UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

UM AMBIENTE INTEGRADO DE FERRAMENTAS PARA O PROJETO DE SISTEMAS A EVENTOS DISCRETOS

FRANCISCO DE ASSIS SOARES DE OLIVEIRA

DISSERTAÇÃO SUBMETIDA À UNIVERSIDADE FEDERAL DE SANTA CATARINA PARA A OBTENÇÃO DO GRAU DE MESTRE EM ENGENHARIA ELÉTRICA

FLORIANÓPOLIS, 01/07/1994.

UM AMBIENTE INTEGRADO DE FERRAMENTAS PARA O PROJETO DE SISTEMAS A EVENTOS DISCRETOS

FRANCISCO DE ASSIS SOARES DE OLIVEIRA

ESTA DISSERTAÇÃO FOI JULGADA ADEQUADA PARA A OBTENÇÃO DO TÍTULO:

MESTRE EM ENGENHARIA

ESPECIALIDADE: ENGENHARIA ELÉTRICA, ÁREA DE CONCENTRAÇÃO: SISTEMAS DE CONTROLE E AUTOMAÇÃO INDUSTRIAL, E APROVADA EM SUA FORMA FINAL PELO CURSO DE PÓS-GRADUAÇÃO.

Prof.ª Janette Cardoso, Drª
Departamento de Eng. Elétrica, UFSC
Orientador

Prof. Roberto de Souza Salgado, Ph. D.
Coordenador do Curso de Pós-Graduação em Engenharia Elétrica

BANCA EXAMINADORA:

Prof.ª Janette Cardoso, Drª
Presidente

Prof. Jean-Marie Farines, Dr. Ing.
Departamento de Eng. Elétrica, UFSC

Prof. Marcelo Stemmer, Dr.
Departamento de Eng. Elétrica, UFSC

Dedico este trabalho à minha esposa, Maria Lúcia.

AGRADECIMENTOS

Em primeiro lugar agradeço a Deus, com a plena convicção de que foi feita Sua vontade e de que tudo concorre para o bem daqueles que Nele acreditam.

À professora Janette Cardoso, pela paciência, dedicação e incentivo demonstrados durante sua orientação e que tornaram possível a conclusão deste trabalho.

Aos demais membros do Laboratório de Controle e Microinformática (LCMI), especialmente àqueles com quem convivi mais de perto, não somente pelas valiosas discussões técnicas, mas sobretudo pelas manifestações de companheirismo e amizade.

À PETROBRÁS, *uma história brasileira de sucesso*, pela oportunidade e apoio que me foram dados para a realização deste trabalho.

RESUMO

Os Sistemas a Eventos Discretos (SED) industriais atuais são em geral caracterizados pelo alto grau de integração e automação de suas funções. A complexidade desses sistemas leva à sua decomposição numa hierarquia com vários níveis de controle. Este trabalho apresenta o Ambiente Integrado de Ferramentas (AIF), concebido para auxiliar o projeto de sistemas de controle para Sistemas a Eventos Discretos, nos níveis de Coordenação e Controle Local. A Rede de Petri Interpretada foi o modelo teórico utilizado na representação do sistema de controle e do sistema controlado, permitindo, também, representar todas as interações entre os dois sistemas. O AIF é composto por quatro módulos (Edição, Análise, Simulação e Tradução) e é utilizado em todas as fases do projeto. Sua utilização é ilustrada num exemplo de automação de uma estação coletora de petróleo.

ABSTRACT

Modern Discrete Event Industrial Systems are in general characterized by the high degree of integration and automation of their functions. Their complexity leads to split up the whole system into hierarchical systems with various control levels. This work presents the Integrated Environment of Tools (IET) to help the design of control systems for Discrete Event Industrial Systems, at their Coordination and Local levels. The Interpreted Petri net was the theoretical model used to model the control system and the plant, allowing to represent all the interactions between the two systems. The IET is composed by four modules (Edition, Analysis, Simulation and Translation) and provides help for all phases of the design. Its utilization is illustrated in an example of automation of a petroleum collecting station.

SUMÁRIO

LISTA DE FIGURAS

CAPÍTULO 1

INTRODUÇÃO

As atuais exigências de adequação a padrões internacionais de qualidade, de redução de custos e de aumento na confiabilidade e na segurança operacional têm levado o setor industrial, de forma crescente e inexorável, a otimizar e automatizar seus processos de produção.

Por sua vez, a automação industrial, cada vez mais baseada em máquinas com comando numérico computadorizado, controladores lógico-programáveis e robôs, requer para o desenvolvimento dos sistemas de controle e supervisão a utilização de uma metodologia que permita representar corretamente os elementos envolvidos e suas interações, sem guardar dependência com a tecnologia empregada em cada um. Devido às exigências que são impostas à qualidade dos serviços que esses automatismos devem oferecer - em particular aquelas relativas aos aspectos de confiabilidade e segurança operacional - a concepção desses sistemas de controle não costuma ser uma tarefa trivial. Deve-se garantir nos sistemas automatizados a verificação de aspectos tais como sincronização, exclusão mútua e alocação de recursos. Assim, os métodos e ferramentas utilizados durante o processo de desenvolvimento de um sistema têm uma importância fundamental.

É usual decompor-se o ciclo de desenvolvimento de um sistema de controle nas seguintes etapas:

- Especificação formal: partindo-se da especificação informal, resultante da interação entre o cliente e/ou usuário e o projetista do sistema, elabora-se uma nova especificação, utilizando uma *Técnica de Descrição Formal (TDF)*;

- Validação da especificação formal: consiste em mostrar que a especificação obtida corresponde aos requisitos do cliente ou do futuro usuário. Dentre as atividades englobadas pela validação pode-se citar: a análise, a simulação e a emulação;

- Implementação: partindo da especificação validada do sistema, obtém-se, se possível de modo direto e automático, uma realização do mesmo. No caso do presente trabalho essa realização consiste em gerar um *software* de controle e supervisão;

- Teste de conformidade: implementado o sistema, verifica-se, através de testes, se seu funcionamento está conforme a especificação inicial.

Em um processo real de desenvolvimento ocorrerão, naturalmente, muitas interações entre as etapas (e.g.: uma especificação deve ser modificada se sua análise detectar incorreções).

A escolha da ferramenta de especificação (também chamada de modelo de especificação) adequada deve basear-se no atendimento a alguns requisitos básicos, sendo os principais:

- *expressividade*, ou seja, a capacidade de representar todas as características do sistema estudado;

- capacidade de *abstração*, que permite desconsiderar aspectos irrelevantes para a descrição de um sistema e obter uma especificação independente da implementação (especificação abstrata);

- a especificação obtida deve ser facilmente validável. O nível de exigência aqui é variado: pode-se desejar apenas uma documentação clara e concisa; pode-se querer utilizar a especificação para simular o comportamento (lógico ou temporal) do sistema ou pode-se desejar que a especificação obtida seja analisável e que permita provar o atendimento a certas restrições;

- deve haver métodos de implementação direta a partir da especificação obtida. Com isso tem-se uma diminuição do número de erros que podem eventualmente aparecer devido à mudança de ferramenta de representação entre o projeto e a implementação. De fato, pouco adiantaria ter validado a especificação se a mesma tivesse que ser revista quando da realização do sistema.

As ferramentas que apresentam essas características servem de base para as chamadas Técnicas de Descrição Formal (TDF). Em suma, uma TDF caracteriza-se pelo seu poder de expressão, que permite obter uma especificação clara, legível e sem ambigüidades, e pelo seu poder de análise, que facilita a detecção de erros.

Essas características permitem a construção de ferramentas automáticas utilizáveis nas etapas de desenvolvimento de um sistema de controle. A proposta deste trabalho é apresentar o AIF - Ambiente Integrado de Ferramentas, um ambiente automatizado integrando ferramentas de especificação, validação (por análise e simulação) e implementação, para auxiliar no projeto de sistemas de controle para Sistemas a Eventos Discretos. Foram integradas à esse ambiente as ferramentas: ARP - Analisador de Redes de Petri e SPP - Sistema de Produção Proposicional, que traduz uma Rede de Petri descrita na forma de regras num programa em código C; ambas ferramentas foram desenvolvidas no LCMI/UFSC. A ferramenta de simulação foi especificada e implementada de maneira a simular o sistema de controle e a planta interagindo. Este ambiente pode ser utilizado tanto no projeto do sistema de controle de uma planta, quanto no projeto da própria planta (verificando, por exemplo, se uma determinada configuração (ou distribuição) de máquinas em um chão de fábrica é coerente).

O AIF foi construído em torno de uma mesma Técnica de Descrição Formal: a Rede de Petri. Entre as diversas abordagens (modelos de base ou formalismos) existentes, tais como Máquinas de Estados Finitos, Redes de Petri, Gramáticas Formais, Álgebras de Processos, Tipos Abstratos de Dados, Lógica Temporal e modelos híbridos, a Rede de Petri foi escolhida como modelo de base pelas seguintes razões:

i. pode ser utilizada em várias etapas do ciclo de desenvolvimento de um sistema (especificação, modelagem, validação e implementação);

ii. certas características exigidas na especificação informal são traduzidas diretamente no modelo (exclusão mútua, sincronização, conflito, alocação de recursos, seqüências, etc.) e podem ser verificadas;

iii. a modelagem é facilitada graças ao refinamento (*top-down*) e à composição de módulos (*bottom-up*) utilizando sub-redes *bem-formadas*;

iv. a implementação pode ser feita diretamente utilizando um jogador de fichas;

v. existência de ferramentas baseadas no modelo Rede de Petri (ou compatíveis) que poderiam ser integradas ao ambiente.

A apresentação deste trabalho está organizada conforme descrito a seguir.

No segundo capítulo são dadas as características dos Sistemas a Eventos Discretos e ressalta-se a importância da simulação no projeto desses sistemas.

O terceiro capítulo é dedicado à apresentação do modelo Rede de Petri: os conceitos básicos, as diferentes formas de análise e os modelos de Rede de Petri Interpretada e Rede de Petri Temporal.

O quarto capítulo é consagrado à descrição da arquitetura do Ambiente Integrado de Ferramentas, formada por quatro módulos: Edição, Análise, Simulação e Tradução.

No capítulo quinto o módulo de Simulação é detalhado, descrevendo-se como é feita a simulação do controle e da planta usando a Rede de Petri Interpretada.

Como exemplo de aplicação do ambiente em quetão mostra-se, no sexto capítulo, sua utilização no projeto de automação de uma estação de coleta de petróleo similar às existentes nos campos terrestres explorados pela PETROBRÁS no estado do Rio Grande do Norte.

Por fim, no sétimo capítulo, são apresentadas as conclusões e perspectivas deste trabalho.

CAPÍTULO 2

SISTEMAS A EVENTOS DISCRETOS E SUA SIMULAÇÃO

2.1 INTRODUÇÃO

Neste capítulo serão, inicialmente, apresentados os sistemas para os quais se desenvolveu o Ambiente Integrado de Ferramentas: os Sistemas a Eventos Discretos. Em seguida, serão vistos alguns aspectos da simulação de sistemas a eventos discretos: sua importância no projeto de tais sistemas, bem como os modelos de representação e as abordagens de implementação.

2.2 SISTEMAS A EVENTOS DISCRETOS

De modo geral, um sistema discreto é um sistema onde as mudanças de estado se produzem em determinados instantes de tempo. As evoluções de um Sistema a Eventos Discretos (SED) são provocadas por eventos físicos que se produzem em instantes separados por intervalos de tempo que podem ser irregulares [Ram89]. As variáveis de estado são modificadas de forma abrupta; os valores das variáveis no próximo estado podem ser calculados diretamente a partir dos valores dos estados precedentes, sem considerar o tempo entre os dois instantes. Exemplos de tais eventos físicos são: a transmissão de um pacote num sistema de comunicação, a chegada ou saída de um cliente numa fila, o fim de operação numa peça num sistema de manufatura, o sinal de um pressostato indicando que a pressão do fluido na descarga de uma bomba atingiu um valor alto, etc..

A natureza discreta dos SED's faz com que os modelos matemáticos convencionais, baseados em equações diferenciais, não sejam adequados para tratá-los. Por outro lado, sua importância faz com que seja fundamental encontrar soluções para problemas relacionados ao seu controle [Zil93].

Sistemas como os Sistemas de Manufatura, processos industriais como os de uma refinaria de petróleo, etc., possuem uma parte importante de sinais que se caracterizam como eventos discretos. A parte contínua de tais sistemas (e.g.: velocidade da furadeira na operação de uma peça, temperatura de uma coluna de destilação fracionada) é controlada de forma automatizada usando as técnicas de controle convencional. Entretanto, freqüentemente, a parte discreta é controlada por operadores.

A utilização de novas técnicas de produção dentro de conceito de *Automação Integrada* permite diminuir os ciclos de concepção e produção dos SED's, aumentando a qualidade do serviço e a flexibilidade. Ao mesmo tempo, há uma melhora na eficácia e na satisfação dos

operadores humanos envolvidos no processo de automação, e um aumento na utilização de equipamentos [Whi89]. Este desenvolvimento é fruto da evolução dos computadores, mais rápidos, potentes e baratos, bem como das técnicas de engenharia de *software* que permitem uma utilização ordenada e com melhor desempenho.

2.2.1 ESTRUTURA HIERÁRQUICA

Atualmente, os Sistemas a Eventos Discretos mais tratados na literatura são os Sistemas Flexíveis de Manufatura (FMS). Tais sistemas possuem um elevado grau de complexidade e por este motivo diferentes autores propõem uma decomposição através de níveis de abstração hierarquizados [Buz82] [Alb81]. Esta abordagem hierárquica é válida também em outros tipos de Sistemas a Eventos Discretos, como o controle de plantas industriais, supervisão de tráfego, concepção de *softwares*, etc.. Cada nível opera sobre um certo horizonte de tempo ou de complexidade, que diminui quando se desce na hierarquia. Nos níveis mais altos o sistema global é considerado de maneira agregada e o horizonte de tempo é maior. Na hierarquia, cada nível é uma desagregação dos níveis superiores. As restrições de tempo real são introduzidas progressivamente e são mais severas à medida que se desce para os níveis mais baixos. Várias abordagens são possíveis para a definição do número de níveis e do papel de cada um, sendo comum encontrar-se a decomposição nos cinco níveis descritos a seguir [Car90][Sil90]:

- Planejamento:

Determina, a partir da demanda dos clientes, o número de produtos que serão produzidos por semana ou mês, e para cada produto, sua data de entrada e saída do sistema. O plano de fabricação não é gerado em todos os seus detalhes, mas é feita uma alocação preliminar dos recursos (máquinas) de modo a reduzir a explosão combinatória no nível Escalonamento.

- Escalonamento:

O objetivo deste segundo nível é o de permitir satisfazer os objetivos de fabricação fixados no nível Planejamento. O calendário de fabricação é detalhado considerando individualmente cada operação sobre cada produto. O resultado é uma seqüência de datas para a execução de cada operação sobre cada máquina.

- Coordenação global:

Sua principal função é a de controlar a fabricação de maneira que esta se comporte conforme previsto no plano de fabricação detalhado nos dois níveis anteriores e atualizar a representação de estado do sistema em tempo real. Este nível deve, considerando a representação atualizada do sistema, tomar decisões em tempo real relativas à distribuição das operações sobre as máquinas, de modo a respeitar o plano. Deve, portanto, ser capaz de tratar, num mesmo instante, diversas atividades e mensagens, com um tempo de resposta imposto pelos níveis Escalonamento e Controle Local.

- Coordenação de subsistemas:

Sua principal função é o controle em tempo real dos subsistemas num nível de detalhe maior que o do nível Coordenação Global. De modo geral, trata-se do gerenciamento da cooperação de um conjunto de máquinas objetivando a realização de um conjunto de atividades. É o responsável pelo encadeamento de operações elementares constituindo estas atividades, como é o caso, por exemplo, das células de usinagem e de montagem, e dos sistemas de transporte. É neste nível que se encontra a função supervisão.

- Controle Local:

Este nível implementa o controle em tempo real das máquinas, esteiras, válvulas, etc., interagindo diretamente com os sensores e os atuadores. Todos os procedimentos de urgência são executados neste nível, sendo que as decisões em tempo real são tomadas nos dois níveis anteriores.

O objeto deste trabalho diz respeito ao projeto de Sistemas a Eventos Discretos nos níveis de Coordenação e Controle Local.

2.2.2 MODELOS DE ESPECIFICAÇÃO

Existem várias abordagens para modelar e especificar Sistemas a Eventos Discretos, mas nenhuma conseguiu se firmar como universal. Segundo [Ho87], isto se deve ao fato de que nenhum dos modelos considera simultaneamente todos os aspectos apresentados a seguir [Zil93]:

- natureza descontínua dos eventos discretos;
- natureza contínua das medidas de desempenho;
- importância da formulação probabilista;
- necessidade de análise hierárquica;
- presença de dinâmica;
- viabilidade do uso de soluções computacionais.

Nos modelos lógicos, tais como o modelo de Wonham e Ramadge, Lógica Temporal e Rede de Petri, o tempo é ignorado, considerando-se somente a ordem dos eventos. A representação clássica de um sistema com número finito de estados consiste em enumerar todos os estados possíveis e em descrever os eventos do tipo mudança de estado (descrever os estados seguintes de cada estado).

No modelo de Ramadge e Wonham [Ram89] o objetivo é o de incluir os conceitos da teoria de controle tais como controlabilidade, observabilidade, agregação, e o controle descentralizado e hierárquico no controle de SED's. Este modelo verifica se é possível modificar, através do controle, o conjunto de trajetórias admissíveis de modo que estas possuam a

propriedade desejada. A principal característica deste modelo é a separação entre o conceito do sistema a controlar do controle em malha fechada. A partir das restrições é construído um supervisor, eliminando as trajetórias não desejadas (e. g.: dois processos não podem escrever ao mesmo tempo em uma área de memória compartilhada).

As Lógicas Temporais [Man83] são extensões do cálculo proposicional obtidas pela adição de modalidades que permitem formalizar noções como: *é possível que, é inevitável que, é sempre verdade que*, etc.. Tais lógicas podem ser utilizadas tanto na especificação quanto na verificação. Na especificação, o usuário enuncia o funcionamento do sistema e as propriedades através de fórmulas da lógica temporal. O modelo assim obtido tem a vantagem de satisfazer as propriedades, mas nada garante que ele seja realista. A descrição lógica deve ser suficientemente precisa para que o modelo seja utilizável. Na verificação, o usuário traduz as propriedades desejadas usando fórmulas lógicas e verifica se são satisfeitas na solução proposta. Esta etapa é realizada automaticamente através de um controlador de modelos.

O modelo Rede de Petri [Pet66] permite representar a causalidade, a seqüência de ações, a decisão (escolha entre duas seqüências em conflito) e a independência (paralelismo). As restrições são descritas diretamente sobre o modelo, como por exemplo, a exclusão mútua. A distribuição das fichas nos lugares da rede (marcação) fornece, a cada instante, o estado do sistema. A evolução da marcação é provocada pelo disparo das transições. A Rede de Petri é um modelo formal que permite a validação por análise e por simulação. Além disso, pode ser implementada diretamente utilizando um programa *jogador de fichas*.

