Um Sistema Tutor Multi-Agentes no Domínio de Redes de Petri

Gustavo Meneses Góis

Dissertação submetida à Coordenação do Curso de Pós-Gradução em Informática da Universidade Federal da Paraíba - Campus II como parte dos requisitos necessários para obtenção do grau de Mestre em Informática.

Área de Concentração: Redes de Petri

Angelo Perkusich
(orientador)
Evandro de Barros Costa
(orientador)

Campina Grande, Paraíba, Brasil

Ficha Catalográfica

GÓIS, Gustavo Meneses

G616S

Um Sistema Tutor Multi-Agentes do Domínio de Redes de Petri.

Dissertação (Mestrado) – UFPB/CCT/COPIN, Campina Grande, Agosto de 2000.

91p. Il.

Orientador: Ângelo Perkusich

1. Redes de Petri
2. Sistemas Tutores Inteligentes
3. Sistemas Multi-Agentes.

CDU 519.711

"UM SISTEMA TUTOR MULTI-AGENTES NO DOMÍNIO DE REDES DE PETRI"

GUSTAVO MENESES GÓIS

DISSERTAÇÃO APROVADA EM 30.08.2000

PROF. ANGELO PERKUSICH, D.Sc
Orientador

PROF. EVANDRO DE BARROS COSTA, D.Sc
Orientador

Maria de Fatima Q. Vieira Turrell
PROFª MARIA DE FÁTIMA Q. V. QUEIROZ , Ph.D
Examinadora

PROF. TOMAZ DE CARVALHO BARROS, D.Sc
Examinador

CAMPINA GRANDE – PB

Dedicatória

Aos meus pais, pelo estímulo e apoio incondicional desde a primeira hora, pela paciência e grande amizade com que sempre me ouviram. As palavras são pequenas para refletir o quanto eles são responsáveis por toda e qualquer vitória por mim já alcançada, ou que possa vir a alcançar na minha vida.

À minha esposa e ao meu filho, suas presenças são uma fonte constante de energia para todas as minhas realizações, e sem eles, nenhuma das conquistas teria sentido.

Agradecimentos

Ao Professor Angelo Perkusich. Sua capacidade em transmitir conhecimentos, aliada à sua competência, boa vontade e coragem demonstradas durante a orientação deste trabalho foram fundamentais para esta realização.

Ao professor Evandro Barros Costa, que mesmo à distância, participou de forma efetiva no desenvolvimento deste trabalho.

Ao professor Jorge Abrantes Figueiredo, tive a honra de tê-lo como orientador em bolsas de iniciação científica, além de professor de algumas disciplinas, tornando-se para mim um referencial de postura e competência na minha vida profissional.

À CAPES, já que sem o seu auxílio financeiro não teria sido possível a realização deste trabalho.

A todos que trabalham, de uma forma ou de outra, na Universidade Federal da Paraíba - Campus II, Campina Grande, e propiciam um ambiente adequado ao estudo. Em especial, os meus agradecimentos às funcionárias da COPIN Aninha e Vera, que com sua simpatia e boa vontade resolvem de forma bastante competente todos os problemas que estão sob suas responsabilidades.

Ao meu grupo de amigos, o "Dream Team", em especial às minhas amigas Ana Karla e Márcia, que em momentos bastante difíceis encorajaram-me a continuar e concluir o desenvolvimento deste trabalho.

À Banca Examinadora, por ter aceito o convite, e pelas considerações importantes sobre o trabalho.

E, por último, mas não menos importante, a todos os colegas, amigos e professores, em especial ao grupo de redes de Petri, que contribuíram positivamente para a realização deste trabalho.

Resumo

Este trabalho apresenta a definição, modelagem e análise de um Sistema Tutor Multi-Agente no domínio de redes de Petri. A concepção deste sistema baseia-se num modelo de ambiente interativo de aprendizagem com uma abordagem Multi-Agentes, denominado MATHEMA. Neste contexto detalhamos a definição de um modelo de conhecimento do domínio de redes de Petri que permite uma estruturação mais adequada e, consequentemente, uma melhor investigação sobre o seu conteúdo. Para a modelagem e análise do Sistema Tutor Multi-Agentes são aplicadas as redes de Petri Coloridas, e os resultados são detalhadamente apresentados.

Abstract

This works presents the definition, modeling and analisys of a Multi-Agent Tutoring System in the Petri Nets domain. The design of this system is based on a model of a multi-agent interactive learning environment named MATHEMA. Therefore, we detail the definition of the Petri Net domain knowledge in such a way that it is possible to get an adequate knowledge struture, thus allowing a better investigation and understanding of this domain. To model and analyze the Multi-Agent Tutoring System in the Petri Nets domain we applied Colored Petri Nets, and the results are presented in details.

Conteúdo

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Este trabalho está inserido na área de Inteligência Artificial Distribuída (IAD) [2; 39], mais especificamente no que diz respeito aos Sistemas Tutores Inteligentes Artificiais que utilizam uma abordagem baseada em Agentes.

Para analisarmos a evolução dos softwares educacionais, é interessante estudarmos a evolução da utilização da informática em nossas atividades pessoais. É fácil perceber o crescimento acelerado da utilização dos computadores pela sociedade nas últimas décadas, causado principalmente pelo desenvolvimento da informática, aliado à sua crescente utilização nos mais variados setores da sociedade, o que provocou uma grande mudança no modo das pessoas agirem e se relacionar. Um exemplo disto está na utilização da INTERNET, que possibilita que a informação se distribua pelo mundo com uma velocidade espantosa.

A educação tradicional não poderia deixar de sofrer algumas alterações diante de todas estas mudanças; já que ela é diretamente atingida justamente por tratar com um elemento chave de todas estas mudanças: a *informação*. Atualmente, o maior objetivo das pesquisas nesta área não é substituir todo o processo educacional desenvolvido ao longo do tempo, mas, adequar este processo tradicional às tecnologias computacionais existentes. Este desafio está sendo tratado pela área de informática na educação, que aliada às áreas de Inteligência Artificial Distribuída, Psicologia Cognitiva e Educação, busca o desenvolvimento de técnicas que permitam o desenvolvimento de softwaresvoltados exclusivamente para a área educacional [26].

Estes tipos de software vêm evoluindo ao longo do tempo, incorporando cada vez

mais os requisitos necessários a um sistema educacional. Um exemplo de software educacional são os chamado Sistemas Tutores Inteligentes (STI's), que surgiram na década de 70, fruto de pesquisas na área de Inteligência Artificial (IA), Psicologia cognitiva e Educação. Da IA, aproveitou-se os métodos relacionados à representação e manipulação do conhecimento sobre um dado domínio; da Psicologia cognitiva buscou-se as definições sobre o modelo do estudante, ou seja, *a quem* ensinar; e, por fim, da Educação utilizou-se o conhecimento de *como* e *quando* ensinar. Estes sistemas educacionais evoluíram e passaram a incorporar técnicas de IAD, utilizando uma abordagem de Sistemas Multi-Agentes para a sua concepção e desenvolvimento.

A IAD é uma sub-área da Inteligência Artificial que tem como principal objetivo o desenvolvimento de métodos e técnicas que sirvam de auxílio à resolução de problemas complexos. Para isto, as pesquisas em IAD buscam o desenvolvimento de métodos e técnicas para decompor estes problemas complexos em sub-problemas mais simples de serem resolvidos separadamente. Daí, estes sub-problemas são processados por entidades (agentes) que potencialmente têm capacidade para resolvê-los. Os métodos e técnicas de IAD estão sendo utilizados nas mais diversas áreas, como: processamento de linguagem natural, manufatura, robótica, etc. Neste trabalho, são utilizados os conceitos de IAD aplicados ao desenvolvimento de softwares educacionais.

Este trabalho tem como objetivos a (i) definição, (ii) modelagem e (iii) análise de um Sistema Tutor Multi-Agente no domínio de redes de Petri (STMA-RP). Redes de Petri [28] é uma ferramenta formal de modelagem que é aplicada apropriadamente a diversos tipos de sistemas. Para a modelagem do Sistema Tutor Multi-Agentes utilizamos uma extensão de redes de Petri denominada redes de Petri Coloridas [22]. A definição de redes pode ser encontrada no Apêndice A.

Para a concepção do STMA-RP, utilizamos como arcabouço teórico, um modelo de ambiente interativo de aprendizagem com uma abordagem Multi-Agentes, denominado MATHEMA [8; 11; 13; 9]. A questão básica no MATHEMA é envolver um determinado *aprendiz* na resolução de diversos problemas, e, a partir daí, desempenhar o papel de Tutor assistente. Para o contexto deste trabalho, a principal contribuição do MATHEMA foi a definição de um modelo de conhecimento sobre um domínio, permitindo uma estruturação mais adequada, e, consequentemente, uma melhor investigação sobre o

seu conteúdo.

A seguir, apresentamos a estrutura da dissertação.

Estrutura da dissertação

No Capítulo 2 apresenta-se o embasamento teórico necessário para o entendimento dos capítulos seguintes. Inicialmente faz-se um estudo sobre a evolução dos softwares educacionais, desde os Sistemas de Instrução Assistida por Computador (CAI's), até os Ambientes Interativos de Aprendizagem (ILE's). A seguir, apresentam-se os conceitos de IA, detendo-se nos Sistemas Tutores Inteligentes. Depois são apresentados os conceitos relativos à IAD, dando ênfase especial às definições de Agentes e Sistemas Multi-Agentes.

No Capítulo 3 descreve-se o ambiente MATHEMA, dando ênfase à definição do modelo de conhecimento, e apresenta-se a definição do modelo de conhecimento para o domínio de redes de Petri. A seguir, apresenta-se as etapas de construção da Sociedade de Agentes Tutores Artificiais (SATA), bem como a SATA referente ao domínio de redes de Petri. Por fim, descrevem-se os modelos definidos para o ambiente MATHEMA, referentes aos agentes e aos elementos necessários à sua operacionalização.

No Capítulo 4 trata-se dos modelos de redes de Petri coloridas para o ambiente descrito no Capítulo 3, e a análise realizada sobre estes modelos.

Por fim, no Capítulo 5 apresentam-se as conclusões e os trabalhos futuros.

Capítulo 2

Conceitos Gerais

Neste capítulo, apresentamos alguns conceitos que servirão de suporte para o entendimento dos capítulos seguintes. Inicialmente, apresenta-se uma visão sobre a evolução dos softwares educacionais. A seguir, apresentam-se os conceitos relativos à Inteligência Artificial, detendo-se nas definições de Sistemas Tutores Inteligentes. Prosseguindo, são detalhados conceitos relativos a um ramo de pesquisa da Inteligência Artificial, a Inteligência Artificial Distribuída (IAD). Por fim, apresentam-se algumas noções de Agentes e Sistemas Multi-Agentes.

2.1 Evolução dos Sofwares Educacionais

Os primeiros sistemas a surgirem como uma categoria de software educacional foram os Computer Aided Instruction - CAI (Sistemas de Instrução Assistida por Computador) [3]. A principal característica deste tipo de sistema é a sua busca pela redução do processo de aprendizagem, a um modelo causal do tipo estímulo-resposta. Na prática, o sistema propõe ao estudante uma série de questões sobre uma unidade de ensino. O estudante responde às questões e o sistema devolve imediatamente as realimentações correspondentes. Com isso, os alunos podem *aprender* em seu próprio ritmo.

Uma das limitações dos sistemas CAI é a incapacidade de percepção das características cognitivas individuais dos estudantes, como por exemplo: conhecimento prévio do domínio, estilo e capacidade de aprendizagem. Todos os estudantes recebem uma mesma unidade de ensino, na mesma forma e seqüência; assim, os estudantes terminam

por assumir uma postura passiva diante do processo de aprendizagem. Resumindo, os sistemas CAI não eram mais do que uma versão eletrônica dos livros de aprendizagem.

Na década de 60, o pesquisador Seymour Papert e sua equipe do *Massachusets Institute of Technology* (MIT) propôs os sistemas denominados *Micromundos*. Estes sistemas apresentavam uma proposta pedagógica oposta à apresentada pelos CAIs, já que os Micromundos têm como objetivo a aprendizagem pela ação, numa perspectiva de construção do conhecimento.

Como exemplo, temos o projeto LOGO [29] que apresenta um micromundo gráfico. Neste micromundo há um objeto representado por uma tartaruga que interage com o aluno de forma a ajudá-lo na tarefa de resolução de problemas.

Os CAIs apoiam o processo de aprendizagem através da simples transmissão do conhecimento, enquanto os Micromundos apoiam o processo de aprendizagem na construção do conhecimento por parte do aluno, inspirando-se no *construtivismo* difundido pelas idéias de Piaget [33], e no *sócio-construtivismo* apoiado por Vygotsky [44].

Os *simuladores* e os *jogos* educacionais são exemplos de softwares educacionais que possuem alguns conceitos comuns aos CAIs e aos Micromundos. A simulação busca a representação do comportamento de um objeto real. Nos simuladores o sistema interage com o usuário permitindo que este experimente o resultado de suas ações perante suas decisões sobre o sistema que está sendo simulado. Por sua vez, os jogos buscam a exploração auto-dirigida ao invés da instrução explícita e direta [43].

Diversos pesquisadores defendem que os jogos podem servir de apoio ao desenvolvimento de diversas características como: negociação, persuasão, cooperação, etc. No entanto, eles alertam que muitas vezes suas interfaces devem ser vistas com bastante cuidado, para que a atenção do aluno não seja muito desviada da percepção dos conceitos envolvidos nas atividades interativas. Dependendo do nível de intervenção oferecido por um sistema para a simulação ou jogo, este será mais parecido com um CAI, ou mais próximo de um micromundo.

Um outro exemplo de software educacional é o *tutorial*, utilizado como ferramenta de apoio ao ensino tradicional, facilitando a aquisição de conhecimentos por parte do aprendiz.

Com o tempo, os sistemas CAIs foram agregando os recursos de Inteligência Artifi-

cial e os resultados da Psicologia Cognitiva e Educação [26], dando origem aos sistemas *Intelligent* CAI (ICAI) ou Sistemas Tutores Inteligentes (STI). Estes sistemas têm como característica básica a representação de conhecimentos relacionados às questões do tipo: o *que* ensinar, a *quem* ensinar e *como* ensinar. O objetivo principal dos STIs é oferecer instrução individualizada aos aprendizes. Na Seção 2.2 apresenta-se mais informações sobre STIs.

Atualmente, observa-se a tendência de obtenção de modelos computacionais apoiados no conceito de cooperação. Neste sentido, busca-se a evolução dos STIs para os *Interactive/Intelligent Learning Environment* - ILE (Ambientes Interativos/Inteligentes de Aprendizagem) ou ainda Sistemas Tutores Cooperativos. Estes sistemas utilizam conceitos comuns aos tutores inteligentes e aos micromundos.

Ao mesmo tempo que os ambientes educacionais apoiados por computador passaram a dar ênfase à tecnologia de computação distribuída, sendo utilizados como suporte para a tecnologia de *groupware* e de CSCW (*Computer Supported Cooperative Work*) [1] , os STIs e os ILEs incorporaram resultados da Inteligência Artificial Distribuída, utilizando modelos de trabalho cooperativos através de uma abordagem de Sistemas Multi-agentes [8]. A seguir, apresenta-se os Sistemas Tutores Inteligentes.

2.2 Sistemas Tutores Inteligentes

Os STIs são programas de computador que são desenvolvidos com propósitos educacionais e que incorporam técnicas de IA, geralmente utilizando-se da tecnologia dos sistemas especialistas. Sua base de conhecimento é construída por um especialista, com base no conhecimento do tema a ser ensinado.

De acordo com Jonassen e Wang [25], três questões devem ser consideradas para que um STI seja considerado inteligente:

- O conteúdo do tema ou especialidade deve ser codificado, de modo que o sistema possa acessar as informações, fazer inferências ou resolver problemas.
- O sistema deve ser capaz de avaliar a aquisição deste conhecimento pelo estudante.

- As estratégias tutoriais devem ser projetadas a fim de reduzir a distância entre o conhecimento do especialista e o conhecimento do estudante.

A seguir falaremos de dois exemplos clássicos na área de pesquisa de STIs.

Wenger [45] afirma que o início das pesquisas no campo de STI se deu no início da década de 70, com o desenvolvimento do sistema SCHOLAR [7]. Este sistema, projetado e construído pelo pesquisador Jaime Carbonell, em um laboratório de pesquisa em Cambridge (Massachusetts), serviu de base para futuros trabalhos nesta área.

O SCHOLAR tinha como objetivo o ensino de geografia da América do Sul, utilizando como representação do conhecimento uma rede semântica [39] em cujos nodos estão os objetos e conceitos geográficos. Desta forma, o sistema faz uso de procedimentos de inferência para uma interação tutorial simples. Estes objetos e conceitos são organizados hierarquicamente, permitindo que inferências simples possam ser feitas pela propagação das propriedades hierárquicas. Por exemplo, sabendo que o Chile está na América do Sul e que Santiago está no Chile, pode-se concluir que Santiago está na América do Sul.

Uma das dificuldades encontradas pelo SCHOLAR foi a incapacidade de fazer inferências satisfatórias sobre o comportamento do aluno, não permitindo que pudesse se ajustar melhor as estratégias de ensino.

Um outro exemplo de um STI é o SOPHIE - SOPHisticated Instrucional Environment [45] - desenvolvido por John Seely Brown, Richard Burton, e seus colegas na *Bolt Beranek and Newman, Inc.* [6], que tinha como objetivo o desenvolvimento da iniciativa do estudante durante uma interação tutorial. Seu objetivo principal era criar um ambiente de aprendizagem através do qual os estudantes seriam incentivados a buscar idéias sobre suas próprias conjecturas ou hipóteses em situações de resolução de problemas.

Ao contrário do SCHOLAR, este sistema utiliza como representação do conhecimento o modelo de simulação, ao invés de uma rede semântica. O programa apresenta ao estudante a simulação de uma parte de um equipamento eletrônico com defeito. O estudante deve diagnosticar o problema fornecendo as medidas adequadas ou formulando algumas questões específicas. O sistema é projetado para responder questões hipotéticas sobre o sistema sendo simulado, e também avaliar hipóteses. O sistema tem

um modelo do conhecimento para resolução de problemas em seu domínio, assim como numerosas estratégias heurísticas para responder às questões dos estudantes, criticar suas hipóteses, e sugerir teorias alternativas. O SOPHIE permite que os estudantes tenham uma relação um-para-um com o especialista, auxiliando o surgimento de suas próprias idéias. Assim como o SCHOLAR, o projeto SOPHIE também foi marcante para estudos futuros na área, gerando uma longa e diversificada linha de pesquisa.

Wenger [45] diz que a função principal de um STI é agir como um *veículo de comunicação*. Vários trabalhos mais recentes reforçam este ponto, dando ênfase à comunicação. Portanto, é importante considerar que independente do paradigma utilizado, o objetivo fundamental de todo STI é comunicar o conhecimento e/ou habilidades para o estudante resolver problemas dentro de um determinado domínio.

