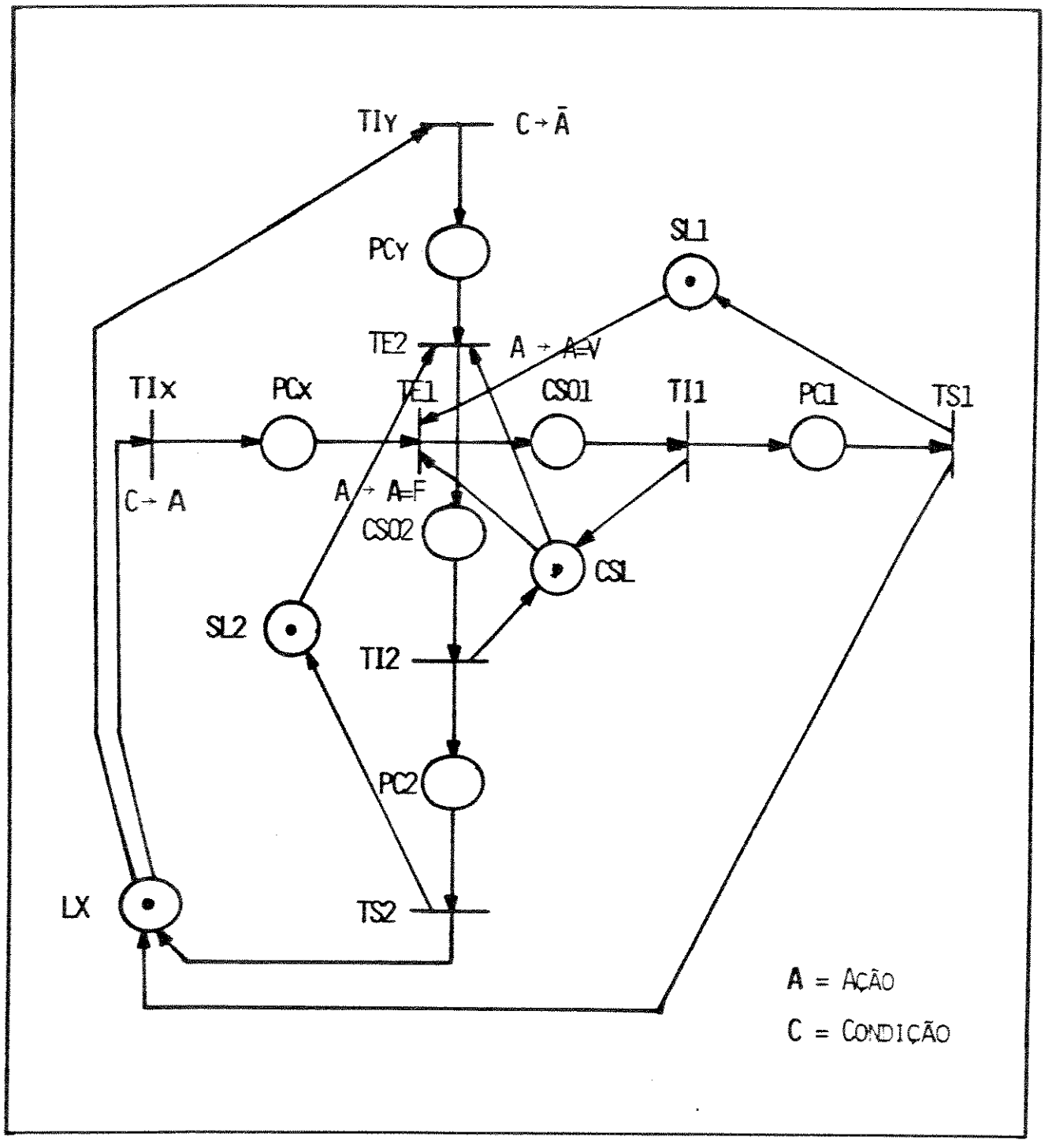


Figura 7.2.4a - Exemplo de Rede de Petri Simulador

7.2.4.1. Linguagem DESY/2

A Figura 7.2.4b mostra a descrição do exemplo efetuada através da linguagem DESY/2.

```
NOEUDS
      PCx,PCy,SL1,SL2,CSO1,CSO2,CSL,PC1,PC2,LX: PLACE;
      TS1,TS2,TIx,TIy,TI1,TI2,TE,TE2: TRANSITION;

ARCS
     CHEMIN  =  (TIy,PCy,TE2,CSO2,TI2,PC2,TS2,LX,TIy),
                (TIx,PCx,TE1,CSO1,TI1,PC1,TS1,LX,TIx),
                (TS1,SL1,TE1),
                (TS2,SL2,TE2),
                (TI1,CSL,TE2),
                (TI2,CSL,TE1);

VAR
     A: BOOLEEN;

INTERPRETATION
     ACTION TE1= A:=FAUX;
     ACTION TE2= A:=VRAI;

     COND TIx= A;
     COND TIy= (PAS A);

FIN.
```

Figura 7.2.4b - Descrição de uma Rede de Petri
Linguagem DESY/2

A partir desta descrição o Programa de Análise Sintática e Semântica (DES2) gera uma tabela a ser utilizada pelo módulo de simulação.

Comentário

A instrução *nó* descreve os lugares e transições da rede.

A instrução *arco* descreve as interligações entre os lugares e as transições.

A instrução *var* descreve o tipo das variáveis utilizadas.

A instrução *interpretação* reune um conjunto de expressões associadas a transições que neste trabalho são utilizadas na solução de conflitos e sincronização.

A instrução *fim* encerra a descrição da rede.

7.2.4.2. Linguagem DESAL

A Figura 7.2.4c mostra a descrição das condições de simulação efetuada através da linguagem DESAL.

```
PLACE
  MARQUAGE LX,SL1,SL2,CSL := 1;
  TEMPS CSO1,CSO2,PCy,PCx,PC1,PC2,SL1,SL2,CSL,LX := 1;
ECHEANCIER
  STATISTIQUE DEBUT 0;
  STATISTIQUE FIN 60;
  OBSERVER 0*1;
FIN.
```

Figura 7.2.4c - Descrição das condições de simulação Linguagem DESAL

A partir desta descrição um programa de Análise Sintática e Semântica (DES2) gera uma tabela a ser utilizada pelo módulo de simulação.

Comentário

A instrução *lugar* descreve a marcação inicial dos lugares e os tempos associados a cada lugar.

A instrução *calendário* reune um conjunto de instruções que define o início e fim da simulação bem como os intervalos de observação do sistema.

A Figura 7.2.4d mostra o resultado da simulação efetuada pelo programa *PSM*.

```
TEMPS INITIAL :       0
TEMPS FINAL   :      60

TRANSITIONS :
              TS1    7 TIRS, TRANSIT :     8.571
              TS2    8 TIRS, TRANSIT :     7.500
              TIx    8 TIRS, TRANSIT :     7.500
              TIy    7 TIRS, TRANSIT :     8.571
              TI1    7 TIRS, TRANSIT :     8.571
              TI2    8 TIRS, TRANSIT :     7.500
              TE1    7 TIRS, TRANSIT :     8.571
              TE2    8 TIRS, TRANSIT :     7.500

PLACES :
          PCx   TEMPS=  1
            MAMAX=    1     MOYJ = 0.12     NFM=    7
            TMD  =    0     MOYTD= 0.00
          PCy   TEMPS=  1
            MAMAX=    1     MOYJ = 0.14     NFM=    8
            TMD  =    0     MOYTD= 0.00
          SL1   TEMPS=  1
            MAMAX=    1     MOYJ = 0.75     NFM=    7
            TMD  =    5     MOYTD= 5.00
          SL2   TEMPS=  1
            MAMAX=    1     MOYJ = 0.73     NFM=    8
            TMD  =    5     MOYTD= 4.50
          CSO1  TEMPS=  1
            MAMAX=    1     MOYJ = 0.13     NFM=    7
            TMD  =    0     MOYTD= 0.00
          CSO2  TEMPS=  1
            MAMAX=    1     MOYJ = 0.14     NFM=    8
            TMD  =    0     MOYTD= 0.00
          CSL   TEMPS=  1
            MAMAX=    1     MOYJ = 0.74     NFM=   15
            TMD  =    2     MOYTD= 1.93
          PC1   TEMPS=  1
            MAMAX=    1     MOYJ = 0.13     NFM=    7
            TMD  =    0     MOYTD= 0.00
          PC2   TEMPS=  1
            MAMAX=    1     MOYJ = 0.14     NFM=    8
            TMD  =    0     MOYTD= 0.00
          LX    TEMPS=  1
            MAMAX=    1     MOYJ = 0.25     NFM=   15
            TMD  =    0     MOYTD= 0.25
```

Figura 7.2.4d - Resultado da simulação

O resultado da simulação apresesenta os seguintes valores:

- Tempo de simulação (TEMPS INITIAL e TEMPS FINAL)
- Número de disparo de cada transição (TRANSITIONS)
- Tempo associado a cada lugar (TEMPS)
- Número máximo de marcas em cada lugar (MAMAX)
- Número médio de marcas em cada lugar (MOYJ)
- Número de vezes que um lugar foi marcado (NFMD)
- Tempo máximo de marcação de um lugar (TMD)
- Tempo médio de marcação de um lugar (MOYTD).

7.3. TRADUÇÃO REDE DE PETRI/ARQUIVO DE ENTRADA DO SIMULADOR DE REDES DE PETRI (PETSIM)

Este módulo tem por objetivo traduzir a descrição de uma Rede de Petri para um arquivo de dados de entrada do simulador de Redes de Petri.

Esta descrição da Rede de Petri é obtida a partir de especificações de um Sistema de Software Tempo Real utilizando a *Interface Interativa de Especificação do Sistema* e traduzindo para um arquivo de entrada do Módulo de Descrição da Rede de Petri (DESY/2) do simulador PSI.

O primeiro programa deste módulo gera uma base de dados específica para o simulador, traduzindo a base de dados de descrição da Rede de Petri.

O segundo programa gera um arquivo de entrada no padrão de Linguagem DESY/2 do simulador de Redes de Petri descrito no item 7.2.

O simulador possui um programa de análise sintática e semântica (DES2) que irá compilar os comandos da Linguagem DESY/2.

Os dois últimos programas permitem deletar a base de dados do simulador e imprimir uma Rede de Petri nos moldes de um programa descrito pela Linguagem DESY/2 do simulador.

7.4. TRADUÇÃO DESCRIÇÃO DA DINAMICA/ARQUIVO DE ENTRADA DO SIMULADOR DE REDES DE PETRI (CONDSIM)

Este módulo tem por objetivo traduzir a descrição das condições de simulação de um modelo de Rede de Petri para um arquivo de dados de entrada do simulador de Redes de Petri.

Esta descrição das condições de simulação é obtida a partir das especificações de um Sistema de Software com Características Tempo Real utilizando a *Interface Interativa de Descrição da Dinâmica do Sistema* e traduzindo para um arquivo de entrada do Módulo de Descrição das Condições de Simulação (DESAL) do simulador PSI.

O primeiro programa deste módulo gera um arquivo de entrada no padrão da Linguagem DESAL do simulador de Redes de Petri descrito no item 7.2.

O simulador possui um programa de análise sintática (DES2) que irá compilar os comandos da Linguagem DESAL.

O segundo e último programa permite imprimir as condições de smulação nos moldes de um programa descrito pela Linguagem DESAL do simulador.

CAPITULO VIII

ESTRUTURA DO SISTEMA E BASE DE DADOS

8.1. INTODUÇÃO

O presente capítulo tem por objetivo apresentar a estrutura do trabalho computacional realizado como parte experimental da tese.

O primeiro item apresenta a estrutura do sistema e o segundo item apresenta a estrutura das bases de dados geradas.

8.2. ESTRUTURA DO SISTEMA

O sistema foi estruturado de forma modular sendo que cada módulo é decomposto em sub-módulos e assim sucessivamente até atingir o nível dos programas básicos de manipulação das correspondentes bases de dados.

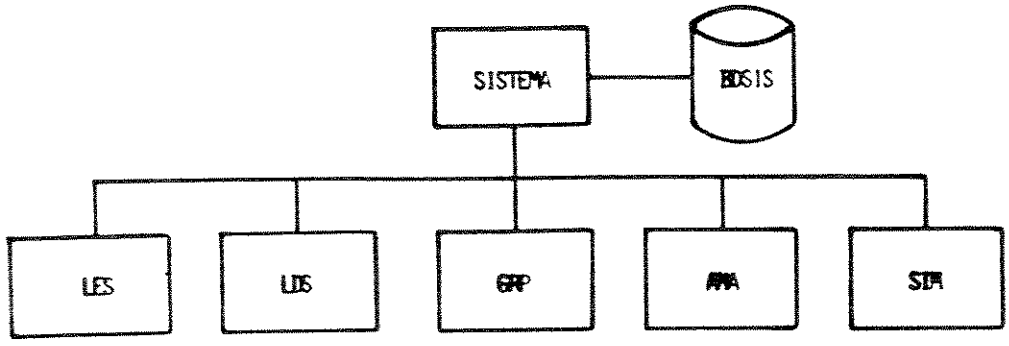
Os programas básicos permitem inserir, mostrar, alterar e imprimir as informações constantes nas bases de dados.

Os módulos de tradução para o analisador e simulador possuem programas básicos que permitem traduzir as base de dados e gerar arquivos de entrada para o analisador e simulador.

A estrutura modular do sistema permite uma melhor organização do trabalho, bem como facilitar as atividades de codificação, depuração e testes do sistema.

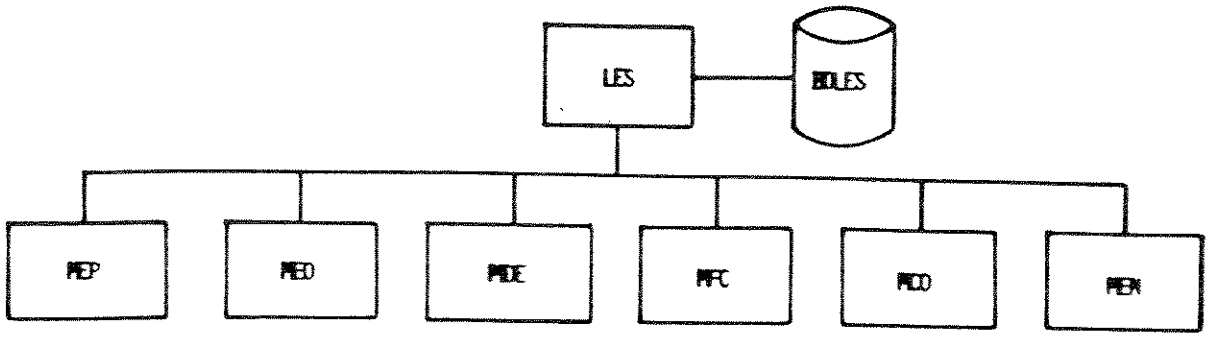
Esta estrutura facilita também as possíveis alterações que possam vir a ser feitas no sistema, decorrentes de reavaliações e/ou evoluções dos conceitos teóricos que serviram de suporte para a concepção do sistema.

As figuras a seguir ilustram a estrutura do sistema e de seus níveis de decomposição.



LES = INTERFACE DE ESPECIFICAÇÃO DE SISTEMAS
LDS = INTERFACE DE DESCRIÇÃO DA DINÂMICA DO SISTEMA
GRP = GERADOR DE REDES DE PETRI
ANA = ANÁLISE
SIM = SIMULAÇÃO

FIGURA 8.2 - ESTRUTURA DO SISTEMA



MEP = MÓDULO DE ELEMENTOS DE PROJETO
MED = MÓDULO DE ELEMENTOS OPERACIONAIS
MDE = MÓDULO DE DECOMPOSIÇÃO

MFC = MÓDULO DE FLUXO DE CONTROLE
MCO = MÓDULO DE COMUNICAÇÃO
MEN = MÓDULO DE ENLACE

FIGURA 8.2.1 - INTERFACE DE ESPECIFICAÇÃO DE SISTEMAS (LES)

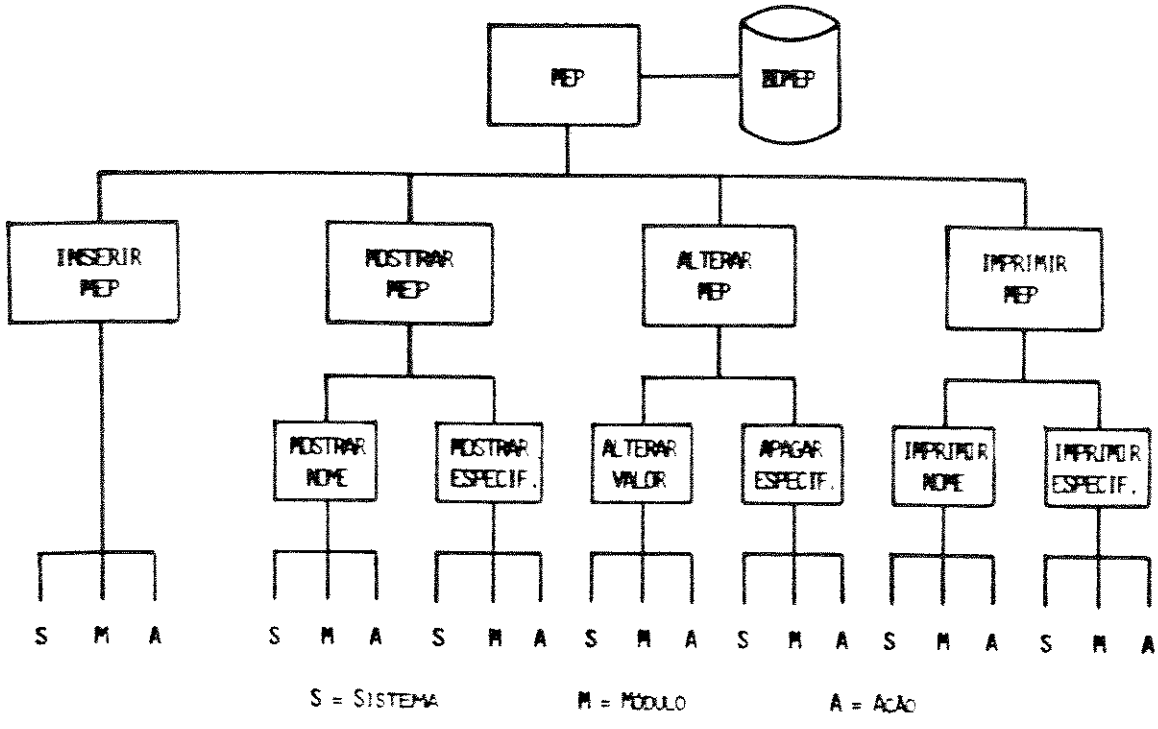


FIGURA 8.2.1.1 - MÓDULO DE ELEMENTOS DE PROJETO (MEP)