Neste trabalho escolheu-se a Rede de Petri como modelo de base. Tal modelo pode ser utilizado nas diversas fases do ciclo de vida de um sistema: especificação, modelagem, validação e implementação, enquanto que, por exemplo, a Lógica Temporal permite apenas a especificação e a verificação. A Rede de Petri permite traduzir certas características diretamente sobre o modelo. É o caso, por exemplo, do paralelismo verdadeiro, que não é diferenciado claramente do não determinismo no modelo de Wonham e Ramadge. Além disto, a tarefa de modelagem é facilitada graças ao refinamento e a composição de módulos, utilizando sub-redes bem formadas [Bra83]. A Rede de Petri possui, também, extensões que permitem tratar o tempo [Mer74] [Mol81].

2.3 SIMULAÇÃO A EVENTOS DISCRETOS

A simulação é a ferramenta de análise por excelência no projeto de Sistemas a Eventos Discretos. Isso se deve à complexidade de tais sistemas e, sobretudo, à inexistência de ferramentas matemáticas que permitam seu estudo completo. No caso, por exemplo, dos sistemas de manufatura, uma vez obtida a especificação informal do cliente deve-se projetar tanto a parte mecânica do sistema (máquinas) quanto a política de produção desejada.

Nesse caso, a simulação tem por objetivo estabelecer as ordens de grandeza características do sistema, a fim de se poder fazer uma comparação entre os diferentes projetos e escolher o melhor dentre eles. Para essa fase do projeto existem simuladores tais como SLAM e SIMAN [Ala84], mas seus modelos de base não permitem a análise formal. A escolha da Rede de Petri como modelo de base permitiu que, além da capacidade de especificação e análise formal, fosse também incorporada ao Ambiente Integrado de Ferramentas a capacidade de simulação.

Nesta seção veremos alguns aspectos da implementação da simulação a eventos discretos. As abordagens variam com os objetivos da simulação. Por exemplo, se o objetivo é analisar a evolução das conseqüências da natureza distribuída e assíncrona de um sistema real, é mais conveniente uma implementação baseada em processos comunicantes, enquanto que uma implementação centralizada é suficiente se o objetivo é obter estatísticas relativas ao desempenho do sistema.

Modelos de representação

O modelo de representação é o elemento mais importante de uma ferramenta de simulação. Naturalmente, no caso da simulação a eventos discretos, o modelo deve também ser discreto. A construção de um modelo discreto é baseada nos conceitos de *evento*, *atividade* e *processo* para representar o fluxo de produtos, a interação entre as entidades e os recursos [Bak90] [Ben85]:

- um *evento* é uma modificação discreta do estado do sistema em um determinado instante;

- uma *atividade* representa uma evolução contínua do sistema, em que o detalhe não é importante, mas que ao cabo de um certo tempo finito produzirá um evento (final de atividade). Uma atividade inicia-se sempre por um evento (início de atividade);

- um *processo* é uma seqüência ordenada no tempo de eventos e atividades.

Baseados nestes conceitos, existem três modelos de simulação:

- Modelo baseado em eventos: o sistema é modelado definindo as mudanças de estado produzidas pela ocorrência dos eventos. A tarefa do projetista consiste em determinar as datas de ocorrência dos eventos futuros que podem acontecer devido a um evento que se produziu no sistema em um dado estado;

- Modelo baseado em atividades: esta abordagem, dual da precedente, consiste em descrever as atividades nas quais estão envolvidas as entidades do sistema e em definir as condições que provocam o início de cada atividade. As condições de início e fim de atividade são testadas ciclicamente;

- Modelo baseado em processos: consiste em descrever a lógica de mudança de estado de uma seqüência de eventos e não cada evento ou atividade, combinando as características das duas abordagens precedentes.

Para citar alguns simuladores a eventos discretos: o SLAM utiliza o modelo baseado em processos, o CASP baseado em eventos e o SIMULA baseado em atividades.

A linguagem utilizada pode ser uma linguagem existente (e. g.: Fortran no caso do SIMAN e SLAM; e Algol no caso do SIMULA), ou pode-se criar uma linguagem de simulação, baseada, por exemplo, no modelo Rede de Petri, que permite também especificar diretamente o sistema e realizar a análise formal dessa especificação.

Implementação

Do ponto de vista da implementação, a abordagem seqüencial é a mais utilizada na simulação a eventos discretos. Sua característica principal é a existência de um relógio único e de um escalonador único. O *escalonador* é uma lista de pares do tipo [evento, data de ocorrência], ordenados segundo sua data de ocorrência. A simulação evolui segundo a cadência de um relógio. Esse relógio permite representar o tempo físico do sistema e ordenar no tempo (ordem cronológica) os eventos do escalonador. Tem-se dois tipos de simulação segundo a forma como é feita a evolução do relógio:

- Simulação dirigida pelos eventos

O programa de simulação, a cada ciclo, realiza a seqüência:

1. executar primeiro evento do escalonador;

2. retirar este evento do escalonador;

3. avançar o relógio de modo a marcar a data do próximo evento.

Somente as datas dos eventos são acessíveis. Os eventos, exceto o evento atual, são potenciais, pois podem ser modificados durante a simulação.

- Simulação dirigida pelo relógio

O tempo é avançado por incrementos e, a cada incremento, são executados todos os eventos da data atual (que possuem data de ocorrência igual à do relógio). Neste caso, deve-se tomar cuidado na escolha do incremento de tempo de modo a minimizar a taxa de falhas na procura de eventos (nenhum evento com data de ocorrência igual a do relógio).

A abordagem de simulação a eventos discretos distribuída difere da abordagem seqüencial pelo fato de que o programa de simulação é executado por um conjunto de

processadores assíncronos comunicando-se por passagem de mensagens. Cada mensagem é um par $[T,m]$ onde T é uma marca de tempo e m o conteúdo da mensagem. Cada processador executa uma parte do programa de simulação (um processo). Cada processo possui seu próprio relógio e seu próprio escalonador [Bak90] [Cha79] [Bry77].

Neste trabalho será utilizada a simulação na abordagem seqüencial e dirigida por eventos.

CAPÍTULO 3

A REDE DE PETRI

3.1 INTRODUÇÃO

Idealizada por Carl Adam Petri [Pet66], a Rede de Petri (RdP) pode ser definida como uma ferramenta gráfica e matemática adequada à descrição e ao estudo de sistemas dinâmicos a eventos discretos, ou seja, sistemas cujos estados só mudam em determinados instantes de tempo, em função da ocorrência de um evento; e.g.: sistemas comandados por controladores lógico-programáveis, sistemas de controle para automação da manufatura, sistemas de comunicação e de tráfego. Este modelo surgiu da intenção de se desenvolver uma base teórica e conceitual para a descrição, de uma maneira uniforme e exata, do maior número possível de fenômenos relacionados à transmissão da informação e à transformação da informação [Rei85].

Dentre suas principais características destacam-se:

- poder de expressão, notadamente na representação de relações de dependência e independência causal em um conjunto de eventos (e.g.: eventos seqüenciais, eventos paralelos, eventos mutuamente exclusivos);

- representação dos sistemas em diferentes níveis de abstração usando a mesma linguagem de descrição;

- possibilidade de verificar propriedades do sistema modelado e efetuar provas de correção [Pet81] [Est85], permitindo também a validação por simulação [Ben85];

- viabilidade de construção de ferramentas automatizadas para verificação, validação e análise de desempenho do sistema modelado.

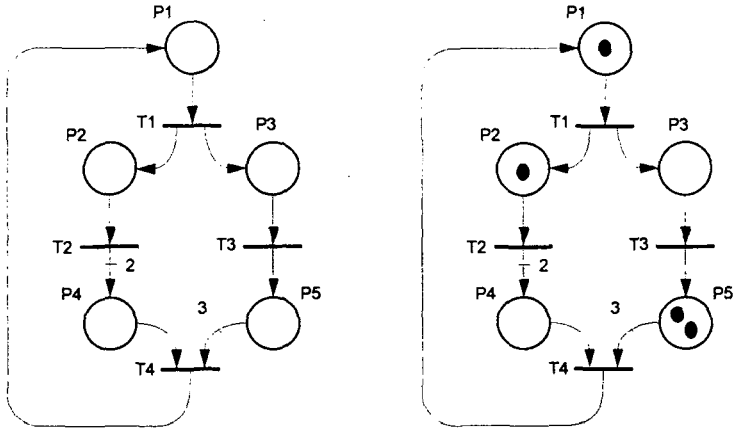
3.2 CONCEITOS BÁSICOS

Lugares, transições e arcos

Na qualidade de ferramenta gráfica, a Rede de Petri permite a representação dos sistemas através de um grafo orientado com dois tipos de nós (figura 3.1): os lugares e as transições. Um *lugar* é representado por um círculo e uma *transição*, por uma barra. Os lugares e transições são ligados por *arcos* orientados. Cada arco liga sempre um lugar a uma transição ou uma transição a um lugar e tem associado a si um *peso w* (um número inteiro maior ou igual a 1, escrito ao lado do arco), que indica a multiplicidade do mesmo (como se *w* arcos estivessem indo de um nó a outro); pesos não indicados são subentendidos como sendo unitários (caso da RdP dita clássica

ou ordinária). Os lugares de uma Rede de Petri podem ser interpretados como condições e as transições correspondem aos eventos.

A figura 3.1.a representa uma RdP com 5 lugares, 4 transições e 10 arcos (8 com peso unitário, um com peso 2 e outro com peso 3). Tomando-se, por exemplo, a transição T3, diz-se que P3 é seu *lugar de entrada* (o arco está orientado de P3 para T3) e P5, seu *lugar de saída*; de maneira similar, pode-se falar em transição de entrada e de saída em relação a um lugar.



(a) (b)

Fig. 3.1 - Rede de Petri: (a) Não marcada; (b) Marcada

Marcação

Os lugares representam estados parciais do sistema modelado e para possibilitar a representação de um estado global do sistema atribui-se a cada lugar um número inteiro (não negativo) de marcas (pequenos pontos pretos), chamadas de *fichas*. O número de fichas em um lugar Pi, denotado por M(Pi), é chamado de marcação do lugar. A *marcação* da rede (ou simplesmente marcação), denotada por M, é definida por um vetor cujas componentes são as marcações dos lugares da rede. Para a figura 3.1.b temos M={1,1,0,1,2}.

Assim, a marcação em um certo instante define o estado da RdP ou, mais precisamente, o estado do sistema modelado pela RdP. A evolução do estado corresponde, portanto, à evolução da marcação; essa evolução é produzida pelos disparos das transições habilitadas.

Regra de disparo de uma transição

O comportamento de muitos sistemas pode ser descrito em termos de seus estados e da evolução desses estados. De modo a simular o comportamento dinâmico do sistema modelado (evolução ao longo do espaço de estados acessíveis), a marcação da RdP (e, por conseguinte, o

estado do sistema modelado) deve ser modificada de acordo com a seguinte regra (regra de habilitação e disparo de uma transição):

i. cada um dos lugares de entrada da transição deve possuir um número de fichas maior ou igual ao peso do respectivo arco; diz-se então que a transição está *sensibilizada* pela marcação, ou *disparável* (por enquanto esses termos podem ser considerados como sinônimos);

ii. o *disparo* de uma transição consiste em retirar de cada lugar de entrada um número de fichas igual ao peso do respectivo arco e em adicionar a cada lugar de saída um número de fichas igual ao peso respectivo arco.

Na figura 3.2 estão ilustrados quatro casos de aplicação da regra de disparo. Deve ser notado que, no caso (b), o lugar P3 ficou com duas fichas porque já havia uma antes do disparo e que, no caso (d), a transição não está sensibilizada, pois o lugar P2 deveria ter pelo menos duas fichas.

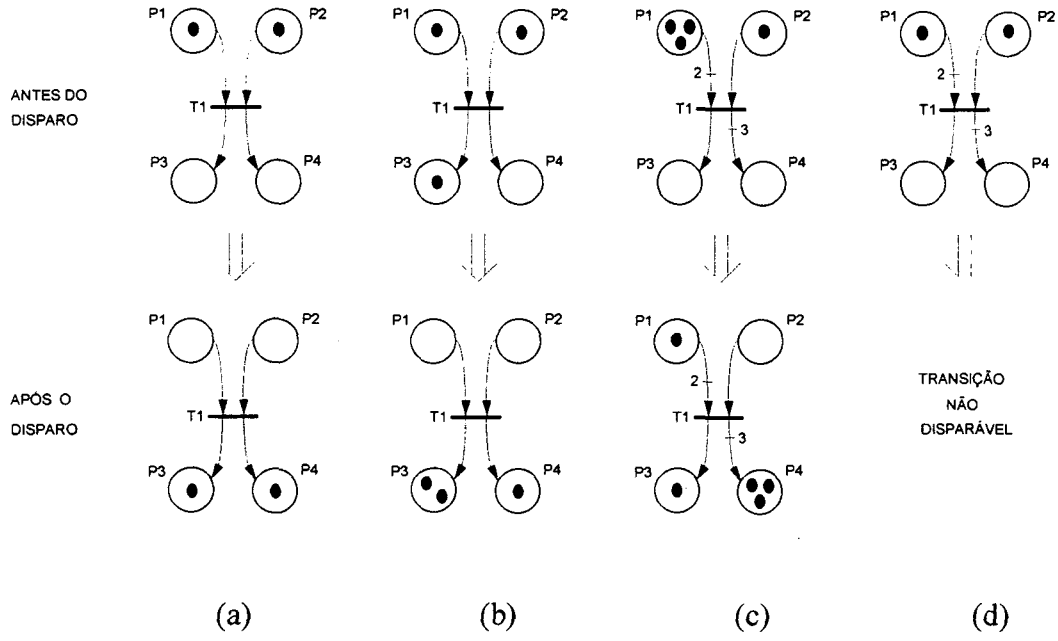


Fig. 3.2 - Exemplos de disparos de transições

Redes de Petri autônomas e não autônomas

Uma RdP *autônoma* descreve o funcionamento de um sistema cuja evolução é independente de eventos externos. Em outras palavras, uma RdP autônoma permite uma abordagem qualitativa: descreve *o que* acontece, mas não *quando* acontece.

O exemplo da figura 3.3 mostra uma RdP autônoma descrevendo o ciclo das estações do ano: a próxima transição a ser disparada será T1 (passagem da primavera para o verão), mas não há indicação sobre o momento em que isso ocorrerá.

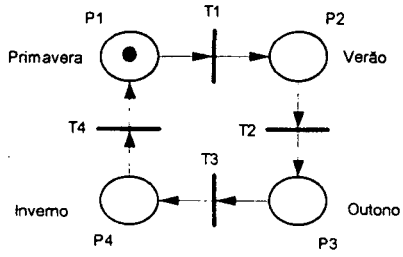


Fig. 3.3 - Rede de Petri autônoma

Uma RdP *não autônoma* descreve o funcionamento de um sistema cuja evolução é condicionada por eventos internos ou externos, inclusive o tempo. Portanto, uma RdP não autônoma permite representar *quando* uma transição será disparada. Suas características tornam-na adequada à avaliação de desempenho e ao estudo dos automatismos lógicos e dos sistemas de controle em tempo real.

A Rede de Petri da figura 3.4 representa o mesmo ciclo das estações do ano. A marcação agora representa o verão e a única transição sensibilizada é T2. Mas, nesse caso, o disparo dessa transição, ou seja, a passagem para o outono, somente ocorrerá *quando* houver decorrido o intervalo de tempo de 3 meses. É, portanto, uma RdP *não autônoma*.

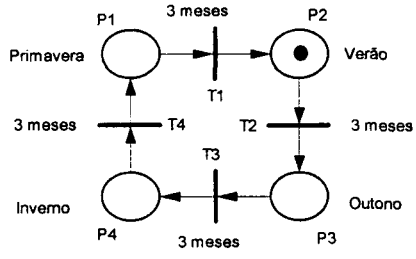


Fig. 3.4 - Rede de Petri não autônoma

3.3 REDE DE PETRI INTERPRETADA

Um sistema de controle subentende a existência de um sistema a controlar; por exemplo, um processo industrial. A interação entre o sistema de controle e o ambiente externo que caracteriza o processo faz-se através de uma interface que permite a troca de informações entre ambos, na forma de *variáveis externas* (booleanas): as que são recebidas do processo (através dos sensores) e as que são enviadas ao mesmo (através dos atuadores). Podem existir, também, informações utilizadas apenas internamente pelo sistema de controle e que são representadas pelas chamadas *variáveis internas* (inteiras ou booleanas): as variáveis auxiliares e as variáveis de sincronização. As primeiras têm o mesmo papel das variáveis auxiliares utilizadas nas linguagens de programação; as variáveis de sincronização são variáveis auxiliares especiais introduzidas com o objetivo de impor modificações na estratégia de controle.

Ao se iniciar a concepção de um sistema de controle, ou seja, na fase de descrição funcional , pode ser conveniente (ou mesmo necessário, dependendo de sua complexidade), que a ferramenta de modelagem permita uma separação clara entre a estratégia de controle e o tratamento das informações, não só para facilitar sua representação e torná-la mais inteligível, mas também para permitir sua interação com o ambiente externo (processo).

A Rede de Petri Interpretada (RdPI), também conhecida como RdP etiquetada, surgiu para atender a essas necessidades. É uma rede do tipo *não autônoma* que distribui a representação do sistema de controle em duas partes: a parte de controle (uma Rede de Petri clássica do tipo Lugar/Transição) e a parte de dados, na forma de etiquetas associadas às transições. Cada etiqueta fica ao lado de uma transição, entre parênteses, e é formada por duas partes, separadas por uma vírgula: a parte à esquerda da vírgula é formada por expressões lógicas envolvendo variáveis externas e/ou internas e representam condições suplementares para o disparo de uma transição sensibilizada; a parte à direta da vírgula é formada por operações de atribuição envolvendo variáveis externas e/ou internas e representam ações a serem efetuadas quando do disparo da transição. A ausência da parte condição implica numa condição sempre verdadeira e a ausência da parte ação implica numa ação vazia. A Rede de Petri Interpretada da figura 3.5 representa a operação de uma máquina em função dos sensores S (indicando presença de peça) e F (indicando fim de operação). Quando S=1, é enviada a ordem *liga_maq* se o lugar P1 estiver marcado, e quando F=1, é enviada a ordem *desliga_maq* se o lugar P2 estiver marcado.

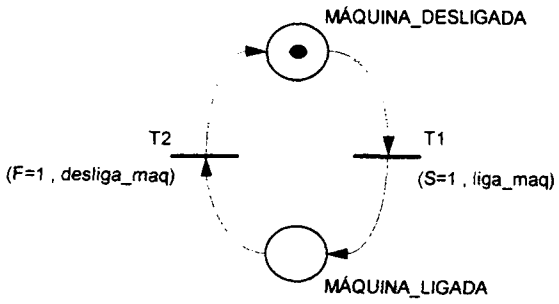


Fig. 3.5 - Rede de Petri Interpretada

O comportamento da Rede de Petri Interpretada é função da interação existente entre a parte de dados (variáveis internas ou externas) e a parte de controle (estrutura da rede, ou rede subjacente). Uma análise preliminar deve ser feita sobre a rede subjacente, para verificar se esta possui as boas propriedades. Essa análise é importante porque as evoluções das marcações da Rede de Petri Interpretada são restrições de evoluções das marcações da rede antes de sua interpretação (rede subjacente). Isto significa que o conjunto de marcações acessíveis após a interpretação está incluído no conjunto de marcações acessíveis antes da interpretação. De fato, para que uma transição possa ser disparada na Rede de Petri Interpretada, é necessário que ela esteja sensibilizada na rede subjacente. No entanto, uma transição pode estar sensibilizada na rede subjacente e não ser disparável na rede interpretada, bastando para isso que a parte condição da etiqueta associada seja falsa. Não obstante, deve-se ressaltar que a análise da rede subjacente fornece grande ajuda ao projetista porque permite detectar eventuais incoerências ainda no início do projeto [Val91]. Entretanto, a rede como um todo, incluindo a estrutura de dados, não pode ser totalmente analisada por métodos analíticos. Uma maneira alternativa de realizar a análise global é através de simulações.