Um STI possui 4 funções operacionais básicas, sendo determinados por quatro componentes principais ou modelos:

- Modelo do especialista (conhecimento do domínio) - Representa o objeto da comunicação.
- Modelo do estudante - Representa o receptor durante o processo da transmissão do conhecimento.
- Modelo pedagógico - Representa os métodos e técnicas didáticas utilizadas no processo da transmissão do conhecimento.
- Modelo da interface com o estudante - É a forma como a comunicação será realizada com o meio externo ao sistema.

O desenvolvimento de um STI requer a aplicação integrada dos quatro modelos, cujas inter-relações podem ser vizualizados na Figura 2.1. É importante ressaltar que esta arquitetura mostrada na figura 2.1 não é consensual, porém seus elementos encontram-se na maioria das arquiteturas existentes.

Durante uma sessão educacional, o sistema monitora a performance do estudante e tenta apurar o conhecimento que o estudante detém. Este processo de diagnóstico é realizado pela comparação do estado de conhecimento atual do estudante com o conhecimento contido no modelo do especialista. Os resultados desta comparação são

passados para o modelo pedagógico, onde as decisões são tomadas sobre qual, quando, e como a informação será transmitida através da interface do sistema com o estudante.

A seguir, falaremos sobre cada um destes modelos.

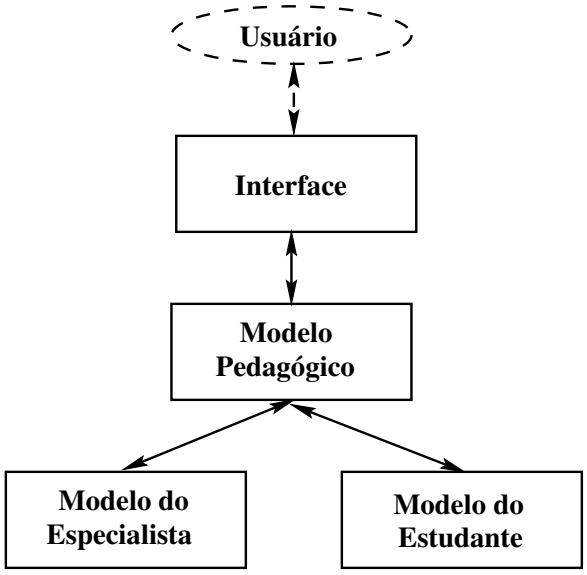


Figura 2.1: Arquitetura de um Sistema Tutor Inteligente

2.2.1 Modelo do Especialista

O modelo do especialista é fundamentalmente uma base de conhecimento, contendo informações sobre um determinado domínio, que é organizada de alguma maneira para representar o conhecimento de um especialista. É, geralmente, considerado o componente central de qualquer STI. Em resumo, este modelo incorpora a maior parte da *inteligência* do sistema na forma do conhecimento necessário para solucionar problemas referentes a um certo domínio [30].

Esta base de conhecimento contém os elementos necessários para que o estudante adquira o conhecimento sobre um domínio e os procedimentos necessários para que ele possa utilizá-los na resolução dos problemas em uma determinada área deste domínio. Para isto, este conhecimento deve ser mapeado em símbolos, de modo que o computador possa armazená-lo e manipulá-lo durante as interações com o estudante [29].

Uma das principais tarefas no desenvolvimento destes sistemas está na *aquisição de conhecimentos* [14], assim, esta tarefa necessita de uma boa cooperação entre o

projetista e o especialista. Por ser uma tarefa bastante trabalhosa, muitas vezes o Modelo do Especialista apresenta-se incompleto em muitos sistemas.

Um fator que deve ser cuidadosamente considerado é a forma na qual o conhecimento é armazenado. Nenhuma forma geral parece ser totalmente adequada para representar o conhecimento, mas tipos diferentes de raciocínio e de conhecimento, requerem diferentes representações para um uso eficiente e eficaz [35]. Portanto, a escolha da representação de conhecimento em um sistema tutorial depende do tipo de conhecimento a ser armazenado e da utilização pretendida. Alguns métodos de IA utilizados para representar o conhecimento do domínio, incluem o desenvolvimento de redes semânticas, a aplicação de regras de produção, representações procedimentais, e a construção de *frames* e *scripts*.

Resumindo, podemos afirmar que o *comportamento inteligente*, requer um conhecimento rico e suficientemente estruturado, de modo a facilitar os tipos desejados de raciocínio e as capacidades cognitivas envolvidas no processo de ensino e aprendizagem.

2.2.2 Modelo do Estudante

Jonassem e Wang [25] afirmam que a dimensão mais significativa em um sistema tutorial é a sua capacidade para modelar o conhecimento do estudante, pois a partir disto, o sistema pode se apresentar *personalizado* e *inteligente*.

Este modelo deve abranger todos os aspectos do conhecimento e do comportamento do estudante que sejam relevantes para o seu desempenho e aprendizagem. Entretanto, a construção de um modelo como este é uma tarefa bastante complexa para um sistema computadorizado. Um computador apresenta-se incapaz neste aspecto, quando comparado com a capacidade das pessoas em combinar informações em uma grande variedade de meios, como por exemplo: o tom de voz ou expressões faciais.

O modelo do estudante deve ser dinâmico, contendo o conhecimento e as capacidades do estudante, seu comportamento de aprendizagem passado, os métodos de apresentação aos quais ele responde melhor, e sua área de interesse dentro do domínio. Munido destas informações, o sistema pode atingir um nível desejável e um método de apresentação adequado, adaptando a instrução à competência e habilidade de cada estudante.

2.2.3 Modelo Pedagógico

A tarefa de ensino é guiada por estratégias e técnicas que são selecionadas e combinadas dinamicamente em reação às atitudes e necessidades dos estudantes, de modo que o assunto abordado seja compreensível e interessante para o aluno. Desta forma, a tarefa do modelo pedagógico é bastante complexa.

Este modelo contém o conhecimento necessário para tomar decisões sobre quais táticas de ensino devem ser empregadas dentre aquelas disponíveis no sistema. O Modelo Pedagógico diagnostica as necessidades de aprendizagem do estudante com base nas informações do modelo do estudante e na solução do professor contida no modelo do especialista. Em geral, as decisões são sobre qual informação apresentar ao estudante, quando e como apresentá-la.

As decisões pedagógicas são tomadas em um contexto de um ambiente educacional que determina o grau de controle sobre a atividade e sobre a interação possuídos respectivamente pelo sistema tutorial e pelo estudante [45]. Todas estas decisões são sutis. A ordem e a maneira pela qual os tópicos serão tratados poderão produzir experiências de aprendizagem diferentes. Por exemplo, em uma orientação tutorial, algumas vezes é mais eficaz deixar o estudante pesquisar à vontade do que interrompê-lo, enquanto outras vezes, esta liberdade pode deixá-lo *perdido*.

Portanto, um processo de aprendizagem depende de uma série de fatores e o sistema tutorial não deve inibir a motivação pessoal do estudante ou o seu senso de descobrimento. Este processo pedagógico requer grande versatilidade.

2.2.4 Modelo de Interface com o Estudante

Uma interface adequada é necessária para qualquer tipo de sistema, e os sistemas tutoriais não fogem a regra. É justamente através da interface que o STI realiza duas de suas principais funções: apresentação do material de ensino e a monitoração do progresso do estudante de acordo com o retorno do aluno.

É interessante que o estudante não precise realizar muito esforço na tentativa de adaptação à interface, mas sim, que a utilização do sistema seja feita de uma forma intuitiva e agradável. O esforço do estudante deve ser dirigido para o aprendizado das

lições, e não para o aprendizado do sistema. [38].

Isto é possível através de um bom projeto de interface, e, atualmente, muitos princípios baseados nas teorias cognitivas têm sido propostos para projetos de interface, como resultado de pesquisas na área da interação homem-máquina.

O aluno comunica-se com o tutor através de restrições na linguagem a fim de viabilizar a comunicação. Este aspecto relacionado com interfaces, e tradicionalmente relacionado com as pesquisas em IA, é o processamento da linguagem natural. Esta área possui um vasto campo a ser pesquisado e através do seu desenvolvimento será possível atingirmos um grau maior de amigabilidade com os computadores.

2.3 Inteligência Artificial Distribuída (IAD)

IAD [2; 39] é uma área de pesquisa de IA que tem como principal objetivo o desenvolvimento de métodos e técnicas que sirvam de auxílio à resolução de problemas complexos, que necessitam de conhecimentos sobre diferentes domínios.

Ao contrário de IA, que baseia-se em um *comportamento humano individual*, concentrando sua atenção na *representação do conhecimento* e *métodos de inferência*, IAD baseia-se em um *comportamento social*, cuja ênfase está nas *ações* e *interações* dos elementos que participam do processo de resolução do problema. Podemos dividir a IAD em duas áreas: Resolução Distribuída de Problemas (RDP) e Sistemas Multi-Agentes (SMA).

Os principais tópicos abordados em RDP tratam de questões sobre gerenciamento de informações, como a decomposição de tarefas e síntese da solução. Por exemplo, um problema pode ser dividido em diferentes (mas não independentes) sub-problemas, que podem ser tratados por diferentes agentes. Daí, estas soluções podem ser sintetizadas na solução do problema original.

Por sua vez, SMA permitem que estes sub-problemas sejam tratados por diferentes agentes, que possuem seus próprios interesses e metas. Nas duas áreas temos o conceito de *agentes*. A seguir, falaremos sobre agentes e SMA, cujos conceitos são essenciais para um melhor entendimento deste trabalho.

Agentes e SMA

Recentemente, o paradigma de agentes tem se tornado extremamente popular. Algumas das razões para o suscesso deste paradigma está na sua flexibilidade, modularidade e aplicabilidade geral para a grande classe de problemas [19].

Existem muitas definições do termo agente, e não há um consenso sobre o seu real significado. Russel e Norvig, em [36], afirmam que um agente é somente *alguma coisa* que *age* e *percebe*. Já Franklin e Graesser, em [17], apresentam uma série de definições, de diversos pesquisadores, sobre o termo agente. Hayes [19] afirma que um agente pode ser definido como uma entidade (humana ou computacional) que é capaz de atingir seus objetivos, e que faz parte de uma *comunidade*, possuindo algum tipo de influência uns sobre os outros.

Desta última definição, podemos perceber dois dos principais conceitos de agentes, que resultam na diferença dos sistemas baseados em agentes, dos demais:

- Agentes podem realizar atividades de forma autônoma - No mínimo, eles devem ser capazes de realizarem algumas instruções sem a ajuda de outros agentes. Adicionalmente, eles podem ser capazes de tomar decisões sobre eles mesmos, com vários níveis de complexidade.

- Agentes fazem parte de uma comunidade - Apesar de alguns agentes possuirem um alto grau de autonomia, eles nunca são totalmente independentes, pois eles compartilham um ambiente, e, desta forma, podem competir por recursos, intencionalmente ou não.

A definição mais utilizada pela literatura, e que adequa-se ao contexto de agentes utilizado neste trabalho, é a de Ferber [16], que diz que:

> *Agente é uma entidade real ou virtual que emerge num ambiente onde pode tomar algumas ações, que é capaz de perceber e representar parcialmente este ambiente, que é capaz de comunicar-se com outros agentes e que possui um comportamento autônomo que é uma consequência de sua observação, seu conhecimento e suas interações com outros agentes.*

As pesquisas em SMA buscam o desenvolvimento de técnicas que sirvam de auxílio ao desenvolvimento de sistemas complexos utilizando diversos agentes e vários mecanismos que coordenam o comportamento destes agentes.

Existem muitas vantagens na utilização de SMA. Uma das vantagens é o *paralelismo*, já que através de múltiplos agentes, um sistema pode otimizar sua operacionalização oferecendo uma paralelização de suas tarefas. Ao dividir um domínio em diversos componentes, diferentes tarefas podem ser manipuladas por diversos agentes.

Outra vantagem é a robustez, pois como os controles e as responsabilidades são suficientementes compartilhadas pelos diferentes agentes, o sistema pode controlar as falhas de um ou mais agentes. Apesar de não ser extremamente necessário que um SMA seja implementado sobre múltiplos processadores, para que o sistema ofereca um bom grau de robustez, é recomendável que seus agentes sejam distribuídos sobre diferentes máquinas.

Outro ponto favorável à utilização de SMA é a modularidade, facilitando a inserção de novos agentes em um SMA, bem como a mudança das capacidades e parâmetros do sistema.

A seguir, tem-se a descrição do ambiente MATHEMA, que serviu como um arcabouço para o desenvolvimento do sistema tutor multi-agentes no domínio de redes de Petri.

Capítulo 3

Sistema Tutor Multi-Agente no domínio de redes de Petri (STMA-RP)

Neste capítulo apresentamos conceitos relativos a um ambiente de ensino/aprendizagem utilizado como arcabouço conceitual neste trabalho, o MATHEMA [8; 11; 13; 9]. Paralelamente às suas definições, apresentamos as definições do Sistema Tutor Multi-Agente no domínio de redes de Petri (STMA-RP).

Inicialmente, apresentamos o MATHEMA, dando uma ênfase especial ao seu modelo de conhecimento. A seguir apresentamos o modelo de conhecimento definido pelo MATHEMA aplicado ao domínio de redes de Petri. Em seguida, enumeramos os passos necessários à construção da Sociedade de Agentes Tutores Artificiais (SATA), bem como sua definição. Após esta definição, apresentamos os agentes que compõem a SATA no domínio de redes de Petri. Por fim, mostramos o Modelo do Agente definido pelo MATHEMA.

3.1 Modelo do Conhecimento

Conforme dito na introdução, utilizamos como arcabouço conceitual para a definição de um Sistema Tutor Multi-Agentes em redes de Petri o ambiente de ensino/aprendizagem MATHEMA. De forma resumida, podemos definir o MATHEMA como sendo um modelo de ambiente interativo de aprendizagem baseado no computador, cujo propósito é apoiar atividades que promovam uma melhor interação entre o sistema, que se com-

porta como tutor, e o aprendiz, durante o processo de aprendizagem. As atividades de ensino-aprendizagem têm início a partir do processo de interações cooperativas que envolvem os seus componentes, aprendizes e tutores.

A questão básica no MATHEMA é envolver o aprendiz na resolução de diversos problemas e, a partir daí, desempenhar um papel de tutor assistente. A aprendizagem, por sua vez, é decorrente de atividades provenientes do processo de resolução de problemas, significando aquisição de conhecimento.

Para isto, concentrou-se esforços na elaboração de um método que oferecesse ao sistema tutor, um modelo adequado sobre o conhecimento de um dado domínio, levando-se em conta um compromisso entre sua riqueza e estruturação. Tudo isto levou à definição de um sistema tutor, utilizando técnicas de IAD, segundo uma abordagem baseada em Sistemas Multi-Agentes.

O resultado da busca deste modelo sobre um domínio de conhecimento foi a definição de um esquema de modelagem que oferecesse uma forma eficaz de investigação sobre determinado objeto de conhecimento. Segundo este esquema, o conhecimento sobre determinado domínio é abordado através de duas formas de visualização: uma visão externa e uma visão interna.

Seguindo uma visão externa, temos a realização de um particionamento de um determinado domínio de conhecimento em diferentes subdomínios, de acordo com alguma visão particular sobre este domínio. O objetivo disto é a busca de um particionamento e um corpo de conhecimento que lhe seja subordinado no momento de sua operacionalização, de modo que se alcance um conhecimento com especialidades distribuídas em três dimensões de conhecimento: uma para contexto, uma para profundidade e outra para lateralidade.

O contexto funciona como um ponto de vista sobre determinado domínio de conhecimento. Desta forma, para um domínio de conhecimento D_k, temos diversos contextos $\langle C_1, C_2, \cdots, C_n \rangle$ relacionados a este domínio. Para cada contexto definido C_i, onde $1 \leq i \leq n$, podemos ter diferentes profundidades relacionadas ao contexto $\langle P_{i1}, P_{i2}, \cdots, P_{im} \rangle$. A profundidade é definida a partir da tentativa de aprimoramento na linguagem de percepção. Por fim, para cada par definido de contexto e profundidade $\langle C_i, P_{ij} \rangle$, onde $1 \leq j \leq m$ podemos ter diversas lateralidades $\langle L_{ij1}, L_{ij2}, \cdots, L_{ijt} \rangle$,

onde $t \geq 0$, que funcionam como os conhecimentos de suporte sobre um determinado domínio.

Esta visão multidimensional possibilita estabelecer visões contextuais de um determinado domínio de conhecimento. Cada visão contextual pode vir acompanhada de várias alternativas de variação do ponto de vista de profundidade e lateralidade em relação a cada uma destas visões contextualizadas.

A título de exemplificação, podemos utilizar esta visão multidimensional sobre o domínio da Geometria Euclidiana Plana. Desta forma, poderíamos ter como contextos: uma visão métrica e uma visão trigonométrica. Para a visão métrica, poderíamos ter as seguintes profundidades: Triângulos Retângulos e Triângulos quaisquer. Por fim, quanto às lateralidades, fixando o contexto sobre uma visão métrica e a profundidade de triângulos retângulos, poderíamos ter os conhecimentos sobre Produtos Notáveis e Equação do 2_o grau. Em [8] temos outros exemplos de utilização deste modelo multidimensional sobre um domínio.

Utilizando uma abordagem sobre o esquema definido anteriormente segundo uma visão interna, temos uma forma de estruturação para cada subdomínio definido. Para cada d_{ij} em D, onde i e j representam, respectivamente, o i-ésimo contexto e a j-ésima profundidade associada a d_{ij}, ou fixando os domínios dl_{ijk} em DL tal como estabelecido acima, passa-se a olhá-los internamente como constituídos por um conjunto de unidades pedagógicas, definidas de acordo com os objetivos de ensino/aprendizagem específicos que estão associados a um curriculum. Simbolicamente, tem-se:

$Currculo = \{up_1, up_2, ..., up_n\}$,

onde $Currculo$ denota um currículo definido para um certo d_{ij} ou dl_{ijk}, sendo que cada up_i denota uma unidade pedagógica do $Currculo$. Estas unidades estão relacionadas segundo uma ordem definida com base em critérios pedagógicos. Cada up_i relaciona um conjunto de problemas, onde, por sua vez, para cada problema, está associado um conhecimento de suporte à sua resolução, incluindo: conceitos e resultados.

Neste trabalho, utilizamos os conceitos deste particionamento, através de uma visão interna e externa, do modelo de conhecimento, aplicado ao domínio de redes de Petri.

Modelo de conhecimento em redes de Petri

Redes de Petri é um modelo formal que é capaz de modelar apropriadamente sistemas assíncronos e com alto índice de paralelismo. Pode ser muito bem utilizada para a realização de atividades de análise, a fim de garantir que os requisitos do sistema apresentados em sua definição estão presentes no modelo construído em redes de Petri. Uma de suas vantagens está na notação gráfica, que permite uma melhor visualização dos modelos.