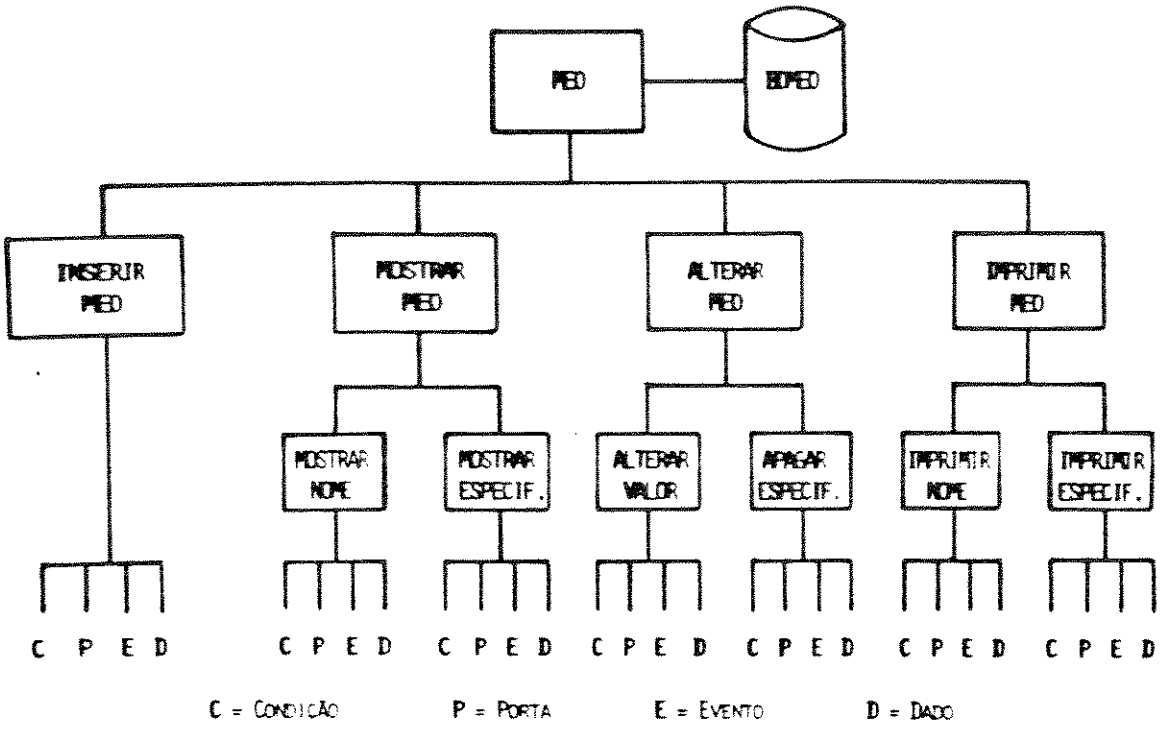


FIGURA 8.2.1.2 - MÓDULO DE ELEMENTOS OPERACIONAIS (MED)

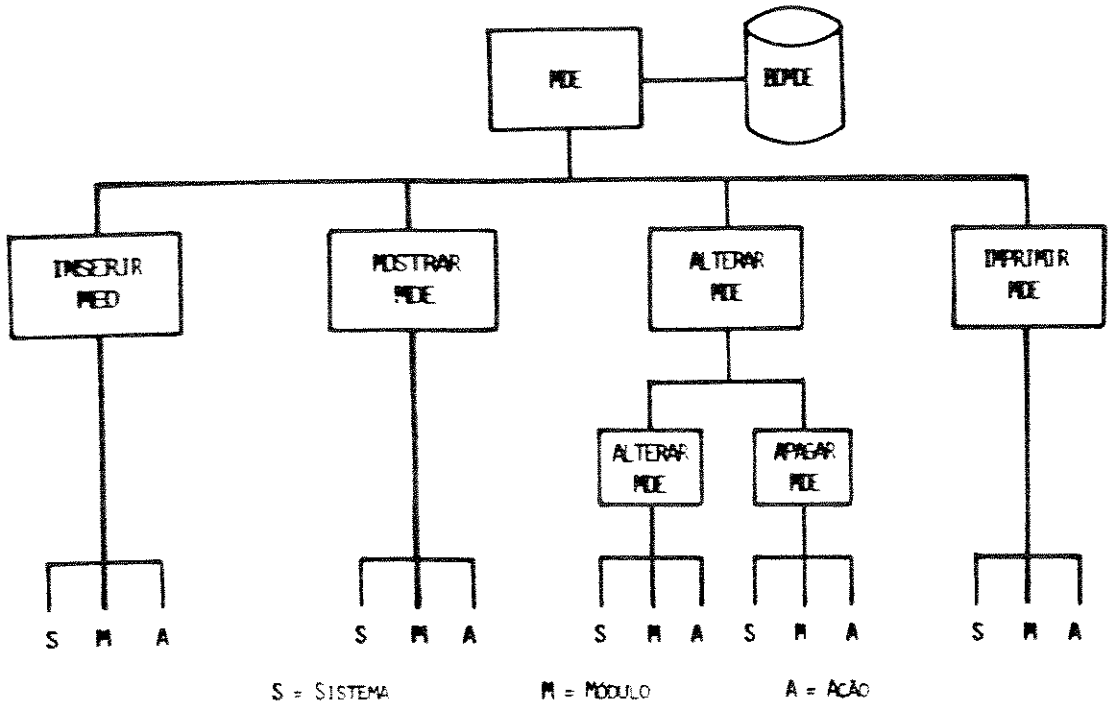


FIGURA 8.2.1.3 - MÓDULO DE DECOMPOSIÇÃO (MDE)

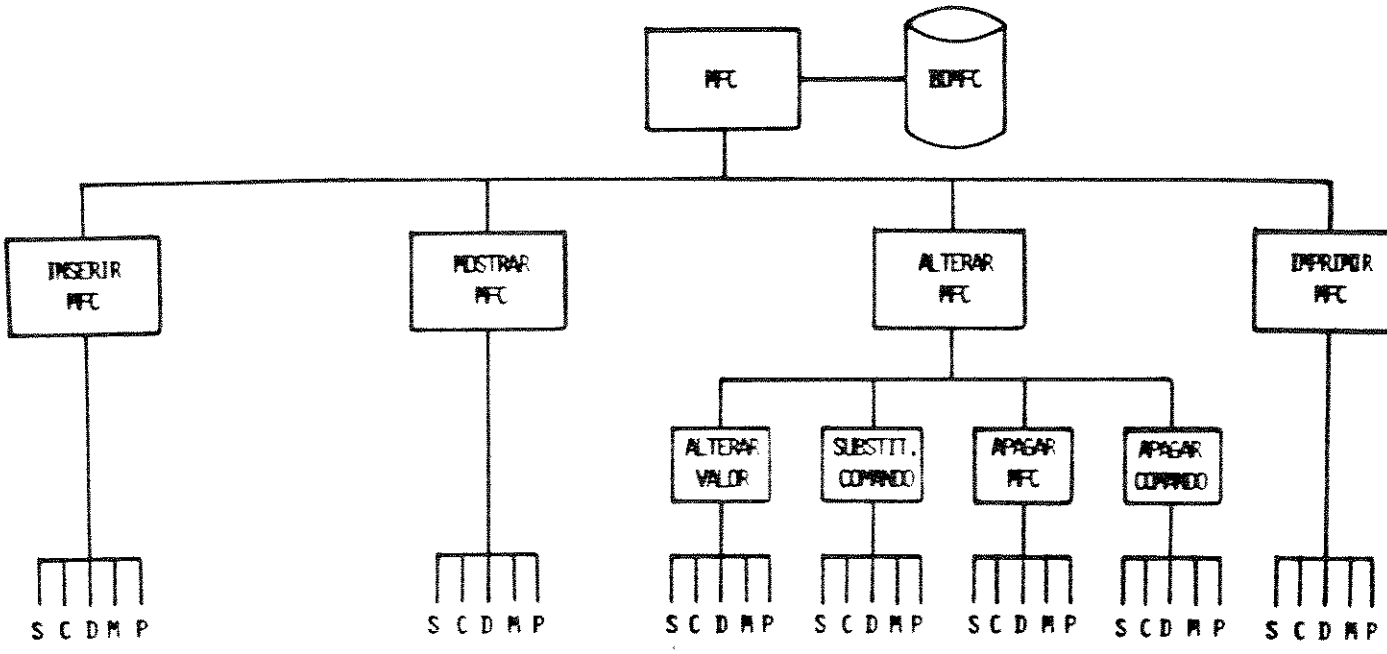


FIGURA 8.2.1.4 - MÓDULO DE FLUXO DE CONTROLE (MFC)

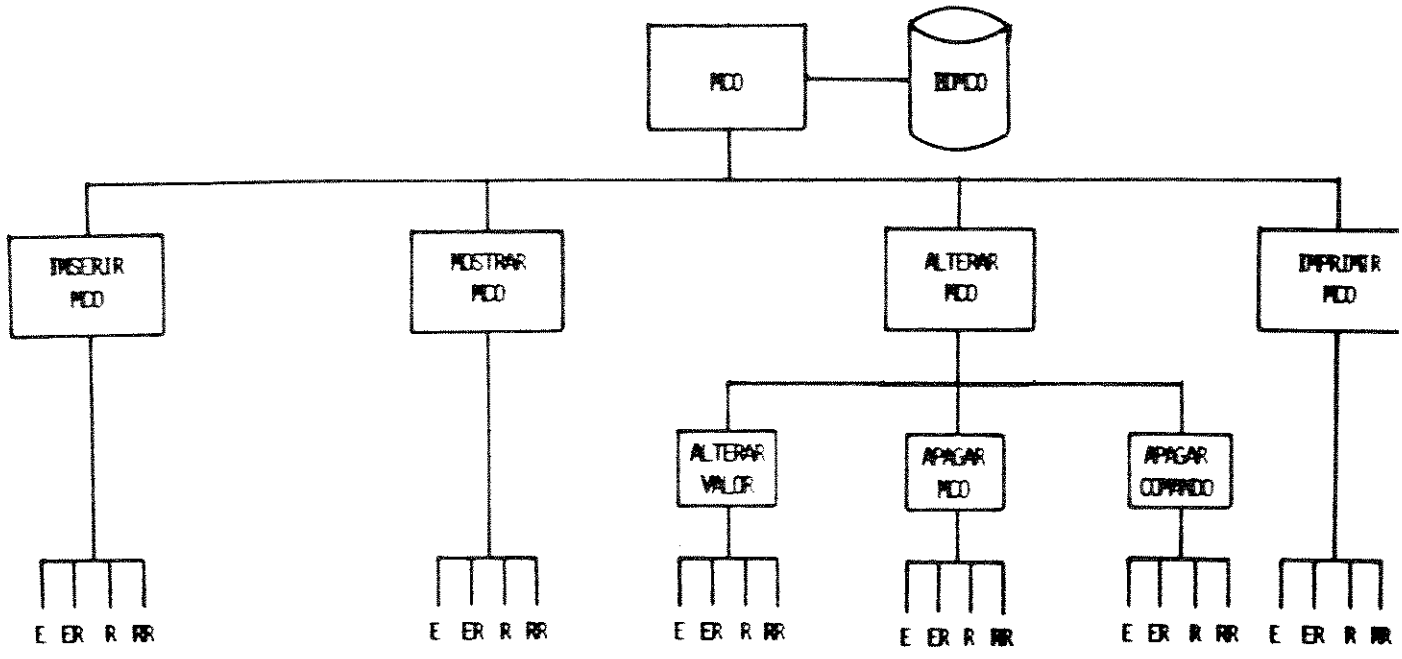


FIGURA 8.2.1.5 - MÓDULO DE COMUNICAÇÃO (MCD)

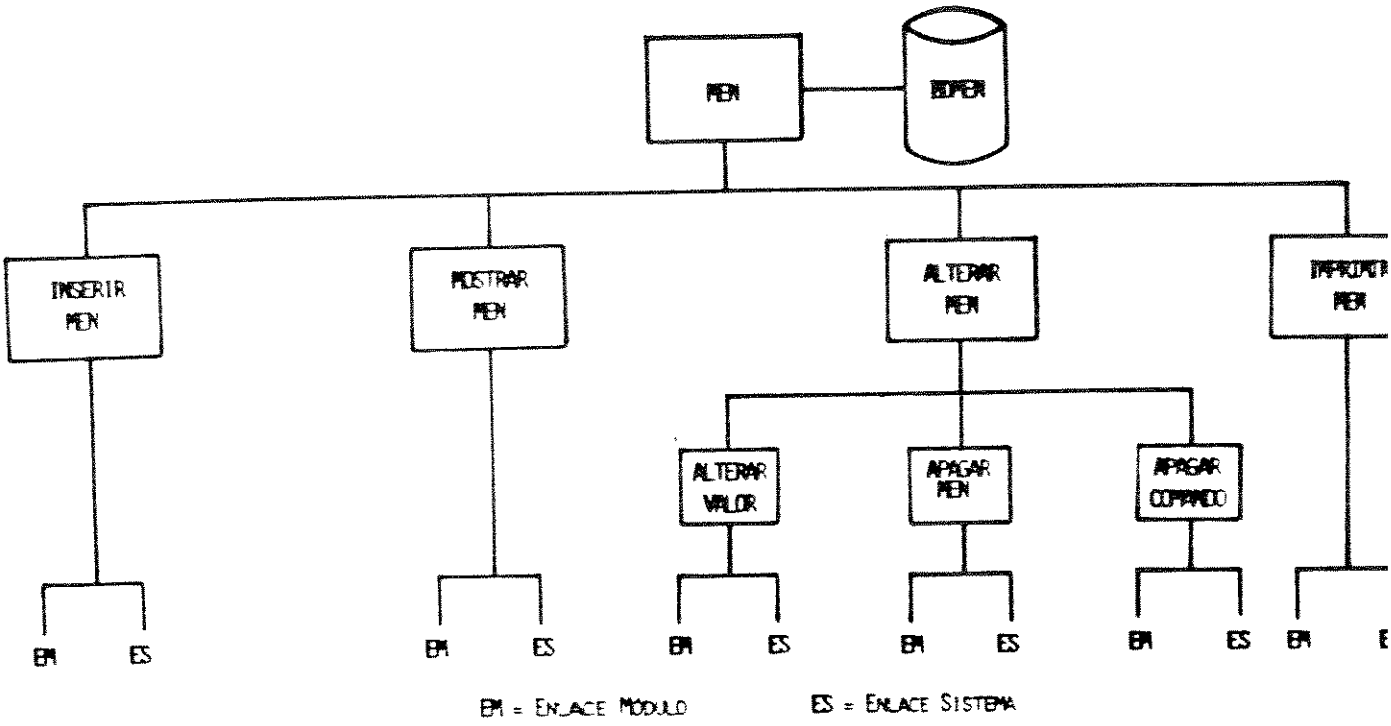
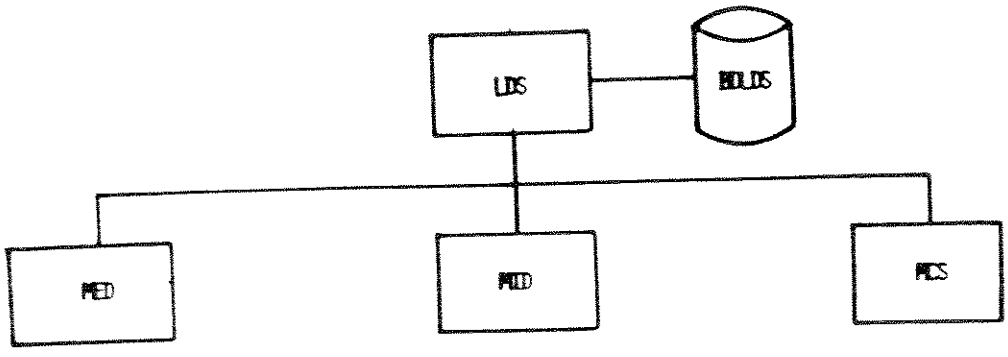


FIGURA 8.2.1.6 - MÓDULO DE ENLACE (MEN)



MED = MÓDULO DE ESPECIFICAÇÃO DA DINÂMICA
MID = MÓDULO DE INTERPRETAÇÃO DA DINÂMICA
MCS = MÓDULO DE CONDIÇÕES DE SIMULAÇÃO

FIGURA 8.2.2 - INTERFACE DE DESCRIÇÃO DA DINÂMICA DO SISTEMA (LDS)

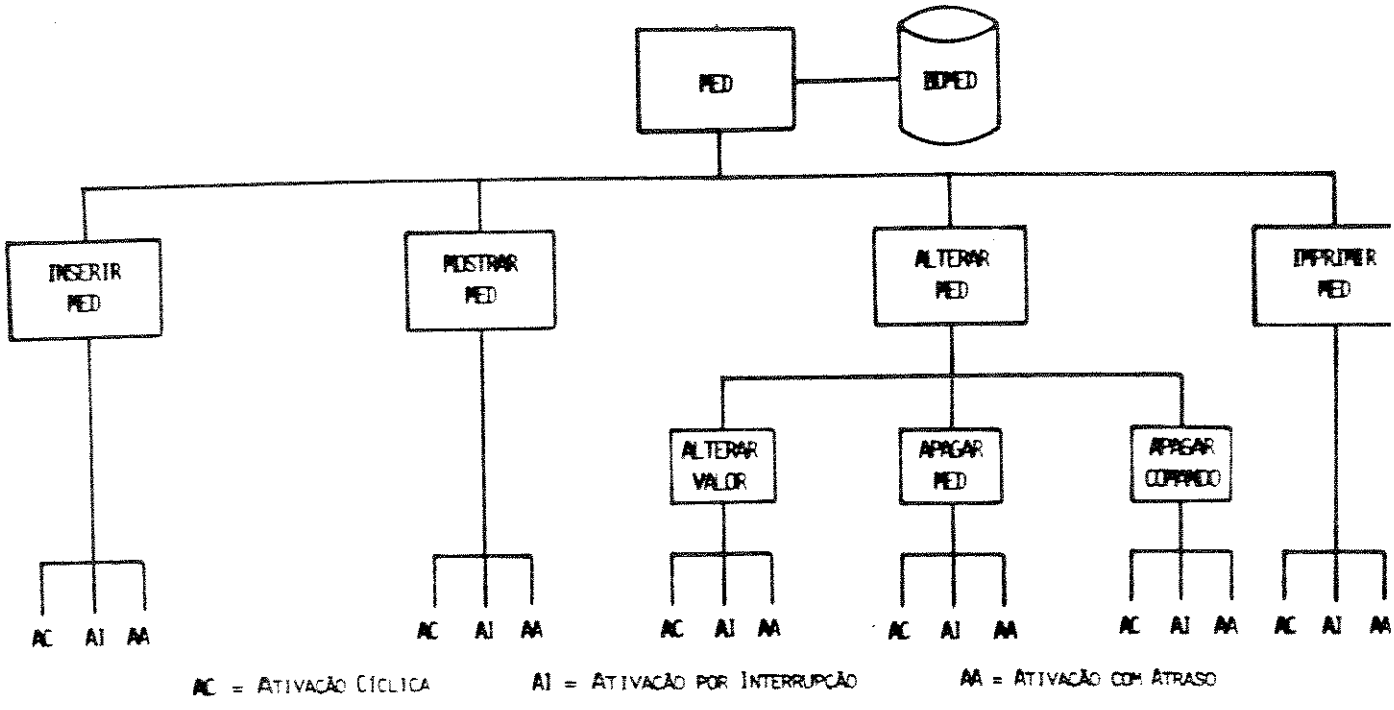


FIGURA 8.2.2.1 - MÓDULO DE ESPECIFICAÇÃO DA DINÂMICA (MED)