No restante desse trabalho a palavra sistema, quando aparecer isoladamente, deve ser subentendida como sistema de controle.

3.4 Associação do Tempo à Rede de Petri

Até agora foi visto que a Rede de Petri é uma ferramenta de modelagem que permite evidenciar conflitos potenciais (possibilidade de disparos simultâneos) e representar problemas de paralelismo e sincronização. Em uma seqüência de eventos o tempo aparece de maneira implícita e qualitativa (já que um evento deve-se produzir após um outro). No entanto, para estudar o comportamento dinâmico (supervisão, avaliação de desempenho, simulação, etc) dos sistemas discretos faz-se necessária a introdução do tempo de maneira explícita e quantitativa. Assim, o modelo RdP foi estendido para levar em conta o tempo, gerando novas classes de RdP, das quais se destacam as redes *temporais* e as redes *temporizadas*.

Rede de Petri Temporal

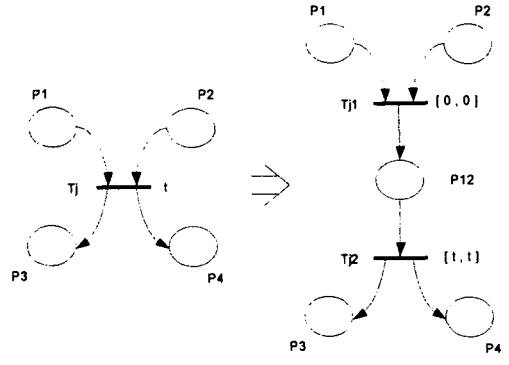
Introduzida por Merlin [Mer74], caracteriza-se pela associação de um intervalo de tempo [*tmin* , *tmax*] a cada transição. O valor *tmin* estabelece o tempo mínimo de sensibilização da transição antes de seu disparo; *tmax* é tempo máximo em que pode ocorrer o disparo da transição. Deve ser observado que enquanto aguarda o instante de disparo a transição pode deixar de ser sensibilizada, uma vez que as fichas dos seus lugares de entrada permanecem disponíveis e, portanto, passíveis de serem utilizadas por outras transições ligadas a esses lugares.

Rede de Petri Temporizada

Também chamada de rede a transição temporizada (para diferenciar das redes a lugar e a arco temporizado), surgiu através de Ramchandani [Ram74] e caracteriza-se pela associação de uma duração de disparo, *td*, a cada transição. Uma vez sensibilizada a transição inicia-se seu disparo que começa com a reserva das fichas dos lugares de entrada (as fichas ficam indisponíveis para outras transições); findo o tempo associado, as fichas são colocadas nos lugares de saída. Nessa classe de rede uma transição representa uma atividade de duração *td*.

Comparando as duas classes, pode-se constatar que a rede temporal possui maior poder de expressão, uma vez que pode representar qualquer rede temporizada, conforme mostrado na figura 3.6 e permite a representação de mecanismos particulares como o *timeout* e o *watchdog* (devido à disponibilidade das fichas nos lugares de entrada de uma transição sensibilizada).

Nesse trabalho utiliza-se, sobretudo, um caso particular da rede temporal fazendo *tmin* = *tmax*, que permite representar o mecanismo de *timeout*.



(a) (b)

Fig. 3.6 - Rede de Petri: (a) Temporizada; (b) Temporal equivalente

3.5 Implementação de uma Rede de Petri

Vários métodos têm sido utilizados para implementar especificações baseadas nas Redes de Petri. Alguns utilizam a chamada abordagem *compilada*, que consiste em transformar a RdP em um conjunto de equações booleanas diretamente executáveis por autômatos programáveis comerciais. A transformação da rede em equações booleanas se faz transição por transição ou lugar por lugar. Tal abordagem, apesar de ser relativamente eficaz nos casos simples, pode ser inadequada nos casos onde a parte seqüencial é muito importante.

Uma outra abordagem, igualmente clássica, consiste na utilização da noção de *jogador de fichas*. O grafo é transformado em uma estrutura de dados que é interpretada em tempo real por um programa *jogador*, independente da aplicação. É chamada de abordagem *interpretada*. O programa jogador, com base no estado do sistema (marcação da rede), examina um certo número de transições e, caso estejam habilitadas, efetua seus disparos evoluindo no espaço de estados até atingir um estado em que não haja mais transições habilitadas, o chamado estado estável, onde fica aguardando eventos externos que habilitarão novas transições. No capítulo 5 tal abordagem é apresentada com mais detalhes.

3.6 A Rede de Petri e os sistemas de regras

A Rede de Petri pode ser vista como um sistema a base de regras. De fato, o conjunto de transições de uma RdP é equivalente a um conjunto de regras de produção [Val89]. As matrizes *Pre* e *Post* [Bra83], exploradas coluna por coluna (isto é, o conjunto de transições com suas regras de disparo) descrevem a base de regras: as condições correspondem à matriz *Pre* e as ações à matriz *Post*. A marcação - distribuição das fichas nos lugares - corresponde, em um dado instante, à base de fatos. A resolução dos conflitos deriva das noções de transições paralelas (a ordem de tiro é indiferente) e de transições em conflito (o resultado depende da ordem dos tiros); se duas transições estão em conflito efetivo, e uma delas é disparada de modo aleatório, a outra não será disparada [Val91].

No caso de uma RdP Interpretada, que possui condições e ações suplementares associadas às transições, a parte condição das regras é formada pela matriz *Pre* associada a essas novas condições. O termo conseqüente é a matriz *Post* à qual se adicionam as novas ações. O conjunto dos fatos é a marcação com os estados de todas as variáveis que aparecem nas condições e ações suplementares.

O jogador de fichas da Rede de Petri corresponde então ao motor de inferência do Sistema de Regras de Produção que escolhe qual a transição (regra) que deve ser disparada para cada marcação da rede (estado da base de fatos).

3.7 VALIDAÇÃO DE SISTEMAS MODELADOS POR REDES DE PETRI

Validar um sistema consiste em mostrar que seu funcionamento está correto e também está perfeitamente de acordo com o desejado pelo cliente. É bem sabido que os erros existentes na especificação funcional de um sistema devem ser descobertos o mais cedo possível (quanto mais tarde descobertos, maiores os custos de correção). Assim, é desejável que a ferramenta de modelagem possua características que viabilizem essa validação na fase inicial do projeto de um sistema de controle.

A Rede de Petri, por ser uma ferramenta de descrição formal, permite que seja feita uma validação da especificação funcional do sistema antes de sua implementação (ou simulação).

Dentre as técnicas de validação utilizadas, duas são particularmente adequadas às características das RdP's: a *validação por análise* e a *validação por simulação*.

3.7.1 VALIDAÇÃO POR ANÁLISE

O modelo de um sistema será de pouco uso se não for analisável, uma vez que com a análise do mesmo espera-se obter importantes informações sobre o sistema modelado. Quando a ferramenta de modelagem é uma ferramenta formal, como é o caso da RdP, a análise da especificação pode ser suficientemente rica para detectar erros de concepção.

Assim, a validação por análise consiste em verificar se a RdP que representa o sistema possui as chamadas *boas propriedades* e em mostrar que as restrições específicas impostas pelo problema estão sendo atendidas.

Boas propriedades

São também conhecidas como propriedades comportamentais, pois dependem da marcação inicial da rede (ou do estado inicial do sistema representado). As propriedades descritas a seguir não guardam nenhuma relação de dependência entre si:

Limitação (*Boundedness*)

Uma RdP é dita k-limitada ou simplesmente limitada se o número de fichas em cada lugar for menor ou igual a um número finito k, para qualquer marcação acessível a partir da marcação inicial. Se k for igual a 1, a rede é chamada de binária ou segura (*safe*).

Como um sistema real é necessariamente um sistema limitado, é claro que toda especificação realizável do sistema deve estar baseada em uma RdP limitada.

Vivacidade (*Liveness*)

Uma RdP é viva, para uma dada marcação inicial, se para toda marcação acessível for possível percorrer uma seqüência de disparos que contenha todas as transições da rede.

O conceito de rede viva está diretamente ligado à não existência de bloqueios (*deadlocks*) na evolução do sistema modelado.

Reversibilidade (*Reversibility*)

Uma RdP é reversível (diz-se também própria ou *reinicializável*), para uma dada marcação inicial, se para toda marcação acessível for possível percorrer uma seqüência de disparos que conduza de volta à marcação inicial.

A maior parte dos sistemas de controle tem funcionamento repetitivo, o que implica que as RdP que os representam devem ser reversíveis.

Propriedades específicas

Na realidade as propriedades específicas correspondem à semântica do sistema que está sendo modelado e traduzem-se na forma de restrições impostas ao projetista. É necessário, portanto, que a ferramenta de modelagem forneça meios de se provar que estas restrições estão sendo atendidas na especificação funcional. Como um exemplo de propriedade específica podemos citar a necessidade de garantir a exclusão mútua entre duas tarefas que precisam ter acesso a um mesmo recurso que não pode ser utilizado simultaneamente pelas mesmas.

Métodos de análise das propriedades

Serão descritos a seguir os métodos comumente empregados na determinação das propriedades que podem ser encontradas em sistemas representados por RdP.

i. Enumeração das marcações

É o chamado método básico para a determinação das propriedades. Consiste em construir o grafo de todas as marcações acessíveis a partir da marcação inicial, considerando o disparo de todas as transições sensibilizadas. O grafo é inspecionado utilizando diretamente a definição da cada propriedade. Os nós do grafo correspondem às marcações e os arcos às transições disparadas.

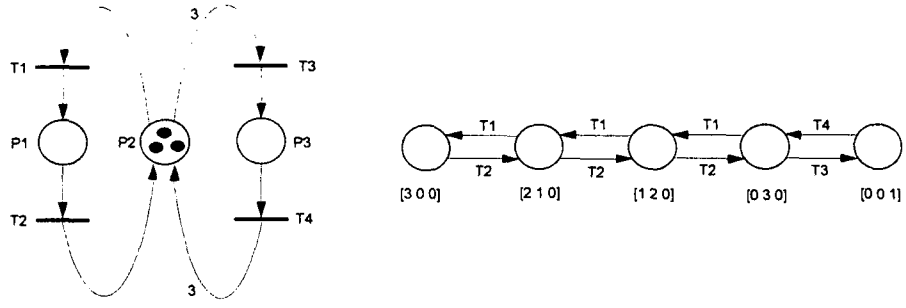
Todas as propriedades podem ser verificadas sobre o grafo:

- Limitação: para qualquer marcação do grafo, a marcação de todo lugar da rede é menor que um número k;

- Vivacidade: uma rede marcada é viva se todas as suas transições são vivas. Uma transição t é viva se para toda marcação do grafo, existe uma seqüência s de transições tal que t esteja sensibilizada;

- Reversibilidade: para qualquer marcação do grafo, existe uma seqüência s de transições que possa levar à marcação inicial.

Por exemplo, a partir da Rede de Petri da figura 3.7a foi gerado o grafo da figura 3.7b, que permite verificar que a rede é 3-limitada, viva e *reinicializável*.



(a) (b)

Fig. 3.7 - Processo Leitor-Escritor: (a) Rede de Petri, (b) Grafo

As limitações desse método são as seguintes:

- só as redes limitadas podem ser analisadas;

- as propriedades são fortemente ligadas à marcação inicial;

- mesmo que a rede seja limitada, o número de marcações acessíveis pode ser muito grande (é o que se chama de explosão combinatória do número de estados).

Uma maneira de se combater a explosão do número de estados é através da redução da rede analisada, método que será descrito a seguir.

ii. Análise por redução

Consiste em diminuir o tamanho (suprimindo lugares e transições) da RdP inicialmente construída, e conseqüentemente o tamanho do seu grafo de estados, sem mudar suas propriedades. A condição necessária e suficiente para que a rede original tenha as boas propriedades é que a rede reduzida também as tenha. Como inconvenientes desse método podemos citar:

- certas redes se reduzem mal ou não se reduzem de modo algum;

- perde-se rapidamente o significado dos lugares e das transições, o que faz com que a ausência de uma propriedade na rede reduzida não permita saber porque a original não a tem (é um método não construtivo);

- a rede reduzida nem sempre permite que se estude as propriedades específicas da rede original.

iii. Análise através de invariantes

É também chamada de análise estrutural, uma vez que ela se baseia na topologia (estrutura gráfica) da rede e é independente da marcação inicial. Esta análise vale para qualquer marcação inicial ou que diz respeito a certas seqüências de disparo a partir de uma dada marcação inicial. Existem dois tipos de invariantes: o de lugar e o de transição.

A idéia por trás do invariante de lugar consiste em achar um conjunto de lugares tais que a soma (eventualmente ponderada) do número de fichas contidas nesses lugares seja uma constante, ou seja, que ela permaneça inalterada apesar dos disparos das transições. Um invariante particular é definido por um vetor f que tem por dimensão o número de lugares da rede e onde cada componente representa um peso que se deve associar a um determinado lugar para garantir que a soma ponderada das fichas nos lugares seja constante. A determinação dos invariantes de lugar requer o cálculo das soluções do sistema de equações lineares $f.C = 0$ (tendo como incógnitas as marcações dos lugares), onde $C =$ *Post* - *Pre* é a matriz de incidência [Bra83].

Para a rede da figura 3.7a (problema dos leitores-escritores), obtém-se como solução o invariante de lugar $f = [1\ 1\ 3]$ que aplicado a uma marcação genérica e à marcação inicial $M^T = [0\ 3\ 0]$ resulta na seguinte equação (válida para qualquer marcação acessível a partir da marcação inicial):

$$M(P1)+M(P2)+3.M(P3)=3$$

Como o número de fichas em cada lugar é um inteiro não negativo, pode-se tirar da equação obtida as seguintes conclusões :

i. $M(P1) \leq 3$ ii. $M(P3) \leq 1$ iii. $M(P1).M(P3) = 0$

A primeira inequação mostra que até três leitores podem estar ativos ao mesmo tempo; a segunda mostra que no máximo um escritor pode estar ativo em um instante considerado e, por fim, a terceira expressão prova a exclusão mútua entre leitores e escritores.

Além das propriedades específicas, como no exemplo acima, o invariante de lugar permite obter informações sobre a propriedade de limitação: se $f(p)=1$, o lugar p é limitado, entretanto se $f(p)=0$, nada se pode afirmar.

Um invariante de transição define uma seqüência de disparos de transições que, desde que seja efetivamente realizável, não modifica a marcação da rede (a marcação final é igual à de partida). Essas seqüências correspondem portanto a comportamentos cíclicos da rede e a seqüências normais de funcionamento do sistema modelado. Um invariante de transição particular é definido por um vetor s que tem por dimensão o número de transições da rede e onde cada componente indica o número de vezes que a transição correspondente deve ser disparada. Para a determinação dos invariantes de transição é necessário calcular a solução do sistema de equações

lineares C.s = 0. Se s(t)=0, ou seja, se a componente associada a uma determinada transição t for nula, isso implica que a transição t é não viva. Entretanto, não se pode garantir que t é viva se s(t)=1.

No caso da rede da figura 3.7a, obtém-se como soluções os seguintes invariantes de transição: $s1^T$ = [1 1 0 0], $s2^T$ = [0 0 1 1] e $s3^T$ = [1 1 1 1], que correspondem, respectivamente, às seguintes seqüências cíclicas de disparo: T1-T2, T3-T4 e T1-T2-T3-T4, conforme pode ser comprovado pelo grafo da figura 3.7b.

Dentre as vantagens da análise através de invariantes podemos citar:

- não requer a enumeração das marcações acessíveis;

- permite estudar redes não limitadas e redes cuja marcação inicial não esteja completamente especificada;

- pode ser usado para a análise de sistemas de grande porte, uma vez que permite uma abordagem modular;

- seu uso durante o projeto de um sistema, utilizando a técnica *bottom-up* (composição de sub-redes) normalmente conduz a um projeto melhor estruturado.

As inconveniências do método são as seguintes:

- de um modo geral, não é conclusivo com relação à análise das propriedades viva e limitada;

- requer, para sua realização, habilidades consideravelmente maiores que as necessárias para os outros métodos, pois apesar de poderem ser calculados de modo automático, é preciso escolher qual a combinação adequada à prova da propriedade em questão e fazer a interpretação, que nem sempre é óbvia.

3.7.2 Análise de sistemas modelados por Redes de Petri Interpretadas

Os métodos de validação por análise vistos no item anterior são indicados para sistemas modelados por Redes de Petri autônomas (sem interação com o ambiente externo). Neste caso as evoluções das marcações não são sincronizadas com a ocorrência de eventos externos e há uma separação clara entre a parte de controle e a parte de dados, já que ambas são representadas utilizando apenas os elementos básicos de uma RdP: lugares, transições, arcos e fichas.

Mas o que muda na análise se o sistema estiver representado por uma RdP Interpretada ?

Neste caso deve-se, como procedimento geral, analisar inicialmente a rede subjacente, onde não são consideradas as informações das etiquetas mas apenas a estrutura da rede, dada pelas matrizes *Pre* e *Post*.

Esta análise é importante porque as evoluções das marcações da Rede de Petri Interpretada são restrições de evoluções das marcações da rede antes de sua interpretação (rede subjacente). De fato, para que uma transição possa ser disparada na Rede de Petri Interpretada, deve de toda maneira estar sensibilizada na rede subjacente. Mas uma transição pode estar sensibilizada na rede subjacente, mas não na rede interpretada se a variável externa associada à transição, indicando o estado de um sensor por exemplo, é falsa. Portanto, após a análise da rede subjacente, deve-se a seguir mostrar que a introdução da parte de dados não introduz nenhum mau funcionamento na rede, o que pode ser feito, por exemplo, através de uma ferramenta de simulação.

A primeira parte, que funciona como uma validação parcial do sistema, pode ser feita utilizando os métodos de análise já vistos e de maneira totalmente automatizada através de ferramentas computacionais.

Para verificar se a parte de dados complementa bem a representação do sistema deve-se mostrar que a rede interpretada é *determinista* e *determinada*. Em ambos os casos o ponto de partida é o conjunto de marcações acessíveis da rede autônoma.

Para mostrar que uma rede interpretada é *determinista* deve-se procurar as transições em conflito efetivo na rede subjacente, ou seja, que estão sensibilizadas pela mesma marcação mas o disparo de uma dessensibiliza as outras. Em seguida, para cada conjunto de transições em conflito, garante-se que as partes *condição* das etiquetas associadas são disjuntas, ou seja, que quando uma delas for verdadeira as outras serão falsas. Resolvendo-se, dessa forma, todos os conflitos existentes na rede subjacente obtém-se uma rede interpretada *determinista*.

Numa rede interpretada dita *determinada* as partes *ação* das etiquetas associadas a transições paralelas (sensibilizadas pela mesma marcação da rede subjacente e com disparos independentes: cada uma pode disparar ao mesmo tempo, antes ou depois das outras) não podem conter operações de atribuição envolvendo a mesma variável, já que isso poderia acarretar numa inconsistência dos dados. Assim, para garantir que uma rede interpretada é *determinada* deve-se procurar as transições que utilizem as mesmas variáveis na parte ação de suas etiquetas e mostrar, através do grafo de marcações acessíveis da rede subjacente, que elas não são paralelas.