Apesar de suas características, o modelo clássico de redes de Petri pode não se revelar o mais apropriado para modelar sistemas que são encontrados no mundo real, devido à sua complexidade. As redes que modelam estes tipos de sistemas normalmente são grandes e complexas. Para isto, foram criados extensões de redes de Petri, que consideram aspectos relacionados com a capacidade de modelagem funcional (redes de Petri de alto nível), e aspectos relacionados às restrições temporais dos modelos. Uma descrição mais completa de redes de Petri pode ser encontrada no Apêndice A desta dissertação, e sua leitura torna-se necessária para o entendimento da modelagem do domínio de conhecimento realizada neste trabalho.

Aplicando o esquema de particionamento do domínio de conhecimento definido pelo MATHEMA, ao domínio de redes de Petri, obtivemos a modelagem do domínio de conhecimento em redes de Petri. Com esta modelagem, é possível obter diferentes visões para o domínio de redes de Petri, sendo elas: contexto, profundidade e lateralidade. A seguir apresentamos este modelo.

Domínio: Redes de Petri

Contextos:

C_1**:** Teoria de Conjuntos

C_2**:** Visão Algébrica

C_3**:** Visão Gráfica

C_4**:** Composição de Sistemas

Profundidades:

P_{i1}**:** Redes de Petri Lugar/Transição

P_{i2}: Redes de Petri Coloridas

para $i = 1, 2, 3, 4$

Lateralidades:

L_{1j1}: Operações sobre conjuntos (União, intersecção, etc.)

L_{2j1}: Álgebra Linear (operações sobre matrizes, etc.)

L_{4j1}: Fusão de Lugares

L_{4j2}: Fusão de Transições

para $j = 1, 2$

3.2 Definição da Sociedade de Agentes Tutores Artificiais (SATA)

Após a definição deste modelo de conhecimento sobre um domínio, foi naturalmente adotado uma abordagem baseada em agentes na concepção do Sistema Tutor, para fins de implementação. Consequentemente, buscou-se técnicas e funcionalidades de IAD, segundo uma abordagem de Sistemas Multi-Agentes (SMA). Em [8] podemos encontrar diversos benefícios na utilização de SMA tanto para a concepção quanto para o desenvolvimento deste ambiente de ensino/aprendizagem.

De uma forma abstrata podemos interpretar este ambiente de aprendizagem de acordo com o modelo apresentado na Figura 3.1, onde, o Sistema Tutor prepara e envia uma mensagem com conteúdo $\langle X \rangle$ para o Aprendiz, e este interpreta e reage produzindo e devolvendo-lhe uma mensagem com conteúdo $\langle Y \rangle$.

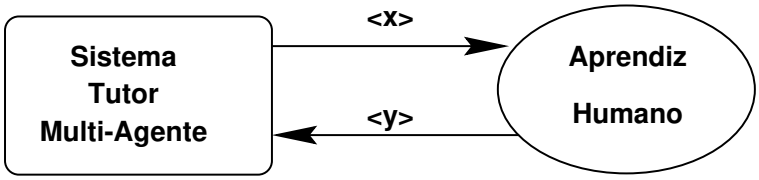


Figura 3.1: Modelo simplificado do Ambiente de Aprendizagem

Uma vez adotada uma solução baseada em SMAs, a construção da Sociedade de Agentes Tutores Artificiais (SATA), é feita de forma simples e direta. Para cada

$< C_1, P_{11} >$	$\Rightarrow$	d_{11}	$\Rightarrow$	AT_{11} - Teoria de conjuntos em redes de Petri Lugar/Transição
L_{111}	$\Rightarrow$	d_{111}	$\Rightarrow$	AT_{111} - Operações sobre conjuntos em redes de Petri Lugar/Transição
$< C_1, P_{12} >$	$\Rightarrow$	d_{12}	$\Rightarrow$	AT_{12} - Teoria de conjuntos em redes de petri coloridas
L_{121}	$\Rightarrow$	d_{121}	$\Rightarrow$	AT_{121} - Operações sobre conjuntos em redes de Petri coloridas
$< C_2, P_{21} >$	$\Rightarrow$	d_{21}	$\Rightarrow$	AT_{21} - Visão algébrica de redes de Petri Lugar/Transição
L_{211}	$\Rightarrow$	d_{211}	$\Rightarrow$	AT_{211} - Álgebra Linear em redes de Petri Lugar/Transição
$< C_2, P_{22} >$	$\Rightarrow$	d_{22}	$\Rightarrow$	AT_{22} - Visão algébrica de redes de Petri coloridas
L_{221}	$\Rightarrow$	d_{221}	$\Rightarrow$	AT_{221} - Álgebra Linear em redes de Petri coloridas
$< C_3, P_{31} >$	$\Rightarrow$	d_{31}	$\Rightarrow$	AT_{31} - Visão gráfica de redes de Petri Lugar/Transição
$< C_3, P_{32} >$	$\Rightarrow$	d_{32}	$\Rightarrow$	AT_{32} - Visão gráfica de redes de Petri coloridas
$< C_4, P_{11} >$	$\Rightarrow$	d_{41}	$\Rightarrow$	AT_{41} - Composição de sistemas utilizando redes de Petri Lugar/Transiçã
$< C_4, P_{12} >$	$\Rightarrow$	d_{42}	$\Rightarrow$	AT_{42} - Composição de sistemas utilizando redes de Petri coloridas
L_{411}	$\Rightarrow$	d_{411}	$\Rightarrow$	AT_{411} - Fusão de lugares utilizando redes de Petri Lugar/transição
L_{412}	$\Rightarrow$	d_{412}	$\Rightarrow$	AT_{412} - Fusão de transição utilizando redes de Petri coloridas

Tabela 3.1: Definição dos Agentes

$d_{ij} \in D$, define-se um agente tutor $AT_{ij}(d_{IJ} \rightarrow AT_{ij})$. Da mesma forma, para cada sub-domínio de conhecimento lateral dl_{ijk} define-se um agente tutor $AT_{ijk}(dl_{ijk} \rightarrow AT_{ijk})$ Assim, temos dois tipos de agentes: Agentes Tutores e Agentes Tutores Laterais.

Definição 3.1 *O conjunto dos agentes tutores* (ATs) *relativo a um domínio D:*

$$AT = \bigcup_{j=1}^{m} AT_{ij}, i = 1, \cdots, n$$

Definição 3.2 *O conjunto dos agentes tutores* $(ATLs)$ *relativo a um domínio DL:*

$$ATL = \bigcup_{k=1}^{t} ATL_{ijk}, i = 1, \cdots, n; j = 1, \cdots, m$$

A sociedade de agentes tutores é obtida a partir da união destes dois tipos de agentes:

Definição 3.3 $SATA = AT \cup ATL$

Aplicando os mesmo conceitos de particionamento do domínio descritos no início desta seção, para o domínio de redes de Petri, temos a definição da SATA-RP, que pode ser visualizada na tabela 3.2.

3.3 Definição do ambiente

O ambiente MATHEMA foi definido a partir da adição de novos elementos, de natureza humana e computacional, ao modelo mostrado na Figura 3.1. O princípio geral do MATHEMA é envolver um aprendiz humano em situações de aprendizagem, numa relação de interação com uma sociedade de agentes tutores artificiais, a partir de situações de resolução de problemas. Esses agentes tutores podem eventualmente cooperarem entre si ou com uma sociedade de especialistas humanos, a fim de promover a aquisição de conhecimento por parte do aprendiz. A seguir, temos a definição da arquitetura do MATHEMA, conforme ilustrado na Figura 3.2.

Definição 3.4 *A arquitetura do MATHEMA é definida pela seguinte tupla:*

$M_{arq} = \langle AH, SATA, SEH, AI, AM, ME \rangle$,

onde:

- Aprendiz Humano (AH) - elemento que possui o interesse de aprender algo sobre um determinado domínio. Possui o papel ativo durante a resolução de problemas, sendo apoiado pela assistência especializada da $SATA$.
- Sociedade de Agentes Tutores Artificiais ($SATA$) - conjunto de agentes que podem cooperar entre si, a fim de viabilizar a aquisição de determinado conhecimento ao aprendiz. Cada agente modela um subdomínio relacionado ao domínio de conhecimento.
- Sociedade de Especialistas Humanos (SEH) - funciona com um suporte à SATA. Implementa os mecanismos de inclusão e exclusão de agentes, bem como alterações no conhecimento dos agentes. Pode observar o comportamento dos agentes e incrementar a capacidade cognitiva destes.

- Agente de Interface (AI)- serve com uma *ponte* entre o aprendiz e a $SATA$. Implementa os mecanismos que viabilizam a comunicação entre estes dois elementos. Inicialmente cabe a este elemento a seleção do agente supervisor, que guiará o aprendiz durante a sessão de aprendizagem. Para isto, assume-se que o AI sabe da existência e capacidade dos agentes da $SATA$.

- Agente de Manutenção (AM) - funciona como uma ponte entre a SEH e a $SATA$. Implementa os mecanismos que viabilizam as operações de manutenção sobre a $SATA$ pelo SEH.

- Motivador Externo (ME)- entidades Humanas que servem como motivadores do Aprendiz para a utilização do MATHEMA. Podem ser professores, colegas, etc. Futuramente, essas entidades poderão ser unidas ao Aprendiz, comportando-se como um trabalho cooperativo.

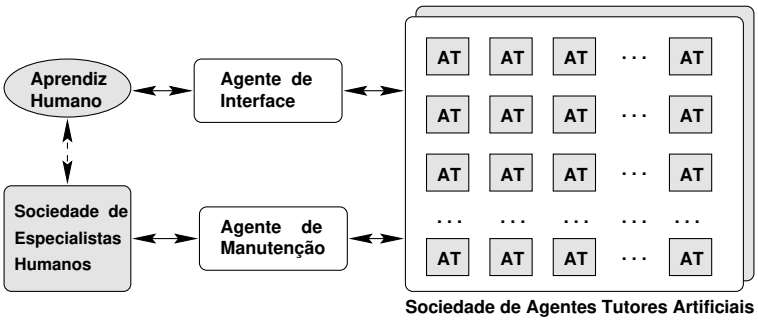


Figura 3.2: Arquitetura do MATHEMA

Analisando a Figura 3.2 podemos perceber as possíveis interações entre os elementos que compõe o MATHEMA. Para um melhor entendimento de como estas interações ocorrem, é apresentado um possível cenário de funcionamento do ambiente. A seguir, apresentamos alguns cenários aplicados ao domínio de redes de Petri.

Vamos supor que um Aprendiz Humano, possivelmente incentivado por um Motivador Humano Externo, decide iniciar uma sessão de aprendizagem. A partir deste momento é iniciada uma interação entre o aprendiz e o Agente de Interface (AI). De acordo com as informações passadas pelo aprendiz sobre os seus objetivos, o AI o auxilia a escolher seu agente supervisor na SATA. Então, daqui pra frente, inicia-se um processo de interação cooperativa e didática entre o aprendiz e o Agente Supervisor

(AS). O Agente Supervisor passa a ser o responsável pelo Aprendiz durante o seu processo de aprendizagem, oferecendo-lhe uma orientação e suporte pedagógico necessários à promoção do seu aprendizado.

Durante esta interação, podem ocorrer diferentes situações. A mais simples é a que envolve apenas a interação entre o aprendiz e o seu Agente Supervisor, numa situação de resolução de problemas. No entanto, pode acontecer a seguinte situação: durante o processo de solução de um problema, o AS pode perceber a necessidade da participação de outros agentes. Em algum momento, entretanto, é possível que a complexidade na interação evolua para uma situação mais extremada na qual agentes na SATA não conseguem atender à requisição do Aprendiz. Nesse caso, o Agente Supervisor primeiramente notifica o aprendiz sobre a impossibilidade em atendê-lo, pelo menos momentaneamente, aconselhando-o a retornar noutra ocasião, na qual a SATA estará apta a resolver o problema. Em seguida, ele informa a SEH sobre o ocorrido, através de serviços oferecidos pelo Agente de Manutenção, que tomará as medidades necessárias para reparar o ocorrido, e, caso seja necessário, realiza uma operação de manutenção na SATA. A seguir, veremos possíveis cenários de interação destes elementos relacionados ao STMA-RP definido neste trabalho.

Cenários de interações no STMA-RP

Nesta seção, veremos os cenários mais significativos que poderão ocorrer nas interações entre o aprendiz e o STMA-RP. Estes cenários foram escolhidos de modo a abranger todas as possíveis interações entre o aprendiz e o STMA-RP. Como exemplo, vamos considerar a utilização deste ambiente para auxílio no processo de modelagem de sistemas estruturados, utilizando redes de Petri Lugar/Transição. A título de exemplificação, utilizaremos como exemplo de sistema, o projeto e controle de sistemas de controle de tráfego, por se tratar de um problema estruturado e conhecido [12; 32; 10].

O problema do projetista, considerado em nosso ambiente, é projetar o sistema de bloqueio[1] para sistemas de veículos (trens, metrôs, veículos auto-guiados, etc.).

[1]Para o caso de sistemas ferroviários e metroviários o sistema de bloqueio é tradicionalmente conhecido como sistema de sinalização.

A maneira natural de resolver este problema é, inicialmente, dividir a via (férrea) em partes, denominadas blocos [4]. Cada bloco é controlado por um controlador. O controlador envia comandos que permitem a mudança de vias pelos veículos, bem como, também pode enviar sinais de forma a garantir a segurança do tráfego dos veículos. Estes sinais indicam se é permitido que um veículo saia de uma seção (sinal verde), ou se um veículo deve parar (sinal vermelho). Para projetar os controladores, a melhor solução é fazer uso dos blocos de construção. Cada bloco de construção corresponde a diferentes tipos de seções no sistema. Uma seção é um modelo para os diferentes padrões em um sistema de veículos. A partir das seções é possível modelar o sistema através de uma abordagem modular. Para isto, foram definidas 3 tipos de seções: elementares, básicas e conexões.

Figura 3.3: Seções Elementares

Seções elementares, mostradas na Figura 3.3, são as menores entidades no sistema, só permitem movimentos unidirecionais. As linhas pontilhadas representam os caminhos possíveis de um veículo. Na entrada de uma seção existe um sinal (representado pelo círculo vazio), e uma seta que indica a direção do veículo. No fim de uma seção temos um sensor (representado por um círculo preto), que gera um sinal quando um veículo sai da seção.

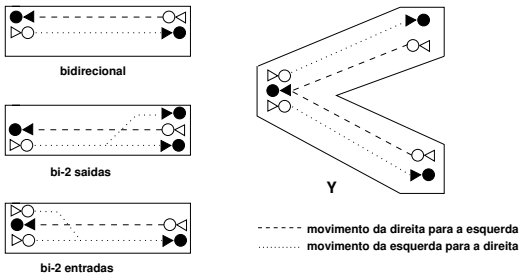


Figura 3.4: Seções Básicas

Seções básicas, mostradas na Figura 3.4, possuem uma complexidade funcional maior do que as seções elementares. Elas permitem o movimento dos veículos em

ambas as direções, porém não permitem que um veículo mude sua direção ao entrar em uma seção. Todas as seções básicas podem ser construídas através de composição das 3 seções elementares.

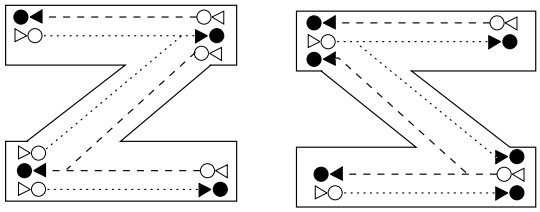


Figura 3.5: Conexões

As conexões permitem que veículos mudem de uma via para outra. Definimos 2 tipos de conexões, conforme mostrado na Figura 3.5.

O problema do projetista é: desenvolver o sistema de bloqueio para sistemas de veículos [10]. O sistema a ser desenvolvido deve desempenhar algumas funções, dentre as quais o roteamento e o bloqueio dos veículos que garantam a segurança do trânsito dos veículos pelas vias.

Agora, vamos considerar os seguintes cenários que poderiam ocorrer entre o aprendiz e o STMA-RP, considerando o seu objetivo como: buscar auxílio à modelagem de um sistema de controle de tráfego. Os agentes relacionados nos cenários, correspondem aos agentes definidos na tabela 3.2.

Situação 1: Diagnóstico

AT_{11} : Defina formalmente uma rede de Petri (RP) lugar/transição

Aprendiz : Apresenta a seguinte rede de Petri[2]:

- $P = \{p_1, p_2\}$
- $T = \{t_1\}$
- $F = \{(p_1, t_1), (t_1, p_2), (p_1, p_2)\}$
- $W = \{[(p_1, t_1), 1], [(t_1, p_2), 1], [(p_1, p_2), 1]\}$
- $M_0 = \{(p_1, 1), (p_2, 0)\}$

[2]Conforme descrito no Apêndice A, consideramos uma rede de Petri formada pela tupla: $PN =< P, T, F, W, M_0 >$

AT_{11} : Realiza o diagnóstico sobre a resposta do aprendiz e conclui que não está correto. Verifica que o erro está na função F, na definição do relacionamento (p_1, p_2), interligando 2 lugares. Retorna para o aprendiz que F é definida como: $F \subseteq (P \times T) \cup (T \times P)$, e que não pode haver um arco de um lugar para outro lugar.

Logo após a situação 1, poderíamos ter o seguinte cenário:

Situação 2: Dica

Aprendiz : O que significa "$\subseteq$"?

AT_{11} : Identifica que este assunto não é da sua competência. Consulta seu Conhecimento Social (CS) e verifica que este assunto é da responsabilidade do agente AT_{111}. Assim, o agente AT_{11} inicia uma cooperação com o agente AT_{111} com o intuito de resolver a questão colocada pelo Aprendiz. Daí, o agente AT_{111} retorna a seguinte dica ao agente AT_{11}, que por sua vez, exibe a dica ao Aprendiz:

Dica: O símbolo "$\subseteq$"significa "está contido ou é igual".

Logo após a situação 2, poderíamos ter o seguinte cenário:

Situação 3: Instrução

Aprendiz : Informa que não entendeu a dica apresentada anteriormente

AT_{11} : Pergunta ao aprendiz se ele deseja mais um *exemplo* ou uma *instrução* sobre o assunto.

Aprendiz : Informa que deseja uma instrução.

AT_{11} : Inicia novamente uma sessão de cooperação com o agente AT_{111}, relativo à tarefa de *explanação* sobre o assunto abordado pelo aprendiz. Daí, o agente AT_{111} envia a seguinte explanação ao agente AT_{11}, que por sua vez envia esta explanação ao Aprendiz:

Explanação: "Dizemos que um conjunto A está contido ou é igual a um conjunto B, quando, para cada elemento pertencente ao conjunto A, existe

um elemento equivalente ao conjunto B; ou, se o conjunto A é igual ao conjunto B."