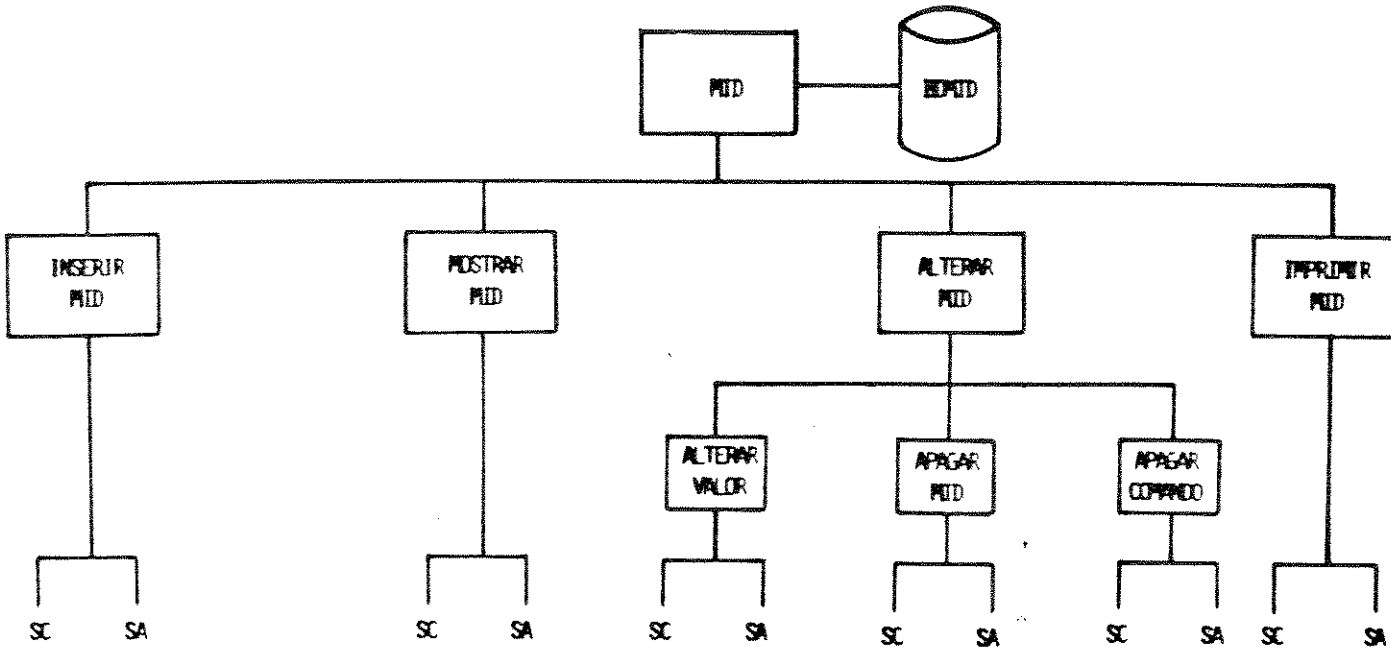


FIGURA 8.2.2.2 – MÓDULO DE INTERPRETAÇÃO DA DINÂMICA (MID)

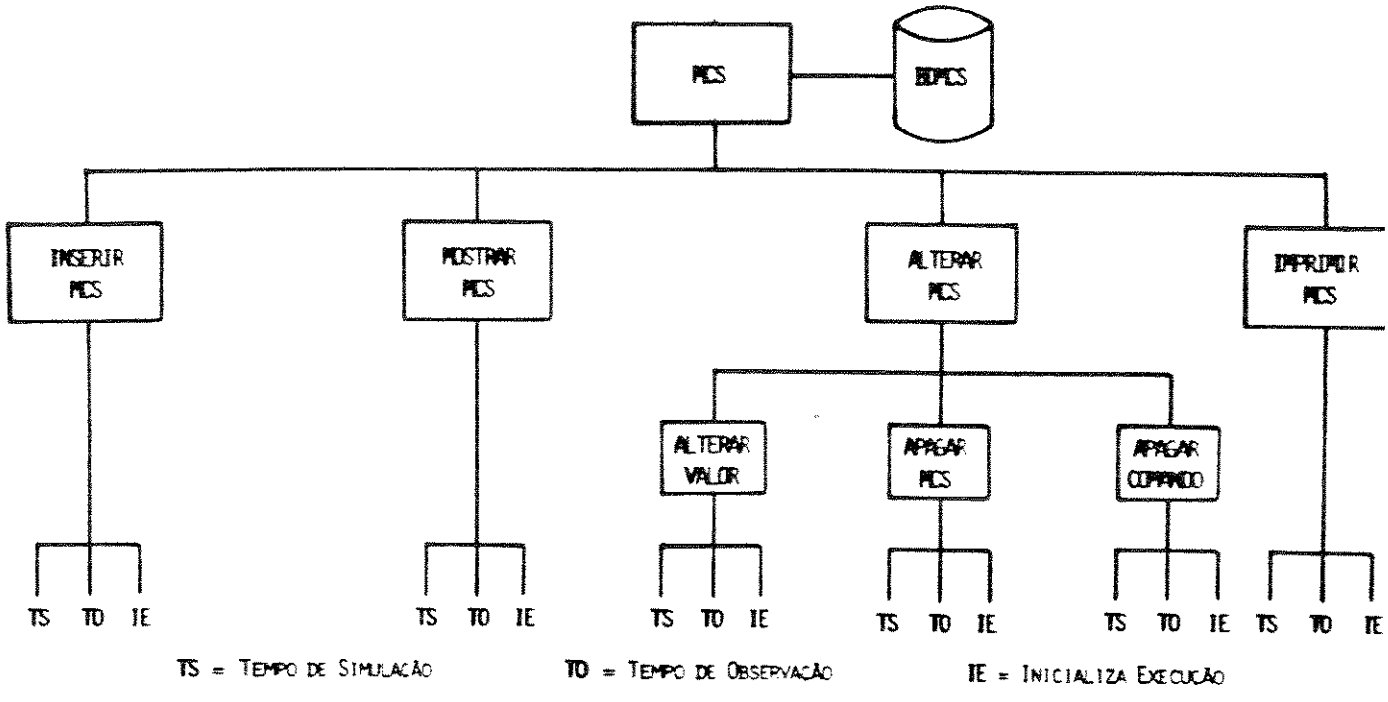


FIGURA 8.2.2.3 - MÓDULO DE CONDIÇÕES DE SIMULAÇÃO (MCS)

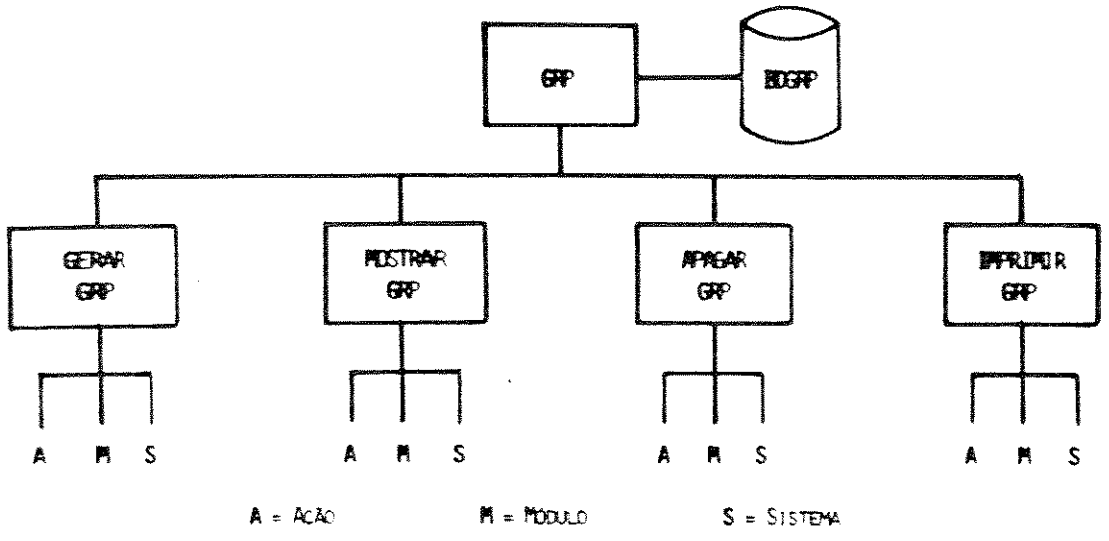


FIGURA 8.2.3 - GERAÇÃO DE REDE DE PETRI

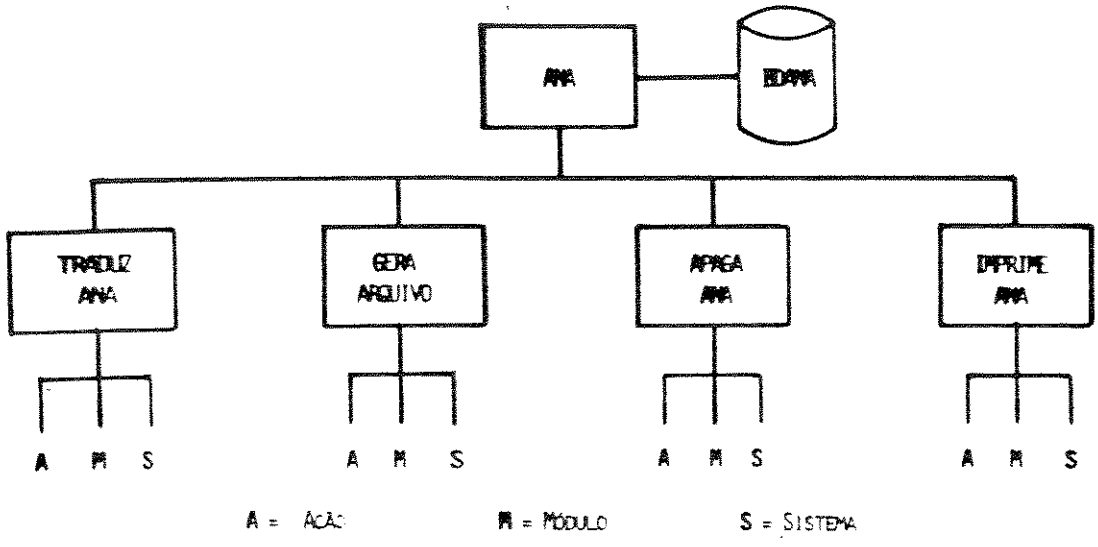
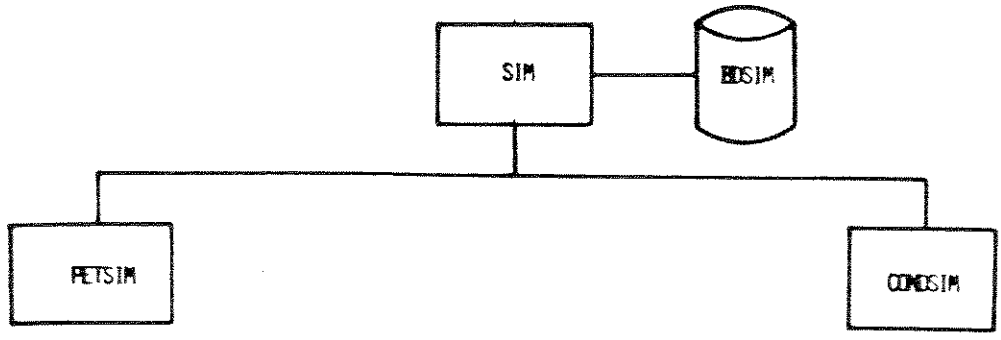


FIGURA 8.2.4 - ANALISADOR (ANA)



PETSIM - DESCRIÇÃO DA REDE DE PETRI
CONDSIM - DESCRIÇÃO DAS CONDIÇÕES DE SIMULAÇÃO

FIGURA 8.2.5 - SIMULADOR (SIM)

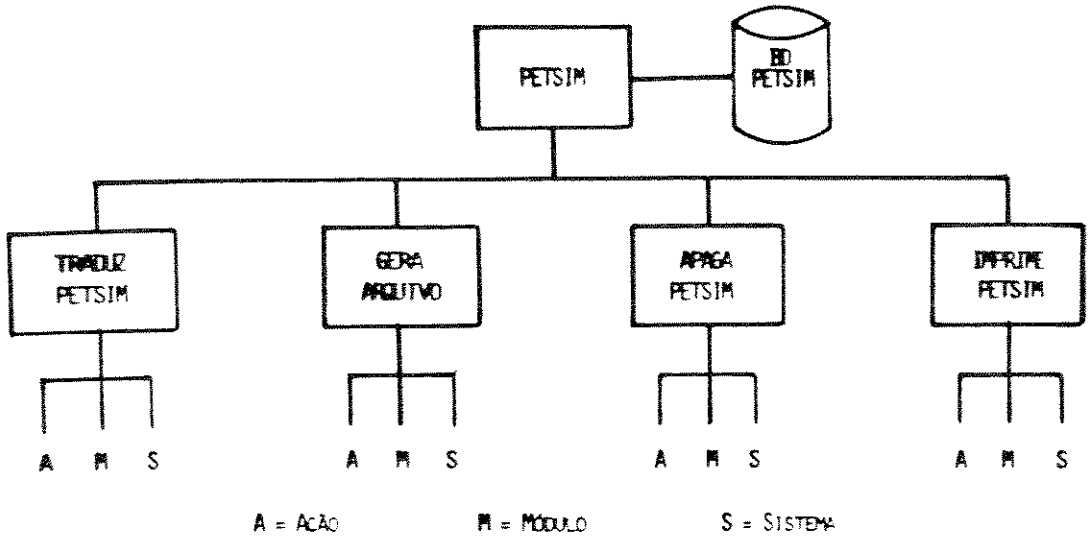


FIGURA 8.2.5.1 - DESCRIÇÃO DA REDE DE PETRI (PETSIM)

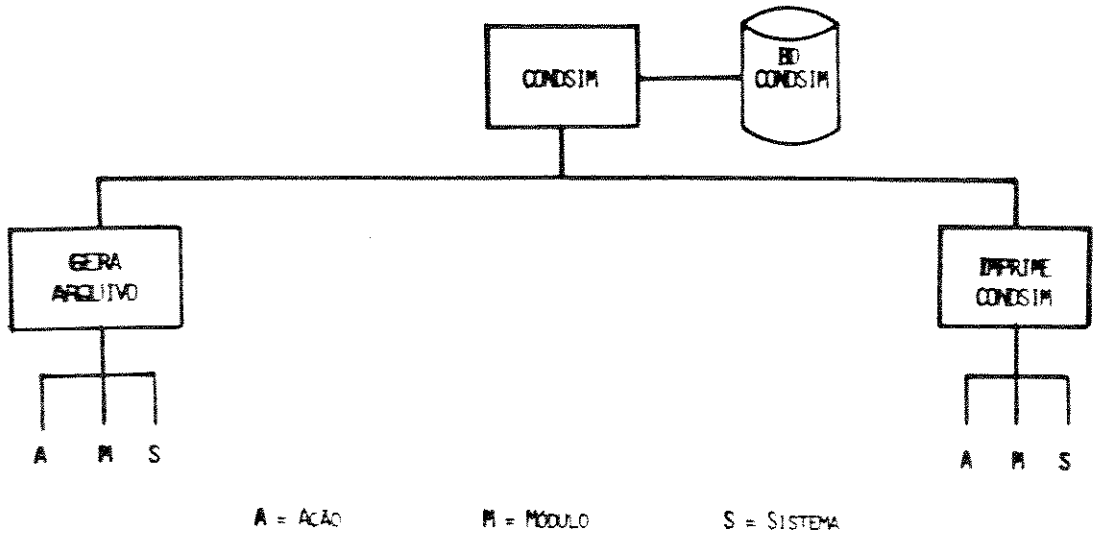


FIGURA 8.2.5.2 - DESCRIÇÃO DAS CONDIÇÕES DE SIMULAÇÃO (CONDSIM)

8.3. BASES DE DADOS

A base de dados foi estruturada de forma modular atendendo os critérios de modularidade do sistema.

Cada módulo do sistema gera portanto uma base de dados que armazena as informações especificas daquele módulo.

As bases de dados foram concebidas segundo o modelo *Entidade-Relacionamento*.

As *Entidades* são os elementos de projeto (Sistema, Módulo e Ação) e os Elementos Operacionais (Condição, Porta, Evento e Dado).

Cada uma das entidades possui associado um conjunto de *Atributos* (nome, objetivo, data, versão e valor) que constituem a Ficha de Descrição do Elemento.

A estrutura de implementação do software permite facilmente a inclusão de novos *Atributos* que venham a ser definidos posteriormente, com base em reavaliações e/ou evoluções dos conceitos teóricos que serviram de suporte a este trabalho.