Até agora nada foi dito sobre a rede interpretada onde aparecem variáveis de sincronização. O procedimento nesse caso consiste em localizar as transições onde tais variáveis aparecem na parte condição das etiquetas e em mostrar que não há possibilidade de bloqueios. No entanto, a inspeção do conjunto de estados acessíveis pode não ser viável, já que no caso das redes interpretadas cada estado acessível é formado por uma marcação acessível da rede subjacente e uma combinação das variáveis de sincronização, o que pode levar a uma explosão combinatória do número de estados ou mesmo a um número infinito de estados (mesmo que o número de marcações acessíveis da rede subjacente seja finito).

A melhor maneira de evitar a explosão combinatória é basear a análise na utilização de invariantes. Um invariante, nesse caso, é uma hipótese (envolvendo as variáveis de sincronização e o número de fichas dos lugares) que se quer mostrar válida para todos os estados do sistema. Para tanto, deve-se obter os invariantes lineares de lugar a partir da estrutura da rede subjacente e completar esses invariantes com outros que envolvam as variáveis de sincronização.

3.7.3 VALIDAÇÃO POR SIMULAÇÃO

No caso da RdP autônoma clássica (ordinária ou ponderada), a validação por análise permite verificar várias propriedades do sistema modelado. Entretanto, para outras classes e extensões, tais como a RdP Interpretada e a RdP Temporizada, isso não é mais possível. Surge então a necessidade de utilizar outras técnicas. Uma das técnicas mais empregadas é a da simulação, seja para detectar eventuais erros de modelagem ou para realizar cenários que permitam uma melhor compreensão do comportamento do sistema modelado.

Apesar de ser uma técnica não exaustiva, é possível obter uma validação por simulação mais refinada se se puder simular o conjunto sistema de controle/sistema controlado. Com base nesse argumento, pode-se fazer esse tipo de validação de três maneiras:

- conectando-se o controlador (com o programa de controle carregado) ao sistema a controlar e realizando ensaios: é o chamado teste *in situ*;

- conectando-se o controlador a uma emulador simulando, em tempo real, o comportamento do sistema a controlar: é o teste *hors-situ*;

- simulando o conjunto controlador/sistema a controlar em um mesmo computador: é o teste por *simulação*.

A primeira solução é a mais precisa, mas corre-se o risco de enfrentar situações que possam provocar danos materiais e até mesmo humanos. Esse tipo de validação só deve ser realizado quando houver a certeza de que todos os erros perigosos já foram detectados e corrigidos.

Na segunda solução não há o risco de situações perigosas e pode-se ser tão preciso quanto se queira. No entanto, a necessidade de se construir um emulador do sistema a controlar pode impor alguns entraves ao seu uso, principalmente no caso de aplicações simples ou durante a fase de detalhamento do programa de controle.

A última solução tem a vantagem de ser realizada em um mesmo computador, através de um *software* de simulação, sem precisar imobilizar os equipamentos reais. É particularmente indicada para as pequenas aplicações.

3.8 CONCLUSÃO

Esse capítulo foi dedicado à apresentação da Rede de Petri, o modelo formal de representação que serviu de base para o ambiente descrito no presente trabalho.

Inicialmente foram introduzidos os conceitos de base, comuns a qualquer classe de Rede de Petri: lugares, transições, arcos, marcações e a regra de disparo de uma transição.

Para representar a influência de eventos externos ao sistema modelado e sua evolução em função do tempo, foram abordadas duas extensões ao modelo de base: as Redes de Petri Interpretadas e as Redes de Petri Temporal e Temporizada.

No tocante à implementação de uma RdP, foi enfatizada a abordagem que utiliza a técnica do *jogador de fichas* e feita uma analogia entre seu funcionamento e o do *motor de inferência* utilizado pelos Sistemas de Regras de Produção utilizados na área de Inteligência Artificial.

Por fim, tratou-se de duas técnicas de validação que podem ser utilizadas em sistemas modelados por Redes de Petri: a validação por análise e a validação por simulação.

CAPÍTULO 4

O AMBIENTE INTEGRADO DE FERRAMENTAS

4.1 INTRODUÇÃO

O desenvolvimento da tecnologia computacional tem levado a um crescimento acelerado de métodos, técnicas e ferramentas que visam a solucionar problemas associados à concepção e à operação de Sistemas a Eventos Discretos (SED) tais como: redes de computadores, sistemas operacionais, automação da manufatura, robótica, sistemas de controle de tráfego, sistemas de gerenciamento de bases de dados, sistemas logísticos, dentre outros. Tais sistemas podem ser caracterizados como sistemas autônomos ou sistemas que interagem com o ambiente externo através de estímulos e respostas [Far93].

Dada a complexidade de um SED, a realização das etapas de análise e síntese do sistema de controle, com garantia de resultados satisfatórios, requer a adoção de metodologias baseadas em modelos formais que permitam: a representação completa, consistente e sem ambigüidades do sistema de controle; a análise e a validação dessa representação, tanto do ponto de vista do comportamento quanto do desempenho; o desenvolvimento de ferramentas automatizadas para utilização em todas as fases do ciclo de vida do sistema de controle (especificação, validação, implementação e teste) e na realização de protótipos.

No capítulo 3 foi apresentado o modelo formal de representação adotado nesse trabalho: a Rede de Petri. Mostrou-se que tal modelo atende aos requisitos exigidos no projeto do sistema de controle de um SED, pois é adequado a todas as etapas do ciclo de vida de seu desenvolvimento.

Neste capítulo será descrito o Ambiente Integrado de Ferramentas (AIF), cujo objetivo é fornecer ao projetista um ambiente que disponha de ferramentas baseadas no modelo formal utilizado na descrição do sistema e que o auxilie na realização das diferentes etapas necessárias à sua concepção, desde a especificação até a implementação.

4.2 DESCRIÇÃO GERAL DO AIF

Neste trabalho, o centro de interesse da utilização do Ambiente Integrado de Ferramentas é o apoio ao projeto do sistema de controle de um SED. Uma validação mais refinada do sistema de controle implica no estudo do conjunto sistema de controle-sistema controlado (planta ou processo).

O AIF foi concebido como um ambiente onde tanto a planta quanto o controle podem ser modelados por Redes de Petri Interpretadas, analisados e então simulados em conjunto. A análise permite ao projetista detectar, para cada rede, a existência de bloqueios (*deadlocks*), verificar exclusão mútua e testar as boas propriedades da rede subjacente. A simulação permite verificar a interação entre o controle e a planta. Como o comportamento da planta é também emulado, após esse passo a Rede de Petri pode controlar diretamente a planta a partir de sua interpretação por um programa jogador de fichas ou da sua tradução num programa em linguagem C.

Atualmente, os ambientes integrados de desenvolvimento costumam apresentar vários tipos de integração [Far93]: no nível de apresentação, dando uma visão uniforme ao usuário, qualquer que seja o modelo e a ferramenta utilizados; no nível de dados, permitindo o compartilhamento e a troca de dados entre ferramentas (por exemplo, se uma ferramenta é baseada em autômatos e outra em Rede de Petri); no nível de controle, permitindo controlar a ativação de ferramentas; no nível de compatibilização, permitindo a conversão e a transmissão de arquivos de um ambiente de execução a outro.

O ambiente integrado aqui descrito está baseado num único modelo de representação, a Rede de Petri. No entanto, como o ambiente também utiliza ferramentas existentes - o ARP, Analisador de Redes de Petri [Maz90] e o SPP [Pal91], tradutor de Redes de Petri escritas na forma de regras de produção em um programa em código C - haverá, naturalmente, diferentes formas de representação interna do sistema modelado através de uma Rede de Petri. Assim, a integração ocorrerá no nível de controle, viabilizando a ativação das diferentes ferramentas, e no nível de compatibilização, possibilitando as conversões entre as diferentes representações internas utilizadas pelas ferramentas.

A arquitetura do AIF, representada na figura 4.1, é constituída de quatro módulos: Edição, Análise, Simulação e Tradução, descritos resumidamente a seguir.

Módulo de Edição

O módulo encarregado da leitura das redes e da criação das estruturas de dados internas é o módulo de Edição. Tais estruturas só serão criadas se as descrições das redes estiverem sintática e semanticamente corretas.

Módulo de Análise

O módulo de Análise permite a análise através dos métodos de enumeração de marcações, procura de invariantes (de lugar e de transição), equivalência de linguagem, simulação de uma rede e avaliação de desempenho. Embora o AIF tenha sido concebido para tratar com sistemas modelados por RdP Interpretadas, o estudo da rede subjacente (sem as etiquetas associadas às transições) deve ser visto como uma etapa inicial (pré-validação) no processo de

validação de tais sistemas. Esse módulo utiliza o *software* ARP retrocitado.

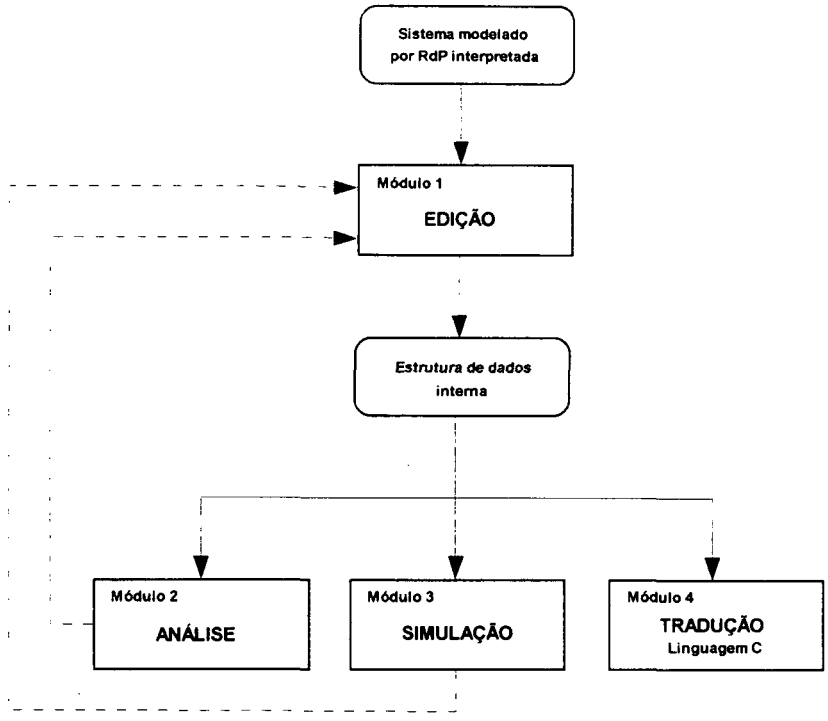


Fig. 4.1 - Arquitetura do AIF

Módulo de Simulação

Este módulo simula o controle e emula a planta, estando ambos representados através de Redes de Petri Interpretadas. A emulação da planta consiste em criar um modelo executável da mesma que, ao interagir com o sistema de controle durante a simulação, comporte-se como se fosse a planta real; tal modelagem do comportamento da planta aparecerá para o sistema de controle como uma série de reações sucessivas separadas por intervalos de tempo normalmente conhecidos e as Redes de Petri se adaptam muito bem a esse tipo de modelagem. As condições suplementares associadas às transições na rede que representa o sistema de controle são as mensagens enviadas pelos sensores; as ações associadas às transições dessa rede são os sinais de comando enviados à planta. No módulo de Simulação, os sensores e as tarefas ativadas pelos atuadores são emulados pela rede que descreve a planta. Este módulo será descrito em detalhes no próximo capítulo.

Módulo de Tradução

Fechando o ciclo de desenvolvimento de um sistema de controle, o módulo de Tradução permite que a partir da especificação validada (isto é, após sua análise utilizando os módulos de Análise e Simulação) do sistema de controle, seja gerado um programa-fonte em linguagem C.

Como indicado na figura 4.1, se após a validação por análise ou simulação for detectado algum erro na especificação a rede pode ser modificada e revalidada.

Os próximos itens descrevem mais detalhadamente os módulos de Edição, Análise e Tradução. O módulo de Simulação é objeto do próximo capítulo.

4.3 Módulo de Edição

Esse módulo engloba todas as funções que estão ligadas, direta ou indiretamente aos arquivos que contêm as descrições das redes. São elas:

- Leitura de uma rede existente: mostra, em ordem alfabética, a lista dos arquivos que contêm as descrições das redes existentes. Escolhido o arquivo, é verificada a correção sintática e semântica da descrição da rede. Se a descrição estiver correta, a rede é colocada em uso, ou seja, as estruturas de dados criadas durante a leitura estão aptas para serem utilizadas pelo ambiente;

- Criação de uma rede nova: pode ser feita através de dois modos. Em um deles (modo interativo) as informações necessárias à criação da rede são solicitadas ao usuário e à medida que vão sendo fornecidas é feita uma verificação na consistência dos dados, evitando que sejam cometidos erros sintáticos ou semânticos; é particularmente indicado para redes de pequeno porte (e.g.: 10 lugares e 10 transições) ou para a criação parcial de uma rede. No outro (modo editor), é utilizado um editor de textos, através do qual o usuário entra com a descrição da rede utilizando a sintaxe da linguagem de descrição. Este editor é bem adequado à sua finalidade; no entanto, a criação do arquivo contendo a descrição da rede pode ser feita através de qualquer editor de textos (padrão ASCII);

- Edição de uma rede existente: caso o usuário queira modificar uma rede existente, seja para corrigir algum erro detectado na leitura ou para complementar sua descrição, pode fazer uso dessa opção, utilizando o mesmo editor de textos para a criação de redes novas no modo editor;

- Acesso ao sistema operacional: permite que se acesse temporariamente o sistema operacional sem precisar terminar a execução do AIF. Isso pode ser útil, por exemplo, quando se desejar mudar o diretório e/ou o acionador de disco corrente.

A linguagem de descrição utilizada está exemplificada na figura 4.2, que representa a descrição correspondente à Rede de Petri Interpretada da figura 2.5.

```
REDE FIG_2.5;

NODOS

  P1
  : Lugar(1);

  P2
  : Lugar(0);

  T1, T2
  : Transicao;

ESTRUTURA

  T1:   (P1) , (P2);
  T2:   (P2) , (P1);

FIM


VARIAVEIS

  F,S, Liga_maq, Desliga_maq
  : Externa(0);

 CONDICOES_ACOES

  T1:   (S=1) , (Liga_maq=1);
  T2:   (F=1) , (Desliga_maq=1);

FINAL
```

Fig. 4.2 - Linguagem de descrição

Os lugares são definidos após a palavra-chave NODOS agrupados segundo a marcação inicial: o lugar P1 tem 1 ficha - Lugar(1) - enquanto os demais não estão marcados - Lugar(0). As transições são também listadas dentro de NODOS. Para cada transição *t*, os arcos de entrada são indicados na ESTRUTURA: os lugares de entrada são colocados no primeiro par de parênteses separados por vírgulas, e os de saída no segundo par de parênteses.

A palavra-chave FIM indica ao módulo de Análise que apenas esta parte do arquivo, que descreve a rede subjacente, será utilizada. O restante do arquivo, iniciando na palavra-chave VARIAVEIS e terminando na palavra-chave FINAL, representa a parte das etiquetas da rede e será utilizada pelos módulos Simulação e Tradução. Em VARIAVEIS são indicados se as variáveis (booleanas) são internas ou externas e qual o valor inicial. Para cada transição é indicada a lista de condições (primeiro par de parênteses) e ações (segundo par de parênteses) sobre as variáveis internas ou externas, separadas por vírgula. Se houver mais de uma condição (ou ação), as mesmas devem ser separadas por vírgulas dentro dos respectivos parênteses.

A figura 4.3 mostra o menu referente a este módulo.

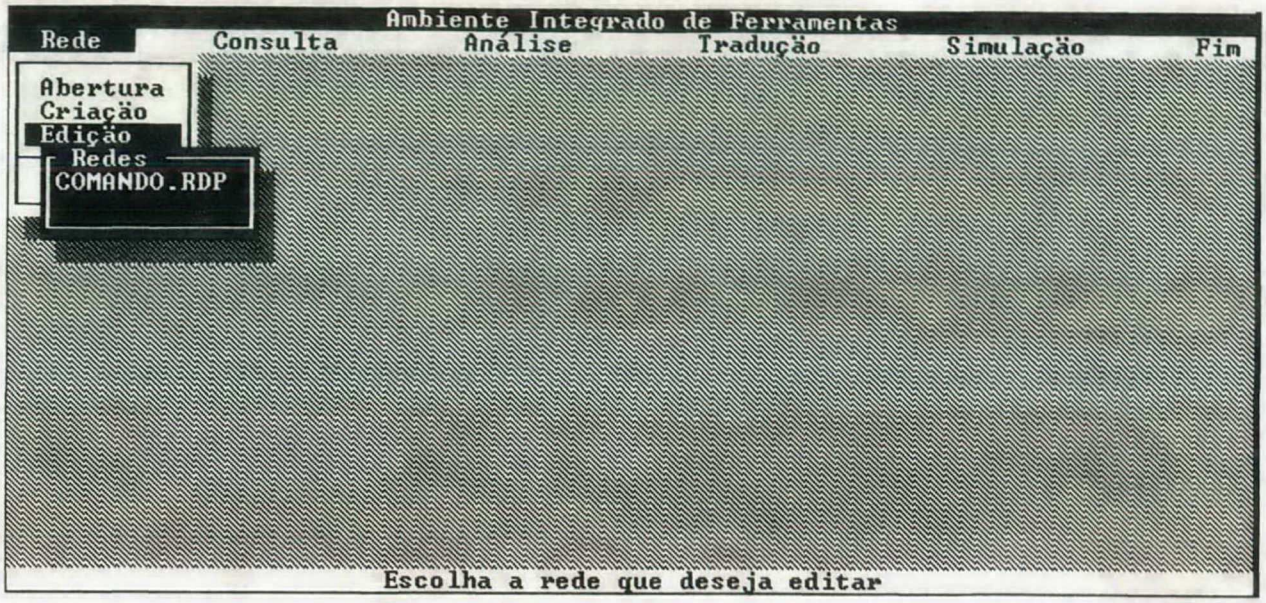


Fig. 4.3 - Menu do Módulo de Edição

4.4 MÓDULO DE ANÁLISE

O módulo de Análise utiliza um programa externo e independente, denominado ARP (Analisador de Redes de Petri) [Maz90] [Far90], que foi integrado ao AIF e, em conseqüência, pode ser chamado a partir dele e aceita os mesmos arquivos de entrada.

Apesar de não tratar com RdP Interpretada, o ARP será de grande valia no estudo da rede subjacente, ou seja, a rede que resulta quando são desconsideradas as informações associadas às etiquetas das transições de uma RdP Interpretada. Tal estudo deve ser visto como o primeiro passo rumo à validação de um sistema descrito por uma RdP Interpretada, uma vez que algumas propriedades da rede subjacente são também válidas para a rede interpretada (e.g.: uma rede subjacente binária implica em uma rede interpretada binária).

O fato da linguagem de descrição adotada no AIF ter sido baseada na linguagem de descrição do ARP, permite que esse aceite os mesmos arquivos gerados por aquele (a parte das etiquetas é automaticamente desconsiderada).

Em sua versão atual (versão 2.3), o ARP trabalha com Redes de Petri do tipo Lugar/Transição (que engloba as redes binárias, ordinárias e ponderadas) com temporização e temporização estendida. Suas funcionalidades estão distribuídas pelos seguintes módulos:

- Edição/compilação das redes;
- Análise das propriedades comportamentais (Limitação, Vivacidade e Reversibilidade) através da construção e inspeção do grafo de marcações acessíveis;

- Cálculo dos invariantes de lugar e de transição;
- Verificação por equivalência de linguagem;
- Simulação;
- Avaliação de desempenho (para redes com temporização e temporização estendida com probabilidades).