Logo após a situação 3, poderíamos ter o seguinte cenário:

Situação 4: Resolução de Problemas

Aprendiz : Ache os invariantes de lugar e transição de uma RP lugar/transição PN, representada pela matriz de incidência[3]:

$$C = \begin{vmatrix} -1 & 1 & 2 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & -2 & -1 & 1 \\ 1 & 0 & 0 & 0 & -1 \end{vmatrix}$$

AT_{11} : Identifica que este assunto não é de sua responsabilidade. Ao consultar o seu CS verifica que este assunto é de responsabilidade do agente AT_{21}. Então, passa o controle para o agente AT_{21}, ou seja, a partir daqui este agente passa a ser o Agente Supervisor.

AT_{21} : Tenta resolver o problema, mas *percebe* que necessita de conhecimentos sobre operações com matrizes. Ao consultar o seu CS verifica que o agente AT_{211} é o responsável sobre este assunto. Invoca um pedido de cooperação com o agente AT_{211}.

AT_{211} : Retorna ao agente AT_{21} os conhecimentos sobre operações com matrizes.

AT_{21} : Retorna ao aprendiz a seguinte resposta do problema proposto:[4]

Os P-invariantes são:

$$y_1 = (1,1,1,0,0,0,0,0,0)^T \Rightarrow M(p_1) + M(p_2) + M(p_3) = 1 \qquad (3.1)$$

$$y_2 = (0,0,0,1,1,1,0,0,0)^T \Rightarrow M(p_4) + M(p_5) + M(p_6) = 1 \qquad (3.2)$$

[3]Conceitos sobre invariantes de redes de Petri podem ser encontrados no apêndice A.

[4]o agente poderia retornar ao aprendiz a incapacidade de resolver tal problema, caso isto acontecesse. Logo depois, informaria à Sociedade de Especialistas Humanos tal incapacidade.

$$y_3 = (0,0,1,0,0,1,1,0,0)^T \Rightarrow M(p_3) + M(p_6) + M(p_7) = 1 \quad (3.3)$$

$$y_4 = (0,0,0,0,0,0,0,1,1)^T \Rightarrow M(p_8) + M(p_9) = 1 \quad (3.4)$$

onde, os suportes para cada invariante são:

$||y_1|| = \{p_1, p_2, p_3\}$, $||y_2|| = \{p_4, p_5, p_6\}$, $||y_3|| = \{p_3, p_6, p_7\}$ e $||y_4|| = \{p_8, p_9\}$

Logo após o a situação 4, poderíamos ter o seguinte cenário:

Situação 5: Modelagem de um sistema de controle de tráfego

Aprendiz : O projetista define o sistema de tráfego através de um grafo indicando pontos de conexões e direções, como apresentado na Figura 3.6, e requisita que o sistema retorne a rede de Petri que modele este tipo de sistema.

AT_{21} : Este agente identifica que este assunto não é de sua competência. Ao consultar o seu conhecimento social, verifica que este assunto é de competência do agente AT_{41}. Desta forma, passa o controle da sessão para este agente, portanto, daqui para a frente este agente passa a ser o Agente Supervisor do aprendiz.

AT_{41} : Este agente particiona este grafo em seções básicas, onde cada seção possui uma rede de Petri equivalente. Barros [4] mostra que a rede de Petri do modelo global, pode ser obtida através da composição destas redes de Petri mais simples, e, prova que o modelo global conserva as mesmas propriedades dos modelos mais simples[5]. Como exemplo, temos a rede de Petri da Figura 3.7, que modela o funcionamento de uma seção simples (unidirecional), mostrada na Figura 3.3. Conforme dito anteriormente, as seções básicas podem ser obtidas através da composição das seções elementares. Assim, temos que a seção bidirecional é obtida através da composição de seções unidirecionais. Métodos de composição de redes de Petri, utilizam técnicas de fusão de lugar e fusão de transição, para a costrução de sistemas mais

[5]Mais detalhes podem ser encontrados no Apêndice A

estruturados. Então, o agente AT_{41} invoca a cooperação do agente AT_{411}, para que este obtenha a rede de Petri resultante da fusão dos lugares *ec* e *sl*, da rede de Petri mostrada na Figura 3.7.

AT_{411} : Este agente recebe as redes que devem ser fundidas, e os respectivos lugares onde serão realizados a fusão. Então, realiza a tarefa e retorna a rede Petri resultante desta fusão, que pode ser visualizada na Figura 3.8.

AT_{41} : Através dos métodos de composição, e com a ajuda dos agentes AT_{41} e do agente AT_{42}, este agente vai compondo a rede de Petri que modela o comportamento do sistema proposto pelo aprendiz, e então, envia a resposta ao aprendiz.

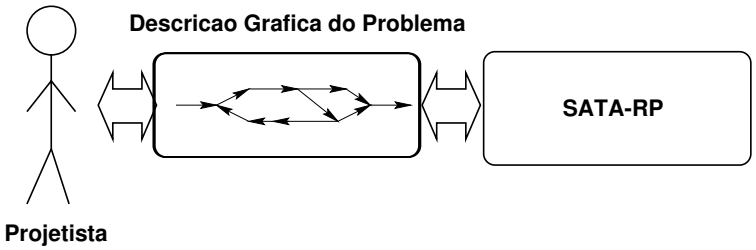


Figura 3.6: Descrição gráfica do problema do projetista

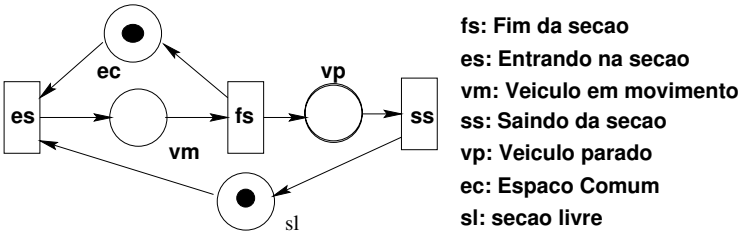


Figura 3.7: Modelo de redes de Petri para a seção unidirecional

3.4 Modelo dos Agentes

Como dito anteriormente, cada um dos agentes que compõe a SATA, é definido como uma entidade especializada em algum subdomínio específico, tendo assim, o conhecimento necessário para realizar tarefas pedagógicas nesse subdomínio. Esses agentes

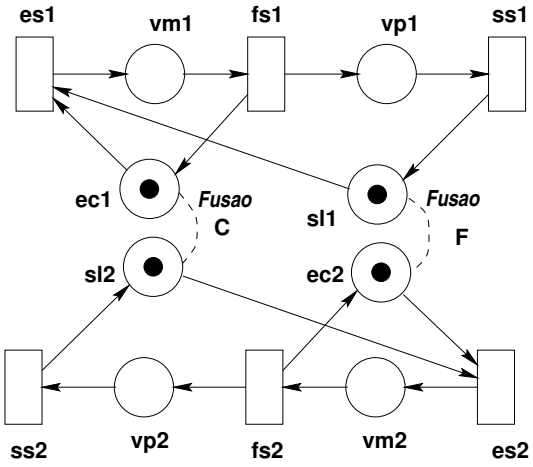


Figura 3.8: Modelo de redes de Petri para a seção bidirecional

desempenham papéis que incluem principalmente os de tutores inteligentes cooperativos, no momento de suas interações com um aprendiz. Para isto, foram criadas condições que permitissem a organização, comunicação e cooperação entre os agentes, durante o processo de resolução de uma tarefa.

Cada agente possui um conjunto de habilidades que são utilizadas apropriadamente durante a execução das tarefas. Estas habilidades dizem respeito a um método para resolver uma determinada classe de tarefa. Já o método corresponde a uma das três categorias de atividades pedagógicas: resolução de problema, diagnóstico ou instrução. A tarefa, é composta pela informação da habilidade que pode executá-la e pelos recursos necessários à sua execução.

Para viabilizar a interação entre os agentes, foram definidos mecanismos que oferecessem aos agentes a possibilidade de ter pleno conhecimento do seu *conhecimento*, e do *conhecimento* dos outros agentes. Estes mecanismos são definidos como *autoconhecimento* e *conhecimento social*, respectivamente.

Os agentes tutores são organizados como agentes cognitivos, no sentido de que ele simula tipos de interações existentes em organizações sociais, como em um departamento de uma empresa.

De um ponto de vista de controle, temos que as atividades cooperativas do MATHEMA são essencialmente distribuídas. Durante as interações entre o Aprendiz e o sistema, a entidade encarregada de guiar o aprendiz durante a sessão de aprendizagem é o agente supervisor. A partir do momento que um agente supervisor necessita de coope-

ração, e identifica os agentes que o auxiliarão nesta atividade, os agentes que o estão auxiliando passam a ter o controle nessa instância de cooperação. Resumidamente, podemos dizer que o controle não está sob a responsabilidade de um agente específico, mas todos podem eventualmente vir a exercê-lo.

A fim de viabilizar as atividades de cooperação entre os agentes, foram definidos protocolos que especificam como as interações entre os agentes podem ocorrer. Estes protocolos formam um modelo de cooperação híbrido, no sentido de que utiliza tanto um modelo de organização Mestre-Escravo, quanto um modelo baseado em Licitação.

O modelo mestre-escravo é utilizado na seguinte situação: um determinado agente (mestre) tem a responsabilidade de executar uma tarefa, no entanto necessita da cooperação de algum outro agente (escravo) para sua execução. Assim, ele envia a tarefa para este agente identificado (escravo), e, quando o agente requisitado conclui a tarefa, ele retorna os resultados ao agente que lhe requisitou (mestre).

Já o modelo de licitação é utilizado quando determinado agente tem uma tarefa a ser resolvida, porém, não tem condições de definir *qual* agente está habilitado para resolvê-la. Então, este agente envia um anúncio da tarefa para toda a sociedade ou para uma parte dela, já que este agente sabe dos endereços dos outros agentes. Os agentes que avaliarem possuir condições de execução da tarefa, enviam uma proposta de volta. Caso mais de um agente avalie ter condições de executar a tarefa e responder favoravelmente, o anunciante escolhe, com base em critérios de afinidade um dentre os possíveis executores da tarefa, e ativa o modelo Mestre-Escravo com ele. Caso apenas um agente se proponha, o anunciante, da mesma forma, ativa o modelo mestre-escravo. No caso mais extremado, quando nenhum agente dá um retorno positivo, chega-se numa situação de impossibilidade de cooperação com a sociedade de agentes, tendo assim que se recorrer ao mundo externo, a Sociedade de Especialistas Humanos.

As interações entre os agentes podem ocorrer caso exista algum mecanismo para promover a comunicação entre eles. Nesse sentido, a comunicação foi definida como um suporte para o desenvolvimento das atividades interativas. O modelo de comunicação utilizado pelos agentes na SATA define um mecanismo de comunicação baseado em troca assíncrona de mensagens. Esse modelo permite que cada agente possa se comunicar com os demais agentes da sociedade. Desta forma, podemos definir quatro

requisitos que devem ser satisfeitos pelo modelo de comunicação:

- definição de um meio de comunicação pelo qual as mensagens possam trafegar;
- definição de uma linguagem de comunicação comum que assegure um entendimento mútuo entre o emissor e o receptor de uma mensagem;
- definição de tipos de endereçamento que contemplem as necessidades do sistema;
- definição de um esquema que esclareça a informação do agente interlocutor que envia uma mensagem.

Maiores detalhes sobre os modelos dos agentes podem ser encontrados em [8] A seguir, apresentamos a arquitetura dos agentes tutores.

Arquitetura dos Agentes Tutores

Para a definição da arquitetura dos agentes tutores foram considerados as funcionalidades de cada um dos seus componentes isoladamente, e em seguida uma visão funcional integrada dos sistemas. A apresentação da arquitetura está estruturada de forma a oferecer uma visão em dois níveis distintos de abstração: o nível macro e o nível micro. Inicialmente veremos o nível macro.

Em um nível macro, temos a definição de um modelo conceitual de arquitetura que foi elaborado para um agente tutor. Este modelo define sua estrutura através de uma composição hierárquica de três componentes: os sistemas tutor, social e de distribuição. A seguir, temos a descrição de cada um destes sistemas.

- Sistema Tutor - interage diretamente com o aprendiz humano, cabendo a ele a execução das atividades tutoriais. Isoladamente pode ser visto como um sistema tutor inteligente. É nesse sistema onde estão os conhecimentos que o agente possui para resolver problemas e para efetuar outras operações pedagógicas no domínio de aplicação.
- Sistema Social - viabiliza o comportamento cooperativo entre os agentes tutores. Ele é composto por bases de conhecimento e mecanismos de raciocínio necessários

para realizar tal comportamento. Além disso, ele oferece recursos para cooperação com a Sociedade de Especialistas Humanos (SEH), através do agente de manutenção.

- Sistema de Distribuição - manipula as mensagens enviadas e recebidas pelo agente tutor, através do meio de comunicação. De uma forma resumida, podemos dizer que este sistema viabiliza a execução daquilo que o Sistema Social decide.

Em um nível micro, temos uma visão interior da arquitetura de um agente tutor, descrevendo os módulos de cada um dos sistemas mencionados anteriormente, destacando-se apenas os mais relevantes na viabilização da interação entre agentes, e destes com a SEH. A seguir, temos a descrição de cada um destes sistemas, que podem ser visualizados na Figura 3.9. Nesta figura podemos perceber a existência de alguns módulos destacados com tons de cinza mais escuro, que dizem respeito às bases de conhecimento de apoio a atividade cooperativa. Já os outros módulos representam mecanismos operacionais.

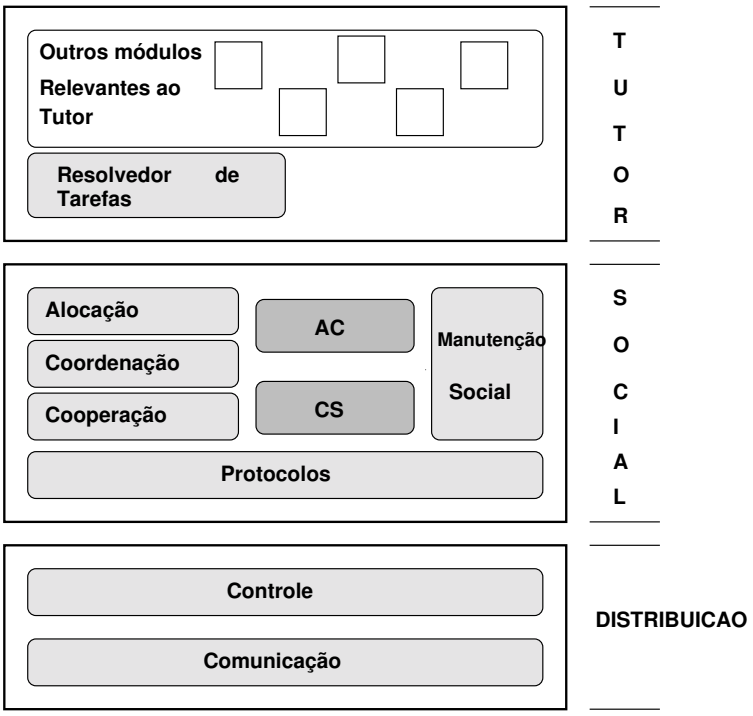


Figura 3.9: Arquitetura do Agente Tutor - visão micro

- Resolvedor de Tarefas - a principal tarefa deste módulo é a resolução de problemas. Os agentes tutores possuem suas especialidades em resolver tarefas em domínios bem específicos. Entretanto, podem existir certas tarefas em que suas soluções envolvam mais do que a especialidade de um agente tutor. Cabe ao Resolvedor de Tarefas, realizar as tarefas que são de sua competência e, casos seja necessário, identificar as que não são de sua competência. Este módulo conta com várias funcionalidades, entre elas inclui-se um método de decomposição de tarefas, que opera sobre tarefas descritas apenas na linguagem algébrica que foi definida para representar os domínios utilizados.

  Cada agente possui um mecanismo de raciocínio para operar sobre o modelo de suas capacidades e o modelo das capacidades dos outros agentes. Para fazer isso, ele conta com três módulos, que são aqui denominados: Autoconhecimento, Conhecimento Social e Alocação.

- Autoconhecimento (AC) - representa o que o agente sabe sobre as suas próprias habilidades e conhecimentos. Este módulo é utilizado quando um agente necessita decidir se ele possui conhecimento para resolver uma certa tarefa pedagógica.

- Conhecimento Social (CS) - é um modelo no qual o agente representa explicitamente o conhecimento sobre os outros agentes tutores na sociedade. Este módulo é usado quando o conhecimento do agente é insuficiente, para resolver uma dada tarefa. Assim, o agente utiliza o CS para encontrar agentes que possam executar a tarefa. O CS é definido pelo conjunto de ACs dos outros agentes tutores, e, devido ao dinamismo inerente à sociedade, este modelo pode ser alterado no decorrer das interações.

- Alocação - aloca agentes identificados como aptos a resolver uma determinada tarefa que foi submetida pelo Sistema Tutor para cooperação.

- Coordenação - interpreta a estrutura de tarefas, com base numa dependência causal definida, determinando a cada momento a tarefa a ser repassada para o próximo módulo, no caso Cooperação, visando a execução desta tarefa. A idéia é assegurar a execução coerente da tarefa completa. Além disso, é este módulo que

garante a distribuição e coleta correta dos seus recursos. Para tanto, baseia-se na estrutura da decomposição da tarefa. O módulo Coordenação opera sincronizando (paralelizando ou serializando) as execuções das subtarefas. O insucesso na cooperação obtido para uma determinada subtarefa particular Tn acarreta no insucesso na cooperação para a tarefa T, como um todo. Quando isto acontece, o módulo de coordenação espera a finalização das cooperações em andamento, não realiza as pendentes, e aborta o processo. O módulo Coordenação é funcionalmente justificado devido à sua necessidade para tratar os possíveis pedidos de cooperação a partir de tarefas decompostas em subtarefas. Se não existisse tal possibilidade, este módulo não seria necessário na arquitetura. Com isto, fica fácil entender que o trabalho do módulo Coordenação é útil apenas no tratamento das situações que envolvem tarefas decompostas; caso contrário, para uma única tarefa, a função do módulo torna-se praticamente inexistente.

- Cooperação é o módulo que promove a execução de uma tarefa. Considerando que a execução de uma tarefa não é assegurada diretamente ou mesmo garantida de ser consolidada, cabe a esse módulo exaurir todas as possibilidades de cooperação, até alcançar uma situação conclusiva. Quando se trata de uma situação na qual mais de um agente está apto a cooperar sobre uma dada tarefa, gera-se então um situação de conflito. Isso porque o agente que solicitou a cooperação terá que escolher um entre os agentes aptos. Assim, é também função do módulo Cooperação resolver um tal conflito, isto é, escolher um agente. Além disso, o módulo Cooperação dispõe de um mecanismo para a ativar uma instância de execução de um dado protocolo, no momento do encaminhamento da atividade cooperativa ao agente que vai cooperar.