Os *Relacionamentos* são estruturas que agrupam as entidades segundo diferentes critérios de organização, tais como:

- Decomposição (MDE)
- Fluxo de Controle (MFC)
- Comunicação (MCO)
- Enlace (MEN)
- Especificações da Dinâmica (MED)
- Interpretação da Dinâmica (MID)
- Condições de Simulação (MCS)

Como exemplo, o módulo de *Fluxo de Controle* gera uma base de dados com uma estrutura em árvore do tipo pai-filho; sendo o pai a ação principal a ser programada, e os filhos os elementos (entidades) que irão compor o fluxo de controle desta ação.

Esta estrutura utiliza também três parâmetros adicionais: tipo, nível e número.

O parâmetro *Tipo* identifica a natureza do comando que pode ser Sequencial, Cíclico, Desvio Condicional, Desvio Múltiplo e Paralelo.

O parâmetro *Nível* agrupa os diferentes elementos (entidades/filhos) que irão compor um comando.

O parâmetro *Número* é utilizado na geração da Rede de Petri, e define a *localização* do comando na *estrutura global* do fluxo de controle da ação principal objeto da programação.

Os demais módulos do sistema geram bases de dados segundo esta mesma estrutura, ou seja, uma árvore do tipo pai-filho e parâmetros adicionais que permitem representar, identificar e agrupar os dados armazenados.

Os itens a seguir apresentam as bases de dados geradas em cada módulo do sistema, necessáricas tanto para a descrição de um Sistema de Software com Características Tempo Real como para a posterior Geração Automática das Redes de Petri.

O conjunto de informações armazenadas são *suficientes* para a geração automática das Redes de Petri segundo os padrões de entrada do analisador e simulador utilizados na validação.

O conjunto de informações para a descrição de um Sistema de Software com Características Tempo Real podem ser modificadas com base em reavaliações e/ou evoluções dos conceitos teóricos que serviram de suporte a este trabalho.

8.3.1. MÓDULO DE ELEMENTOS DE PROJETO (BDMEP)

Sistema

sistema(Cod_sis,Nome_sis,nome)
sistema(Cod_sis,Obj_sis,objetivo)
sistema(Cod_sis,Data_sis,data)
sistema(Cod_sis,Hora_sis,hora)
sistema(Cod_sis,Ver_sis,versao)

Módulo

modulo(Cod_mod,Nome_mod,nome)
modulo(Cod_mod,Obj_mod,objetivo)
modulo(Cod_mod,Data_mod,data)
modulo(Cod_mod,Hora_mod,hora)
modulo(Cod_mod,Ver_mod,versao)

Ação

acao(Cod_acao,Nome_acao,nome)
acao(Cod_acao,Obj_acao,objetivo)
acao(Cod_acao,Data_acao,data)
acao(Cod_acao,Hora_acao,hora)
acao(Cod_acao,Ver_acao,versao)

As entidades Cod_xx são geradas automaticamente pelo sistema através de regras que gerenciam a criação dos códigos.

Os atributos Nome_xx, Obj_xx, Ver_xx são solicitados interativamente pelo sistema através de fichas de descrição, as quais são preenchidas campo por campo pelo usuário, utilizando o editor de texto do sistema.

Os atributos Data_xx e Hora_xx são fornecidos automaticamente pelo sistema.

8.3.2. MÓDULO DE ELEMENTOS OPERACIONAIS (BDMEO)

Condição

condicao(Cod_cond,Nome_cond,nome)
condicao(Cod_cond,Obj_cond,objetivo)
condicao(Cod_cond,Data_cond,data)
condicao(Cond_cond,Hora_cond,hora)
condicao(Cod_cond,Ver_cond,versao)

Porta

porta(Cod_porta,Nome_porta,nome)
porta(Cod_porta,Obj_porta,objetivo)
porta(Cod_porta,Data_porta,data)
porta(Cod_porta,Hora_porta,hora)
porta(Cod_porta,Ver_porta,versao)

Evento

evento(Cod_even,Nome_even,nome)
evento(Cod_even,Obj_even,objetivo)
evento(Cod_even,Data_even,data)
evento(Cod_even,Hora_even,hora)
evento(Cod_even,Ver_even,versao)

Dado

dado(Cod_dado,Nome_dado,nome)
dado(Cod_dado,Obj_dado,objetivo)
dado(Cod_dado,Data_dado,data)
dado(Cod_dado,Hora_dado,hora)
dado(Cod_dado,Ver_dado,versao)
dado(Cod_dado,Valor_dado,valor)

Os comentários a respeito dos detatalhes de implementação neste caso são os mesmos do módulo de elementos de projeto.

8.3.3. MÓDULO de DECOMPOSIÇÃO (BDMDE)

Sistema

decsis(Cod_sis,Cod_mod,modulo)

Módulo

decmod(Cod_mod,Cod_acao,acao)

Ação

decacao(Cod_acao,Cod_acao1,acao)

As entidades-filho Cod_mod, Cod_acao e Cod_acao1 são solicitadas interativamente pelo sistema. O usuário fornece um código através do editor de texto, e o sistema realiza um cheque de consistência quanto à existência ou não do código fornecido. O sistema irá também informar o usuário através de mensagens sobre a não existência do código fornecido.

8.3.4. MÓDULO DE FLUXO DE CONTROLE (BDMFC)

Comando Sequencial

```
mfc(Cod_acao,Cod_acao1 ,sequencial,nivel,numero)
```

Cod_acao = código ação principal
Cod_acao1 = código ação decomposta
sequencial = tipo
nivel = número que agrupa elementos de um comando
numero = número para ligação no Fluxo de Controle RP

Comando Ciclico

```
mfc(Cod_acao,Cod_cond,ciclico1,nivel,numero)
mfc(Cod_acao,Cod_acao1 ,ciclico2,nivel,numero)
```

Cod_acao = código ação principal
Cod_cond = código condição decomposta
Cod_acao1 = código ação cíclica
ciclico = tipo
nivel = número que agrupa elementos de um comando
numero = número para ligação no Fluxo de Controle RP

Comando Desvio Condicional

```
mfc(Cod_acao,Cod_cond,desvcond1,nivel,numero)
mfc(Cod_acao,Cod_acao1 ,desvcond2,nivel,numero)
mfc(Cod_acao,Cod_acao2,desvcond3,nivel,numero)
```

Cod_acao = código ação principal
Cod_cond = código da condição (IF)
Cod_acao1 = código ação 1 (THEN)
Cod_acao2 = código ação 2 (ELSE)
desvcond = tipo

nivel = número que agrupa elementos de um comando
numero = número para ligação no Fluxo de Controle RP

Neste caso o sistema realiza um cheque de consistência no código (Cod_acao2) fornecido pelo usuário, evitando redundância com o código (Cod_acao1) e consequentemente *repetição* de uma ação do mesmo comando.

Comando Desvio Múltiplo

```
mfc(Cod_acao,Cod_cond,desvmult1,nivel,numero)
mfc(Cod_acao,Cod_acao1,desvmult2,nivel,numero)
mfc(Cod_acao, Cod_acao2,desvmult2,nivel,numero)
.
.
.
mfc(Cod_acao,Cod_acaon,desvmult2,nivel,numero)
```

Cod_acao = código ação principal
Cod_cond = código da condição (CASE)
Cod_acao1 = código ação decomposta
.
.
.
Cod_acaon = código ação decomposta n
desvmult = tipo
nivel = número que agrupa elementos de um comando
numero = número para ligação no Fluxo de Controle RP

Neste caso o sistema realiza um cheque de consistência no código (Cod_acao_xx) fornecido pelo usuário evitando redundância com os demais códigos fornecidos anteriormente, e com consequente *repetição* de uma ação no mesmo comando.

Comando Paralelo

mfc(Cod_acao,Cod_acao1,paralela1,nivel,numero)

Cod_acao = código ação principal
Cod_acao1 = código ação decomposta 1
paralela1 = tipo
nivel = número que agrupa elementos de um comando
numero = número para ligação no Fluxo de Controle RP

As entidades-filho contidas em cada um dos comandos são solicitadas interativamente pelo sistema. O usuário fornece um código através do editor de texto, e o sistema realiza um cheque de consistência quanto à existência ou não do código fornecido. O sistema irá também informar o usuário através de mensagens sobre a não existência do código fornecido.

O parâmetro *Tipo* é fornecido pelo usuário através de uma escolha múltipla em um menu que contém os comandos do fluxo de controle.

Os parâmetros *nível* e *número* são gerados e gerenciados pelo sistema através de regras que gerenciam a criação de códigos.

8.3.5. MÓDULO DE COMUNICAÇÃO (BDMCO)

Comando Envia

```
mco(Cod_acao,Cod_dado,envia1,nivel,numero)
mco(Cod_acao,Cod_acao1,envia2,nivel,numero)
mco(Cod_acao,Cod_porta,envia3,nivel,numero)
```

Cod_acao = código ação principal
Cod_dado = código da mensagem enviada
Cod_acao1 = código da ação que envia a mensagem
Cod_porta = código da porta de saída
envia = tipo
nivel = número que agrupa os elementos de um comando
numero = número para ligação no Fluxo de Controle RP

Comando Envia com Resposta

```
mco(Cod_acao,Cod_dado,envresp2,nivel,numero)
mco(Cod_acao,Cod_acao1,envresp2,nivel,numero)
mco(Cod_acao,Cod_porta,envresp3,nivel,numero)
mco(Cod_acao,Cod_porta1,envresp4,nivel,numero)
```

Cod_acao = código ação principal
Cod_dado = código da mensagem enviada
Cod_acao1 = código da ação que envia a mensagem
Cod_porta = código da porta de saída
Cod_porta1 = código da porta de entrada
envresp = tipo
nivel = número agrupa os elementos de um comando
numero = número para ligação no Fluxo de Controle RP

Comando Recebe

```
mco(Cod_acao,Cod_dado,recebe1,nivel,numero)
mco(Cod_acao,Cod_acao1,recebe2,nivel,numero)
mco(Cod_acao,Cod_porta1,recebe3,nivel,numero)
  .                                   .
  .                                   .
mco (Cod_acao,Cod_portan,recebe3,nivel,numero)
```

Cod_acao = código ação principal
Cod_dado = código da mensagem recebida
Cod_acao1 = código da ação que recebe a mensagem
Cod_porta1 = código da porta de entrada 1

.
.
.

Cod_portan = código da porta de entrada N
recebe = tipo
nivel = número agrupa os elementos de um comando
numero = número para ligação no Fluxo de Controle RP

Comando Recebe com Resposta

```
mco(Cod_acao,Cod_dado,recresp1,nivel,numero)
mco(Cod_acao,Cod_acao1,recresp2,nivel,numero)
mco(Cod_acao,Cod_porta,recresp3,nivel,numero)
mco(Cod_acao,Cod_porta1,recresp4,nivel,numero)
.                                   .
.                                   .
.                                   .
mco(Cod_acao,Cod_portan,recresp4,nivel,numero)
```

Cod_acao = código ação principal
Cod_dado = código da mensagem recebida
Cod_acao1 = código da ação que recebe a mensagem
Cod_porta = código da porta de saída para envio de resposta

Cod_porta1 = código porta de entrada 1

.
.
.

Cod_portan = código porta de entrada N
recresp = tipo
nivel = número agrupa elementos de um comando
numero = número para ligação no Fluxo de Controle RP

Os comentários a respeito dos detalhes de implementação neste caso são os mesmos do módulo fluxo de controle.

8.3.6. MÓDULO DE ENLACE (BDMEN)

Enlace Sistema

```
mensis(Cod_sis,Cod_mod1 ,mensis1,nivel)
mensis(Cod_sis,Cod_acao1,mensis2,nivel)
mensis(Cod_sis,Cod_porta1,mensis3,nivel)
mensis(Cod_sis,Cod_mod2,mensis4,nivel)
mensis(Cod_sis,Cod_acao2,mensis5,nivel)
mensis(Cod_sis,Cod_porta2,mensis6,nivel)
```

Cod_sis = código do sistema
Cod_mod1 = código do módulo envia mensagem
Cod_acao1 = código ação envia mensagem
Cod_porta1 = código porta de saída
Cod_mod2 = código do módulo recebe mensagem
Cod_acao2= código da ação recebe mensagem
Cod_porta2 = código da porta de entrada
mensis = tipo
nivel = número agrupa elementos de um comando

Enlace Módulo

```
menmod(Cod_mod,Cod_acao1,menmod1,nivel)
menmod(Cod_mod,Cod_porta1,menmod2,nivel)
menmod(Cod_mod,Cod_acao2,menmod3,nivel)
menmod(Cod_mod,Cod_porta2,menmod4,nivel)
```

Cod_mod = código do módulo
Cod_acao1 = código ação envia mensagem
Cod_porta1 = código porta de saída
Cod_acao2 = código ação recebe mensagem
Cod_porta2 = código porta de entrada
menmod = tipo
nivel = número agrupa elementos de um comando

Os comentários a respeito dos detalhes de implementação neste caso são os mesmos do módulo de fluxo de controle.

8.3.7. MÓDULO DE ESPECIFICAÇÃO DA DINAMICA (MED)

Comando Ativação Cíclica (ação)

```
med(Cod_acao,Cod_acao,ativacic1,nivel)
med(Cod_acao,Cod_even,ativacic2,nivel)
med(Cod_acao,Temp_ativ,ativacic3,nivel)
med(Cod_acao,Int_ativ,ativacic4,nivel)
```

Cod_mod = código da ação
Cod_acao = código da ação a ser executada
Cod_even = código do evento de ativação
Temp_ativ = tempo de ocorrência da primeria ativação
Int_ativ = intervalo de tempo cíclico
ativacic = tipo
nivel = número agrupa elementos de um comando

Comando Ativação por Interrupção (ação)

```
med(Cod_acao,Cod_acao,ativaint1,nivel)
med(Cod_acao,Cod_even,ativaint2,nivel)
med(Cod_acao,Temp_ativ,ativaint3,nivel)
med(Cod_acao,Int_ativ,ativaint4,nivel)
```

Cod_mod = código da ação
Cod_acao = código da ação a ser executada
Cod_even = código do evento de ativação
Temp_ativ = tempo de ocorrência da primeira ativação
Int_ativ = intervalo de tempo aleatório
ativaint = tipo
nivel = número agrupa elementos de um comando

Comando Atraso de Ativação (ação)

```
med(Cod_acao,Cod_acao,ativaatr1,nivel)
med(Cod_acao,Cod_even,ativaatr2,nivel)
med(Cod_acao,Temp_esp,ativaatr3,nivel)
```

Cod_mod = código da ação
Cod_acao = código da ação a ser executada
Cod_even = código do evento de ativação
Temp_esp = tempo de atraso
ativaatr = tipo
nivel = número agrupa elementos de um comando

Os comentários a respeito dos detalhes de implementação neste caso são os mesmos do módulo de fluxo de controle.

8.3.8. MÓDULO DE INTERPRETAÇÃO DA DINAMICA (BDMID)

Comando Solução de Conflito (ação)

mid(Cod_acao,Tipo_interp,solconf1,nivel)
mid(Cod_acao,Tipo_var,solconf2,nivel)
mid(Cod_acao,Cond_intconf,solconf3,nivel)
mid(Cod_acao,Acao_intconf,solconf4,nivel)
mid(Cod_acao,Nome_var,solconf5,nivel)
mid(Cod_acao,Exp,solconf6,nivel)

Cod_acao = código da ação principal
Tipo_interp = tipo de interpretação (action/cond)
Tipo_var = tipo de variável (booleen/entier)
Cond_intconf = condição interpretada (*)
Acao_intconf = ação interpretada (*)
Nome_var = nome da variável
Exp = expressão lógica/algébrica
selconf = tipo
nivel = número agrupa elementos de um comando

(*) O programa tradutor encontra a transição que recebe a interpretação.

Comando Sincronização de Ações (ação)

mid(Cod_acao,Tipo_interp,sincac1,nivel)
mid(Cod_acao,Tipo_var,sincac2,nivel)
mid(Cod_acao,Porta1_sincac,sincac3,nivel)
mid(Cod_acao,Porta2_sincac,sincac4,nivel)
mid(Cod_acao,Nome_var,sincac5,nivel)
mid(Cod_acao,Exp,sincac6,nivel)

Cod_acao = código da ação principal
Tipo_interp = tipo de interpretação (action/cond)
Tipo_var = tipo de variável (booleen/entier)
Porta1_sincac = porta de envio da mensagem
Porta2_sincac = porta de recebimento da mensagem
Nome_var = nome da variável
Exp = expressão lógica/algébrica
sincac = tipo
nivel = número agrupa elementos de um comando

8.3.9. MÓDULO DE CONDIÇÕES DE SIMULAÇÃO (BDMCS)

Comando Tempo de Simulação (ação)

```
mcs(Cod_acao,Inic_sim,temposim1,nivel)
mcs(Cod_acao,Fim_sim,temposim2,nivel)
```

Cod_acao = código da ação
Inic_sim = tempo de início da simulacão
Fim_sim = tempo de fim da simulação
temposim = tipo
nivel = número agrupa elementos de um comando

Comando Tempo de Observação (ação)

```
mcs(Cod_acao,Tempo_obs,tempoobs1,nivel)
mcs(Cod_mod,Int_obs,tempoobs2,nivel)
```

Cod_acao = código do módulo
Tempo_obs = tempo de início da observação
Int_obs = intervalo de observação
tempoobs = tipo
nivel = número agrupa elementos de um comando

Comando Inicializa Execução (ação)

```
mcs(Cod_acao,Inic_exec,inicexec,nivel)
```

Cod_mod = código do módulo
Inic_exec = ações que inicializam em execução
inicexec = tipo
nivel = número agrupa elementos de um comando

Os comentários a respeito dos detalhes de implementação, neste caso, são os mesmos do módulo de fluxo de controle.

8.3.10. MÓDULO DE GERAÇÃO DE REDE DE PETRI (BDGRP)

Lugares

Lugar(Cod_acao,Cod_lugar,nome,tipo)

Cod_acao = código ação principal
Cod_lugar = código lugar (ação, condição, porta)
nome = nome lugar
tipo = fluxo/comunic (fluxo de controle ou comunicação)

Transições de Entrada

Entrada(Cod_acao,Cod_lugar,Cod_trans,tipo)

Cod_acao = código ação principal
Cod_lugar = código elemento
Cod_trans = código transição de entrada
tipo = fluxo/comunic

Transições de Saída

Saída(Cod_acao,Cod_elem,Cod_trans,tipo)

Cod_acao = código ação principal
Cod_lugar = código elemento
Cod_trans = código transição de saída
tipo = fluxo/comunic

Todas as estruturas são *geradas automaticamente* a partir das estruturas definidas nos itens anteriores e que especificam um Sistema de Software com Características Tempo Real.