4.5 Módulo de Tradução

O objetivo deste módulo é a geração, a partir do modelo validado do sistema de controle, de um programa-fonte em linguagem C que, em tempo de execução, tenha o mesmo comportamento do modelo validado. Isso é feito através de um programa chamado SPP (Sistema de Produção Proposicional) [Pal91] .

O SPP utiliza técnicas de inteligência artificial para desenvolver sistemas especialistas. Sendo um programa de aplicação geral, pode resolver problemas em diversas áreas, incluindo a de sistemas de controle em tempo real (ou seja, com restrições de tempo). O modelo de base utilizado para a representação do conhecimento foi o dos Sistemas de Produção (também chamados de Sistemas de Regras de Produção, uma vez que o conhecimento é representado através de regras que ao serem executadas vão produzindo mais conhecimento).

O fato de uma especificação descrita através de uma RdP Interpretada poder ser facilmente traduzida em forma de regras, conforme descrito no capítulo 2 (item 2.6), aliado ao fato do mecanismo de execução dessas regras (utilizado no SPP) ter funcionamento similar ao do mecanismo de disparo de transições de uma RdP Interpretada (utilizado pelo módulo de Simulação do AIF) permitiu a integração do SPP ao AIF.

As opções disponíveis nesse módulo são as seguintes:

- Tradução para a forma de regras: permite a criação de um arquivo contendo a tradução da rede (do sistema de controle) na forma de regras, já na sintaxe requerida pelo SPP. Esse arquivo pode ser editado e modificado pelo usuário.

- Geração do programa-fonte em C: chama o SPP que, utilizando como entrada o arquivo criado pela opção anterior, gera um outro arquivo contendo o código-fonte, em linguagem C, do programa de controle. Não se pode utilizar essa opção sem antes ter traduzido a rede para a forma de regras (opção anterior).

4.5.1 Interface com a Planta

Tendo em vista que o programa-fonte gerado será utilizado para comandar uma planta real, isso implica que deverá haver uma interface de comunicação entre ambos. Para facilitar a

construção dessa interface o SPP dispõe de mecanismos que permitem chamadas a funções externas ao programa de controle. Através dessas funções os valores dos atuadores podem ser passados para a planta e os valores dos sensores podem ser recebidos da mesma.

Recomenda-se que a chamada do módulo de Tradução seja feita somente após a passagem pelos módulos de Análise e Simulação, uma vez que nesse momento espera-se que o modelo que representa o sistema de controle já esteja validado.

4.6 CONCLUSÃO

Esse capítulo foi dedicado à apresentação do AIF, iniciando com seu objetivo e uma visão geral dos módulos que compõem sua arquitetura.

Em seguida foram descritos, com mais detalhes, os módulos de Edição, Análise, e Tradução. Foram integrados ao ambiente os *softwares* ARP de análise de Redes de Petri e o SPP - tradutor de Redes de Petri escrita na forma de regras em código-fonte em linguagem C.

No próximo capítulo será visto em detalhe o módulo de Simulação, em cuja concepção está a ênfase desse trabalho.

CAPÍTULO 5

A SIMULAÇÃO NO AIF

5.1 INTRODUÇÃO

Na simulação de um sistema de controle, um modelo executável do mesmo é desenvolvido e observado. A execução do modelo pode contar com o suporte de uma ferramenta computacional ou pode ser feita de modo totalmente manual (e. g.: numa folha de papel ou até mesmo mentalmente) no caso de sistemas simples. É um método bastante utilizado para o entendimento de um sistema e para a procura de erros, particularmente na fase de sua especificação funcional e nas fases iniciais de sua validação.

Mesmo sendo um método não exaustivo (permite encontrar erros em uma especificação, mas não prova sua total correção), é possível obter uma validação por simulação mais refinada se se puder contar com um suporte computacional e com a emulação do sistema controlado (planta). O apoio do computador permitirá uma simulação mais rápida e sem erros. A utilização de um modelo executável emulando a planta e interagindo com o modelo executável do sistema de controle permitirá detectar as incoerências na troca de informações entre os dois sistemas e, sobretudo, estudar o comportamento em situações de pane.

Foram essas idéias que motivaram a concepção do módulo de simulação do AIF.

5.2 REQUISITOS BÁSICOS PARA A SIMULAÇÃO

Primeiramente é preciso que os sistemas (controle e planta) estejam modelados por RdP Interpretadas. Durante a etapa de criação dos arquivos contendo a descrição de cada rede deve-se garantir idêntica declaração das variáveis externas (mesmo nome e mesmo valor inicial) nos dois arquivos. Assim, essas variáveis, que representam os sensores e os atuadores, serão comuns às duas redes e formarão a interface de comunicação entre elas (figura 5.1). Assim, quando da execução dos modelos, essa interface possibilitará a troca de informações entre os mesmos e simulará a interface real entre o sistema de controle e a planta.

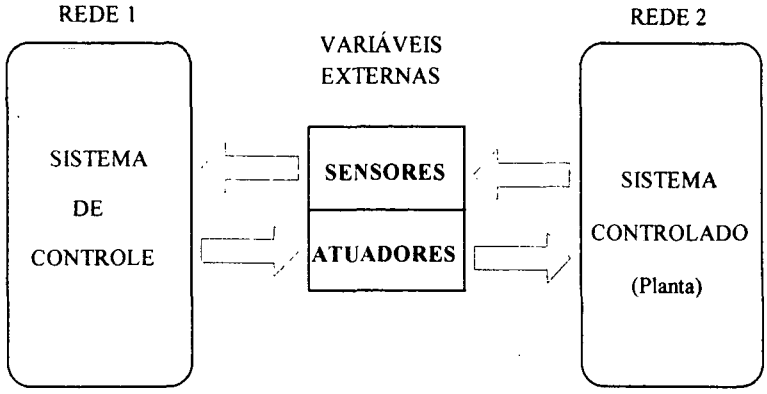


Fig. 5.1 - Interface de comunicação entre os modelos

5.3 ARQUITETURA DO MÓDULO DE SIMULAÇÃO

Os principais elementos que compõem esse módulo estão representados na figura 5.2: *função jogador, função escalonador, tabela de transições temporizadas e função interface com o usuário.*

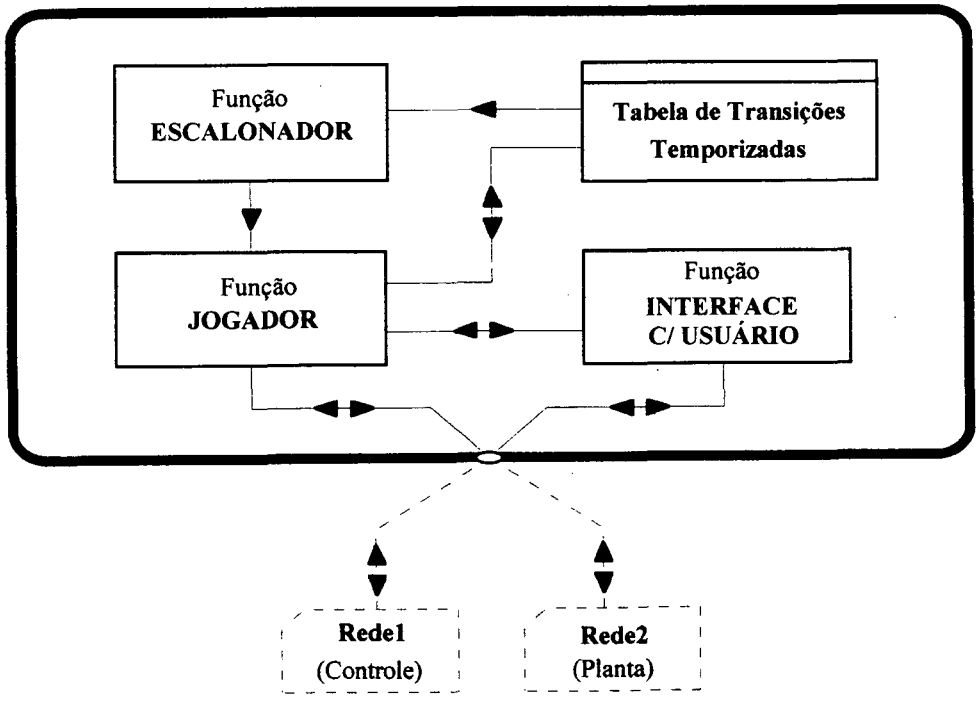


Fig. 5.2 - Arquitetura do Módulo de Simulação

Função jogador

A execução dos modelos fica a cargo da *função jogador*, que é um programa jogador de fichas para RdP do tipo Lugar/Transição (com ou sem etiquetas associadas às transições). O papel dessa função é, para cada rede e partindo do seu estado atual, procurar as transições disparáveis e dispará-las, até atingir um estado onde não haja mais transições disparáveis na rede, o chamado estado estável.

Função escalonador

Cumpre à *função escalonador* gerenciar o tempo (virtual) durante a simulação. Ao ser chamada, ela indexa a *tabela de transições temporizadas* (ordem crescente de instantes de disparo) e, com base nas informações da primeira transição, atualiza o tempo e indica a próxima rede a ser executada.

Tabela de transições temporizadas

É formada por transições das duas redes e contém, para cada entrada, as seguintes informações: nome da transição, rede à qual pertence e instante de disparo previsto. É acessada tanto pela *função escalonador* como pela *função jogador* e é fundamental para a representação do comportamento dinâmico dos sistemas modelados.

Durante o acesso da *função escalonador* à tabela, suas transições são colocadas em ordem crescente de instantes de disparo: se houver duas transições com mesmo instante de disparo, mas pertencendo a redes diferentes, então, por convenção, será dada prioridade à transição que pertencer à rede de controle.

Já a *função jogador* faz dois tipos de acesso à tabela: em um, dispara as transições cujo instante de disparo é igual ao tempo atual; no outro, insere na mesma as transições habilitadas mas com instantes futuros de disparo

Função interface com o usuário

A simulação pode ser interrompida pelo usuário sempre que ele julgar conveniente. Ao fazê-lo, o controle da simulação é passado para a *função interface com o usuário*. Nesse momento os estados dos sistemas modelados e o tempo (virtual) de simulação são congelados, de modo que a simulação poderá ser reiniciada a partir do ponto de interrupção. Tal função coloca à disposição do usuário um menu com uma série de opções cujo objetivo é aumentar a flexibilidade e a eficiência na condução da simulação.

5.4 CICLO DE FUNCIONAMENTO DA SIMULAÇÃO

Tendo sido introduzidas as principais funções que compõem o módulo de simulação, é possível agora descrever o ciclo de funcionamento da simulação. Tal ciclo está representado pela RdP Interpretada da figura 5.3, que será usada como referência para as explicações que se seguem.

Lugar P1: *início da simulação.*

Lugar P2: é solicitada ao usuário a escolha das redes que representam os sistemas de interesse.

Lugar P3: após a checagem sintática e semântica das redes, é verificada a coerência da interface de comunicação entre ambas. Enquanto houver erro na descrição das redes ou na interface de comunicação a simulação não poderá ser iniciada (volta para o lugar P2).

Lugar P4: uma vez corretas as descrições das redes e a interface de comunicação (transição t4), o controle da simulação é passado para a *função interface com o usuário*, que coloca ao seu dispor uma série de opções para a condução da simulação, dentre as quais a que permite iniciar a execução das redes através da chamada à *função jogador.*

Lugar P5: é chamada a *função jogador*, que executa a rede em uso (por convenção, na primeira vez em que essa função é chamada, a rede em uso é sempre a do sistema de controle) levando-a até um estado estável. O usuário pode interromper a execução da rede sempre que o desejar (transição t6); nesse caso o controle da simulação retorna para a *função interface com o usuário* (volta ao lugar P4).

Lugar P6: atingido o estado estável da rede em uso (transição t7) é verificado o número de vezes que a *função jogador* foi chamada desde o início da simulação.

Lugar P7: caso a *função jogador* tenha sido chamada apenas uma vez (transição t9) ou tenha havido algum disparo na última rede executada (transição t10), coloca-se em uso a outra rede (transição t12 ou t13) e chama-se a *função jogador* (volta ao lugar P5). A primeira condição garante que a simulação se iniciará com a execução incondicional das duas redes e a segunda baseia-se na possibilidade dos disparos da última rede terem provocado o aparecimento de transições disparáveis na outra rede.

Lugar P8: caso a *função jogador* tenha sido chamada mais de uma vez (transição t8), verifica-se se houve algum disparo na última rede executada.

Lugar P9: não tendo havido disparos na última rede (transição t11), isso significa que não existem mais transições disparáveis nas duas redes, no instante atual. Então deve ser verificada a existência de transições, potencialmente disparáveis em instantes futuros, na *tabela de transições temporizadas.*

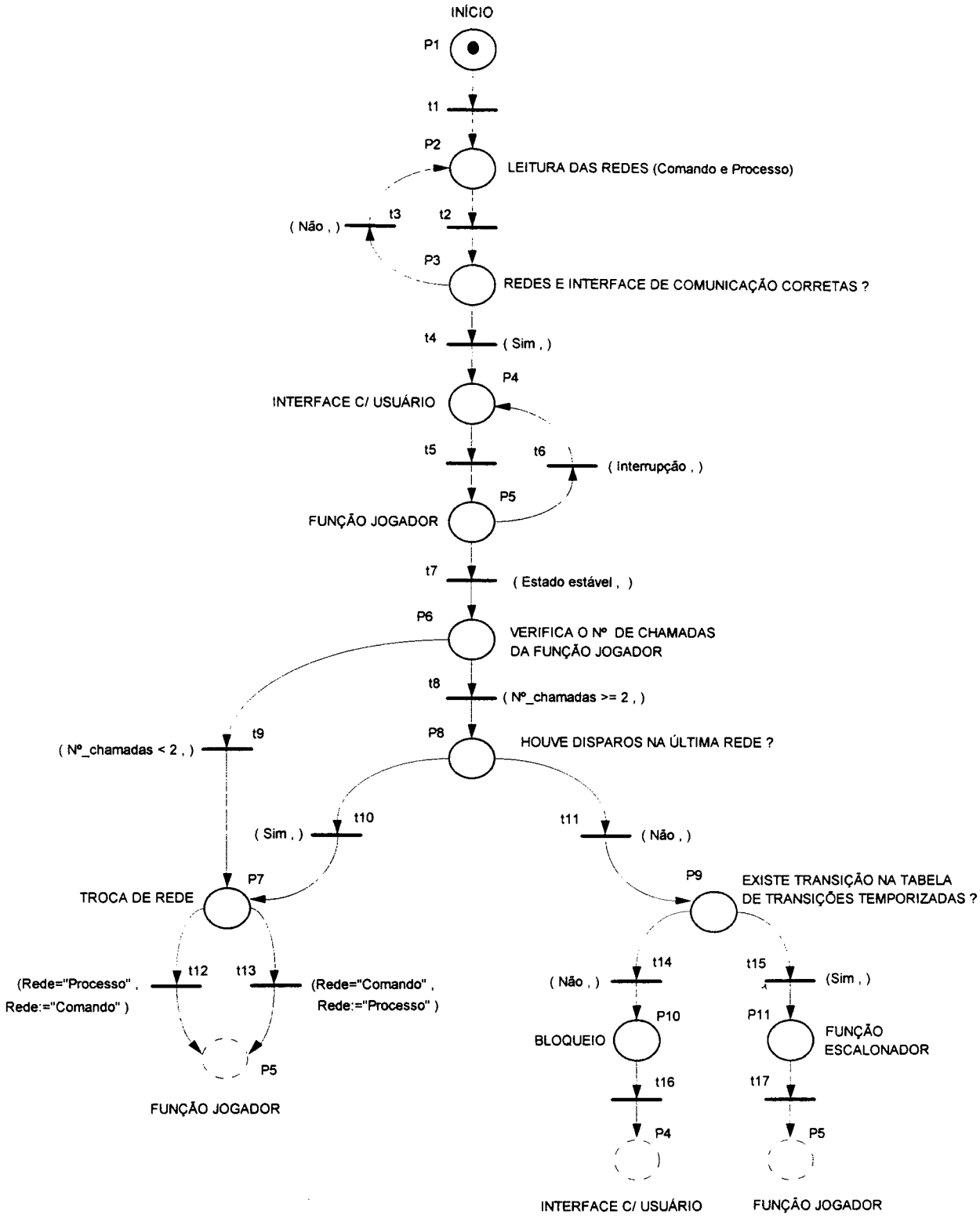


Fig. 5.3 - Ciclo de funcionamento da simulação

Lugar P10: se a *tabela de transições temporizadas* estiver vazia (transição t14) atinge-se a situação de bloqueio na simulação, ou seja, não há transições disparáveis nas duas redes. Nesse caso é enviada uma mensagem ao usuário e o controle da simulação é passado (transição t16) para a *função interface com o usuário* (volta ao lugar P4).

Lugar P11: existindo alguma transição na *tabela de transições temporizadas* (transição t15) é chamada a *função escalonador,* que indexa a tabela, atualiza o tempo de simulação e indica qual rede será executada. Após isso (transição t17), é chamada a *função jogador* (volta ao lugar P5) que executará a rede indicada.

5.4.1 JUSTIFICATIVAS PARA A ESTRATÉGIA ADOTADA

Analisando o ciclo de funcionamento da simulação (figura 5.3), pode-se perceber que durante o seu curso são feitas várias tomadas de decisão. O conjunto dessas decisões constitui a estratégia adotada na condução da simulação. As justificativas que motivaram a elaboração dessa estratégia serão dadas a seguir, na forma de perguntas e respostas.

Por que somente após duas chamadas da *função jogador* é que se verifica se houve disparos na última rede em uso ?

Isso ocorre sempre que se inicia (ou reinicia) um ciclo da simulação e tem por objetivo permitir o correto preenchimento da *tabela de transições temporizadas* com transições das duas redes (se for o caso). De outra forma, se logo após a execução da primeira rede fosse verificado que não houve disparos e que a *tabela de transições temporizadas* não está vazia, então a *função escalonador* atualizaria o tempo de simulação e as transições com instante de disparo igual ao tempo atual seriam disparadas pela *função jogador*, o que poderia implicar num erro, pois nada impediria que a outra rede (que nesse caso ainda não teria sido executada) apresentasse transições habilitadas com instantes de disparos inferiores.

Por que é necessário que não tenha havido disparos na última rede em uso para se poder chamar a *função escalonador* ?

Antes de mais nada, convém deixar claro que o AIF foi concebido para ser executado em computadores baseados em um único processador e que, portanto, não é possível representar o paralelismo verdadeiro que existe na evolução real dos dois sistemas modelados. Como o tempo (virtual e absoluto) de simulação só é atualizado pela *função escalonador*, é necessário, para produzir o efeito de paralelismo (pseudoparalelismo), que antes de sua chamada todas as transições habilitadas (as não temporizadas e as temporizadas com instante de disparo igual ao tempo atual de simulação) de ambas as redes tenham sido disparadas. Garante-se, assim, que todos os eventos (disparos de transições) que ocorrem entre duas chamadas da *função escalonador* ocorrem no mesmo instante de tempo, o que permite interpretá-los como eventos paralelos. O fato de não ter havido disparos na execução da última rede em uso permite que se

conclua que não existem mais transições disparáveis (com a ausência de disparos em uma rede, nenhuma transição pode-se tornar habilitada na outra rede) no tempo atual de simulação e que, caso a *tabela de transições temporizadas* não esteja vazia, deve-se chamar a *função escalonador* para atualizar o tempo.