- Controle - é o módulo que intermedia ações entre o Sistema Social (através do módulo responsável pelas instâncias de diálogo em execução) e o Sistema de Distribuição. Ele conta com um mecanismo de apoio à tomada de decisão para saber se uma dada mensagem está relacionada a uma atividade de cooperação ou de manutenção. Uma vez tomada a decisão, ele encaminha a mensagem ao destino apropriado. O módulo Controle tem a sua importância nos dois sentidos. Quan-

do uma mensagem é recebida, ela precisa ser encaminhada para a instância de diálogo apropriada, cabendo ao módulo Controle efetuar isto. Para tal trabalho, as mensagens que trafegam pelo ambiente de comunicação devem ser dotadas de informações adicionais. O módulo Controle também é dotado de informações e mecanismos que permitem desconsiderar o recebimento de uma determinada mensagem. Isto acontece especificamente devido ao atraso de sua chegada, sendo uma das seguintes situações: ou a instância de diálogo responsável pelo seu tratamento não existe mais; ou o tempo estabelecido para recebê-la, após uma difusão, expirou. No outro sentido, de uma instância de diálogo para o módulo Comunicação, a função do Controle é bastante simples, significando apenas colocar na mensagem a ser enviada as informações de controle sobre a instância em questão, visando o controle adequado no outro lado do diálogo.

- Comunicação - é o módulo encarregado da distribuição e coleta das mensagens, fazendo assim, a mediação entre o sistema de distribuição e o meio de comunicação. A visão do módulo Comunicação cria uma abstração importante na independência de todo o mecanismo existente acima dele com relação ao meio de comunicação utilizado. O módulo Comunicação implementa um modelo de comunicação que é responsável pelas atividades de envio e recebimento das mensagens através do ambiente de comunicação, comum a todos os ATs da sociedade.

Pelos conceitos apresentados neste capítulo, pudemos perceber a utilização do MATHEMA como arcabouço conceitual para a concepção de um Sistema Tutor Multi-Agente no domínio de redes de Petri (STMA-RP). A maior contribuição do MATHEMA ao STMA-RP foi a sua proposta de modelagem sobre um domínio de conhecimento. Em nosso trabalho, aplicamos esta modelagem sobre o domínio de redes de Petri. Outra contribuição do MATHEMA foi a sua arquitetura, utilizada neste trabalho para definir a arquitetura do STMA-RP. A seguir, apresentamos a modelagem do STMA-RP.

Capítulo 4

Modelagem e Análise do Sistema Tutoe Multi-Agente em redes de Petri

Neste capítulo apresentamos os modelos de redes de Petri coloridas do STMA-RP, bem como detalhamos sua análise. Inicialmente apresentamos os passos que seguimos para a construção dos modelos, e a ferramenta Design/CPN utilizada para a edição, simulação e análise dos modelos [23]. A seguir, apresentamos e descrevemos cada modelo, e, por fim, apresentamos resultados de análise. Para a análise dos modelos, definimos possíveis cenários de interações entre os elementos que compõe o modelo, e a seguir, realizamos a simulação e construção dos espaços de estados.

4.1 Introdução

Para a obtenção dos modelos das redes de Petri coloridas foram realizados os seguintes passos:

- Elaboração dos principais requisitos que o sistema deve apresentar;
- Construção do modelo;
- Simulação do modelo, utilizando a ferramenta Design/CPN;
- Integração do modelo do Sistema Tutor ao modelo do Sistema Social [27].

O Design/CPN é um pacote de ferramentas para o desenvolvimento de modelos de redes de Petri Coloridas, que possui quatro ferramentas computacionais integradas em um único pacote:

1. editor gráfico para construir, modificar e analisar sintaticamente os modelos CPN;
2. um simulador de operação interativa ou automática, capaz de definir diferentes critérios para parada e observação da evolução da rede;
3. uma ferramenta para gerar e analisar grafos de ocorrência de modelos CPN;
4. uma ferramenta para análise de desempenho que possibilita a simulação e observação de dados de desempenho de modelos CPN.

O Design/CPN é um dos pacotes de ferramentas mais elaborados para construção, modificação, e análise de redes de Petri coloridas, e tem sido bastante utilizado em diferentes domínios de aplicação.

As etapas descritas anteriormente para a obtenção do modelo não foram realizadas nesta ordem; mas, pelo contrário, muitas vezes foi necessário a volta para a elaboração dos requisitos, após ser realizada a simulação através da ferramenta Design/CPN. Também foram inseridos diversos elementos (lugares e transições) ao modelo, para tornar possível a simulação através da ferramenta. A seguir, temos a descrição dos modelos.

4.2 Descrição do Modelo

Nesta seção descrevemos os modelos para cada um os elementos da arquitetura do MATHEMA que estão ilustrados na Figura 4.1

O modelo construído é uma rede de Petri colorida hierárquica[1], e os seus elementos podem ser visualizados através da Figura 4.2.

A figura 4.2 representa a página hierárquica dos modelos, e permite termos uma visão global dos elementos que compõe o modelo (páginas), bem como os possíveis relacionamentos entre eles. As páginas são representadas pelos retângulos com as bordas

[1] A definição de uma rede de Petri colorida hierárquica pode ser encontrada no Apêndice A deste documento

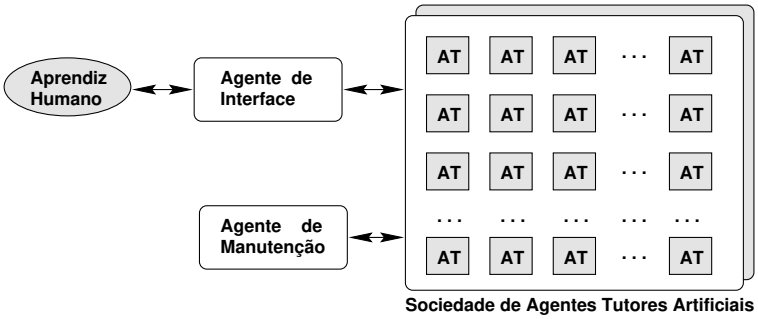


Figura 4.1: Elementos da arquiteura do MATHEMA modelados neste trabalho

arredondadas, e os arcos indicam os possíveis relacionamentos entre os elementos. O retângulo de origem de um arco indica a página que realiza determinada requisição à outra página (retângulo destino do arco).

Convém ressaltar que as redes: *Alocation*, *Coordenation*, *Cooperation*, *Coop_Exec* e *Coop_Encaminha*, são referentes ao Sistema Social. Estas redes representam a funcionalidades do Sistema Social, e maiores detalhes sobre estas redes são descritos em [27]. Conforme dito anteriormente, um dos objetivos deste trabalho é a modelagem e análise do Sistema Tutor. Ainda realizamos a integração dos modelos referentes ao Aprendiz e Sistema Tutor com os modelos referentes ao Sistema Social, para podermos analisar com uma maior abrangência o funcionamento do ambiente como um todo. Desta forma, o funcionamento das redes referentes ao Sistema Social não serão descritas neste documento.

A seguir, apresentamos uma descrição de cada um dos elementos do modelo.

4.2.1 Aprendiz

O modelo do Aprendiz é mostrado na Figura 4.3. Pela Figura 4.2 podemos perceber que esta rede relaciona-se com outras duas redes: *Agente de Interface (AI)* e *Sistema Tutor (ST)*. Esta rede modela as seguintes funcionalidades do Aprendiz:

1. Inicialização do sistema

2. Resolução de um teste para a verificação do nível de conhecimento do Aprendiz sobre o domínio em questão.

3. Receber uma explanação ou dica sobre determinado assunto.

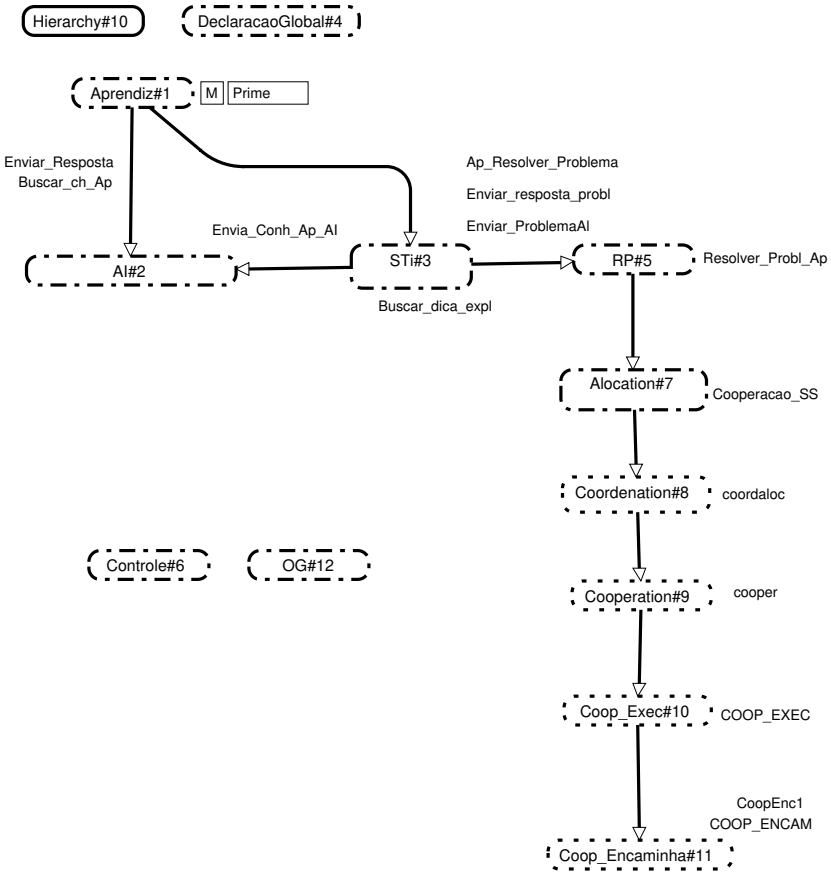


Figura 4.2: Página hierárquica referente aos modelos

4. Resolver problemas proposto pelo sistema.

5. Propor a resolução de problemas para o sistema.

O Aprendiz interage com o Agente de Interface para a realização das tarefas 1 e 2, descritas acima, e, no restante das tarefas, o Aprendiz interage com um Agente Tutor, denominado Agente Supervisor, conforme descrito na Seção 3.3. Mais especificamente, com o Sistema Tutor deste Agente.

4.2.2 Agente de Interface (AI)

O modelo do Agente de Interface é apresentado na Figura 4.4. Conforme podemos ver pela Figura 4.2, esta rede relaciona-se com outras 2 redes: *Aprendiz*, já descrita anteriormente, e, *Sistema Tutor (ST)*. Esta rede modela as seguintes características do Agente de Interface:

1. Percepção do nível de conhecimento do Aprendiz, para a escolha do Agente Supervisor. Para isso, esta rede modela o envio de um problema para ser resolvido pelo aprendiz, e simula o diagnóstico deste problema.

2. A partir deste diagnóstico, seleciona o Agente Supervisor para interagir com o Aprendiz e o sistema.

3. Armazena a informação que indica o nível de conhecimento do aprendiz, para que, desta forma, da próxima vez que o Aprendiz utilize o sistema, o sistema perceba esta situação, e, adeque-se da melhor forma para o Aprendiz. O modelo *AI* interage com o modelo *ST*, que representa o Sistema Tutor, justamente na situação de atualização do nível de conhecimento do Aprendiz.

4.2.3 Sistema Tutor (ST)

Esta rede, que pode ser visualizada na figura 4.5 modela o comportamento do Sistema Tutor. Esta rede interage com outras 3 redes: *Aprendiz* e *Agente de Interface (AI)*, já descritas anteriormente, e, *Resolvedor de Problemas (RP)* do Sistema Tutor. A rede *Sistema Tutor (ST)* modela as seguintes características do Sistema Tutor:

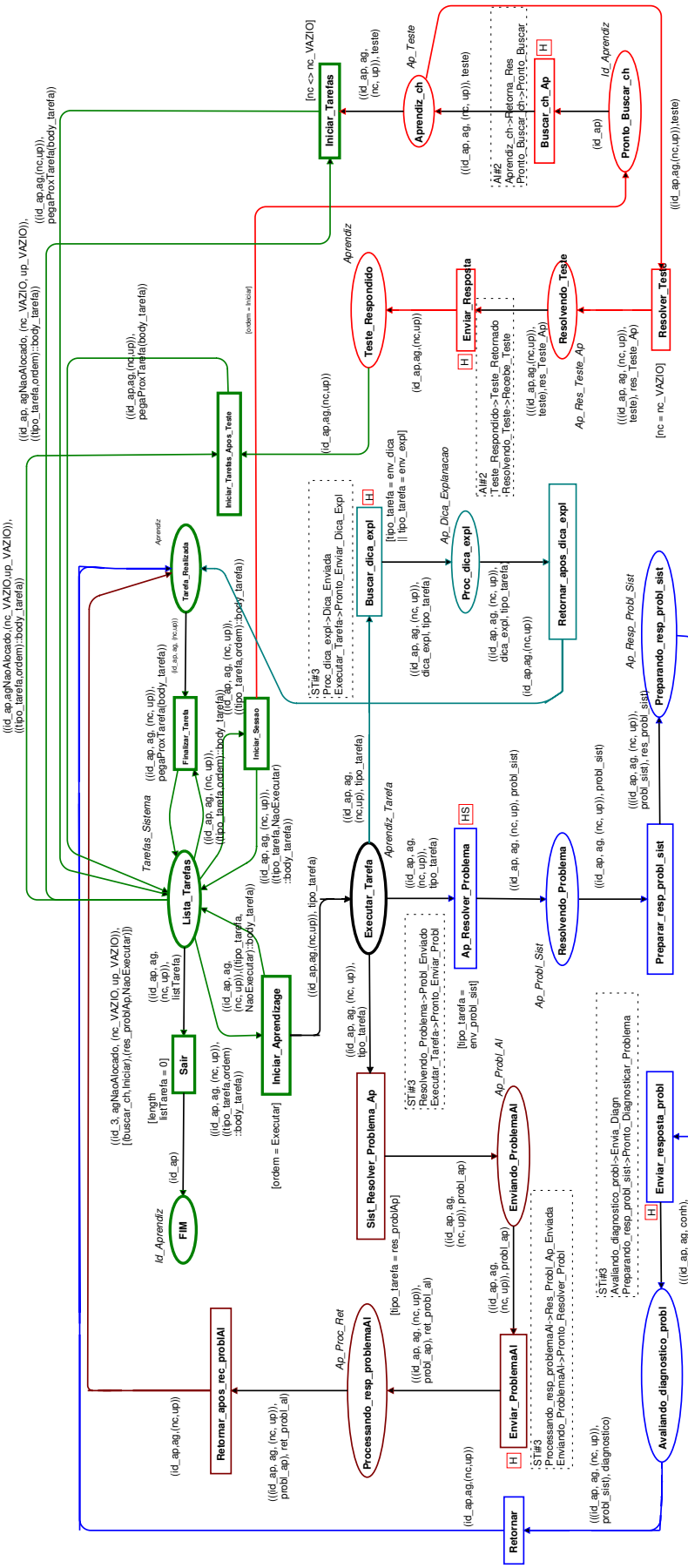


Figura 4.3: Modelo do Aprendiz

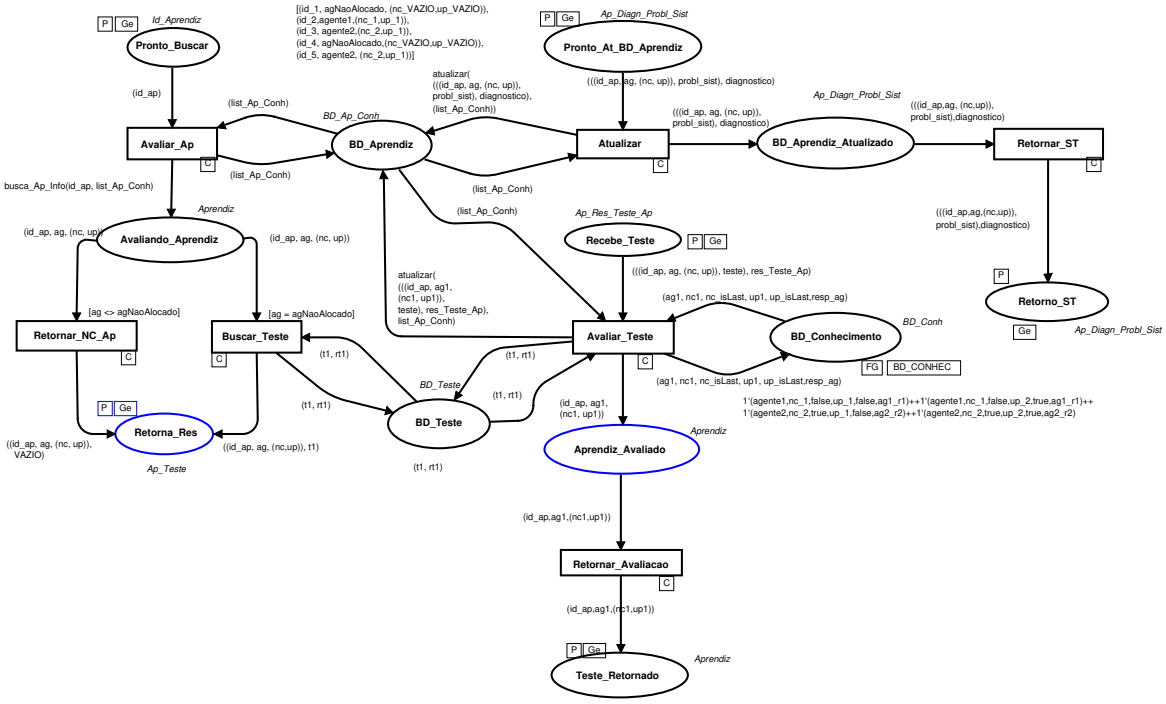


Figura 4.4: Modelo do Agente de Interface

1. Envio de Dicas ou Explanação sobre determinado assunto.

2. Envio de Problemas para serem resolvidos pelo aluno.

3. Diagnóstico da resolução dos problemas propostos ao Aprendiz.

4. Resolução de Problemas propostos pelo Aprendiz para serem resolvidos pelo sistema. No entanto, a responsabilidade desta tarefa cabe ao módulo Resolvedor de Problemas, conforme descrito na seção 3.3. A rede *RP* que modela o Resolvedor de Problemas é descrito a seguir.