8.3.11. MÓDULO DE ANÁLISE (BDANA)

Lugares

lugana(Cod_acao,Cod_lugar,Cod_lugar1,numero)

Cod_acao = código da ação principal
Cod_lugar = código do lugar
Cod_lugar1 = código do lugar no analisador
numero = número lugar no analisador

Transições

transana(Cod_acao,Cod_trans,Cod_trans1,numero)

Cod_acao = código da ação principal
Cod_trans = código da transição
Cod_trans1 = código da transição no analisador
numero = número lugar no analisador

Lugares de Entrada

entana(Cod_acao,Cod_trans1,numero,Cod_lugar1,numero1)

Cod_acao = código da ação principal
Cod_trans1 = código da transição no analisador
numero = número da transição no analisador
Cod_lugar1 = código do lugar de entrada no analisador
numero1 = número do lugar de entrada no analisador

Lugares de Saída

saiana(Cod_acao,Cod_trans1,numero,Cod_lugar1,numero1)

Cod_acao = código da ação principal
Cod_trans1 = código da transição no analisador
numero = número da transição no analisador
Cod_lugar1 = código do lugar de saída no analisador
numero1 = número do lugar de saída no analisador

Marcação Inicial

marcaana(Cod_acao,Cod_lugar,Cod_lugar1,numero)

Cod_acao = código da ação principal
Cod_lugar = código do lugar
Cod_lugar1 = código do lugar no analisador
numero = número do lugar no analisador

Todas as estruturas são *geradas automaticamente* objetivando atender os padrões de entrada do analisador.

8.3.12. MÓDULO DE SIMULAÇÃO (BDSIM)

Lugares

lugsim(Cod_acao,Cod_lugar,Nome_lugar)

Cod_acao = código da ação principal
Cod_lugar = código do lugar (ação, condição, porta)
Nome_lugar = nome do lugar

Transições

transsim(Cod_acao,Cod_trans,Nome_trans)

Cod_acao = código da ação principal
Cod_trans = código da transição
Nome_trans = nome da transição
(transição fonte / modela eventos externos)

Lugares de Entrada

entsim(Cod_acao,Cod_trans,Cod_lugar)

Cod_acao = código da ação principal
Cod_trans = código da transição
Cod_lugar = código do lugar de entrada

Lugares de Saída

saisim(Cod_acao,Cod_trans,Cod_lugar)

Cod_acao = código da ação principal
Cod_trans = código da transição
Cod_lugar = código do lugar de saída

Todas as estruturas são *geradas automaticamente* objetivando atender os padrões de entrada do simulador.

CAPÍTULO IX

RESULTADOS EXPERIMENTAIS

9.1. INTRODUÇÃO

O trabalho prático da tese teve por objetivo fazer um estudo experimental sobre as potencialidades do ambiente de programação Prolog para o desenvolvimento de ferramentas de engenharia de software.

As metas previstas foram atingidas havendo a possibilidade futura de evoluir o trabalho experimental para aplicações práticas.

Tendo em vista o exposto acima, não foi gerada uma versão *compilada* do sistema o que requer um trabalho de depuração do software que extrapola o escopo da finalidade do trabalho

A titulo de demonstração dos resultados experimentais os programas escritos em Prolog foram agrupados em três conjuntos utilizando o *interpretador* do ambiente de programação Prolog.

A partir destes três conjuntos foram obtidos os resultados experimentais descritos nos itens a seguir:

9.2. RESULTADO EXPERIMENTAL 1 (GERAÇÃO AUTOMATICA DE REDES DE PETRI)

O primeiro conjunto gerado através do Interpretador Prolog permite obter os seguintes resultados:

- Descrever o módulo (M1) e as ações (A1 e A5) representados na Figura9.2.1. Esta descrição é feita através da "Interface de Especificação de Sistemas", que permite descrever os Elementos de Projeto e os Elementos Operacionais.

- Programar as ações A1 e A5, através dos correspondentes Fluxos de Controle e Comunicação.

- Gerar de forma automática a Rede de Petri do módulo M1 (Figura 9.2.2).

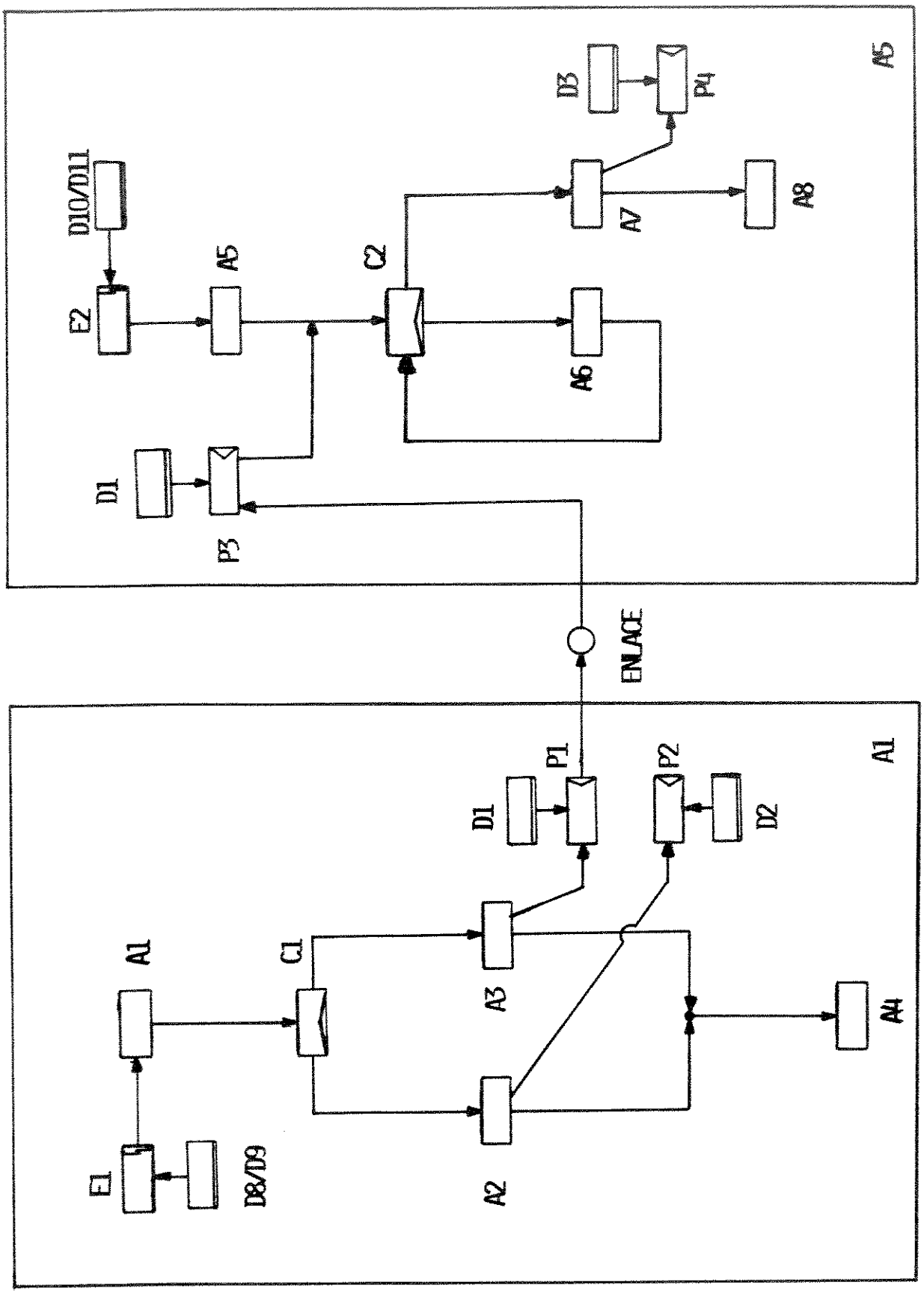


FIGURA 9.2.1 - REPRESENTAÇÃO GRÁFICA DO MÓDULO M1
AÇÕES A1 E A5

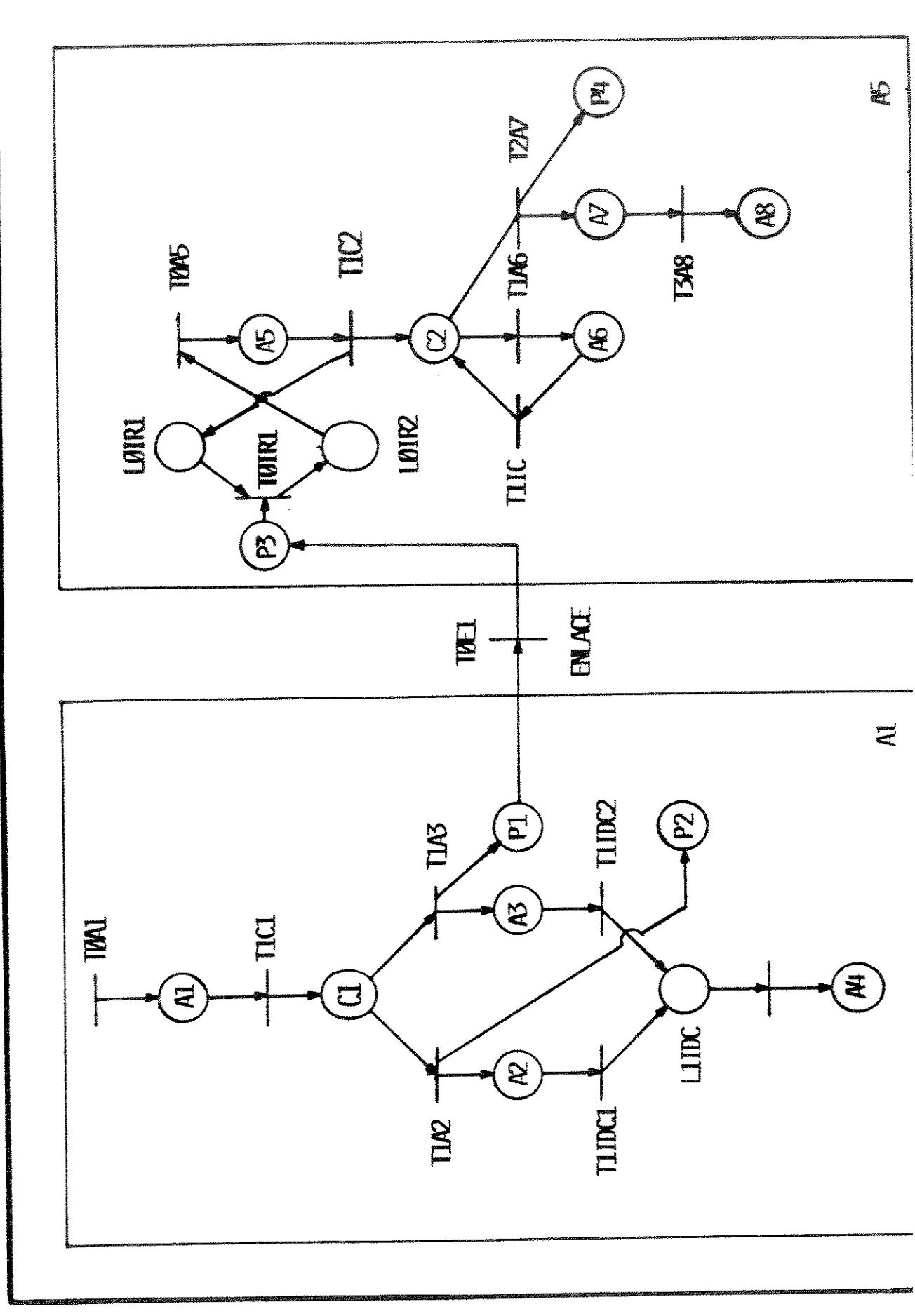


FIGURA 9.2.2 - REDE DE PETRI DO MÓDULO M1

```
**************************NOME SISTEMAS*****************************
CODIGOS SISTEMAS      NOMES SISTEMAS
S1                    Controle de Temperatura
********************************************************************

**************************NOME MODULOS******************************
CODIGOS MODULOS       NOMES MODULOS
M1                    Controle de Temperatura Forno 1
********************************************************************

**************************NOME ACOES********************************
CODIGOS ACOES         NOMES ACOES
A1                    Le Sensor Temperatura 1
A2                    Aciona Alarme 1
A3                    Processa Dados Sensor 1/Forno 1
A4                    Reinicializa Acao Le Sensor Temp 1
A5                    Ligar Forno 1
A6                    Aumentar Temperatura Forno 1
A7                    Gerar Relatorios 1
A8                    Reinicializa Acao Ligar Forno 1
********************************************************************

**************************NOME CONDICAO*****************************
CODIGO CONDICAO       NOME CONDICAO
C1                    Testa Dados Temperatura Sensor 1
C2                    Testa Dados Temperatura Ligar Forno 1
********************************************************************

**************************NOME PORTA********************************
CODIGO PORTA         NOME PORTA
P1                    Envia Mensagem - Ligar Forno 1
P2                    Envia Mensagem Desligar Forno 1
P3                    Recebe Mensagem Ligar Forno 1
P4                    Envia Relatorio 1 para Impressora
********************************************************************

**************************NOME DADO*********************************
CODIGO DADO         NOME DADO
D1                    Mensagem Ligar Forno 1
D2                    Mensagem Desligar Forno 1
D3                    Relatorio 1
********************************************************************
```

FIGURA 9.2.3 - ELEMENTOS DE PROJETO / ELEMENTOS OPERACIONAIS
DESCRIÇÃO DE NOMES

```
**********ESPECIFICACAO DA ACAO**********
CODIGO: A1
NOME: Le Sensor Temperatura 1
OBJETIVO: Le Dados Sensor Temperatura 1
DATA: 23-1-1989          HORA: 12:10
VERSAO: 00
**********************************************************


**********ESPECIFICACAO DA ACAO**********
CODIGO: A2
NOME: Aciona Alarme 1
OBJETIVO: Temperatura Forno 1 - Ultrapassou Limite
DATA: 23-1-1989          HORA: 12:11
VERSAO: 00
**********************************************************


**********ESPECIFICACAO DA ACAO**********
CODIGO: A3
NOME: Processa Dados Sensor 1/Forno 1
OBJETIVO: Processa Dados Lidos no Sensor 1/Forno 1
DATA: 23-1-1989          HORA: 12:12
VERSAO: 00
**********************************************************


**********ESPECIFICACAO DA ACAO**********
CODIGO: A4
NOME: Reinicializa Acao Le Sensor Temp 1
OBJETIVO: Reinicializa Leitura Sensor Temp 1-Forno 1
DATA: 23-1-1989          HORA: 12:12
VERSAO: 00
**********************************************************
```

FIGURA 9.2.4 – ESPECIFICAÇÃO DE AÇÕES
AÇÃO A1

```
**********ESPECIFICACAO DA ACAO**********
CODIGO: A5
NOME: Ligar Forno 1
OBJETIVO: Aciona Dispositivo de Potencia - Forno 1
DATA: 23-1-1989          HORA: 12:13
VERSAO: 00
**********************************************************


**********ESPECIFICACAO DA ACAO**********
CODIGO: A6
NOME: Aumentar Temperatura Forno 1
OBJETIVO: Aciona Controle para Aumentar Temperatura Forno 1
DATA: 23-1-1989          HORA: 12:14
VERSAO: 00
**********************************************************


**********ESPECIFICACAO DA ACAO**********
CODIGO: A7
NOME: Gerar Relatorios 1
OBJETIVO: Gera Relatorios de Aumento de Temperatura-Forno 1
DATA: 23-1-1989          HORA: 12:14
VERSAO: 00
**********************************************************


**********ESPECIFICACAO DA ACAO**********
CODIGO: A8
NOME: Reinicializa Acao Ligar Forno 1
OBJETIVO: Reinicializa Acao Ligar Forno 1
DATA: 23-1-1989          HORA: 12:15
VERSAO: 00
**********************************************************
```

FIGURA 9.2.5 - ESPECIFICAÇÃO DE AÇÕES
AÇÃO A5

```
**********ESPECIFICACAO DA CONDICAO**********
CODIGO: C1
NOME: Testa Dados Temperatura Sensor 1
OBJETIVO: Testa Dados Temperatura Sensor 1 com Valores Limite
DATA: 23-1-1989          HORA: 12:18
VERSAO: 00
**********************************************************


**********ESPECIFICACAO DA CONDICAO**********
CODIGO: C2
NOME: Testa Dados Temperatura Ligar Forno 1
OBJETIVO: Testa Dados Temp - Ligar Forno 1
DATA: 23-1-1989          HORA: 12:18
VERSAO: 00
**********************************************************

**********ESPECIFICACAO DA PORTA**********
CODIGO: P1
NOME: Envia Mensagem - Ligar Forno 1
OBJETIVO: Envia Mensagem para Ligar Forno 1
DATA: 23-1-1989          HORA: 12:19
VERSAO: 00
**********************************************************


**********ESPECIFICACAO DA PORTA**********
CODIGO: P2
NOME: Envia Mensagem Desligar Forno 1
OBJETIVO: Envia Mensagem para Desligar Forno 1
DATA: 23-1-1989          HORA: 12:20
VERSAO: 00
**********************************************************


**********ESPECIFICACAO DA PORTA**********
CODIGO: P3
NOME: Recebe Mensagem Ligar Forno 1
OBJETIVO: Recebe Mensagem para Ligar Forno 1
DATA: 23-1-1989          HORA: 12:20
VERSAO: 00
**********************************************************


**********ESPECIFICACAO DA PORTA**********
CODIGO: P4
NOME: Envia Relatorio 1 para Impressora
OBJETIVO: Envia Relatorio de Aumento de Temperatura
DATA: 23-1-1989          HORA: 12:21
VERSAO: 00
**********************************************************
```

FIGURA 9.2.6 - ESPECIFICAÇÃO DE CONDIÇÃO / PORTA
AÇÕES A1 E A5

```
**********ESPECIFICACAO DO DADO**********
CODIGO: D1
NOME: Mensagem Ligar Forno 1
OBJETIVO: Mensagem para Ligar Forno 1
DATA: 23-1-1989          HORA: 12:21
VERSAO: 00
VALOR: LIGAR
**************************************************


**********ESPECIFICACAO DO DADO**********
CODIGO: D2
NOME: Mensagem Desligar Forno 1
OBJETIVO: Mensagem Para Desligar Forno 1
DATA: 23-1-1989          HORA: 12:22
VERSAO: 00
VALOR: DESLIGAR
**************************************************


**********ESPECIFICACAO DO DADO**********
CODIGO: D3
NOME: Relatorio 1
OBJETIVO: Relatorio de Aumento de Temperatura
DATA: 23-1-1989          HORA: 12:22
VERSAO: 00
VALOR: REL F1
**************************************************
```