Por que a ausência de disparos na última rede em uso e uma *tabela de transições temporizadas* vazia implicam em um bloqueio na simulação ?

Foi visto no caso anterior que o fato de não ter havido disparos na última rede implicava na ausência de transições disparáveis no tempo atual. Isso ainda não caracteriza um bloqueio, pois a existência de transições na *tabela de transições temporizadas* implica em potenciais disparos em instantes futuros. Por outro lado, se, adicionalmente, a tabela estiver vazia não haverá a possibilidade de disparos futuros, caracterizando uma ausência total de transições disparáveis nas duas redes, ou seja, um bloqueio de fato, significando que o diálogo entre o sistema de controle e a planta teve um comportamento não previsto, normalmente provocado por um erro de modelagem.

5.5 FUNÇÃO JOGADOR

O papel da *função jogador* é, através da procura e do disparo das transições habilitadas, fazer evoluir cada rede até um estado estável. Antes de descrever seu algoritmo de funcionamento serão introduzidos os conceitos de transição habilitada e lugar-chave.

5.5.1 TRANSIÇÃO HABILITADA

Em se tratando de RdP Interpretadas, uma transição é dita habilitada ao disparo (ou disparável) se ela estiver sensibilizada pela marcação atual da rede e tiver a parte condição de sua etiqueta avaliada como verdadeira. O conceito de transição sensibilizada (que vale para qualquer rede do tipo Lugar/Transição e que no caso de redes não interpretadas se confunde com o de transição habilitada) diz que uma transição é considerada sensibilizada pela marcação atual da rede quando todos os seus lugares de entrada possuírem um número de fichas maior ou igual ao peso dos respectivos arcos de saída.

5.5.2 LUGAR-CHAVE

A procura de transições sensibilizadas é sem dúvida a etapa mais demorada da *função jogador*, pois pesquisar todo o conjunto de transições implica na realização de muitos testes inúteis (já que nem todos os lugares estarão marcados ou conterão fichas suficientes).

Para otimizar o processo de procura das transições sensibilizadas a *função jogador* utiliza o conceito de lugar-chave [Bar87]. A idéia básica é escolher um lugar-chave para cada transição (deve ser qualquer um de seus lugares de entrada) e, na etapa de procura das transições

sensibilizadas, pesquisar somente aquelas que possuam lugares-chaves que pertençam ao conjunto de lugares marcados da rede. No entanto, o fato de ter seu lugar-chave marcado não garante a sensibilização de uma transição (é condição necessária, mas não suficiente), pois ela pode, por exemplo, ter um outro lugar de entrada não marcado. Para tentar minimizar esse problema utiliza-se o conceito de lugar-chave dinâmico [Bar87]: ao se encontrar uma transição que possui seu lugar-chave marcado mas que não está sensibilizada, escolhe-se como novo lugar-chave dessa transição o primeiro lugar de entrada não marcado encontrado; assim, na próxima etapa de procura, caso o novo lugar-chave ainda continue sem ficha, essa transição não será inutilmente testada.

Diante do exposto, fica claro que o conceito de lugar-chave, apesar de melhorar o desempenho da *função jogador*, não evita totalmente os testes infrutíferos. A aplicação do conceito de lugar-chave é particularmente indicada para a execução de redes binárias (caso dos automatismos lógicos).

5.5.3 ALGORITMO DE FUNCIONAMENTO

O algoritmo de funcionamento da *função jogador* está representado pela RdP da figura 5.4, na qual estão baseadas as explicações dadas a seguir.

Lugar P1: é chamada a *função jogador*, que assume o controle da simulação.

Lugar P2: verifica se a *tabela de transições temporizadas* está vazia.

Lugar P3: caso a *tabela de transições temporizadas* não esteja vazia (transição t3), lê os dados associados à primeira transição da tabela (nome da transição, rede à qual pertence e instante de disparo) e verifica se a rede associada à transição é a rede em uso e se o instante de disparo da transição coincide com o tempo atual da simulação.

Lugar P4: caso a transição lida pertença à rede em uso e seu instante de disparo seja igual ao tempo atual (transição t5), é verificado se a mesma ainda está habilitada ao disparo.

Lugar P5: estando ainda habilitada (transição t6), é feito o disparo da transição.

Lugar P6: caso a transição não esteja mais habilitada ao disparo (transição t7) ou tenha sido disparada (transição t8), é feita sua retirada da tabela. Concluída a retirada (transição t9), deve-se verificar se a *tabela de transições temporizadas* tornou-se vazia (volta ao lugar P2).

Lugar P7: caso a *tabela de transições temporizadas* esteja vazia (transição t2) ou sua primeira transição não pertença à rede em uso ou possua instante de disparo maior que o tempo atual (transição t4), é feita a leitura do próximo lugar da lista de lugares marcados (caso tenha havido algum disparo, a *função jogador* recomeça a leitura pelo primeiro lugar da lista, pois o disparo pode ter modificado a marcação).

Lugar P8: verifica se o lugar marcado lido é lugar-chave de alguma transição.

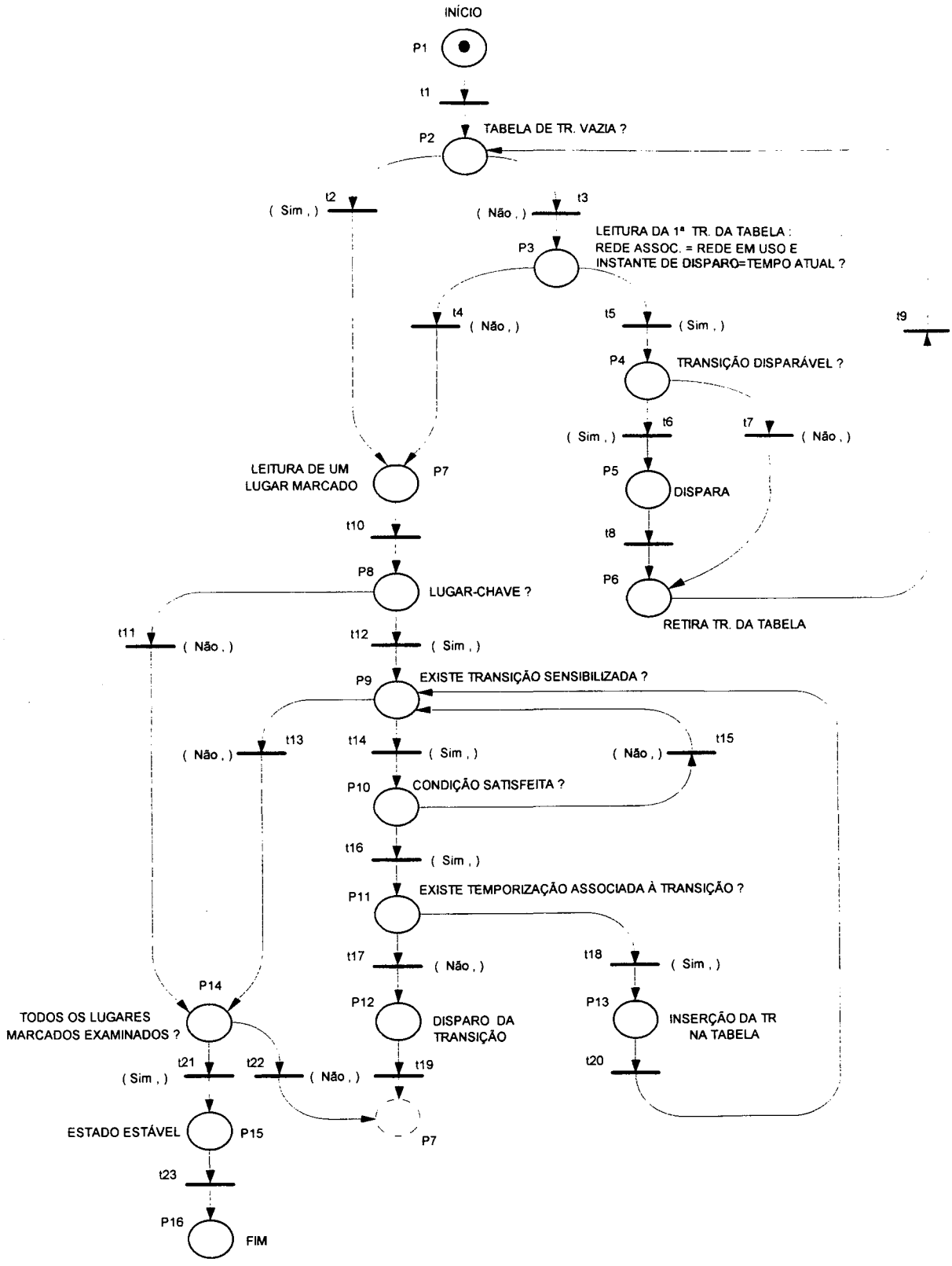


Fig. 5.4 - Algoritmo de funcionamento da função jogador

Lugar P9: sendo o lugar lido um lugar-chave (transição t12), cria uma lista contendo as transições associadas e procura a primeira que estiver sensibilizada pela marcação atual.

Lugar P10: existindo uma transição sensibilizada na lista relativa ao lugar-chave (transição t14), verifica se a parte condição de sua etiqueta é verdadeira. Se não for (transição t15), procura a próxima transição sensibilizada da lista (volta ao lugar P9).

Lugar P11: caso a condição associada à transição seja verdadeira (transição t16), verifica se a mesma possui temporização associada.

Lugar P12: não havendo temporização associada à transição (transição t17), efetua-se o seu disparo. Concluído o disparo (transição t19), reinicia a leitura dos lugares marcados (volta ao lugar P7).

Lugar P13: havendo temporização associada à transição (transição t18), cabe à *função jogador* programar o instante futuro de disparo da mesma e inseri-la na *tabela de transições temporizadas*. Feito isso (transição t20), procura a próxima transição sensibilizada da lista relativa ao lugar-chave (volta ao lugar P9).

Lugar P14: se o lugar marcado lido não for lugar-chave (transição t11) ou se nenhuma das transições que o têm como lugar-chave estiver sensibilizada (transição t13), então verifica se o lugar lido é o último da lista de lugares marcados. Se não for (transição t22), lê o próximo lugar da lista de lugares marcados (volta para o lugar P7).

Lugar P15: tendo sido percorrida toda a lista de lugares marcados (transição t23) chega-se a um estado estável da rede em uso e, portanto, termina sua execução..

5.6 FUNÇÃO ESCALONADOR

Cabe à *função escalonador* organizar a *tabela de transições temporizadas* e, através da mesma, gerenciar a evolução do tempo de simulação. Antes de descrever seu algoritmo de funcionamento serão feitas algumas considerações sobre o tratamento do tempo durante a simulação.

5.6.1 O TEMPO NA SIMULAÇÃO

Durante a simulação pode-se atribuir a qualquer transição das redes envolvidas um valor de temporização, em uma unidade de tempo qualquer. Para a rede que representa o sistema controlado essa temporização permite tornar o comportamento do modelo mais próximo do comportamento do sistema real. Com relação à rede que representa o sistema de controle, permite a utilização de transições do tipo "cão de guarda" (*watchdog*), que só serão disparadas em caso de alguma anormalidade no sistema comandado (o que pode ser útil na simulação de panes).

A evolução do tempo de simulação não guarda nenhuma relação com o passar do tempo real. Em outras palavras, mesmo sendo definido em unidades de tempo, é um tempo simbólico. Sua contagem se inicia a partir do zero e, de forma absoluta, é atualizada somente quando a *função escalonador* é chamada. Assim, quaisquer disparos (em ambas as redes) entre duas chamadas dessa função ocorrem no mesmo instante (virtual) de tempo. Caso nenhuma temporização seja associada às transições das duas redes, todos os disparos ocorrerão no instante zero, uma vez que a *função escalonador* nunca será chamada (a *tabela de transições temporizadas* estará sempre vazia) e, portanto, o tempo de simulação nunca será atualizado.

5.6.2 ALGORITMO DE FUNCIONAMENTO

O algoritmo de funcionamento da *função escalonador* está representado pela RdP da figura 5.5. As explicações que se seguem utilizam essa figura como referência.

Lugar P1: é chamada a *função escalonador*, que assume o controle da simulação.

Lugar P2: a *tabela de transições temporizadas* é indexada por instantes crescentes de disparo.

Lugar P3: é feita a leitura dos dados associados à primeira transição da tabela (nome da transição, rede à qual pertence e instante de disparo). Com base nas informações lidas define-se a nova rede em uso e atualiza-se o tempo de simulação, que passa a ser o próprio instante de disparo da transição.

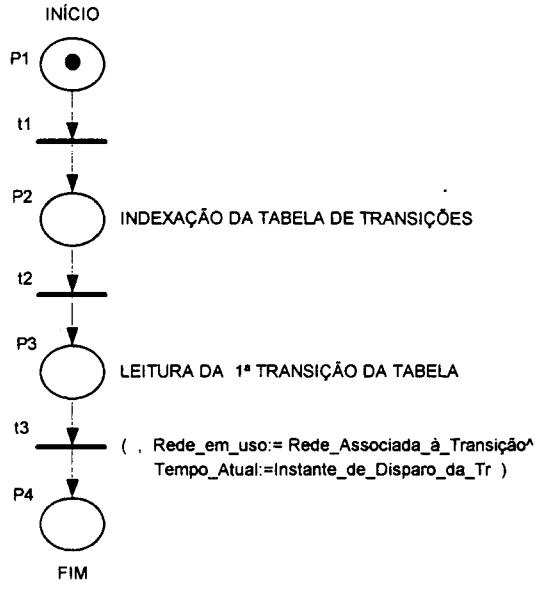


Fig. 5.5 - Algoritmo de funcionamento da função escalonador

5.7 FUNÇÃO INTERFACE COM O USUÁRIO

Para que se possa fazer uma análise detalhada do funcionamento do sistema de controle modelado é necessário dotar a simulação de meios que permitam ao usuário não somente intervir a qualquer momento durante a mesma, mas que coloquem à sua disposição o máximo de opções que garantam flexibilidade e facilidade na sua condução. É esse o papel da *função interface com o usuário*.

Ao ser solicitada uma interrupção garante-se a integridade dos dados que estavam sendo manipulados (concluindo as operações consideradas como indivisíveis, e. g.: o disparo de uma transição) e guardam-se os estados atuais dos sistemas modelados. O controle da simulação é então passado para a *função interface com o usuário*, que coloca à disposição do usuário um menu com as seguintes opções:

I. Iniciar ou continuar a simulação: é utilizada para se iniciar (ao simular pela primeira vez ou após ter escolhido a opção de voltar ao estado inicial) a simulação ou dar continuidade à mesma após uma interrupção. A continuação se dá a partir do ponto de interrupção, levando em conta as modificações introduzidas pelo usuário.

II. Ver a seqüência de disparos: durante a simulação, à medida que cada transição vai sendo disparada, são mostrados na janela de visualização da respectiva rede (há uma janela para cada sistema modelado) o nome da transição e seu instante de disparo. No entanto, é possível, com essa opção, ver o histórico das transições disparadas (com os respectivos instantes) desde o início da simulação até o instante de interrupção.

III. Selecionar o modo de atuação: essa opção permite, a qualquer momento durante a simulação, que se mude seu modo de funcionamento. São três os modos de funcionamento possíveis:

- **Modo automático** (*default*): as transições são disparadas automaticamente até que seja atingido o instante definido pelo usuário (o valor predefinido é de 86400 unidades de tempo, que equivale, considerando uma unidade de tempo igual a 1 segundo, a um tempo virtual de simulação de 24 horas);

- **Modo semi-automático**: a cada estado estável atingido é feita uma pausa na simulação, dando ao usuário a possibilidade de continuar até o próximo estado estável ou interromper a simulação nesse ponto;

- **Modo passo a passo**: é similar ao modo semi-automático, sendo que as pausas na simulação são feitas após o disparo de cada transição (seja ela da rede do sistema de controle ou da rede do sistema controlado).

IV. Acessar as redes: para cada rede envolvida na simulação é possível, através de um segundo menu, a escolha das seguintes opções:

- **Visualizar ou modificar a marcação**: permite ver os lugares da rede (os lugares com número de fichas maior que zero são mostrados primeiro e em ordem alfabética) e os respectivos números de fichas, bem como modificar o número de fichas de qualquer lugar;

- **Visualizar ou modificar as variáveis**: mostra as variáveis internas e externas da rede (as variáveis com valor diferente de zero são mostradas primeiro e em ordem alfabética) com os respectivos valores e permite a modificação dos mesmos;

- **Visualizar ou modificar as temporizações**: mostra as transições da rede (as transições com valor de temporização maior que zero são mostradas primeiro e em ordem alfabética) e o valor de temporização atribuído a cada uma. Todas as transições são iniciadas com temporização nula; fica a critério do usuário a atribuição de novos valores ao longo da simulação;

- **Visualizar o número de disparos**: mostra as transições da rede (em ordem alfabética e decrescente de número de disparos) e o número de vezes que cada uma disparou desde o início da simulação.

V. Voltar ao estado inicial: escolhida essa opção, cada rede é colocada no estado inicial, ou seja, os lugares e as variáveis assumem os valores iniciais definidos no arquivo de descrição da rede. Os valores de temporização não são modificados.

5.8 CONCLUSÃO

Mostrou-se nesse capítulo que a arquitetura do módulo de Simulação foi idealizada tendo como objetivo uma validação por simulação mais refinada do sistema de controle através da interação de seu modelo executável com o modelo executável da planta.

Para facilitar o entendimento do funcionamento interno desse módulo iniciou-se pela apresentação do ciclo de funcionamento da simulação e das justificativas para a estratégia adotada, e, em seguida, foram abordadas as principais funções que o compõem: *função jogador*, *função escalonador* e *função interface com o usuário*.

Com esse capítulo conclui-se a apresentação do AIF, um ambiente desenvolvido para computadores IBM-PC compatíveis e sistema operacional DOS, escrito em linguagem Clipper 5.01. É necessário um mínimo de 270 kbyes de memória RAM para a execução do AIF e não há limite, no programa, para o tamanho das redes de entrada.

CAPÍTULO 6

UTILIZAÇÃO DO AIF EM UM EXEMPLO DE APLICAÇÃO

6.1 INTRODUÇÃO

Esse capítulo ilustra a utilização do AIF em um exemplo prático: a automatização de uma estação de coleta de petróleo similar às existentes nos campos terrestres explorados pela PETROBRÁS no Rio Grande do Norte.

A função de uma estação coletora é receber o petróleo bruto proveniente de poços localizados em sua circunvizinhança, testar a vazão de cada um desses poços e transferir o volume produzido, através de bombas de transferências, por oleodutos que conduzem a estações de armazenamento de maior capacidade [Rol93].

6.2 DESCRIÇÃO DO PROCESSO

O processo, descrito a seguir, está representado na figura 6.1. Um tanque de armazenamento, TQ, recebe petróleo , continuamente, através de uma tubulação ligada aos poços produtores. Quando o óleo atinge um determinado nível no tanque (nível alto) aciona-se uma moto-bomba, B1, que faz a transferência do óleo para uma estação de coleta central. Uma segunda moto-bomba ("stand by"), B2, deve ser acionada, caso a primeira não seja ligada dentro de um intervalo de 2 segundos. Quando o nível do óleo no tanque cair até um determinado ponto (nível baixo), deve-se desligar a moto-bomba para evitar que a mesma corra o risco de succionar em vazio (o que poderia danificá-la). Também deve ocorrer o desligamento da bomba quando houver um aumento excessivo de pressão na descarga da mesma (o que caracteriza um bloqueio na tubulação a jusante). A sinalização dos níveis alto e baixo é feita através de chaves de nível localizadas nessas posições: LSH (nível alto) e LSL (nível baixo); e a sinalização da pressão alta na descarga da bomba é feita através de um pressostato, PSH.