4.2.4 Resolvedor de Problemas (RP)

Esta rede, que pode ser visualizada na Figura 4.6, modela o comportamento do módulo referente ao Resolvedor de Problemas. Este modelo interage com outras 2 redes: *ST*, já descrita, e, *Alocation* que faz parte do modelo que simula o comportamento do Sistema Social de um agente. Esta rede modela a tarefa de alocação de agentes para a realização de uma lista de tarefas enviada pelo Resolvedor de Problemas. A rede

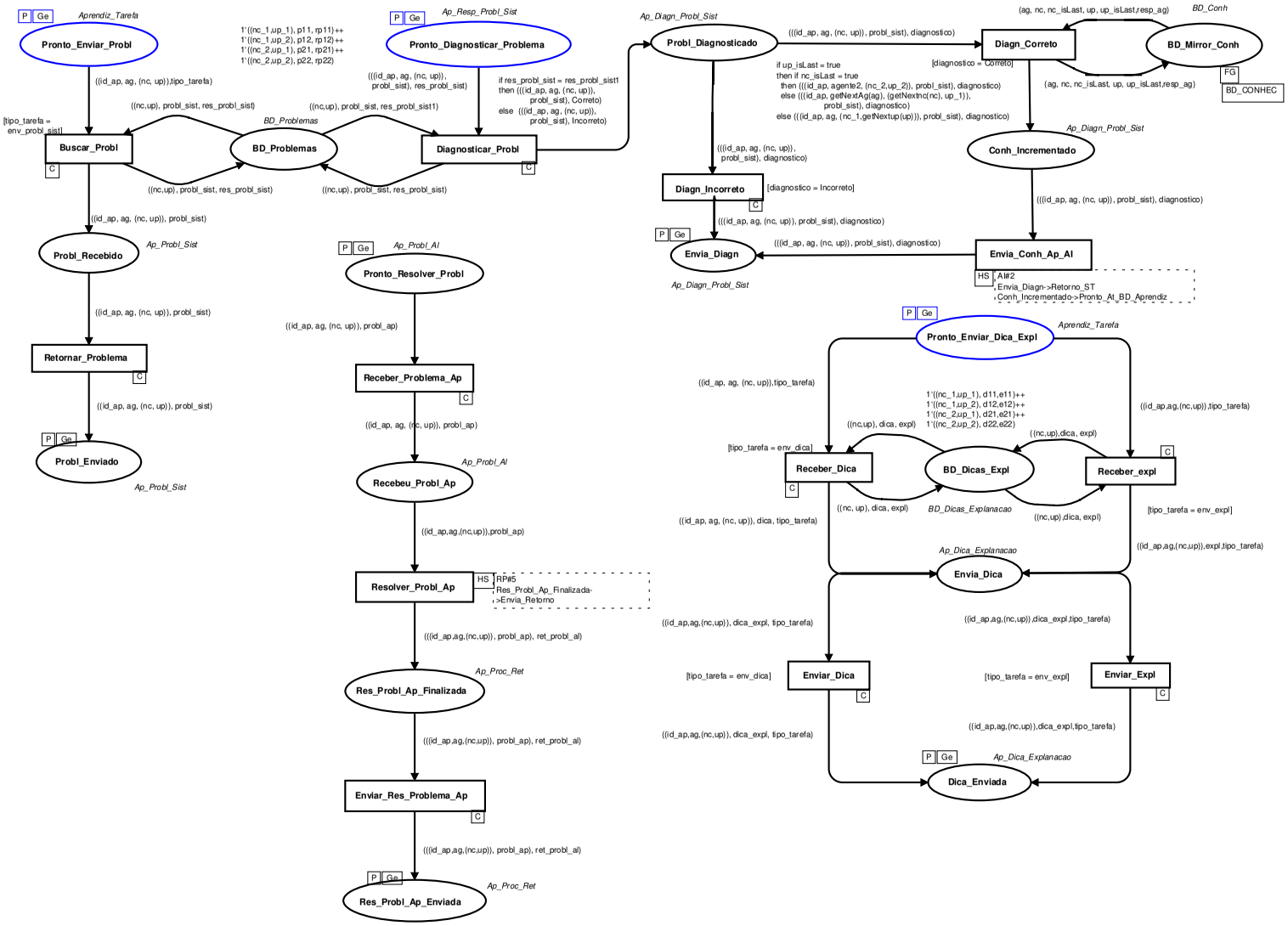


Figura 4.5: Modelo do Sistema Tutor

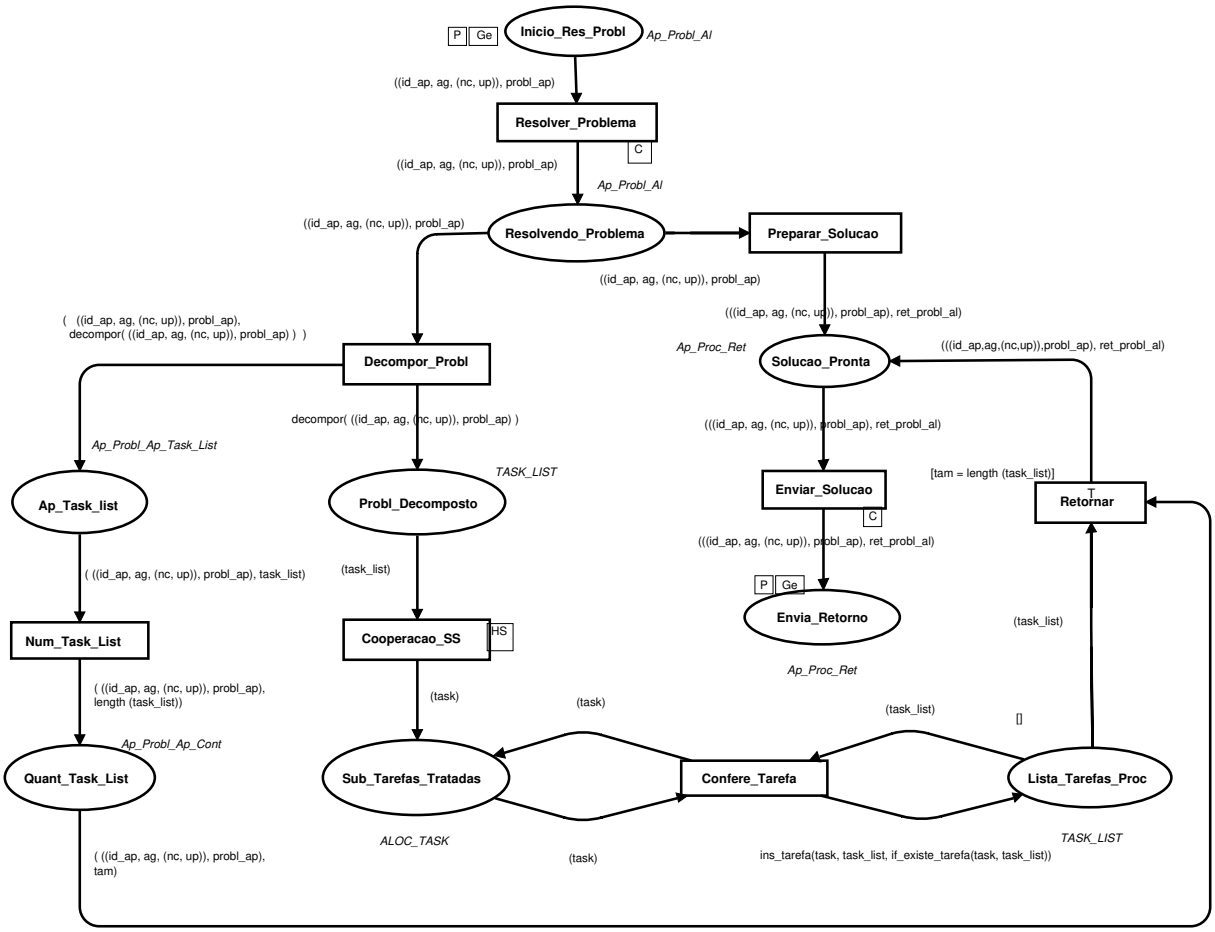


Figura 4.6: Modelo do Resolvedor de Problemas

Resolvedor de Problemas (RP) modela as seguintes carcaterísticas do Resolvedor de Problemas:

1. Resolução de Problemas.

2. Decomposição dos problemas.

3. Caso o Resolvedor de Problemas não seja capaz de resolver o problema, ele envia os subproblemas ao Sistema Social, que encarrega-se de alocar os agentes que potencialmente podem resolver o problema, conforme descrito na seção 3.3.

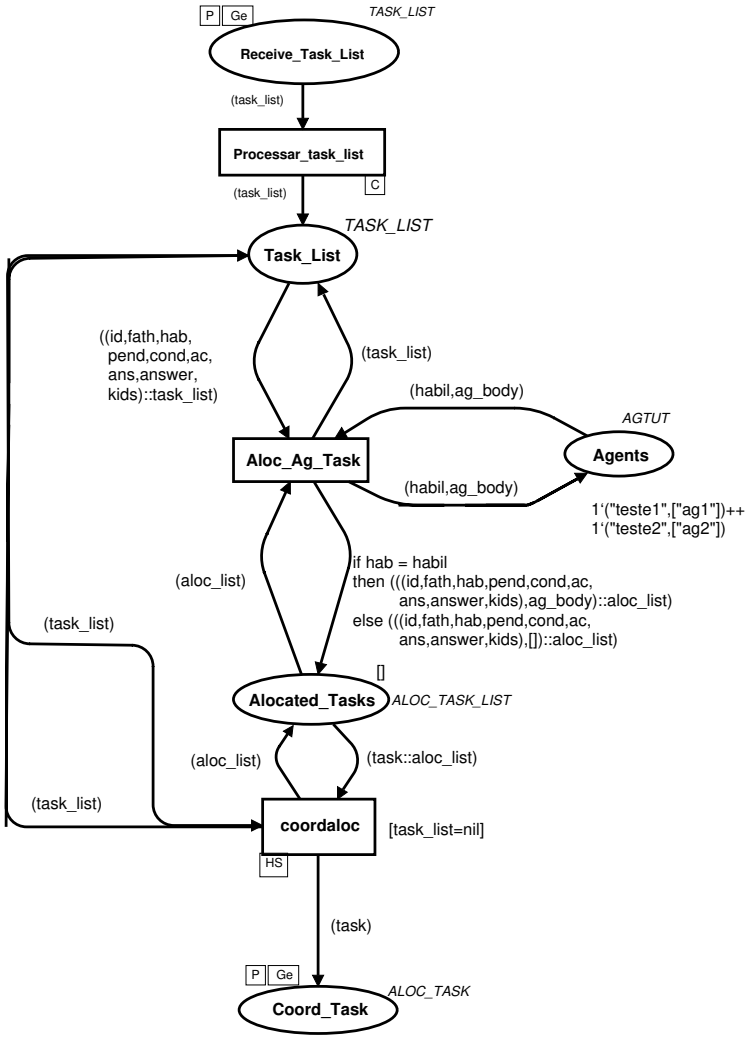


Figura 4.7: Modelo do Sistema Social - Alocação

4.2.5 Sistema Social

Alocação (Alocation)

Esta rede, que pode ser visualizada na Figura 4.7, faz parte do modelo do Sistema Social, e modela a alocação de agentes que podem vir a resolver determinada tarefa.

Coordenação (Coordenation)

Este modelo, que pode ser visualizado na Figura 4.8, tem a tarefa de realizar a coordenação de execução das tarefas pelos Agentes.

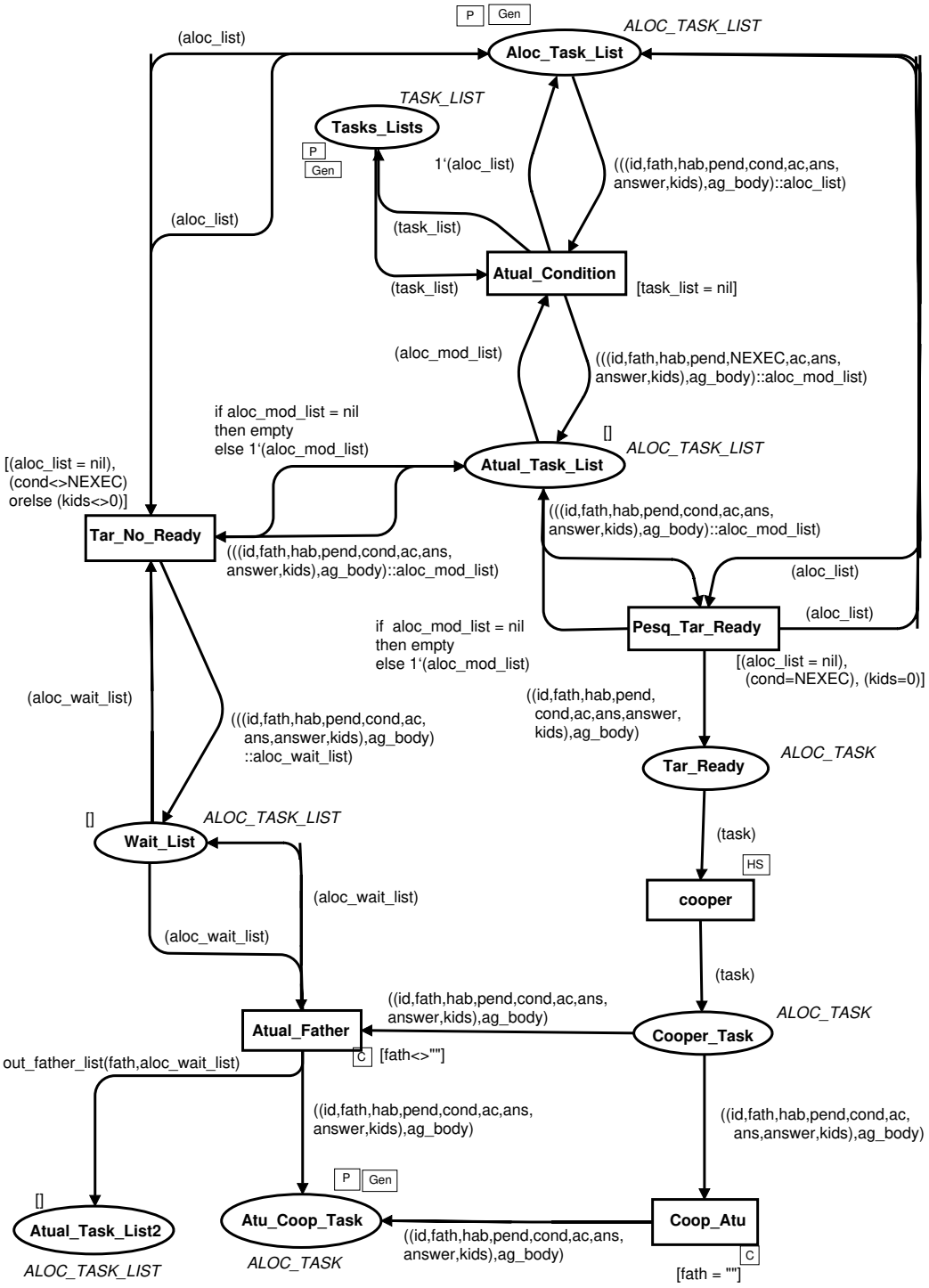


Figura 4.8: Modelo do Sistema Social - Coordenação

Cooperação (Cooperation)

Este modelo, que pode ser visualizado na Figura 4.9, simula a característica de cooperação entre os agentes para a realização de determinada tarefa.

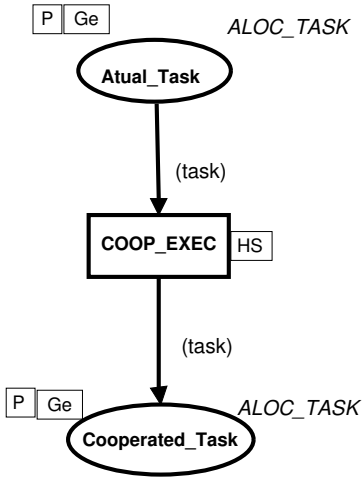


Figura 4.9: Modelo do Sistema Social - Cooperação

Cooperação - execução e encaminhamento (Coop_Exec e Coop_Enc)

Estes modelos, que podem ser visualizado nas Figuras 4.10 e 4.11, respectivamente, modelam o processamento da tarefa pelos agentes.

4.3 Análise do modelo

Para a análise dos modelos foram realizados os seguintes passos:

- Definição dos cenários
- Simulação
- Geração dos diagramas de sequência
- Construção do espaço de estados

Inicialmente definimos os cenários que poderiam ocorrer durante uma sessão de aprendizagem no ambiente. Os cenários foram construídos de modo que pudessem