FIGURA 9.2.7 – ESPECIFICAÇÃO DE DADO
AÇÕES A1 E A5

```
**********FLUXO DE CONTROLE ACAO**********

CODIGO: A1      NOME: Le Sensor Temperatura 1

TIPO: SEQUENCIAL                                    NIVEL:0
CODIGO DA ACAO: A1

TIPO: DESVIO CONDICIONAL                            NIVEL:1
CODIGO CONDICAO: C1
CODIGO ACAO VERDADE: A2
CODIGO ACAO FALSO: A3

TIPO: SEQUENCIAL                                    NIVEL:2
CODIGO DA ACAO: A4

**********FLUXO DE CONTROLE ACAO**********

CODIGO: A5      NOME: Ligar Forno 1

TIPO: SEQUENCIAL                                    NIVEL:0
CODIGO DA ACAO: A5

TIPO: CICLICO                                       NIVEL:1
CODIGO CONDICAO: C2
CODIGO ACAO CICLICA: A6

TIPO: SEQUENCIAL                                    NIVEL:2
CODIGO DA ACAO: A7

TIPO: SEQUENCIAL                                    NIVEL:3
CODIGO DA ACAO: A8
```

FIGURA 9.2.8 - FLUXO DE CONTROLE
AÇÕES A1 E A5

```
**********COMUNICACAO ACAO**********

CODIGO: A1      NOME: Le Sensor Temperatura 1

TIPO: ENVIA                                          NIVEL:0
MENSAGEM ENVIADA: D1
ACAO QUE ENVIA A MENSAGEM: A3
PORTA DE SAIDA: P1

TIPO: ENVIA                                          NIVEL:1
MENSAGEM ENVIADA: D2
ACAO QUE ENVIA A MENSAGEM: A2
PORTA DE SAIDA: P2

**********COMUNICACAO ACAO**********

CODIGO: A5      NOME: Ligar Forno 1

TIPO: RECEBE                                         NIVEL:0
MENSAGEM RECEBIDA: D1
ACAO QUE RECEBE A MENSAGEM: A5
PORTAS DE ENTRADA: P3

TIPO: ENVIA                                          NIVEL:1
MENSAGEM ENVIADA: D3
ACAO QUE ENVIA A MENSAGEM: A7
PORTA DE SAIDA: P4

**********ENLACE MODULO**********
CODIGO: M1
NOME: Controle de Temperatura Forno 1

TIPO: ENLACE MODULO                          NIVEL:0
ACAO QUE ENVIA A MENSAGEM: A1
PORTA DE SAIDA: P1
ACAO QUE RECEBE A MENSAGEM: A5
PORTA DE ENTRADA: P3
```

FIGURA 9.2.9 - COMUNICAÇÃO / ENLACE
AÇÕES A1 E A5

```
**********REDE DE PETRI MODULO**********
CODIGO: M1
NOME: Controle de Temperatura Forno 1


**********REDE DE PETRI ACAO**********
CODIGO: A1          NOME: Le Sensor Temperatura 1

LUGARES FLUXO DE CONTROLE
A1          Le Sensor Temperatura 1
A2          Aciona Alarme 1
A3          Processa Dados Sensor 1/Forno 1
A4          Reinicializa Acao Le Sensor Temp 1
C1          Testa Dados Temperatura Sensor 1
L1IDC          Lugar Interno

TRANSICOES DE ENTRADA
A1          T0A1
C1          T1C1
A2          T1A2
L1IDC          T1IDC1
A3          T1A3
L1IDC          T1IDC2
A4          T2A4

TRANSICOES DE SAIDA
A2          T1IDC1
A3          T1IDC2
A1          T1C1
C1          T1A2
C1          T1A3
L1IDC          T2A4

LUGARES COMUNICACAO
P1          Envia Mensagem - Ligar Forno 1
P2          Envia Mensagem Desligar Forno 1

TRANSICOES DE ENTRADA
P1          T1A3
P2          T1A2

TRANSICOES DE SAIDA
P1          T0E1

**********
```

FIGURA 9.2.10 - REDE DE PETRI / MÓDULO M1
AÇÃO A1

```
**********REDE DE PETRI ACAO**********
CODIGO: A5          NOME: Ligar Forno 1

LUGARES FLUXO DE CONTROLE
A5          Ligar Forno 1
A6          Aumentar Temperatura Forno 1
A7          Gerar Relatorios 1
A8          Reinicializa Acao Ligar Forno 1
C2          Testa Dados Temperatura Ligar Forno 1

TRANSICOES DE ENTRADA
A5          T0A5
C2          T1C2
C2          T1IC
A6          T1A6
A7          T2A7
A8          T3A8

TRANSICOES DE SAIDA
A5          T1C2
C2          T1A6
A6          T1IC
C2          T2A7
A7          T3A8

LUGARES COMUNICACAO
P3          Recebe Mensagem Ligar Forno 1
P4          Envia Relatorio 1 para Imperssora
L0IR1          Lugar Interno 1
L0IR2          Lugar Interno 2

TRANSICOES DE ENTRADA
L0IR1          T1C2
L0IR2          T0IR1
P4          T2A7
P3          T0E1

TRANSICOES DE SAIDA
P3          T0IR1
L0IR1          T0IR1
L0IR2          T0A5

************************************************************
```

FIGURA 9.2.11 - REDE DE PETRI / MÓDULO M1
AÇÃO A5

9.3. RESULTADO EXPERIMENTAL 2 (GERAÇÃO AUTOMÁTICA DE REDES DE PETRI / ANÁLISE)

O segundo conjunto gerado através do Interpretador Prolog permite obter os seguintes resultados:

- Todos os resultados experimentais obtidos no Experimento 1.

- Inserir a descrição das ações que iniciam no estado de execução (marcação inicial) utilizando alguns módulos da "Interface de Descrição da Dinâmica de Sistemas".

- Gerar a base de dados e o arquivo de dados no formato do *Analisador Invarian* desenvolvido na USP/SP [BRES 86].

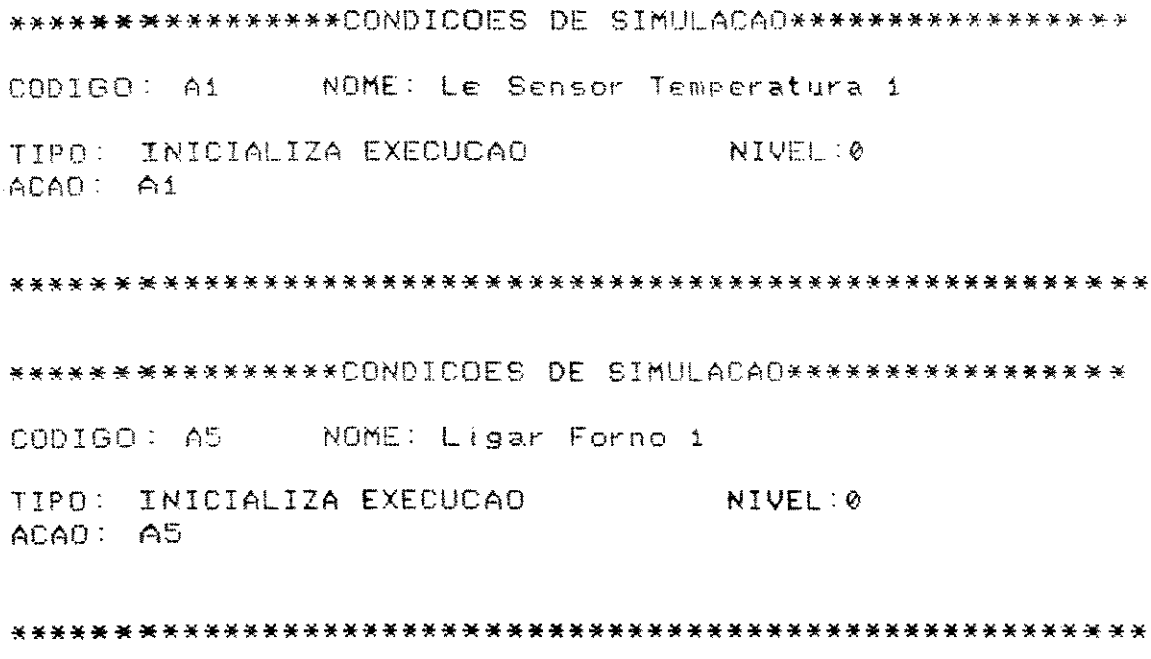


FIGURA 9.3.1 - MARCAÇÃO INICIAL
AÇÕES A1 E A5

```
**********REDE DE PETRI ACAO ANALISADOR**********
CODIGO :  A1          NOME : Le Sensor Temperatura 1

MARCACAO INICIAL:
P1= 1

LUGARES DE ENTRADA E DE SAIDA DAS TRANSICOES
T1
ENTRADAS  :
SAIDAS    :  P1
T2
ENTRADAS  :  P1
SAIDAS    :  P5
T3
ENTRADAS  :  P5
SAIDAS    :  P2   P8
T4
ENTRADAS  :  P2
SAIDAS    :  P6
T5
ENTRADAS  :  P5
SAIDAS    :  P3   P7
T6
ENTRADAS  :  P3
SAIDAS    :  P6
T7
ENTRADAS  :  P6
SAIDAS    :  P4


CONVERSAO DE CODIGOS RP  ANALISADOR-RP SISTEMA

T1/T0A1  T2/T1C1  T3/T1A2  T4/T1IDC1  T5/T1A3  T6/T1IDC2  T7/T2A4
P1/A1  P2/A2  P3/A3  P4/A4  P5/C1  P6/L1IDC  P7/P1  P8/P2
```

```
7  8
1  1  0
1  0  1  0
2  1  0  5  0
3  5  0  2  8  0
4  2  0  6  0
5  5  0  3  7  0
6  3  0  6  0
7  6  0  4  0
```

FIGURA 9.3.2 - REDE DE PETRI / AÇÃO A1
INTERFACE ANALISADOR INVARIAN

```
**********REDE DE PETRI ACAO ANALISADOR**********
CODIGO : A5          NOME : Ligar Forno 1

MARCACAO INICIAL:
P1= 1

LUGARES DE ENTRADA E DE SAIDA DAS TRANSICOES
T1
ENTRADAS :  P9
SAIDAS   :  P1
T2
ENTRADAS :  P1
SAIDAS   :  P5   P8
T3
ENTRADAS :  P2
SAIDAS   :  P5
T4
ENTRADAS :  P5
SAIDAS   :  P2
T5
ENTRADAS :  P5
SAIDAS   :  P3   P7
T6
ENTRADAS :  P3
SAIDAS   :  P4
T7
ENTRADAS :  P6   P8
SAIDAS   :  P9

CONVERSAO DE CODIGOS RP ANALISADOR-RP SISTEMA

T1/T0A5 T2/T1C2 T3/T1IC T4/T1A6 T5/T2A7 T6/T3A8 T7/T0IR1
P1/A5 P2/A6 P3/A7 P4/A8 P5/C2 P6/P3 P7/P4 P8/L0IR1 P9/L0IR2
```

```
7 9
1 1 0
1 9 0 1 0
2 1 0 5 8 0
3 2 0 5 0
4 5 0 2 0
5 5 0 3 7 0
6 3 0 4 0
7 6 8 0 9 0
```

FIGURA 9.3.3 - REDE DE PETRI / AÇÃO A5
INTERFACE ANALISADOR INVARIAN

```
DETERMINACAO DE INVARIANTES E IMPASSES DA REDE DE PETRI ana1.res

MARCACAO INICIAL      P 1=1

LUGARES DE ENTRADA E DE SAIDA DAS TRANSICOES

T  1 -  ENTRADAS
        SAIDAS      P 1
T  2 -  ENTRADAS    P 1
        SAIDAS      P 5
T  3 -  ENTRADAS    P 5
        SAIDAS      P 2 P 8
T  4 -  ENTRADAS    P 2
        SAIDAS      P 6
T  5 -  ENTRADAS    P 5
        SAIDAS      P 3 P 7
T  6 -  ENTRADAS    P 3
        SAIDAS      P 6
T  7 -  ENTRADAS    P 6
        SAIDAS      P 4


INVARIANTES DA REDE DE PETRI : Nao existem invariantes

POSSIVEIS IMPASSES DA REDE DE PETRI : Nao existem impasses
```

```
DETERMINACAO DE INVARIANTES E IMPASSES DA REDE DE PETRI ana2.res

MARCACAO INICIAL     P 1=1

LUGARES DE ENTRADA E DE SAIDA DAS TRANSICOES

T 1 -  ENTRADAS    P 9
       SAIDAS      P 1
T 2 -  ENTRADAS    P 1
       SAIDAS      P 5 P 8
T 3 -  ENTRADAS    P 2
       SAIDAS      P 5
T 4 -  ENTRADAS    P 5
       SAIDAS      P 2
T 5 -  ENTRADAS    P 5
       SAIDAS      P 3 P 7
T 6 -  ENTRADAS    P 3
       SAIDAS      P 4
T 7 -  ENTRADAS    P 6 P 8
       SAIDAS      P 9


INVARIANTES DA REDE DE PETRI

I 1 : (  P 1  P 2  P 3  P 4  P 5  P 6  P 9  )
I 2 : (  P 1  P 2  P 5  P 6  P 7  P 9  )
I 3 : (  P 1  P 8  P 9  )

POSSIVEIS IMPASSES DA REDE DE PETRI

IMPASSE   1

 M[ 1] = 0  M[ 2] = 0  M[ 3] = 0  M[ 4] = 1  M[ 5] = 0  M[ 6] = 0  M[ 7] = 1  M[ 8] = 1
 M[ 9] = 0
```

FIGURA 9.3.4 - REDE DE PETRI, AÇÕES A1 E A5
RESULTADO DE ANÁLISE – PROGRAMA INVARIAN

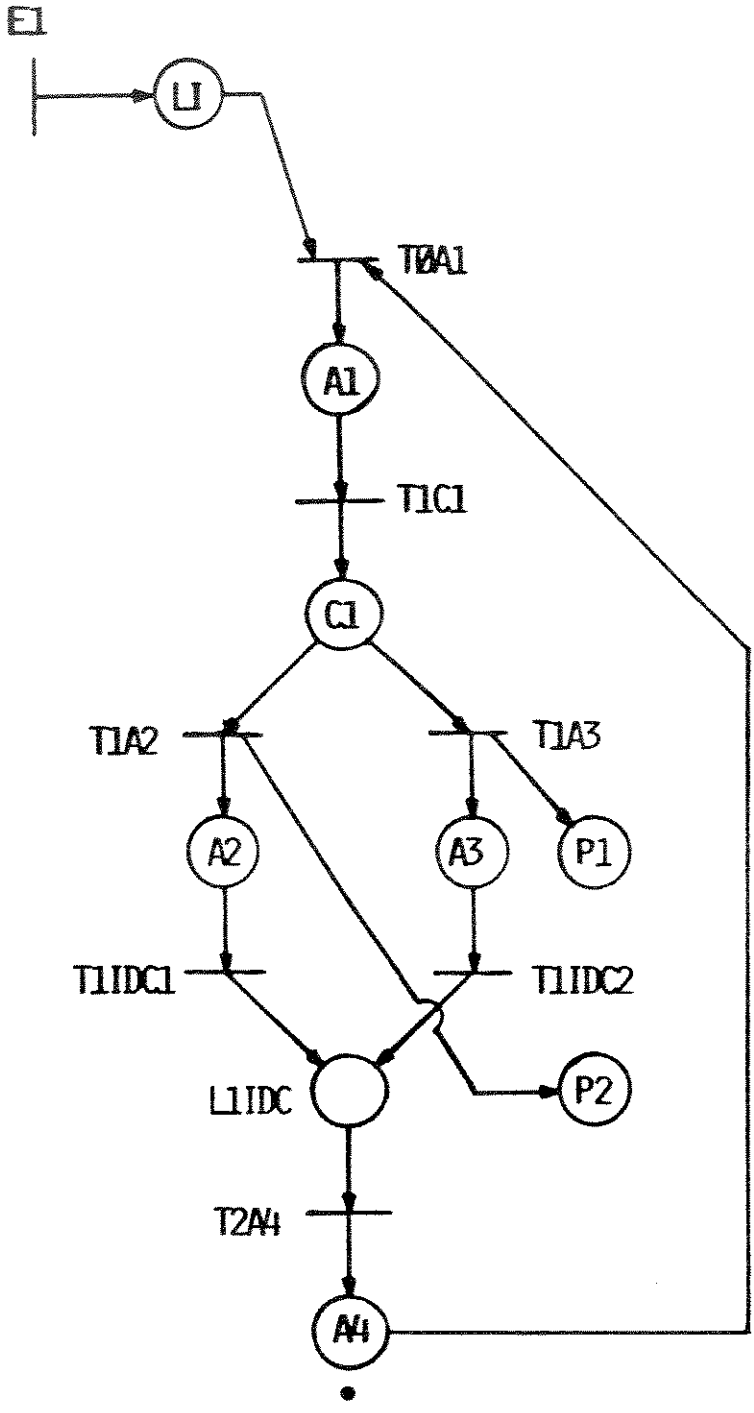
9.4. RESULTADO EXPERIMENTAL 3 (GERAÇÃO AUTOMÁTICA DE REDES DE PETRI / SIMULAÇÃO)

O terceiro conjunto gerado através do Interpretador Prolog permite obter os seguintes resultados:

- Todos os resultados experimentais obtidos no Experimento 1.

- Inserir a descrição das características dinâmicas de uma ação, utilizando a "Interface de Descrição da Dinâmica de Sistemas".