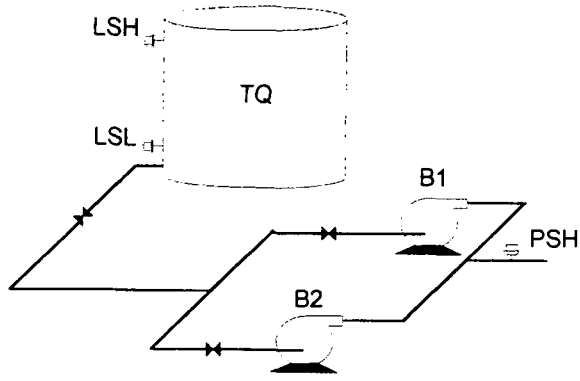


Fig. 6.1 - Estação de coleta de petróleo

6.3 A AUTOMATIZAÇÃO DO PROCESSO

Os três serviços necessários à automatização de uma estação de coleta típica, como a que foi descrita no item anterior, são os seguintes:

I - intertravamento das chaves de nível e do pressostato com as bombas;

II - medição do volume produzido através da contagem dos pulsos elétricos enviados por um medidor de vazão (não representado na figura 6.1);

III - aquisição e envio dos dados para uma estação central, onde é feito o controle supervisório de várias estações.

Para realizar esses serviços utiliza-se um Controlador Lógico Programável (CLP) localizado em cada estação coletora. Os CLP's atualmente utilizados nessas estações de coleta são programáveis apenas em linguagem de diagrama de contatos (*Ladder Diagram*), que apesar de ser familiar e de fácil entendimento para o pessoal técnico envolvido, torna difícil a depuração e a validação do programa de aplicação, o que normalmente acabo sendo feito através de testes com o sistema real.

Foi nesse ponto que surgiu a idéia de se utilizar o AIF [Fas94]: antes de se escrever o programa de aplicação na linguagem do CLP, representá-lo na forma de uma RdP Interpretada, validá-la e, tomando por base a rede validada, traduzi-la para a linguagem de programação do CLP (esse último passo poderia ser feito automaticamente pelo AIF, mas o fato de a grande maioria dos fabricantes de CLP ainda não estar seguindo a norma IEC-1131, que padroniza as linguagens de programação para CLP, torna difícil essa tarefa, pois seria necessário um tradutor para cada linguagem proprietária). No caso da utilização da linguagem de contatos, a escrita do programa a partir da RdP é relativamente direta e tem como vantagem o fato de ser feita a partir de uma especificação já validada.

6.4 UTILIZAÇÃO DO AIF

A título de simplificação, utilizaremos o AIF no serviço I descrito no item anterior, ou seja, na validação da parte do programa de aplicação responsável pelo intertravamento das chaves de nível e do pressostato com as bombas. Inicialmente definiremos a lógica de intertravamento e em seguida mostraremos, passo a passo, a utilização do AIF.

6.4.1 LÓGICA DE INTERTRAVAMENTO

As bombas, B1 e B2, são acionadas alternadamente para cada transferência, ou seja, se na primeira transferência do dia for acionada B1, na segunda será B2 e assim por diante. O sinal de entrada para o CLP que faz com que o mesmo acione uma bomba é o de nível alto (LSH). Na finalização da transferência, o sinal de entrada que provoca o desligamento da bomba acionada é o de nível baixo (LSL). Ao acionar uma determinada bomba o programa aguarda, durante 2 segundos, pelo sinal de confirmação de bomba ligada (proveniente de um contato auxiliar do contactor da mesma); se, transcorridos os 2 segundos, o programa não receber essa confirmação, a bomba é desligada sendo enviado um outro sinal para o acionamento da segunda bomba. O sinal de pressão alta (PSH) enviado pelo pressostato na descarga das bombas, durante a transferência, provoca o desligamento da bomba que estiver em operação. Esse sinal deverá ser ignorado se ocorrer em um tempo inferior a 30 segundos após o acionamento da bomba (para evitar que os transientes de pressão alta, que ocorrem normalmente nessa situação, provoquem uma interrupção desnecessária da transferência).

6.4.2 PASSOS PARA A UTILIZAÇÃO DO AIF

Uma vez definidos o processo e a lógica do sistema de controle, deve-se adotar como procedimento para a utilização do AIF a seqüência de passos descrita a seguir:

1º Passo: Representar o sistema de controle e a planta através de Redes de Petri interpretadas;

2º Passo: Descrever as redes na linguagem de entrada do AIF, criando um arquivo para cada sistema;

3º Passo: Utilizando o módulo de análise do AIF, verificar as propriedades comportamentais de cada rede, detectando, se for o caso, as primeiras incoerências na representação dos sistemas;

4º Passo: Realizar a validação por simulação, através do respectivo módulo do AIF, fazendo a rede do sistema de controle interagir com a rede da planta;

5º Passo: Fazer a tradução automática da rede do sistema de controle validada para a linguagem do controlador (na versão atual do AIF a tradução é feita para a linguagem C).

Os itens seguintes correspondem aos passos descritos.

6.4.3 REPRESENTAÇÃO DOS SISTEMAS ATRAVÉS DE RDP INTERPRETADAS

O sistema de controle e a planta estão representados pelas redes das figuras 6.2 e 6.3 respectivamente.

6.4.4 DESCRIÇÃO DAS REDES NA SINTAXE DO AIF

A partir das representações gráficas dos sistemas na forma de Redes de Petri interpretadas, cria-se para cada sistema um arquivo tipo texto contendo a descrição da respectiva rede na sintaxe requerida pelo AIF. Isso pode ser feito através do módulo de edição do AIF ou através de qualquer editor que permita a criação de textos no padrão ASCII.

Nas figuras 6.4 e 6.5 são mostradas as descrições, na sintaxe do AIF, das redes das figuras 6.2 e 6.3 respectivamente.

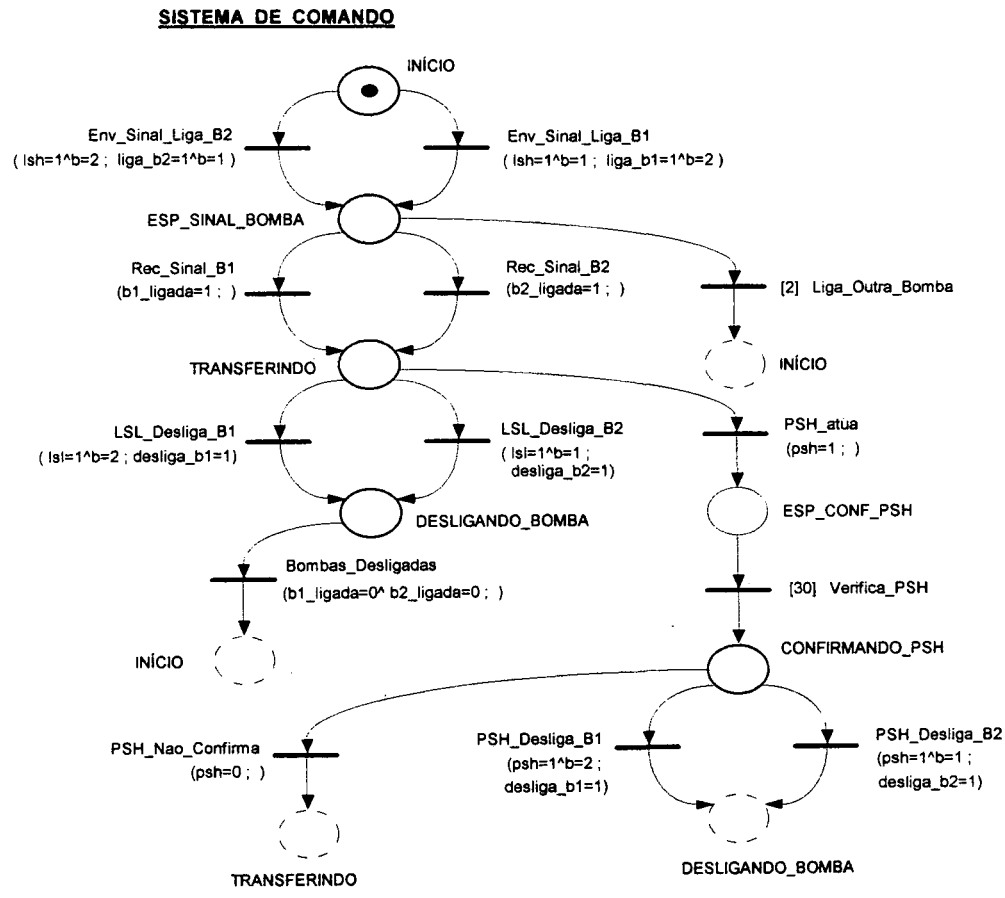


Fig. 6.2 - RdP representando o sistema de controle

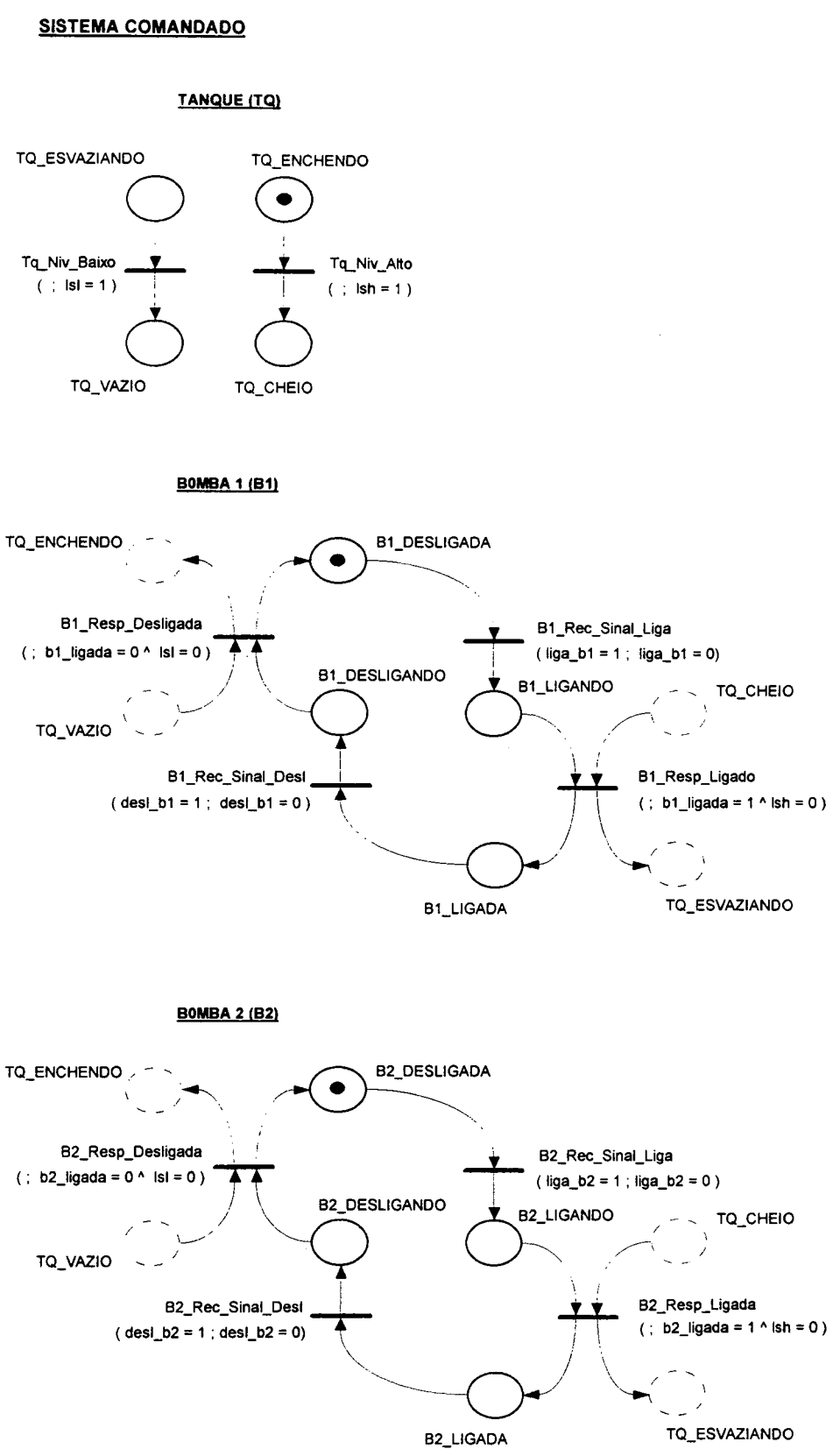


Fig. 6.3 - RdP representando a planta

```
REDE SISTEMA_DE_CONTROLE;

NODOS

  ESP_SINAL_BOMBA, TRANSFERINDO, DESLIGANDO_BOMBA, ESP_CONF_PSH,
  CONFIRMANDO_PSH
  : Lugar(0);

  INICIO
  : Lugar(1);

  ENV_SINAL_LIGA_B1, ENV_SINAL_LIGA_B2, REC_SINAL_B1, REC_SINAL_B2,
  LSL_DESLIGA_B1, LSL_DESLIGA_B2, PSH_ATUA,BOMBAS_DESLIGADAS,
  PSH_NAO_CONFIRMA, PSH_DESLIGA_B1, PSH_DESLIGA_B2
  : Transicao;

  LIGA_OUTRA_BOMBA
  : Transicao [2,2];

  VERIFICA_PSH
  : Transicao [30,30];

ESTRUTURA

  ENV_SINAL_LIGA_B1:   (INICIO) , (ESP_SINAL_BOMBA);
  ENV_SINAL_LIGA_B2:   (INICIO) , (ESP_SINAL_BOMBA);
  REC_SINAL_B1:        (ESP_SINAL_BOMBA) , (TRANSFERINDO);
  REC_SINAL_B2:        (ESP_SINAL_BOMBA) , (TRANSFERINDO);
  LIGA_OUTRA_BOMBA:    (ESP_SINAL_BOMBA) , (INICIO);
  LSL_DESLIGA_B1:      (TRANSFERINDO) , (DESLIGANDO_BOMBA);
  LSL_DESLIGA_B2:      (TRANSFERINDO) , (DESLIGANDO_BOMBA);
  PSH_ATUA:            (TRANSFERINDO) , (ESP_CONF_PSH);
  BOMBAS_DESLIGADAS:   (DESLIGANDO_BOMBA) , (INICIO);
  VERIFICA_PSH:        (ESP_CONF_PSH) , (CONFIRMANDO_PSH);
  PSH_NAO_CONFIRMA:    (CONFIRMANDO_PSH) , (TRANSFERINDO);
  PSH_DESLIGA_B1:      (CONFIRMANDO_PSH) , (DESLIGANDO_BOMBA);
  PSH_DESLIGA_B2:      (CONFIRMANDO_PSH) , (DESLIGANDO_BOMBA);

FIM

VARIAVEIS

  PSH
  : Interna(0);

  B
  : Interna(1);

  LSH, LSL, LIGA_B1, LIGA_B2, B1_LIGADA, B2_LIGADA, DESL_B1, DESL_B2
  : Externa(0);

CONDICOES_ACOES

  ENV_SINAL_LIGA_B1:   (LSH=1,B=1)                    ,  (LIGA_B1=1,B=2);
  ENV_SINAL_LIGA_B2:   (LSH=1,B=2)                    ,  (LIGA_B2=1,B=1);
  REC_SINAL_B1:        (B1_LIGADA=1)                  ,  ( );
  REC_SINAL_B2:        (B2_LIGADA=1)                  ,  ( );
  LSL_DESLIGA_B1:      (LSL=1,B=2)                    ,  (DESL_B1=1);
  LSL_DESLIGA_B2:      (LSL=1,B=1)                    ,  (DESL_B2=1);
  BOMBAS_DESLIGADAS:   (B1_LIGADA=0, B2_LIGADA=0)     ,  ( );
  PSH_ATUA:            (PSH=1)                        ,  ( );
  PSH_NAO_CONFIRMA:    (PSH=0)                        ,  ( );
  PSH_DESLIGA_B1:      (PSH=1,B=2)                    ,  (DESL_B1=1);
  PSH_DESLIGA_B2:      (PSH=1,B=1)                    ,  (DESL_B2=1);

FINAL
```

Fig. 6.4 - RdP do sistema de controle descrita na sintaxe do AIF

```
REDE PLANTA;

NODOS

  TQ_CHEIO, TQ_ESVAZIANDO, TQ_VAZIO, B1_LIGANDO, B2_LIGANDO,
  B1_LIGADA, B2_LIGADA, B1_DESLIGANDO, B2_DESLIGANDO
  : Lugar(0);

  TQ_ENCHENDO, B1_DESLIGADA, B2_DESLIGADA
  : Lugar(1);

  B1_REC_SINAL_LIGA, B2_REC_SINAL_LIGA, B1_RESP_LIGADA, B2_RESP_LIGADA,
  B1_REC_SINAL_DESL, B2_REC_SINAL_DESL, B1_RESP_DESLIGADA, B2_RESP_DESLIGADA
  : Transicao;

  TQ_NIV_ALTO
  : Transicao [200,200];

  TQ_NIV_BAIXO
  : Transicao [100,100];

ESTRUTURA

  TQ_NIV_ALTO:          (TQ_ENCHENDO)                , (TQ_CHEIO);
  TQ_NIV_BAIXO:         (TQ_ESVAZIANDO)              , (TQ_VAZIO);
  B1_REC_SINAL_LIGA:    (B1_DESLIGADA)               , (B1_LIGANDO);
  B1_RESP_LIGADA:       (B1_LIGANDO, TQ_CHEIO)       , (B1_LIGADA, TQ_ESVAZIANDO);
  B1_REC_SINAL_DESL:    (B1_LIGADA)                  , (B1_DESLIGANDO);
  B1_RESP_DESLIGADA:    (B1_DESLIGANDO, TQ_VAZIO)    , (B1_DESLIGADA, TQ_ENCHENDO);
  B2_REC_SINAL_LIGA:    (B2_DESLIGADA)               , (B2_LIGANDO);
  B2_RESP_LIGADA:       (B2_LIGANDO, TQ_CHEIO)       , (B2_LIGADA, TQ_ESVAZIANDO);
  B2_REC_SINAL_DESL:    (B2_LIGADA)                  , (B2_DESLIGANDO);
  B2_RESP_DESLIGADA:    (B2_DESLIGANDO, TQ_VAZIO)    , (B2_DESLIGADA, TQ_ENCHENDO);

FIM


VARIAVEIS

  LSH, LSL, LIGA_B1, LIGA_B2, DESL_B1, DESL_B2, B1_LIGADA, B2_LIGADA
  : Externa(0);

CONDICOES_ACOES

  TQ_NIV_ALTO:          ( )          ,   (LSH=1);
  TQ_NIV_BAIXO:         ( )          ,   (LSL=1);
  B1_REC_SINAL_LIGA:    (LIGA_B1=1)  ,   (LIGA_B1=0);
  B1_RESP_LIGADA:       ( )          ,   (B1_LIGADA=1, LSH=0);
  B1_REC_SINAL_DESL:    (DESL_B1=1)  ,   (DESL_B1=0);
  B1_RESP_DESLIGADA:    ( )          ,   (B1_LIGADA=0, LSL=0);
  B2_REC_SINAL_LIGA:    (LIGA_B2=1)  ,   (LIGA_B2=0);
  B2_RESP_LIGADA:       ( )          ,   (B2_LIGADA=1, LSH=0);
  B2_REC_SINAL_DESL:    (DESL_B2=1)  ,   (DESL_B2=0);
  B2_RESP_DESLIGADA:    ( )          ,   (B2_LIGADA=0, LSL=0);

FINAL
```

Fig. 6.5 - RdP da planta descrita na sintaxe do AIF

6.4.5 RESULTADOS DA ANÁLISE

Feita a descrição das redes na linguagem do AIF, o próximo passo é analisar individualmente cada uma, para verificar suas propriedades e detectar eventuais erros de modelagem. Essa tarefa fica a cargo do módulo de análise e os relatórios gerados pelo mesmo estão mostrados a seguir.

```
I. Relatório da análise na rede do sistema de controle
Enumeracao de estados: rede  CONTROLE (6 estados acessiveis).
Propriedades verificadas:
*--------------------------------------------------------------------------*
A rede em analise e' binaria.
  Lugares Nulos (M = 0): {}
  Lugares Binarios     : {todos(as)}
  Lugares k-Limitados  : {}
  Lugares Nao Limitados: {}

A rede em analise e' estritamente conservativa.

A rede em analise e' viva.
  Tr. vivas        : {todos(as)}
  Tr. quase-vivas  : {todos(as)}
  Tr. nao disparadas: {}

A rede e' reiniciavel.

Nao foram detectados "live-locks" na rede.
Nao foram detectados "dead-locks" na rede.
*--------------------------------------------------------------------------*

II. Relatório da análise na rede da planta

Enumeracao de estados: rede  PLANTA (24 estados acessiveis).
Propriedades verificadas:
*--------------------------------------------------------------------------*
A rede em analise e' binaria.
  Lugares Nulos (M = 0): {}
  Lugares Binarios     : {todos(as)}
  Lugares k-Limitados  : {}
  Lugares Nao Limitados: {}

A rede em analise e' estritamente conservativa.

A rede em analise e' viva.
  Tr. vivas        : {todos(as)}
  Tr. quase-vivas  : {todos(as)}
  Tr. nao disparadas: {}

A rede e' reiniciavel.

Nao foram detectados "live-locks" na rede.
Nao foram detectados "dead-locks" na rede.
*--------------------------------------------------------------------------*
```

6.4.6 RESULTADOS DA SIMULAÇÃO

Concluída a etapa de análise das redes, chama-se o módulo de simulação que permitirá validar a rede que representa o sistema de controle, fazendo-a interagir com a rede que emulará o sistema comandado. Escolhidas as redes pelo usuário, o módulo verifica a coerência da interface de comunicação entre elas (variáveis externas comuns às duas redes que farão o papel de sensores e atuadores).