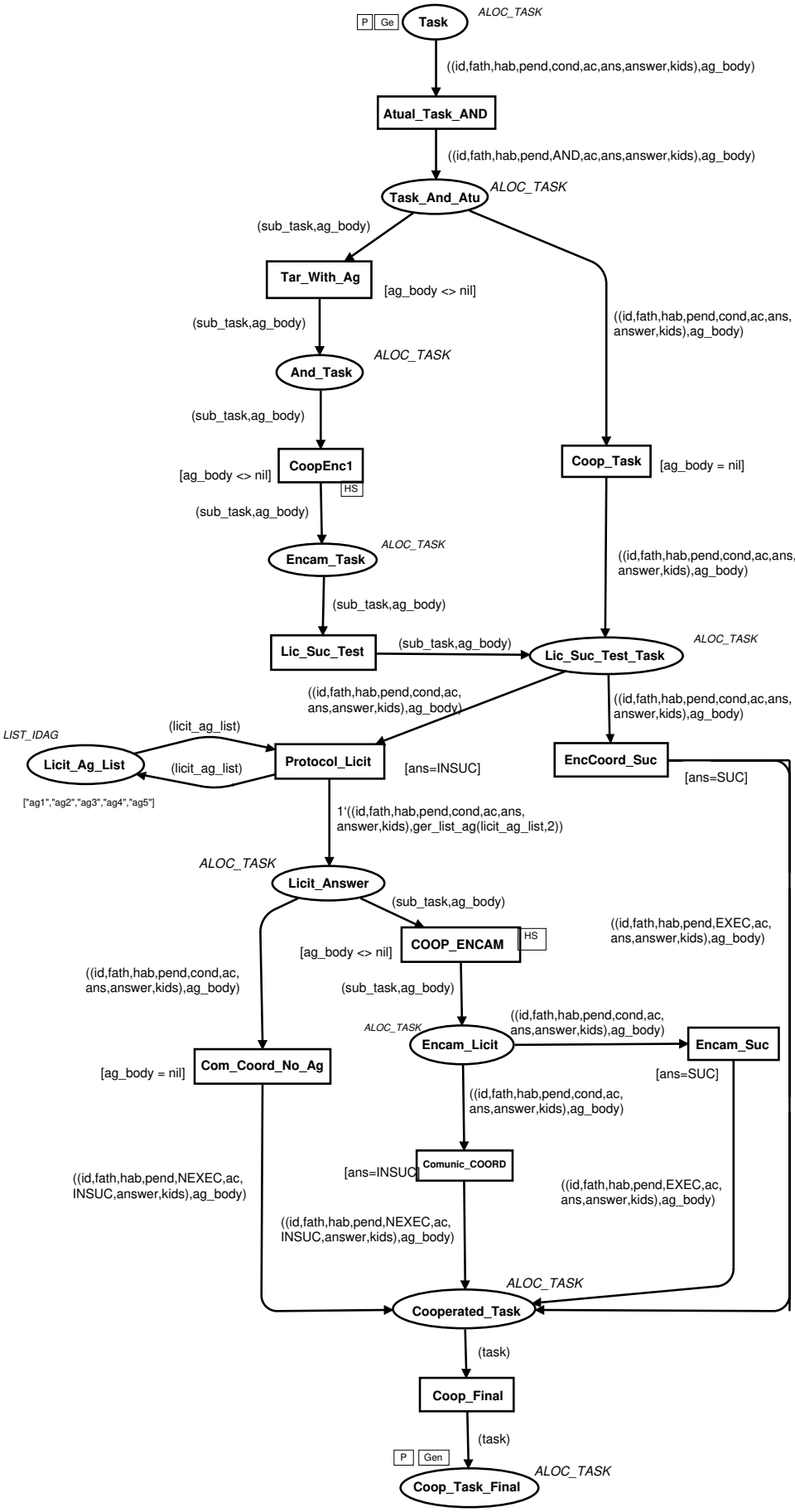


Figura 4.10: Modelo do Sistema Social - Execução da Cooperação

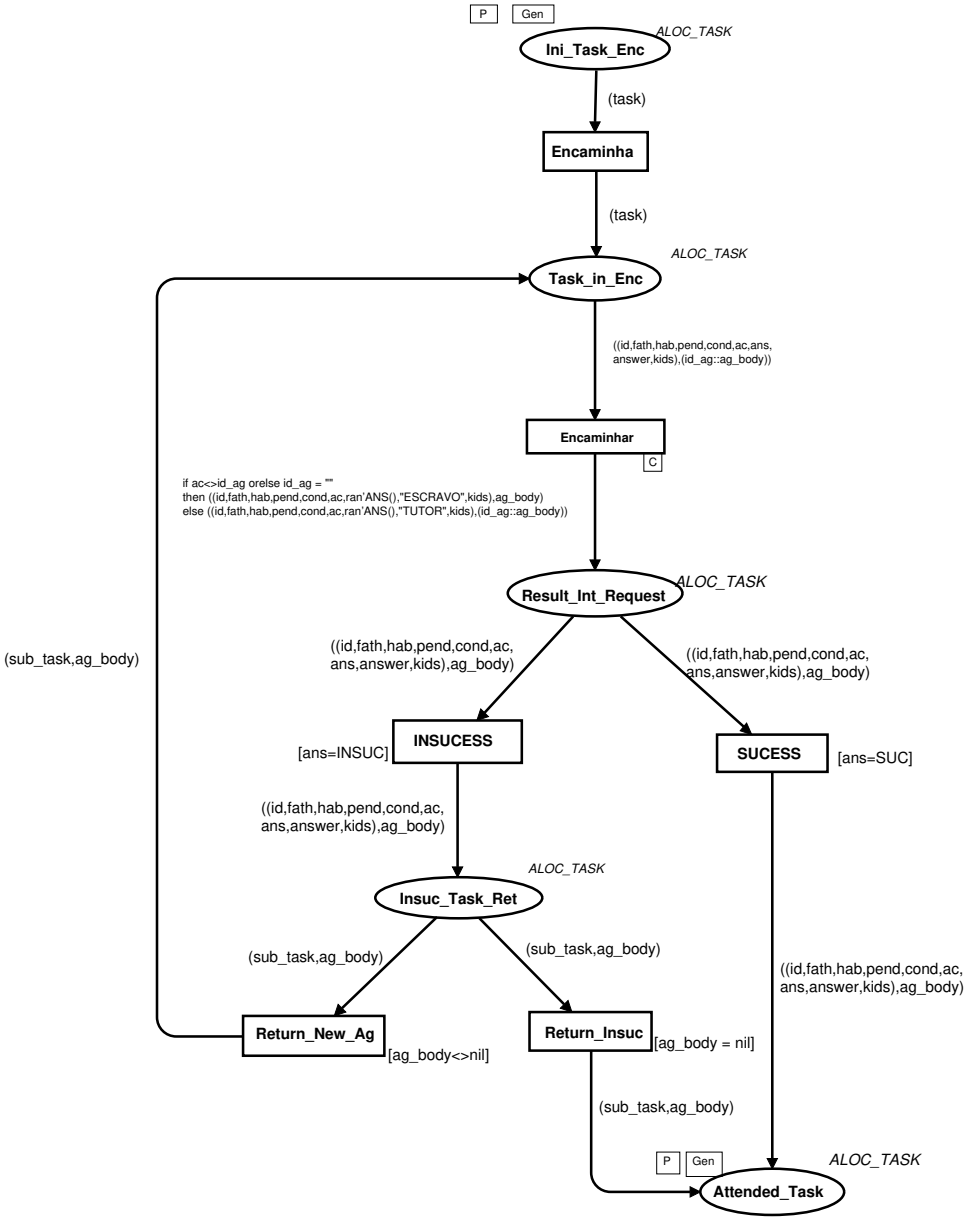


Figura 4.11: Modelo do Sistema Social - Encaminhamento da Tarefa

abranger as interações mais importantes entre o *Aprendiz*, *Agente de Interface*, *Sistema Tutor*, *Resolvedor de Problemas* e o *Sistema Social*. Para isso, agrupamos algumas interações em um só cenário, para facilitar o acompanhamento. É interessante observar que a ordem definida pelo cenário não influi o resultado final. Os cenários são descritos na seção 4.3.2.

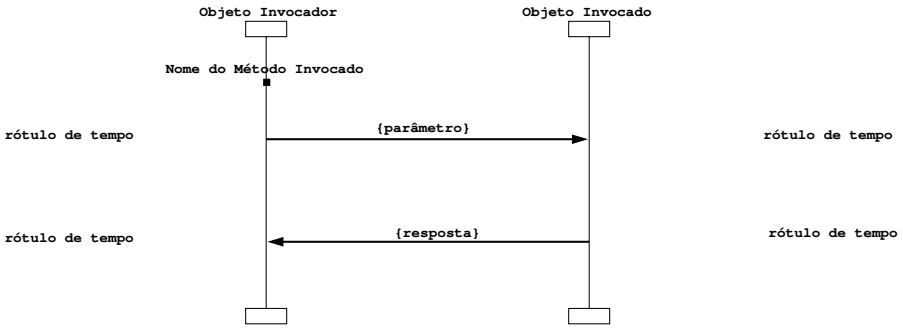


Figura 4.12: Notação do Diagrama de Sequência de Mensagens entre Objetos

A seguir, realizamos a simulacão das redes CPN utizando a ferramenta Design/CPN, já descrita na seção 4.1. Através da simulação foram gerados os diagramas de sequência, cuja notação pode ser vista na figura 4.12. Juntamente com os cenários foi construído o espaço de estados (grafo de ocorrência), para garantir que os resultados alcançados com a sequência de execução particular a um cenário, serão válidos para qualquer sequência de execução.

Para a geração dos cenários nos modelos CPN utilizou-se também uma biblioteca para a geração de diagramas de seqüência de mensagens. Por estar em fase experimental de desenvolvimento esta ferramenta ainda não está integrada ao Design/CPN.

A seguir temos a descrição da simulação dos modelos, e a descrição dos cenários.

4.3.1 Simulação dos Modelos

Inicialmente uma ficha é depositada no lugar *Lista_ Tarefas*, modelando que o sistema deverá responder a uma série de requisições do aprendiz. Esta ficha contém as informações referentes ao identificador do aprendiz, e a lista de tarefas que o aprendiz deseja realizar. A primeira tarefa que o sistema deve realizar é a *inicialização do sistema*, que consiste da verificação do nível de conhecimento do aprendiz. Por isso, a única transição habilitada a partir do depósito de uma ficha no lugar *Lista_ Tarefas* é

a transição *Iniciar_Sessao*. Ao ser disparada, é depositada uma ficha no lugar *Pronto_Buscar_ch*, o que causará a habilitação da transição *Buscar_ch_Ap*. Após esta transição ser disparada, é depositada uma ficha no lugar *Pronto_Buscar*, da página *AI*, referente ao Agente de Interface.

Ao ser depositada uma ficha no lugar *Pronto_Buscar*, a transição *Avaliar_Ap* é habilitada. Esta transição possui o lugar de entrada *BD_Aprendiz*, que armazena a informação do nível de conhecimento dos aprendizes que já utilizaram o sistema. Com isto, o disparo da transição *Avaliar_Ap* simboliza a verificação do sistema referente à utilizações anteriores deste aprendiz no sistema. Após o disparo desta transição, uma ficha é depositada no lugar *Avaliando_Aprendiz*. Neste momento pode ocorrer 2 situações: o aprendiz já utilizou o sistema anteriormente, ou, esta é a primeira utilização do sistema pelo aprendiz.

- Caso o aprendiz nunca tenha utilizado o sistema anteriormente, a transição *Buscar_Teste* é habilitada. Esta transição também possui como lugar de entrada o lugar *BD_Teste*. Este lugar armnazena o teste que o aprendiz deve realizar para a verificação do seu nível de conhecimento sobre o domínio, no contexto deste trabalho, o domínio de redes de Petri. Ao ser disparada a transição *Buscar_Teste*, é depositada uma ficha no lugar *Retorna_Res*, contendo o teste que o aprendiz deve realizar para a verificação do seu nível de conhecimento.

- Cao o aprendiz já tenha utilizado anteriormente o sistema, a única transição habilitada após o depósito de uma ficha no lugar *Avaliando_Aprendiz* é a transição *Retornar_NC_Ap*. Após o disparo desta transição, é depositada uma ficha no lugar *Retorna_Res* com a informação do nível de conhecimento do aprendiz e qual agente será responsável pelo aprendiz daqui para a frente. Este agente é denominado como *agente supervisor*, conforme definido no capítulo 3 deste trabalho.

Ao ser depositado uma ficha no lugar *Retorna_Res*, também é depositado uma ficha no lugar *Aprendiz_ch*, da página *Aprendiz*[2]. A partir daqui, podem ocorrer 2

[2]Explicações referentes ao funcionamento das redes de Petri coloridas estão descritas no Anexo A deste trabalho.

situações: o aprendiz já utilizou o sistema anteriormente e, portanto, já possui um nível de conhecimento definido e um agente supervisor, ou, o aprendiz nunca utilizou o sistem e, desta forma, deve resolver um teste para a verificação do seu nível de conhecimento.

- Caso o aprendiz deva realizar o teste para a verificação do seu nível de conhecimento a transição *Resolver_ Teste* é disparada e uma ficha é depositada no lugar *Resolvendo_ Teste*, simbolizando a resolução do teste pelo aprendiz. A seguir, a transição *Enviar_ Resposta* pode ser disparada, e após o seu disparo é depositada uma ficha no lugar *Recebe_ Teste* da página *AI*. Neste momento, a transição *Aprendiz_ Avaliado* pode ser disparada. Esta transição possui 4 lugares de entrada: *Recebe_ Teste*, *BD_ Teste*, *BD_ Aprendiz* e *BD_ Conhecimento*. O lugar *BD_ Conhecimento* armazena a informação sobre qual agente é responsável por determinado nível de conhecimento. Conforme dito anteriormente, o lugar *BD_ Teste* armazena a informação da resposta correta do teste. Desta forma, o disparo da transição *Avaliar_ Teste* simboliza a atividade de diagnóstico da resposta do teste enviado pelo aprendiz. Após o seu disparo, é depositado uma ficha no lugar *Aprendiz_ Avaliado*, bem como, é depositado uma ficha no lugar *BD_ Aprendiz* com a informação atualizada do nível de conhecimento atual do aprendiz. A seguir, a transição *Retornar_ Avaliação* é habilitada e, após o seu disparo, é depositada uma ficha no lugar *Teste_ Retornado*, o que causa também, o depósito de uma ficha no lugar *Teste_ Respondido* da página *Aprendiz*. A partir daqui, a única transição habilitada é a transição *Iniciar_ Tarefas_ Após_ Teste*, e, após o seu disparo, é depositada uma ficha no lugar *Lista_ Tarefas*, sendo que a tarefa *Inicialização do sistema* é retirada da lista de tarefas a serem realizadas pelo sistema.

- Caso o aprendiz já tenha utilizado o sistema anteriormente e, por isso, já tenha o agente supervisor alocado, não é necessário a resolução do teste, consequentemente a transição *Iniciar_ Tarefas* é habilitada. Após o seu disparo é depositada uma ficha no lugar *Lista_ Tarefas*, com a tarefa *Inicialização do sistema* sendo retirada da lista de tarefas a serem realziadas pelo sistema.

A partir deste momento, o sistema já tem a informação sobre qual o nível de

conhecimento do aprendiz, e, o identificador do seu agente supervisor. A única transição habilitada é a *Iniciar_ Aprendizagem*, e, após o seu disparo, temos o depósito de uma ficha no lugar *Executar_ Tarefa*, que contém a informação da próxima tarefa a ser executada pelo aprendiz. aprendiz pode requisitar que o sistema resolva as seguintes tarefas: receber uma dica/explanação sobre determinado assunto relacionado a um nível de conhecimento, resolver um problema proposto pelo sistema, ou, requisitar que o sistema resolva um problema para o aprendiz.

1. Caso o aprendiz deseje receber uma dica/explanacao sobre determinado assunto relacionado a um nível de conhecimento, a transição *Buscar_ dica_ expl* estará habilitada e, após o seu disparo, uma ficha é depositada no lugar *Pronto_ Enviar_ Dica_ Expl* da página *ST*, referente ao Sistema Tutor. Caso o aprendiz deseje receber uma *dica*, a transição *Receber_ dica* estrá habilitada. Esta transição também tem como lugar de entrada o lugar *BD_ Dicas_ Expl*, que armazena as informações referentes às dicas/explanações para cada nível de conhecimento. Após ser disparada, é depositada uma ficha no lugar *Envia_ Dica*, e, em seguida, a transição *Enviar_ Dica* pode ser disparada, causando o depósito de uma ficha no lugar *Dica_ Enviada*, que contém a dica a ser apresentada. O depósito da ficha em *Dica_ Enviada* causa também o depósito da ficha no lugar *Tarefa_ Realizada* da página *Aprendiz*.

   Caso o aprendiz deseje receber uma *explanação*, o funcionamento é análogo ao descrito para a *dica*, sendo que as transições disparadas são: *Receber_ Expl* e *Enviar_ Expl*, ao contrário de *Receber_ Dica* e *Enviar_ Dica*.

2. Caso o aprendiz deseje receber um problema, a transição *Ap_ Resolver_ Problema* estará habilitada, e, após o seu disparo, uma ficha é depositada no lugar *Pronto_ Enviar_ Probl* da página *ST*. Esta transição também tem como lugar de entrada o lugar *BD_ Problemas*, que armazena os problemas relacionados a cada nível de conhecimento, e suas respectivas respostas. Esta transição simboliza a a busca do problema referente ao nível de conhecimento do aprendiz. Após o seu disparo é depositado uma ficha no lugar *Probl_ Recebido*, contendo o problema que será enviado para o aprendiz. A seguir a transição *Retornar_ Problema* estará habili-

tada e, após o seu disparo, é depositado uma ficha no lugar *Probl_ Enviado*, o que causa também, o depósito de uma ficha no lugar *Resolvendo_ Problema* da página *aprendiz*, simbolizando que o aprendiz está resolvendo o problema proposto pelo sistema.

Daí, a transição *Preparar_ resp_ probl_ sist* ficará habilitada e, após o seu disparo, é depositado uma ficha no lugar *Preparando_ resp_ probl_ sist*, tornando a transição *Enviar_ resposta_ probl* habilitada. Após o disparo desta transição, é depositado uma ficha no lugar *Pronto_ Diagnosticar_ Problema* da página *st*, simbolizando que o aprendiz enviou a resposta do problema ao sistema. A seguir, a transição *Diagnosticar_ Probl* fica habilitada. Esta transição também tem como lugar de entrada o lugar *BD_ Problemas*, que, conforme descrito anteriormente, possui os problemas, e suas respectivas soluções para cada nível de conhecimento. Esta transição simboliza a correção do problema do aprendiz pelo sistema, e após o seu disparo uma ficha é depositada contendo o resultado da correção do problema, que pode ocasionar 2 situações: o problema está correto ou o problema está incorreto.

- Caso o problema esteja incorreto, a transição *Diagn_ Incorreto* ficará habilitada e, após o seu disparo, uma ficha é depositada no lugar *Envia_ Diagn*, o que causará também o depósito de uma ficha no lugar *Avaliando_ diagnostico_ probl* da página *aprendiz*.
- Caso o problema esteja correto, a transição *Diagn_ Correto* estará habilitada. Esta transição também possui como lugar de entrada o lugar *BD_ Mirror_ Conh* que armazena as informações sobre os possíveis níveis de conhecimento, relacionados aos seus respectivos agentes supervisores. Desta forma, esta transição simboliza o incremento do nível de conhecimento do aprendiz, e escolha do novo agente supervisor do aprendiz, visto que o seu nível de conhecimento foi incrementado. Após o disparo desta transição, uma ficha é depositada no lugar *Conh_ Incrementado*, já com o nível de conhecimento do aprendiz incrementado, e a transição *Envia_ Conh_ Ap_ AI* torna-se habilitada. Após o disparo desta transição uma ficha é deposi-

tada no lugar *Pronto_At_BD_Aprendiz* da página *Aprendiz*, habilitando a transição *Atualizar*. Esta transição simboliza a atualização do nível de conhecimento do aprendiz no lugar *BD_Aprendiz*, que, conforme descrito anteriormente, armazena a informação do nível de conhecimento do aprendiz. Ao ser disparada, uma ficha é depositada no lugar *BD_Aprendiz*, já com o novo valor referente ao nível de conhecimento atualizado, e uma ficha é depositada no lugar *BD_Aprendiz_Atualizado*, habilitando a transição *Retornar_ST*. Após o disparo desta transição, uma ficha é depositada no lugar *Retorno_ST*, o que causa, também, o depósito de uma ficha no lugar *Envia_Diagn* da página *ST*. Como dito anteriormente, o depósito de uma no lugar *Envia_Diagn* causa també o depósito de uma ficha no lugar *Avaliando_diagnostico_probl* da página *aprendiz*.

A partir daqui a única transição habilitada é a transição *Retornar*, e, após o seu disparo, é depositada uma ficha no lugar *Tarefa_Realizada*.

3. Caso o aprendiz deseje enviar um problema para ser resolvido pelo sistema, a transiçÃo *Sist_Resolver_Problema_Ap* é a única transição habilitada. Após o seu disparo, uma ficha é depositada no lugar *Enviando_ProblemaAl*, tornando a transição *Enviar_ProblemaAl* habilitada. Após esta transição ser disparada, uma ficha é depositada no lugar *Pronto_Resolver_Probl* da página *ST*. Neste momento a única transição habilitada é a transição *Resolver_Problema_Ap*. Após o seu disparo é depositada uma ficha no lugar *Recebeu_Probl_Ap* habilitando a transição *Resolver_Probl_Ap*. Após o disparo desta transição, é depositada uma ficha no lugar *Inicio_Res_Probl* da página *RP*, referente ao módulo *Resolvedor de Problemas* do *Sistema Tutor*. Neste momento a única transição habilitada é a transição *Resolver_Problema*, e, após o seu disparo, uma ficha é depositada no lugar *Resolvendo_Problema*, simbolizando que o *Resolvedor de Problemas* está resolvendo o problema proposto pelo aprendiz. A partir daqui, pode ocorrer 2 situações: o *Resolvedor de Problemas* pode resolver o problema, ou nãoser capaz de resolver o problema.