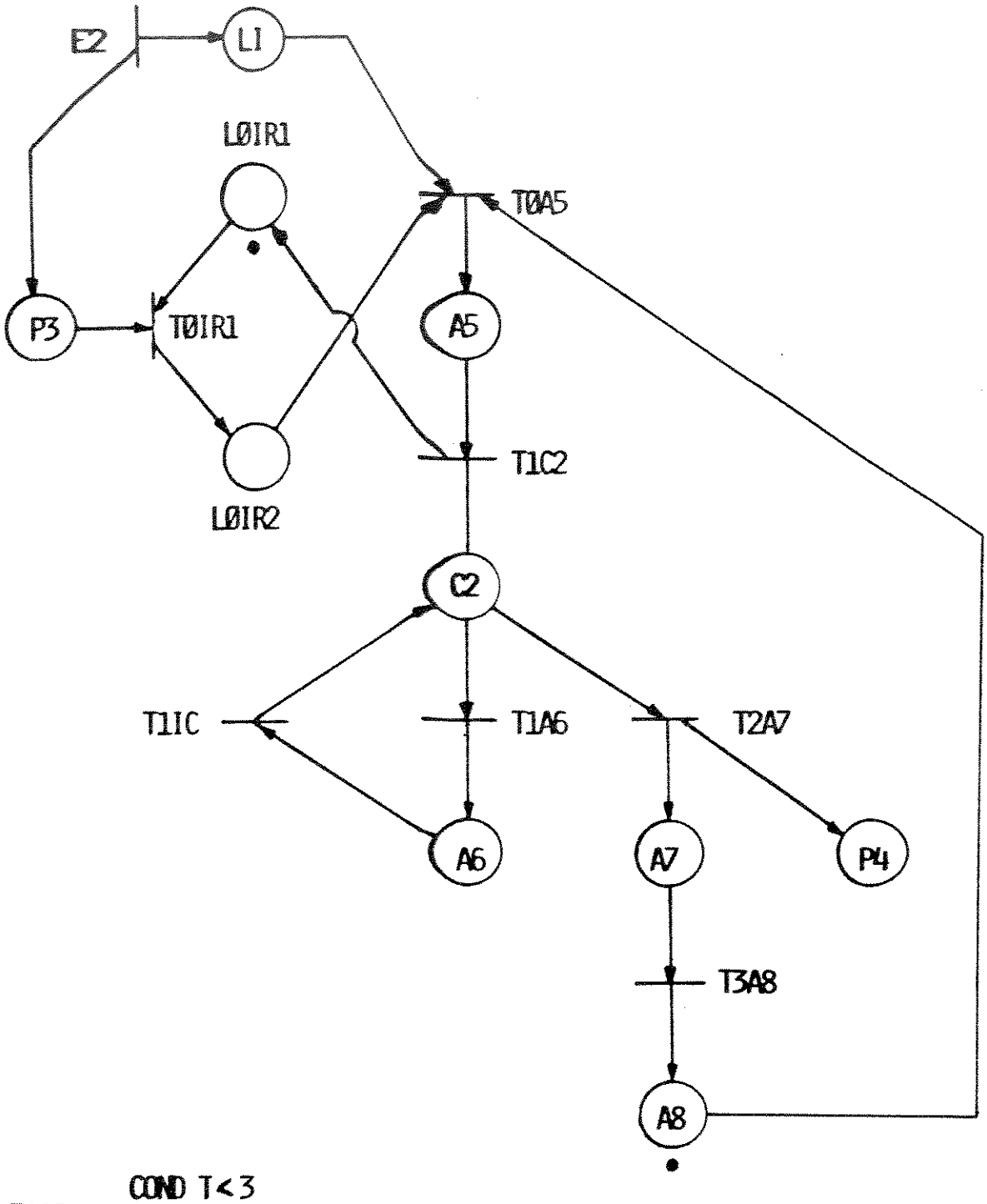
- Gerar a base de dados e o arquivo de dados no formato do *Simulador PPP* desenvolvido no LAAS/Toulouse/França [VALE 84b].



T1A2 COND T>3

T1A3 COND T<=3

FIGURA 9.4.1 - REDE DE PETRI / AÇÃO A1
SIMULAÇÃO



T1A6 COND T<3
ACTION T:=T+1

T2A7 COND T:=3

FIGURA 9.4.2 - REDE DE PETRI / AÇÃO A5
SIMULAÇÃO

```
**********ESPECIFICACAO DO DADO**********
CODIGO: D4
NOME: Inicializa Simulacao
OBJETIVO: Inicio de Simulacao - Acoes A1 e A5
DATA: 24-1-1989          HORA: 13:43
VERSAO: 00
VALOR: 0
**********************************************************


**********ESPECIFICACAO DO DADO**********
CODIGO: D5
NOME: Termina Simulacao
OBJETIVO: Tempo de Termino de Simulacao - Acoes A1 e A5
DATA: 24-1-1989          HORA: 13:44
VERSAO: 00
VALOR: 20
**********************************************************


**********ESPECIFICACAO DO DADO**********
CODIGO: D6
NOME: Inicializa Observacao
OBJETIVO: Inicio de Relatorios Estatisticos - Acoes A1 e A5
DATA: 24-1-1989          HORA: 13:45
VERSAO: 00
VALOR: 0
**********************************************************


**********ESPECIFICACAO DO DADO**********
CODIGO: D7
NOME: Intervalo de Observacao
OBJETIVO: Intervalo de Relatorios Estatisticos - Acoes A1 e A5
DATA: 24-1-1989          HORA: 13:47
VERSAO: 00
VALOR: 1
**********************************************************
```

FIGURA 9.4.3 - DESCRIÇÃO DE DADOS / AÇÕES A1 E A5
SIMULAÇÃO

```
**********ESPECIFICACAO DO DADO**********
CODIGO: D8
NOME: Tempo Primeira Ativacao
OBJETIVO: Tempo que ocorre primeira ativacao - Acao 1
DATA: 24-1-1989          HORA: 13:48
VERSAO: 00
VALOR: 4
**********************************************************


**********ESPECIFICACAO DO DADO**********
CODIGO: D9
NOME: Intervalo de Repeticao Ciclico
OBJETIVO: Ciclo de Ativacao - Acao 1
DATA: 24-1-1989          HORA: 13:48
VERSAO: 00
VALOR: 8
**********************************************************


**********ESPECIFICACAO DO DADO**********
CODIGO: D10
NOME: Intervalo de Tempo de Interrupcao
OBJETIVO: Intervalo de Interrupcao - Acao 5
DATA: 24-1-1989          HORA: 13:49
VERSAO: 00
VALOR: (1,5)
**********************************************************


**********ESPECIFICACAO DO DADO**********
CODIGO: D11
NOME: Intervalo Repeticao Aleatorio
OBJETIVO: Ciclo de Ativacao - Acao 5
DATA: 24-1-1989          HORA: 13:50
VERSAO: 00
VALOR: 5
**********************************************************
```

FIGURA 9.4.4 - DESCRIÇÃO DE DADOS / AÇÕES A1 E A5
SIMULAÇÃO

```
**********ESPECIFICACAO DO EVENTO**********
CODIGO: E1
NOME: Ativa Acao 1 - Ciclica
OBJETIVO: Ativar Acao 1 em Intervalos Ciclicos
DATA: 24-1-1989          HORA: 13:57
VERSAO: 00
**********************************************************
```

```
**********ESPECIFICACAO DO EVENTO**********
CODIGO: E2
NOME: Ativa Acao 5 - Interrupcao
OBJETIVO: Ativa a Acao A5 atraves de Interrupcao
DATA: 24-1-1989          HORA: 13:58
VERSAO: 00
**********************************************************
```

FIGURA 9.4.5 - DESCRIÇÃO DE EVENTOS / AÇÕES A1 E A5
SIMULAÇÃO

```
*****************ESPECIFICACAO DE DINAMICA*****************

CODIGO:  A1      NOME: Le Sensor Temperatura 1

TIPO: ATIVACAO CICLICA                          NIVEL:0
ACAO:  A1
EVENTO:  E1
TEMPO DA PRIMEIRA ATIVACAO: 4
INTERVALO REPETICAO ATIVACAO: 8

************************************************************



*****************ESPECIFICACAO DE DINAMICA*****************

CODIGO:  A5      NOME: Ligar Forno 1

TIPO: ATIVACAO POR INTERRUPCAO                  NIVEL:0
ACAO:  A5
EVENTO:  E2
INTERVALO DA PRIMEIRA ATIVACAO: (1,5)
INTERVALO ATIVACAO ALEATORIA: 5

************************************************************
```

FIGURA 9.4.6 - ESPECIFICAÇÃO DA DINÂMICA - AÇÕES A1 E A5
SIMULAÇÃO

```
**********INTERPRETACAO DA DINAMICA**********

CODIGO: A1      NOME: Le Sensor Temperatura 1

TIPO: SOLUCAO DE CONFLITO                          NIVEL:0
PORTA ENVIA MENSAGEM: C1
PORTA RECEBE MENSAGEM: A2
TIPO INTERPRETACAO: COND
TIPO DA VARIAVEL: ENTIER
NOME VARIAVEL: T
EXPRESSAO LOGICA/ALGEBRICA: T>3

TIPO: SOLUCAO DE CONFLITO                          NIVEL:1
PORTA ENVIA MENSAGEM: C1
PORTA RECEBE MENSAGEM: A3
TIPO INTERPRETACAO: COND
TIPO DA VARIAVEL: ENTIER
NOME VARIAVEL: T
EXPRESSAO LOGICA/ALGEBRICA: T<=3

**********INTERPRETACAO DA DINAMICA**********

CODIGO: A5      NOME: Ligar Forno 1

TIPO: SOLUCAO DE CONFLITO                          NIVEL:0
PORTA ENVIA MENSAGEM: C2
PORTA RECEBE MENSAGEM: A6
TIPO INTERPRETACAO: ACTION
TIPO DA VARIAVEL: ENTIER
NOME VARIAVEL: T
EXPRESSAO LOGICA/ALGEBRICA: T:=T+1

TIPO: SOLUCAO DE CONFLITO                          NIVEL:1
PORTA ENVIA MENSAGEM: C2
PORTA RECEBE MENSAGEM: A6
TIPO INTERPRETACAO: COND
TIPO DA VARIAVEL: ENTIER
NOME VARIAVEL: T
EXPRESSAO LOGICA/ALGEBRICA: T< 3

TIPO: SOLUCAO DE CONFLITO                          NIVEL:2
PORTA ENVIA MENSAGEM: C2
PORTA RECEBE MENSAGEM: A7
TIPO INTERPRETACAO: COND
TIPO DA VARIAVEL: ENTIER
NOME VARIAVEL: T
EXPRESSAO LOGICA/ALGEBRICA: T=3

**********
```

FIGURA 9.4.7 - INTERPRETAÇÃO DA DINÂMICA / AÇÕES A1 E A5
SIMULAÇÃO

```
**********CONDICOES DE SIMULACAO**********

CODIGO: A1      NOME: Le Sensor Temperatura 1

TIPO: TEMPO DE SIMULACAO                NIVEL:0
TEMPO DE INICIO DA SIMULACAO: 0
TEMPO DE FIM DA SIMULACAO: 20

TIPO: TEMPO DE OBSERVACAO               NIVEL:1
TEMPO DE INICIO DA OBSERVACAO: 0
INTERVALO DE OBSERVACAO: 1

TIPO: INICIALIZA EXECUCAO               NIVEL:2
ACAO: A4

**********CONDICOES DE SIMULACAO**********

CODIGO: A5      NOME: Ligar Forno 1

TIPO: TEMPO DE SIMULACAO                NIVEL:0
TEMPO DE INICIO DA SIMULACAO: 0
TEMPO DE FIM DA SIMULACAO: 20

TIPO: TEMPO DE OBSERVACAO               NIVEL:1
TEMPO DE INICIO DA OBSERVACAO: 0
INTERVALO DE OBSERVACAO: 1

TIPO: INICIALIZA EXECUCAO               NIVEL:2
ACAO: A8

**********
```

FIGURA 9.4.8 - CONDIÇÕES DE SIMULAÇÃO / AÇÕES A1 E A5
SIMULAÇÃO

```
(Descricao Rede de Petri de uma ACAO - Simulador )

(Acao principal: Le Sensor Temperatura 1      Codigo: A1)

NOEUDS
      A_1 : PLACE;       (Le Sensor Temperatura 1)
      A_2 : PLACE;       (Aciona Alarme 1)
      A_3 : PLACE;       (Processa Dados Sensor 1/Forno 1)
      A_4 : PLACE;       (Reinicializa Acao Le Sensor Temp 1)
      C_1 : PLACE;       (Testa Dados Temperatura Sensor 1)
      L_1IDC : PLACE;       (Lugar Interno)
      P_1 : PLACE;       (Envia Mensagem - Ligar Forno 1)
      P_2 : PLACE;       (Envia Mensagem Desligar Forno 1)
      L_I : PLACE;       (Lugar Interno para Simulacao)

      T_0A1 : TRANSITION;
      T_1C1 : TRANSITION;
      T_1A2 : TRANSITION;
      T_1IDC1 : TRANSITION;
      T_1A3 : TRANSITION;
      T_1IDC2 : TRANSITION;
      T_2A4 : TRANSITION;
      E_1 : TRANSITION;                (Transicao Fonte)


ARCS
      ENTREE T_1IDC1 = A_2;
      ENTREE T_1IDC2 = A_3;
      ENTREE T_1C1 = A_1;
      ENTREE T_1A2 = C_1;
      ENTREE T_1A3 = C_1;
      ENTREE T_2A4 = L_1IDC;
      ENTREE T_0A1 = L_I,A_4;

      SORTIE T_0A1 = A_1;
      SORTIE T_1C1 = C_1;
      SORTIE T_1A2 = A_2,P_2;
      SORTIE T_1IDC1 = L_1IDC;
      SORTIE T_1A3 = A_3,P_1;
      SORTIE T_1IDC2 = L_1IDC;
      SORTIE T_2A4 = A_4;
      SORTIE E_1 = L_I;


VAR
      T:ENTIER;

INTERPRETATION
      COND T_1A2 = T>3;
      COND T_1A3 = T<=3;

FIN.
```

FIGURA 9.4.9 - ARQUIVO DE ENTRADA SIMULADOR PSI
AÇÃO A1 - DESCRIÇÃO DA REDE DE PETRI

```
{Descricao das Condicoes de Simulacao de uma ACAO-Simulador }

{Acao principal: Le Sensor Temperatura 1      Codigo: A1}

PLACE
     { Acoes que Iniciam em Estado de Execucao }
     MARQUAGE A_4:=1;

ECHEANCIER
     { Tempos de Execucao da Simulacao }
     STATISTIQUE DEBUT 0;
     STATISTIQUE FIN 20;

     { Tempo de Observacao da Simulacao }
     OBSERVER 0*1;


     ACTION
          E_1::TEMPS (4)*8;


FIN.
```

FIGURA 9.4.10 - ARQUIVO DE ENTRADA SIMULADOR PSI
AÇÃO A1 - DESCRIÇÃO DAS CONDIÇÕES DE SIMULAÇÃO

```
TEMPS INITIAL :      0
TEMPS FINAL   :     20

TRANSITIONS :
              T_0A1    3 TIRS, TRANSIT :     6.667
              T_1C1    3 TIRS, TRANSIT :     6.667
              T_1A2    0 TIRS, TRANSIT :     0.000
              T_1IDC1    0 TIRS, TRANSIT :     0.000
              T_1A3    3 TIRS, TRANSIT :     6.667
              T_1IDC2    3 TIRS, TRANSIT :     6.667
              T_2A4    3 TIRS, TRANSIT :     6.667
              E_1    3 TIRS, TRANSIT :     6.667

PLACES :
         A_1 SANS STATISTIQUES
         A_2 SANS STATISTIQUES
         A_3 SANS STATISTIQUES
         A_4 SANS STATISTIQUES
         C_1 SANS STATISTIQUES
         L_1IDC SANS STATISTIQUES
         P_1 SANS STATISTIQUES
         P_2 SANS STATISTIQUES
         L_I SANS STATISTIQUES
```

FIGURA 9.4.11 - RESULTADO SIMULADOR PSI
AÇÃO A1

```
TEMPO    4  TIRO TRANSICAO E_1
     TIROS DESDE ESTADO ESTAVEL :    1
TEMPO    4  TIRO TRANSICAO T_0A1
     PL.DEC. L_I   1
     TIROS DESDE ESTADO ESTAVEL :    1
TEMPO    4  TIRO TRANSICAO T_1C1
     PL.DEC. A_1   1
     TIROS DESDE ESTADO ESTAVEL :    2
TEMPO    4  TIRO TRANSICAO T_1A3
     PL.DEC. C_1   1
     TIROS DESDE ESTADO ESTAVEL :    3
TEMPO    4  TIRO TRANSICAO T_1IDC2
     PL.DEC. A_3   1
     TIROS DESDE ESTADO ESTAVEL :    4
TEMPO    4  TIRO TRANSICAO T_2A4
     PL.DEC. L_1IDC  1
     TIROS DESDE ESTADO ESTAVEL :    5
TEMPO   12  TIRO TRANSICAO E_1
     TIROS DESDE ESTADO ESTAVEL :    1
TEMPO   12  TIRO TRANSICAO T_0A1
     PL.DEC. L_I   1
     TIROS DESDE ESTADO ESTAVEL :    1
TEMPO   12  TIRO TRANSICAO T_1C1
     PL.DEC. A_1   1
     TIROS DESDE ESTADO ESTAVEL :    2
TEMPO   12  TIRO TRANSICAO T_1A3
     PL.DEC. C_1   1
     TIROS DESDE ESTADO ESTAVEL :    3
TEMPO   12  TIRO TRANSICAO T_1IDC2
     PL.DEC. A_3   1
     TIROS DESDE ESTADO ESTAVEL :    4
TEMPO   12  TIRO TRANSICAO T_2A4
     PL.DEC. L_1IDC  1
     TIROS DESDE ESTADO ESTAVEL :    5
TEMPO   20  TIRO TRANSICAO E_1
     TIROS DESDE ESTADO ESTAVEL :    1
TEMPO   20  TIRO TRANSICAO T_0A1
     PL.DEC. L_I   1
     TIROS DESDE ESTADO ESTAVEL :    1
TEMPO   20  TIRO TRANSICAO T_1C1
     PL.DEC. A_1   1
     TIROS DESDE ESTADO ESTAVEL :    2
TEMPO   20  TIRO TRANSICAO T_1A3
     PL.DEC. C_1   1
     TIROS DESDE ESTADO ESTAVEL :    3
TEMPO   20  TIRO TRANSICAO T_1IDC2
     PL.DEC. A_3   1
     TIROS DESDE ESTADO ESTAVEL :    4
TEMPO   20  TIRO TRANSICAO T_2A4
     PL.DEC. L_1IDC  1
     TIROS DESDE ESTADO ESTAVEL :    5
```