A condução da simulação foi baseada em cenários definidos a partir das características informadas na lógica de intertravamento (item 6.4.1). A menos que indicado de outra forma, a simulação em cada cenário se iniciará com o seguinte status:

- marcação inicial das redes conforme indicado nas figuras 6.2 e 6.3;
- valores iniciais das variáveis conforme indicado nas figuras 6.4 e 6.5;
- transições temporizadas:

Rede	Transição	Tempo (virtual)
Controle	Liga_Outra_Bomba	2
	Verifica_PSH	30
Planta	Tq_Nivel_Alto	200
	Tq_Nivel_Baixo	100

A descrição de cada cenário com o resultado da simulação são mostrados a seguir.

Cenário 1 - Sem mudanças nas condições pré-estabelecidas;

SISTEMA DE CONTROLE
Rede: CONTROLE
Lug.:6 Trans.:13 Var.:10

Tr. Disparada	Tempo
ENV_SINAL_LIGA_B1	200
REC_SINAL_B1	200
ENV_SINAL_DESL_B1	300
BOMBAS_DESLIGADAS	300
ENV_SINAL_LIGA_B2	500

SISTEMA CONTROLADO
Rede: PLANTA
Lug.:12 Trans.:10 Var.:8

Tr. Disparada	Tempo
TQ_NIV_ALTO	200
B1_REC_SINAL_LIGA	200
B1_RESP_LIGADA	200
TQ_NIV_BAIXO	300
B1_REC_SINAL_DESL	300
B1_RESP_DESLIGADA	300
TQ_NIV_ALTO	500

Comentário - As bombas são ligadas alternadamente (instantes 200, 500);

Cenário 2 - Mudando o valor inicial (zero) da variável *psh* para 1, simulando pressão alta na descarga das bombas;

SISTEMA DE CONTROLE
Rede: CONTROLE
Lug.:6 Trans.:13 Var.:10

Tr. Disparada	Tempo
ENV_SINAL_LIGA_B1	200
REC_SINAL_B1	200
PSH_ATUA	200
VERIFICA_PSH	230
PSH_DESLIGA_B1	230
BOMBAS_DESLIGADAS	300

SISTEMA CONTROLADO
Rede: PLANTA
Lug.:12 Trans.:10 Var.:8

Tr. Disparada	Tempo
TQ_NIV_ALTO	200
B1_REC_SINAL_LIGA	200
B1_RESP_LIGADA	200
B1_REC_SINAL_DESL	230
TQ_NIV_BAIXO	300
B1_RESP_DESLIGADA	300

Comentário - Como o valor de *psh* permanece em 1 (significando a persistência da pressão alta) ao ser checado após 30 segundos, a bomba em funcionamento (B1) é desligada.

Cenário 3 - Mudando o valor inicial (zero) da variável *psh* para 1 e, após o disparo da transição que indica o recebimento do sinal do pressostato (PSH_Atua), fazendo a variável *psh* igual a zero novamente;

SISTEMA DE CONTROLE
Rede: CONTROLE
Lug.:6 Trans.:13 Var.:10

Tr. Disparada	Tempo
ENV_SINAL_LIGA_B1	200
REC_SINAL_B1	200
PSH_ATUA	200
VERIFICA_PSH	230
PSH_NAO_CONFIRMA	230
ENV_SINAL_DESL_B1	300

SISTEMA CONTROLADO
Rede: PLANTA
Lug.:12 Trans.:10 Var.:8

Tr. Disparada	Tempo
TQ_NIV_ALTO	200
B1_REC_SINAL_LIGA	200
B1_RESP_LIGADA	200
TQ_NIV_BAIXO	300
B1_REC_SINAL_DESL	300
B1_RESP_DESLIGADA	300

Comentário - Como o valor de *psh* retornou a zero entre os instantes 200 e 230 (significando que a pressão alta era apenas transitória), o valor de *psh* é checado (Verifica_PSH) e não confirmado (PSH_Nao_Confirma); assim, a bomba (P1) continua transferindo até ser atingido o nível baixo quando a mesma é desligada pela chave de nível baixo (Env_Sinal_Desl_B1).

6.5 CONCLUSÃO

Esse capítulo mostrou a utilização do AIF no processo de automatização de uma estação de coleta de petróleo usando um CLP.

Os resultados do módulo de Análise mostraram que as redes subjacentes possuíam as propriedades requeridas pelos automatismos lógicos: eram binárias, vivas e reversíveis.

Através do módulo de Simulação pôde-se comprovar a adequação do comportamento dos modelos (sistema de controle e planta) com os requisitos iniciais, transformados em cenários.

Apesar da impossibilidade de tradução automática da rede de controle validada para a linguagem do CLP (diagrama de relés), a contribuição do ambiente está na garantia de que o modelo do sistema de controle representado em forma de RdP Interpretada atende aos requisitos da automação proposta. Outrossim, vale ressaltar que a tradução para a linguagem do CLP, ainda que manual, terá a vantagem de partir de uma especificação pré-validada.

CAPÍTULO 7

CONCLUSÃO

Este trabalho descreve a especificação e a implementação do AIF - Ambiente Integrado de Ferramentas - um ambiente adequado ao desenvolvimento de sistemas de controle para SED's que podem ser podem ser modelados por Redes de Petri Interpretadas.

As ferramentas que integram o AIF o credenciam como um ambiente de auxílio ao projetista em todo o ciclo de desenvolvimento do sistema de controle, sendo particularmente indicado para as fases de descrição, validação e implementação.

A arquitetura do AIF é composta de quatro módulos: módulo de Edição, de Análise, de Simulação e de Tradução.

O módulo de Edição fornece uma interface para a descrição do sistema no modelo RdP Interpretada, permitindo a criação de arquivos usando um editor de texto, a leitura de um arquivo contendo a descrição da rede e a análise sintática e semântica desta.

O módulo de Análise é implementado pelo programa ARP [Maz90]. O ARP realiza a análise da rede subjacente, utilizando os seguintes métodos: grafo de acessibilidade da rede, verificação por equivalência de linguagens, cálculo de invariantes e análise de desempenho.

O módulo de Simulação simula o controle e emula a planta, ambos modelados por Redes de Petri Interpretadas. A emulação da planta permite ao projetista a realização de testes sobre o comportamento do sistema de controle antes de sua implementação. As condições extras associadas às transições representam os sinais enviados pelos sensores; as ações associadas às transições representam os sinais de comando enviados à planta. Os sensores e as tarefas ativadas pelos atuadores são emulados pela rede que representa a planta.

O módulo de Tradução, fazendo uso da equivalência entre a implementação das Redes de Petri através de um jogador de fichas e a de um sistema de regras de produção através de um motor de inferência, traduz a RdP para um conjunto de regras que servem de entrada para o programa que gera o código-fonte em linguagem C.

Foi apresentado neste trabalho um exemplo de aplicação que consiste na automação de uma estação de coleta e transferência de petróleo usando um CLP. O controle, assim como a planta, foi especificado usando o modelo de Rede de Petri Interpretada, analisado e simulado utilizando o ambiente AIF.

Perspectivas

No exemplo de aplicação o código em linguagem C gerado pelo módulo de Tradução não pôde ser utilizado, uma vez que o CLP só aceitava programação em linguagem de contatos de relé (*ladder diagram*). Os CLP's têm experimentado nos últimos anos notável evolução (*hardware* e *software*), sendo grande sua utilização na automação industrial. Foi criada uma norma internacional (IEC 1131) que padroniza as linguagens de programação para CLP. Portanto, uma perspectiva importante para a continuação deste trabalho, sobretudo para aplicações industriais, é a ampliação do módulo de Tradução para gerar, a partir da Rede de Petri validada, código nas linguagens padronizadas pela norma internacional IEC 1131: SFC (Sequencial Function Chart - Grafcet), LD (Ladder Diagram), IL (Instruction List) e ST (Structured Text).

Devido à inexistência de um modelo e de uma metodologia únicos para satisfazer os requisitos dos diferentes problemas de análise e síntese de sistemas, e atender às especificidades de cada aplicação, é interessante integrar ao AIF outros modelos e ferramentas. Por exemplo, enquanto a teoria de Rede de Petri é muito útil na análise dos SED's, a teoria de autômatos é adequada para a síntese, permitindo a realização de uma ferramenta de síntese automática ou baseada em heurística. Outras teorias que permitem tratar com SED's poderiam também ser utilizadas, como Lógica e Abordagem Síncrona.

REFERÊNCIAS BIBLIOGRÁFICAS

[Age79] AGERWALA, T. *Putting Petri Nets to Work.* Computer, p.85-94, dec. 1979.

[Ala84] ALANCHE, P., ANCELIN B. *Evaluation des simulateurs à événements discrets.* Le Nouvel Automatisme, mars 1984.

[Alb81] ALBUS, J.S., BARBARA, A.J., NAGEL, R.N. *Theory and practice of hierarchical control.* In 23nd IEEE Computer Society Int.conf. september 1981.

[Bad93] BADER, F. P. *Utilizing multi-language inernational software standards for distributed process control.* ISA Transactions 32, p 345-354, 1993.

[Bak90] BAKO, B. *Mise en Oeuvre et Simulation du Niveau Coordination de la Commande des Ateliers Flexibles: une approche mixte Reseaux de Petri et Systemes de Regles.* Thèse de Doctorat de l'Université Paul Sabatier, Toulouse, 1990.

[Bar87] BARBALHO, D. S. *Conception et Mise en Oeuvre de la Function Coordination pour une Commande Distribuée d'Atelier.* Thèse de Doctorat de l'Université Paul Sabatier, Toulouse, 1987.

[Ben85] BENZAKOUR, K. *SICLOP: Simulador de Commande Logique et de Procédés.* Thèse de Doctorat 3ième cycle de l'Université Paul Sabatier, Toulouse, 1985.

[Bra83] BRAMS, G. W. *Réseaux de Petri: Théorie et Pratique.* Paris: Masson, 2 v., 1983.

[Bry77] BRYANT, R.E. *Simulation of packet communication arachitecture computer systems.* Technical report MIT, LC, TR-188, Massachussets Institute of Technology, Cambridge, 1977.

[Buz82] BUZACOTT, J. A. *Optimal operating rules for automated manufacturing systems.* IEEE Trans. Automatic Control, 27:80-86, 1982.

[Can90] CANTÚ, Evandro. *Uma Abordagem para a Representação, Simulação e Implementação de Sistemas Baseada na Redes de Petri a Objetos.* Dissertação (Mestrado em Engenharia Elétrica), Universidade Federal de Santa Catarina, 1990.

[Car90] CARDOSO, Janette. *Sur les Réseaux de Petri avec Marquages Flous.* Thèse de Doctorat de l'Université Paul Sabatier, Toulouse, 1990.

[Cha79] CHANDY, K. M., MISRA, J. *Conditional knowledge as a basis for distributed simulation.* Relatório técnico no 5251:TR:87, California Institute of Technologie, 1987.

[Cou80] COURVOISIER, M., VALETTE, R. *Systèmes de Commande en Temps Réel. Description, Analyse et Realization.* Paris: SCM, 1980. ISBN 2-901133-41-X.

[Dav89] DAVID, René, ALLA, Hassane. *Du Grafcet aux réseaux de Petri.* Paris: Hermès, 1989.

[Dia82] DIAZ, M. *Modelling and Analysis of Communicatio and Cooperation Protocols Using Petri Net Based Models.* 2th International Workshop on Protocol Specification, Testing and Verification, Idyllwild, CA, USA, p. 465-510, 1982.

[Est85] ESTEBAN, P. *Sur la Recherche d'algorithmes Simplifiés d'analyse des Réseaux de Petri.* Thèse de Doctorat de l'Université Paul Sabatier, Toulouse, 1985.

[Far89a] FARINES, J. M., MAZIERO, C. A. *O Ambiente ARP de Análise e Simulação de Sistemas Modelados por Rede de Petri.* Anais do III Simpósio Brasileiro de Engenharia de Software, Recife, PE, 1989.

[Far89b] FARINES, J. M., CANTÚ, E., GARNOUSSET, H., MAZIERO, C. A. *ARP: uma Ferramenta de Desenvolvimento de Software em Aplicações Distribuídas.* Seminário Franco-Brasileiro em Sistemas Informáticos Distribuídos, Florianópolis, SC, 1989.

[Far92] FARINES, J. M., CARDOSO, J., CURY, J. E. *Specification and Implementation of an FMS Coordination System Based on High Level Petri net.* Int. Workshop on Intelligent Manufacturing Systems, Dearborn, USA, 1992.

[Fas94] SOARES, F. Assis, CARDOSO, J., CURY, J. E. *An Integrated Environment of Tools for the Design of Manufacturing Systems.* 2th International Workshop IFAC, IMS, 1994.

[Ho87] HO, Y-C. *Basic research, Manufacturing Automation, and putting the cart before the horse.* Editorial, IEEE Trans. on Automatic Control, 32(12):1042-1043 (1987).

[Jen90] JENSEN, Kurt. *Coloured Petri Nets: A High Level Language for Sustem Design and Analysis.* DAIMI PB - 338, Computer Science Department, Aarhus University, nov. 1990.

[Man83] MANNA, B., PNUELI, A. *The temporal logic of branching time*. Acta Informatica, 20, 1983.

[Maz90] MAZIERO, Carlos Alberto. *Um Ambiente para Análise e Simulação de Sistemas Modelados por Redes de Petri*. Dissertação (Mestrado em Engenharia Elétrica), Universidade Federal de Santa Catarina, 1990.

[Mer74] MERLIN, P. *A Study of the Recoverability of Computer Systems*. PhD thesis, University of California, Irvine, 1974.

[Mur89] MURATA, Tadao. *Petri Nets: Properties, Analysis and Applications*. Proceedings of the IEEE,v. 77, n. 4, p.541-580, abr. 1989.

[Pal91] PALADINO, Alvaro Daniel Arioni. *Um Gerador de Programas para Sistemas de Regras de Produção Visando à Eficiência na Execução*. Dissertação (Mestrado em Engenharia Elétrica), Universidade Federal de Santa Catarina, 1991.

[Pet66] PETRI, C. A. *Communication with automata*. Technical report No. RADC-TR065-377, versão inglesa, 1966

[Pet81] PETERSON, J. L. *Petri Net Theory and the Modelling of Systems*. N.J.: Prentice-Hall, 1981.

[Ram74] RAMCHANDANI, C. *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets Models*. PhD thesis, MIT, 1974.

[Ram89] RAMADGE, P. J., WONHAM W. M. *The control of discrete event systems*. Proceedings of the IEEE, 77(1):81-97, 1989.

[Rei85] REISIG, Wolfgang. *Petri Nets*. Berlin: Springer-Verlag, 1985.

[Rol93] ROLIM, T., PAULA, P. *Instrumentação Industrial utilizada na Automação de Estações Coletoras*. 2º Encontro de Engenharia Elétrica da PETROBRÁS, dez, 1993.

[Sib85] SIBERTIN, Christophe. *Le Prototypage des Applications Interactives a l'Aide de Reseaux de Petri*. 6th European Workshop on Applications and Theory of Petri Nets. Helsinki, jun. 1985.

[Sil90] SILVA, M., VALETTE, R. *Petri nets and flexible manufacturing systems*. In Lecture Notes in Computer Science, vol. 424. Springer-Verlag, 1990.

[Val86] VALETTE, R. *Nets in Production Systems*. Lectur Notes in Computer Science 255: Advance in Petri Nets, Part II:191-217,1986.

[Val87] VALETTE, R., ATABAKHCHE, H., BARBALHO, D. *Comparaison entre les Approches fondees sur les Reseaux e Petri et celles utilisant l'Intelligence Artificialle pour la Commande des Ateliers de Production Automatises*. Rapport LAAS 87171, jun. 1987.

[Val88] VALETTE, R. *Les Reseaux de Petri*. Rapport LAAS/CNRS, Toulouse, France, 1988.

[Val89] VALETTE, R., CARDOSO, J., ATABAKHCHE, COURVOISIER, M., LEMAIRE, T. *Petri nets and Production Rules for Decision Levels in FMS Control*. In: IMACS Modeling and Simulation of Systems, 1989.

[Val91] VALETTE, R., CARDOSO, J. *Redes de Petri*. Nota interna do Laboratório de Controle e Microinformática, Universidade Federal de Santa Catarina,1991.

[Whi89] WHITE, K. P., MITCHELL, C. M. *Manufacturing systems enginnering: a second industrial revolution?* IEEE Trans. on Syst. Man and Cyb, 19:161-163, 1989.

[Zil93] ZILLER, R. M. *A abordagem Ramadge-Wonham no controle de sistemas a eventos discretos: contribuições à teoria*. Dissertação (Mestrado em Engenharia Elétrica), Universidade Federal de Santa Catarina, 1993.