FIGURA 9.4.12 - RESULTADO SIMULADOR PSI
AÇÃO A1

```
(Descricao Rede de Petri de uma ACAO - Simulador )

(Acao principal: Ligar Forno 1      Codigo: A5)

NOEUDS
      A_5  : PLACE;       (Ligar Forno 1)
      A_6  : PLACE;       (Aumentar Temperatura Forno 1)
      A_7  : PLACE;       (Gerar Relatorios 1)
      A_8  : PLACE;       (Reinicializa Acao Ligar Forno 1)
      C_2  : PLACE;       (Testa Dados Temperatura Ligar Forno 1)
      P_3  : PLACE;       (Recebe Mensagem - Ligar Forno 1)
      P_4  : PLACE;       (Envia Relatorio 1 para Impressora)
      L_0IR1 : PLACE;        (Lugar Interno 1)
      L_0IR2 : PLACE;        (Lugar Interno 2)
      L_I  : PLACE;       (Lugar Interno Simulacao)

      T_0A5 : TRANSITION;
      T_1C2 : TRANSITION;
      T_1IC : TRANSITION;
      T_1A6 : TRANSITION;
      T_2A7 : TRANSITION;
      T_3A8 : TRANSITION;
      T_0IR1 : TRANSITION;
      E_2  : TRANSITION;     (Transicao Interna Simulacao)


ARCS
      ENTREE T_1C2 = A_5;
      ENTREE T_1A6 = C_2;
      ENTREE T_1IC = A_6;
      ENTREE T_2A7 = C_2;
      ENTREE T_3A8 = A_7;
      ENTREE T_0IR1 = P_3,L_0IR1;
      ENTREE T_0A5 = L_0IR2,L_I,A_8;

      SORTIE T_0A5 = A_5;
      SORTIE T_1C2 = C_2,L_0IR1;
      SORTIE T_1IC = C_2;
      SORTIE T_1A6 = A_6;
      SORTIE T_2A7 = A_7,P_4;
      SORTIE T_3A8 = A_8;
      SORTIE T_0IR1 = L_0IR2;
      SORTIE E_2 = L_I,P_3;


VAR
      T:ENTIER;


INTERPRETATION
      ACTION T_1A6 = T:=T+1;
      COND T_1A6 = T<3;
      COND T_2A7 = T=3;

FIN.
```

FIGURA 9.4.13 - ARQUIVO DE ENTRADA SIMULADOR PSI
AÇÃO A5 - DESCRIÇÃO DA REDE DE PETRI

```
{Descricao das Condicoes de Simulacao de uma ACAO-Simulador }

{Acao principal: Ligar Forno 1      Codigo: A5}

PLACE
      { Acoes que Iniciam em Estado de Execucao }
      MARQUAGE A_8:=1;
      MARQUAGE L_0IR1:=1;


ECHEANCIER
      { Tempos de Execucao da Simulacao }
      STATISTIQUE DEBUT 0;
      STATISTIQUE FIN 20;

      { Tempo de Observacao da Simulacao }
      OBSERVER 0*1;

      ALEATOIRE
            E_2::TEMPS (1,5)*5;


FIN.
```

FIGURA 9.4.14 - ARQUIVO DE ENTRADA SIMULADOR PSI
AÇÃO A5 - DESCRIÇÃO DAS CONDIÇÕES DE SIMULAÇÃO

```
Temps initial :      0
Temps final   :     20

Transitions :
              T_0A5    4 tirs, transit :     5.000
              T_1C2    4 tirs, transit :     5.000
              T_1IC    3 tirs, transit :     6.667
              T_1A6    3 tirs, transit :     6.667
              T_2A7    4 tirs, transit :     5.000
              T_3A8    4 tirs, transit :     5.000
              T_0IR1    4 tirs, transit :     5.000
              E_2    4 tirs, transit :     5.000

Temps: temps associe a la place
Mamax: nb maxi de jetons
Tmd: temps maxi de sejour
Nfm: nb fois place demarquee

Places :
          A_5 sans statistiques
          A_6 sans statistiques
          A_7 sans statistiques
          A_8 sans statistiques
          C_2 sans statistiques
          P_3 sans statistiques
          P_4 sans statistiques
          L_0IR1 sans statistiques
          L_0IR2 sans statistiques
          L_I sans statistiques

 FIN DE STATISTIQUES.
```

FIGURA 9.4.15 - RESULTADO SIMULADOR PSI
AÇÃO A5

```
Programme de simulation sim v:SUN/01/87  laas/pastels
Fichier de stockage du reseau traduit en tableau: sim.tab
Descricao Rede de Petri de uma ACAO - Simulador ***********
**
Reseau
 10 places     8 transitions   0 var b   1 var e
Dim. vecteurs    13    9   80
Fichier conditions de simulations traduit en tableau: sim.ech
Temps       0  trace? a,d,e,m,s,t,z,*
t
Trace?   a,d,e,m,s,t,z,*

Temps       0  trace? a,d,e,m,s,t,z,*

Temps       1  trace? a,d,e,m,s,t,z,*

Temps    2  tir transition E_2
     tirs depuis etat stable :    1
*

Temps    2  tir transition T_0IR1
     pl.dec. P_3  1
     tirs depuis etat stable :    1
*

Temps    2  tir transition T_0IR1
     pl.dec. P_3  1
     tirs depuis etat stable :    1
*

Temps    2  tir transition T_0A5
     pl.dec. L_0IR2  1
     tirs depuis etat stable :    2
*

Temps    2  tir transition T_1C2
     pl.dec. A_5  1
     tirs depuis etat stable :    3
*

Temps    2  tir transition T_1A6
     pl.dec. C_2  1
     tirs depuis etat stable :    4
*

Temps    2  tir transition T_1IC
     pl.dec. A_6  1
     tirs depuis etat stable :    5
*
```

FIGURA 9.4.16 - RESULTADO SIMULADOR PSI
AÇÃO A5

```
Temps    2  tir transition T_1A6
     pl.dec. C_2  1
     tirs depuis etat stable :    6
*

Temps    2  tir transition T_1IC
     pl.dec. A_6  1
     tirs depuis etat stable :    7
*

Temps    2  tir transition T_1A6
     pl.dec. C_2  1
     tirs depuis etat stable :    8
*

Temps    2  tir transition T_1IC
     pl.dec. A_6  1
     tirs depuis etat stable :    9
*

Temps    2  tir transition T_2A7
     pl.dec. C_2  1
     tirs depuis etat stable :   10
*

Temps    2  tir transition T_3A8
     pl.dec. A_7  1
     tirs depuis etat stable :   11
*
```

FIGURA 9.4.IV - RESULTADO SIMULADOR PSI
AÇÃO A5

```
Temps  10  tir transition E_2
     tirs depuis etat stable :    1
*

Temps  10  tir transition T_0IR1
     pl.dec. P_3  1
     tirs depuis etat stable :    1
*

Temps  10  tir transition T_0A5
     pl.dec. L_0IR2  1
     tirs depuis etat stable :    2
*

Temps  10  tir transition T_1C2
     pl.dec. A_5  1
     tirs depuis etat stable :    3
*

Temps  10  tir transition T_2A7
     pl.dec. C_2  1
     tirs depuis etat stable :    4
*

Temps  10  tir transition T_3A8
     pl.dec. A_7  1
     tirs depuis etat stable :    5
*
```

FIGURA 9.4.18 - RESULTADO SIMULADOR PSI
AÇÃO A5

```
Temps  11  tir transition E_2
     tirs depuis etat stable :    1
*

Temps  11  tir transition T_0IR1
     pl.dec. P_3  1
     tirs depuis etat stable :    1
*

     tirs depuis etat stable :    1
*

Temps  11  tir transition T_0A5
     pl.dec. L_0IR2  1
     tirs depuis etat stable :    2
*

Temps  11  tir transition T_1C2
     pl.dec. A_5  1
     tirs depuis etat stable :    3
*

Temps  11  tir transition T_2A7
     pl.dec. C_2  1
     tirs depuis etat stable :    4
*

Temps  11  tir transition T_3A8
     pl.dec. A_7  1
     tirs depuis etat stable :    5
*
```

FIGURA 9.4.19 - RESULTADO SIMULADOR PSI
AÇÃO A5

```
Temps  20  tir transition E_2
     tirs depuis etat stable :    1
*

Temps  20  tir transition T_0IR1
     pl.dec. P_3  1
     tirs depuis etat stable :    1
*

Temps  20  tir transition T_0A5
     pl.dec. L_0IR2  1
     tirs depuis etat stable :    2
*

Temps  20  tir transition T_1C2
     pl.dec. A_5  1
     tirs depuis etat stable :    3
*

Temps  20  tir transition T_2A7
     pl.dec. C_2  1
     tirs depuis etat stable :    4
*


Temps  20  tir transition T_1C2
     pl.dec. A_5  1
     tirs depuis etat stable :    3
*

Temps  20  tir transition T_2A7
     pl.dec. C_2  1
     tirs depuis etat stable :    4
*

Temps  20  tir transition T_3A8
     pl.dec. A_7  1
     tirs depuis etat stable :    5
*

Temps    20  trace? a,d,e,m,s,t,z,*

Fin de simulation
Fichier de listage des statistiques ?
```

FIGURA 9.4.20 - RESULTADO SIMULADOR PSI
AÇÃO A5

CAPITULO X

CONCLUSÕES FINAIS

Este trabalho teve como origem uma proposta de gerar de forma automática uma Rede de Petri com o intuito de validar a especificação de um Sistema de Software com Características Tempo Real.

Esta especificação a princípio deveria ser gerada por um *Sistema de Suporte* ao desenvolvimento de Software Tempo Real e em seguida traduzida para uma base de dados em Prolog.

No ambiente Prolog esta especificação seria traduzida de forma automática em uma Rede de Petri e gerados arquivos de entrada para as ferramentas de Análise e Simulação de Redes de Petri.

Durante a fase de implementação o trabalho evoluiu para a implementação das *Interfaces de Especificação de Sistemas e Descrição da Dinâmica do Sistema*, cujo objetivo é especificar um Sistema de Software com Caracetrísticas Tempo Real, utilizando recursos interativos. Estas interfaces permitem estruturar o conjunto de bases de dados que modelam o sistema e finalmente gerar documentos textuais.

Convém ressaltar novamente aqui, que os conceitos utilizados na implementação destas interfaces foram baseados na *Linguagem de Especificação SPOS-S* - [BIEW 80], [LAUB 81]; e no tópico de *Definição* da Tese de Doutorado "*Ambiente de Programação Distribuída: Definição, Análise e Síntese*" [BRES 85].

A estrutura gerada na base de dados, permite a partir do desenvolvimento de tradutores a interligação com "Sistemas de Desenvolvimento de Software" permitindo visualizar este trabalho como uma *Ferramenta para Geração Automática de Redes de Petri a partir da especificação de um Sistema de Software com características Tempo Real.*

A implementação das *Interfaces de Especificação e Descrição da Dinâmica* abriu a perspectiva de evolução deste trabalho para uma *Ferramenta de Especificação e Validação de Software com*

Características Tempo Real utilizando o ambiente de programação Prolog.

Neste contexto, há a possibilidade de desenvolvimento imediato dos seguintes trabalhos de continuidade:

- Implementação de Interface Gráfica

  A implementação de uma interface gráfica permitiria uma entrada de dados mais eficiente, através de um diagrama de Fluxo de Controle e Dados, bem como permitiria a geração dos seguintes documentos gráficos:

  . Diagramas de Fluxo de Controle e Dados (Entrada e Saída)

  . Gráficos de Rede de Petri (Saída)

  A implementação da interface gráfica requer a mudança de apenas uma das estruturas das interfaces que é voltada para o usuário. A outra estrutura das interfaces, que é voltada para o gerenciamento das bases de dados continuaria sendo utilizada.

- Análise de Erro

  A análise de erro consistiria em dotar o sistema de ferramentas capazes de verificar a compatibilidade das informações geradas de forma textual e/ou gráfica, evitando redundâncias, verificando completeza, descrições não referenciadas, bem como outras análises a serem definidas posteriormente.

Também dentro deste contexto, há a possibilidade de desenvolvimento a médio prazo do seguinte trabalho de continuidade:

- Implementação de Sistema Perito (Técnicas de IA)

A *geração automática* de Redes de Petri a partir da especificação de um Sistema de Software Tempo Real, utiliza já neste trabalho um conjunto de regras que traduz um conjunto de conhecimentos, consistindo em uma contribuição no sentido de automatizar a interligação de diferentes ferramentas de desenvolvimento de um projeto de software.

Tomando este fato como ponto de partida, associado aos recursos oferecidos pelo ambiente de programação Prolog para estruturar e gerenciar base de dados, abre-se a perspectiva de implantação futura de um *Sistema Perito para Desenvolvimento de Software Tempo Real* baseado em conjuntos de regras que utilizem conhecimento.

REFERENCIAS BIBLIOGRÁFICAS

[BIEW 79] BIEWALD, J.; GOEHNER, P.; LAUBER, R.: "EPOS - *A Specification and Design Technique for Computer Controlled Real-Time Automation Systems*". 4th International Conference on Software Engineering, Munich, September 1979.

[BIEW 80] BIEWALD, J.; GOEHNER, P.; SCHELLING, H.: "*Real-Time Features of EPOS: Formulation Evaluation and Documentation.* IFAC/IFAP Workshop on Real-Time Programming. Leibnitz, Austria, April 1980.

[BRES 85] BRESSAN, GRAÇA: "*Ambiente de Programação Distribuída: Definição, Análise e Síntese*". Tese de Doutorado apresentada à EPUSP, 321 pag., 1985.

[CLOC 84] CLOCKSIN, W.F.; MELLISH, C.S.: "*Programming in Prolog*". Second Edition. Springer-Verlag, 293 pag., 1981.

[DALT 86] DALTRINI, BEATRIZ M.: "*Metodologia para Implementação de Sistemas PAC Reconfiguráveis*". Tese de Doutorado apresentada à UNICAMP, 140 pag., 1986.

[GERT 85] GERTLER, JANOS; SEDLAK, JAN: "*Software for Process Control - A Survey*". Automática, vol. 11, pag. 613-625, 1975.

[JINO 86] JINO, MARIO; TRAINA JR., CAETANO; CARVALHO, MARIO B.: "*Automação do Desenvolvimento de Software.* Parte I: *Conceitos Fundamentais, Ferramentas e Automação*". SBA: Controle de Automação, vol. 1, no. 2, pag. 99-109, 1986.

[LAUB 81] LAUBER, R.: "*Computer Aided Development of Real-Time Computer Systems*". 1st SICOP Conference, Rio de Janeiro, Brasil, May 21, 1981.

[LAUT 74] LAUTENBACH, K.; SCHMID, H.: "*Use of Petri Nets for Proving Correctness Process Systems*". Information Processing 74, Proceedings of the 1974 IFIP Congress, Amsterdam, North-Holland, pag. 191-197, 1974.

[MAGA 86] MAGALHÃES, M.F.: "*Software para Tempo Real*". I EScola Brasileiro-Argentina de Informática (I EBAI), Editora da UNICAMP, 82 pag., 1986.

[MENA 85] MENASCHE, M.: "*Parede: An Automated Tool for the Analysis of Timed Petri Nets*". International Workshop on Timed Petri Nets, pag. 162-169, July 1985.

[MURA 77a] MURATA, T.: "*State Equation, Controllability and Maximal Matchings of Petri Nets*". IEEE Transactions on Automatic Control, vol. AC-22, no. 3, pag. 412-416, 1977.

[MURA 77b] MURATA, T.: "*Circuit Theoretic Analysis and Synthesis of Marked Graphs*". IEEE Transactions on Circuits and Systems, vol. CAS-24, no. 7, pag. 400-405, 1977.

[SIMO 86] SIMÕES, MARCOS A.S.: "*Ferramenta de Projeto, Programação e Simulação de Células Flexíveis de Manufatura*". Tese de Mestrado apresentada à UFRJ, 153 pag., 1986.

[PASS 86] PASSOS, CARLOS A.S.: "*Simulação de Sistemas de Manufatura*". Tese de Mestrado apresentada à UNICAMP, 95 pag., 1986.

[PETE 77] PETERSON, J.: "*Petri Nets*". Computing Surveys, vol. 9, no. 3, pag. 223-252, 1977.

[PETE 81] PETERSON, J.: "*Petri Net Theory and Modeling of Systems*". Prentice-Hall, Englewood Cliffs, N.J., 289 pag., 1981.

[VALE 84a] VALETTE, R.; ALANCHE, P.; BENZAKOUR, K.; DOLL, F.; GILLET, P.; RODRIGUES, P.: *"PSS: A Petri Net based Simulator for Flexible Manufacturing Systems"*. 5th European Workshop on Applications and Theory of Petri Nets, pag. 72-85, June 1984.

[VALE 84b] VALETTE, R.; BENZAKOUR, K.: *"Simulateur P.S.S. Rapport Scientifique* 84.065". LAAS-Toulouse, 1984.

APÊNDICE A

LINGUAGEM PROLOG

A.1 - INTRODUÇÃO

O objetivo deste apêndice é apresentar resumidamente as principais características da Linguagem Prolog utilizada na implementação da parte experimental deste trabalho.

A.2 - LINGUAGEM PROLOG

Prolog é uma linguagem de programação de computadores utilizada para resolver problemas que envolvem *objetos* e as *relações* entre objetos [CLOC 84].

A programação de um computador utilizando Prolog consiste de três tópicos:

- Declaração de alguns *fatos* sobre os objetos e seus relacionamentos
- Definir algumas *regras* sobre os objetos e seus relacionamentos
- Efetuar *perguntas* sobre os objetos e seus relacionamentos.

Um programa Prolog é portanto uma coleção de regras e fatos. A solução dos programas decorrem da interpretação destas regras e fatos.

A estrutura de um programa Prolog é diferente de outras linguagens convencionais (Cobol, Fortran, Basic e C) pois em Prolog não há a necessidade de fornecer instruções detalhadas de um determinado fluxo de controle. O programador em Prolog define soluções possíveis para uma tarefa e gera em seguida um conjunto de fatos e regras que irão permitir que o programa selecione a solução correta.

A Linguagem Prolog é utilizada no desenvolvimento de diversos tipos de aplicações incluindo sistemas especialistas, base de dados relacionais, sistemas que utilizam linguagem natural, interpretadores e compiladores.

A.3 - ARITY/PROLOG

A Linguagem Prolog utilizada na elaboração deste trabalho faz parte do ambiente de programação Arity/Prolog Versão 4 desenvolvido pela empresa Arity Corporation U.S.A.

Os módulos componentes deste ambiente de programação são os seguintes:

- Arity/Prolog Compiler and Interpreter
- Arity SQL Development Package
- Arity Expert Systems Development Package
- Arity Screen Design Toolkit
- Arity File Interchange Toolkit

O Centro Tecnológico para Informática CTI/Campinas-SP adquiriu oficialmente a versão 4, utilizada na elaboração deste trabalho.