   - Caso o *Resolvedor de Problemas* seja capaz de resolver o problema, a tran-

sição *Prepara_Solução* é disparada e uma ficha é depositada no lugar *Solucao_Pronta*, contendo a resolução do problema. Daí, a transição *Enviar_Soluçao* torna-se habilitada e, após o seu disparo, uma ficha é depositada no lugar *Envia_Retorno*, o que causa, também, o depósito de uma ficha no lugar *Res_Probl_Ap_Finalizada* na página *ST*.

- Caso o *Resolvedor de Problemas* não seja capaz de resolver o problema, a transição *Decompor_Probl* é disparada, simbolizando a decomposição de tarefas, descrita no capítulo 3. Após o seu disparo é depositado uma ficha no lugar *Ap_Task_list*, contendo a lista de subtarefas da tarefa decomposta, habilitando a transição *Num_Task_List*. Após o disparo desta transição uma ficha é depositada no lugar *Quant_Task_List*, contendo a informação da quantidade de subtarefas. Após o disparo da transição *Decompor_Probl*, uma ficha também é depositada no lugar *Probl_Decomposto*, também contendo a lista das subtarefas. Daí a transição *Cooperaçao_SS* pode ser disparada, simbolizando a requisição do módulo *Sistema Social*, que se encarregará de alocar os agentes responsáveis por resolver as subtarefas[3]. A medida que cada subtarefa é processada pelos agentes responsáveis, as fichas contendo estas subtarefas são depositadas no lugar *Sub_Tarefas*. Com isto, a transição *Confere_Tarefa* é habilitada e, após o seu disparo, uma ficha é depositada no lugar *Lista_tarefas_Proc*. A função *ins_tarefa*, escrita na linguagem ML, encarrega-se de só depositar as subtarefas que já não foram inseridas no lugar *Lista_Tarefas_Proc*. Somente após todas as subtarefas terem sido processadas é que a transição *Retornar* torna-se habilitada, e, após o seu disparo, uma ficha é depositada no lugar *Solucao_Pronta*. A seguir, a transição *Enviar_Solução* torna-se habilitada e, após o seu disparo, uma ficha é depositada no lugar *Envia_Retorno*, causando também o depósito de uma ficha no lugar *Res_Probl_Ap_Finalizada* da página *ST*.

  Ao ser depositado uma ficha no lugar *Res_Probl_Ap_Finalizada*, a transição *Enviar_Res_Problema_Ap* é habilitada, e, após o seu disparo, é depo-

[3] O funcionamento das redes referentes ao Sistema Social pode ser acompanhado pelas figuras presentes no anexo A

sitada uma ficha no lugar *Res_ Probl_ Ap_ Enviada*, o que causa também, o depósito de uma ficha no lugar *Processando_ resp_ problemaAl* da página *aprendiz*. A seguir, a transição *Retornar_ apos_ rec_ problAl* é habilitada, e, após o seu disparo, é depositada uma ficha no lugar *Tarefa_ Realizada*.

Ao ser depositada uma ficha no lugar *Lista_ Tarefas* a transição *Finalizar_ Tarefa* pode ser disparada, simbolizando a realização de uma tarefa, e, após o seu disparo, uma ficha é depositada no lugar *Lista_ Tarefas*, sendo que a tarefa que foi realizada é retirada da lista das tarefas a serem realizadas pelo sistema.

4.3.2 Cenários

A seguir, apresenta-se os diagramas de sequência de mensagens gerados pela ferramenta Design/CPN, para cada um dos cenários.

Cenário: Inicializar o sistema, Receber uma Explanação, Resolver um Problema)

Neste cenário, veja diagrama de sequência de mensagens na Figura 4.13, podemos observar a inicialização do sistema, o envio de uma explanação e a resolução de um problema pelo *Aprendiz*. Para isto, consideramos a seguinte situação: O *aprendiz* inicia a sessão de aprendizagem com o depósito de uma ficha contendo o seu identificador (id_1)[4], foram introduzidos alguns lugares e transições para possibilitar uma parada na execução, e, consequentemente, a análise do modelo. No entanto, podemos considerar aqui, como única condição de inicialização do sistema a informação do id_ap, que serve como um identificador do *aprendiz*. O *Agente de Interface (AI)* percebe que um novo *aprendiz* está disposto a realizar uma sessão, e verifica se este *aprendiz* já utilizou o sistema alguma vez. Neste caso, o *aprendiz* nunca utilizou o sistema, por isso, o *AI* encarrega-se de enviar um teste para verificar o *nível de conhecimento* do *aprendiz*. Além do *identificador*, a ficha contém os valores não inicializados do *agente supervisor*, que será responsável inicialmente pela sessão de ensino ao *aprendiz*, e os

[4]Conforme explicado na seção 4.1

valores referentes ao *conhecimento* atual do aprendiz. Conforme mostrado na seção 3.1, de um ponto de vista interno, o conhecimento é composto por *unidades pedagógicas*.

Ao receber o teste, o *aprendiz* resolve e envia a resposta para o *AI*. Com base na resposta do teste pelo *aprendiz*, o *AI* determina o seu *conhecimento* sobre o domínio em questão, e qual agente será responsável pela condução da sessão de aprendizagem, denominado *Agente Supervisor*. Isto é representado pela ficha $(id\_1, agente1, nc\_1, up\_2)$, que informa o *identificador*, o *agente supervisor*, o *nível de conhecimento* e a *unidade pedagógica* do *aprendiz*, respectivamente.

A seguir, o *aprendiz* manifesta o desejo de receber uma *explanação* sobre um assunto diretamente relacionado ao seu *conhecimento* atual. O *agente supervisor* verifica este pedido e envia a *explanação*, simbolizada pelo valor $e12$, na ficha $(id\_1, agente1, nc\_1, up\_2, e12)$.

Depois, o *aprendiz* decide resolver um *problema* relacionado ao seu *conhecimento* atual. O *agente supervisor* percebe este desejo do *aprendiz* através do depósito da ficha contendo o *identificador* do aluno, o seu prórpio *identificador* (*agente supervisor*), e o *conhecimento* do *aprendiz*. De posse destas informações, o *agente supervisor* envia o *problema* ao *aprendiz*, representado pelo valor $p12$ presente na ficha $(id\_1, agente1, nc\_1, up\_2, p12)$.

O *Aprendiz* resolve o problema e retorna uma resposta. O *agente supervisor* realiza o diagnóstico da resposta do aluno. Neste caso, o *aprendiz* não obteve êxito na resolução deste problema, o que é simbolizado pelo valor $Incorreto$, presente na ficha $(id\_1, agente1, nc\_1, up\_2, p12, Incorreto)$.

Cenário: Inicializar o sistema, Receber uma dica, Resolver um Problema e Incremenetar o conhecimento do *aprendiz*

Neste cenário, veja diagrama de sequência de mensagens na Figura 4.14, podemos observar a inicialização do sistema, o envio de uma dica, a resolução de um problema pelo *aprendiz*, e o incremento em seu conhecimento sobre o domínio de redes de Petri. A sua execução é análoga ao funcionamento do cenário4.3.2, com 2 diferenças. A primeira é que neste cenário o *aprendiz* manifesta o desejo de receber uma dica, e não uma explanação como no exemplo anterior; no entanto, o comportamento do envio da

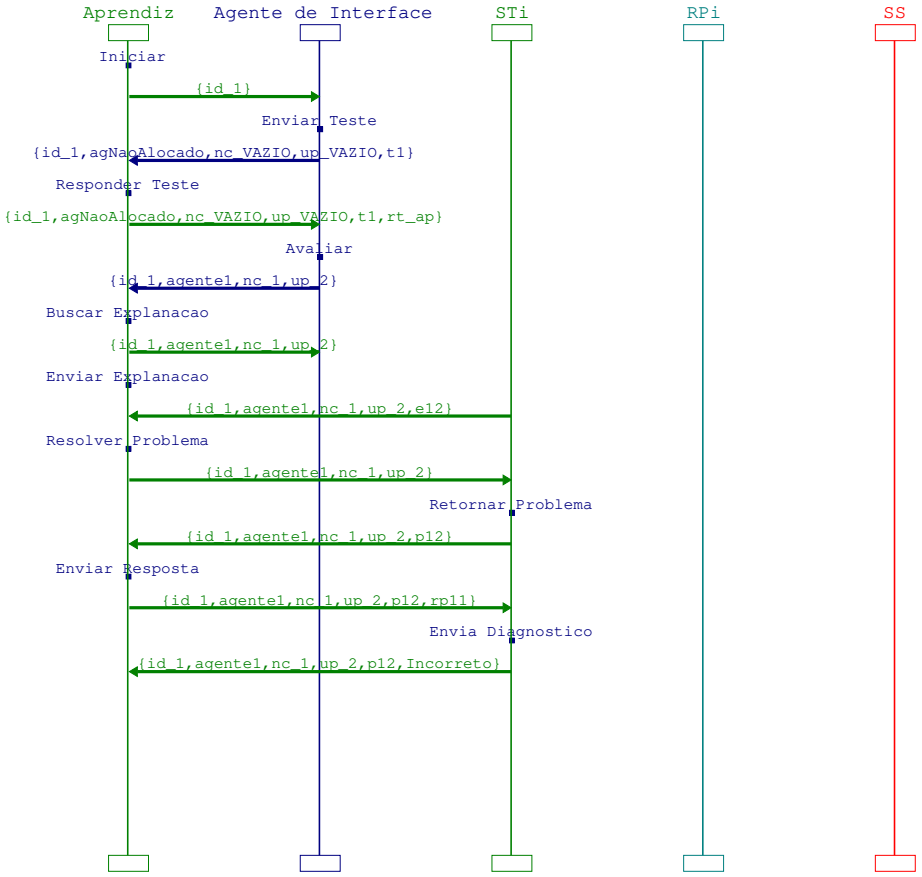


Figura 4.13: Inicialização e Resolução de Problemas

explanação é análogo ao envio da dica. A outra diferença é que neste cenário o *aprendiz* logra êxito na resolução do problema, incrementando o seu nível de conhecimento, fazendo com que o *agente supervisor* informe ao *agente de interface* o ocorrido, para que este atualize o lugar $BD\_Aprendiz$. Este lugar, como dito na seção 3.3, armazena a informação do conhecimento atual do *aprendiz*, para que da próxima vez que der início à sessão de aprendizagem, o *aprendiz* recomece do ponto onde estava na última sessão, e, assim, não precise iniciar todo o processo novamente.

Cenário: Inicializar o sistema, Resolver um problema proposto pelo *Aprendiz*

Neste cenário, veja diagrama de sequência de mensagens na Figura 4.15, podemos observar a inicialização do sistema, e a tentativa de resolução de um problema proposto pelo

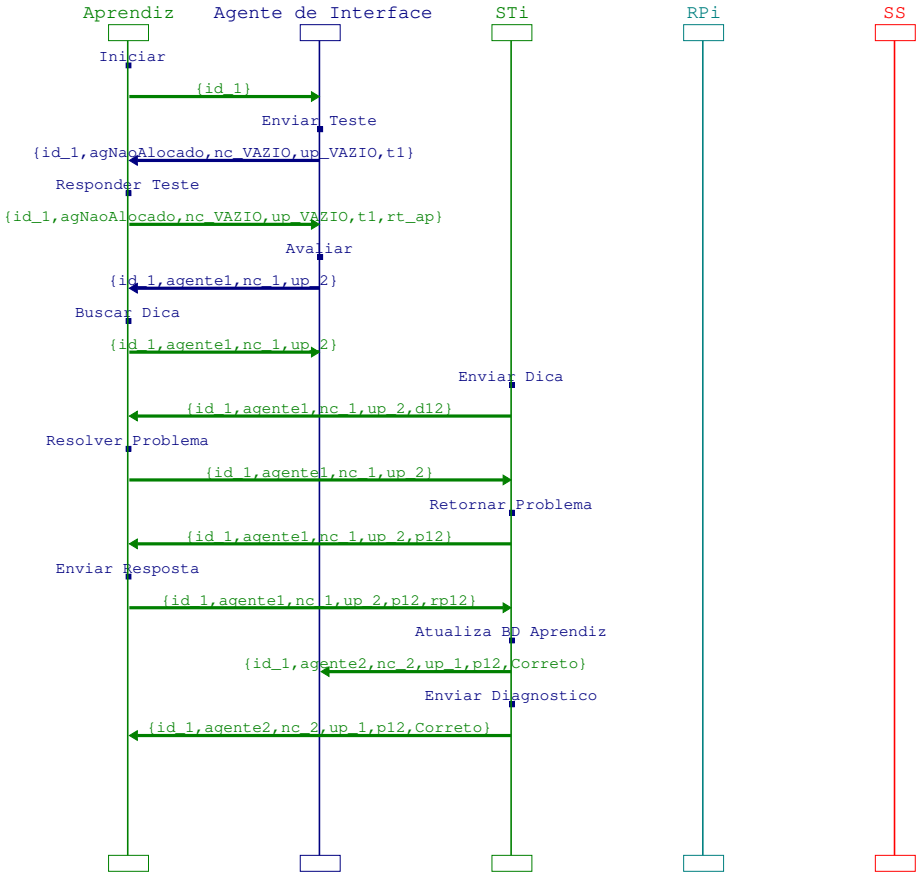


Figura 4.14: Inicialização e Resolução de Problemas

aprendiz. A inicialização do sistema ocorre da mesma forma descrita na seção 4.3.2, portanto não será descrito neste cenário. Aqui, temos a resolução de um problema proposto pelo *aprendiz*. O *agente supervisor* recebe esta requisição de solução de um problema, que é tratada inicialmente pelo *Sistema Tutor*. O *Sistema Tutor*, por sua vez, encaminha esta requisição para o *Resolvedor de Problemas*, que, conforme descrito na seção 3.3 é quem de fato *resolve* os problemas. Neste cenário, o *Resolvedor de Problemas* resolve apenas parcialmente o problema, conforme pode ser verificado pelo valor $Parcialmente$ presente na ficha $(id\_1, agente1, nc\_1, up\_2, p\_ap, Parcialmente)$.

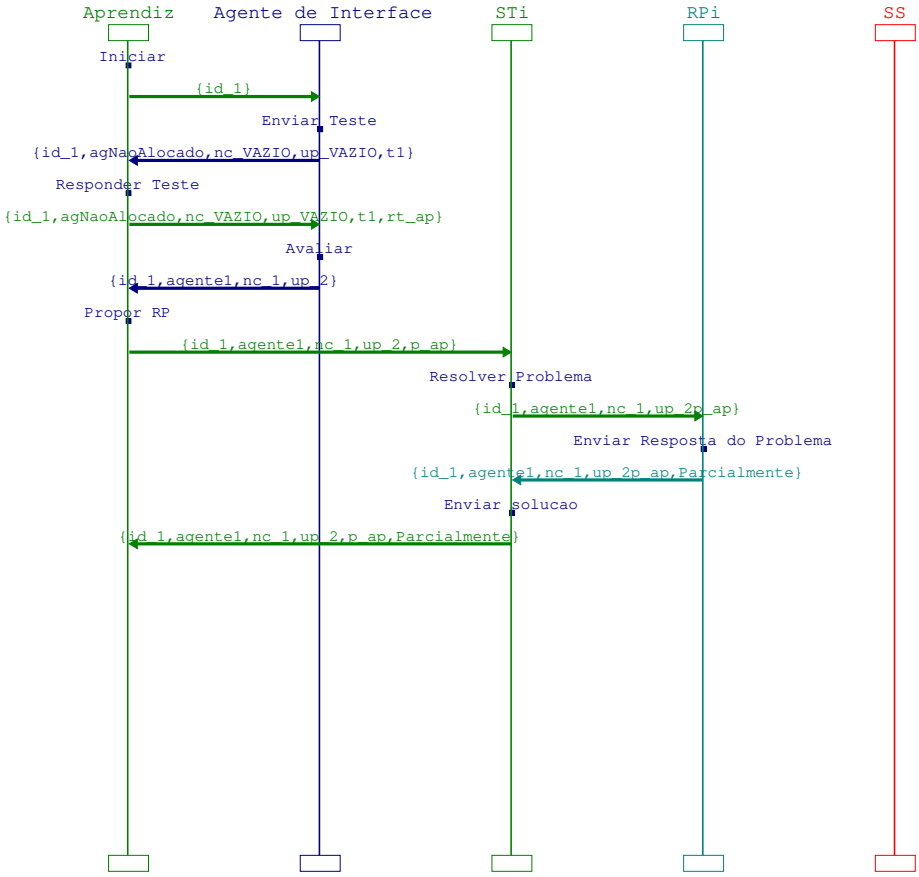


Figura 4.15: Inicialização e Resolução de Problemas do Aprendiz

Cenário: Inicializar o sistema, Recuperação da informação do aprendiz, Resolver um problema proposto pelo *Aprendiz*, Realizar Cooperação

Neste cenário, veja diagrama de sequência de mensagens na Figura 4.16, podemos observar a inicialização do sistema, a recuperação da informação do aprendiz, a resolução de um problema do aprendiz, e a utilização do *Sistema Social* para realizar a cooperação com os demais agentes da sociedade. As interações entre o *Sistema Tutor* e o *Sistema Social* já foram detalhadamente explicadas na seção 3.3, portanto não entraremos em detalhes novamente. A inicialização do sistema é realizada da mesma forma, com o depósito da ficha contendo o *identificador* do aprendiz. A diferença está que, desta vez, o *Agente de Interface* percebe que o *aprendiz* já utilizou o sistema anteriormente. Consequentemente, o *Agente de Interface* retorna o último estado do *aprendiz*, simbolizado pela ficha $(id\_2, agente1, nc\_1, up\_1, t1)$, o que indica que o *aprendiz* cujo

identificador é *id*_2, já utilizou o sistema e que o seu nível de conhecimento é *nc*_1, e estava na unidade pedagógica *up*_1. Neste caso, não é necessário que o *aprendiz* realize nenhum teste para a avaliação do seu nível de conhecimento. Conforme descrito no cenário 4.3.2, o *Resolvedor de Problemas* do *Agente Supervisor* encarrega-se de resolver o problema. No entanto, no cenário presente, o *Resolvedor de Tarefas* não é capaz de resolver sozinho a tarefa. Então, ele decompõe a tarefa e a envia para o *Sistema Social*. O *Sistema Social*, de posse das subtarefas, procura os agentes responsáveis por resolver as sub-tarefas, obedecendo o protocolo mestre-escravo, descrito na seção 3.4. Ao ter a solução das subtarefas (ou a notificação da impossibilidade de resolução pela SATA), o *Sistema Social* envia as subtarefas para o *Resolvedor de Problemas* para que este envie um retorno do problema. No caso deste cenário, o *Resolvedor de Problemas* conclui que é incapaz de resolver o problema.

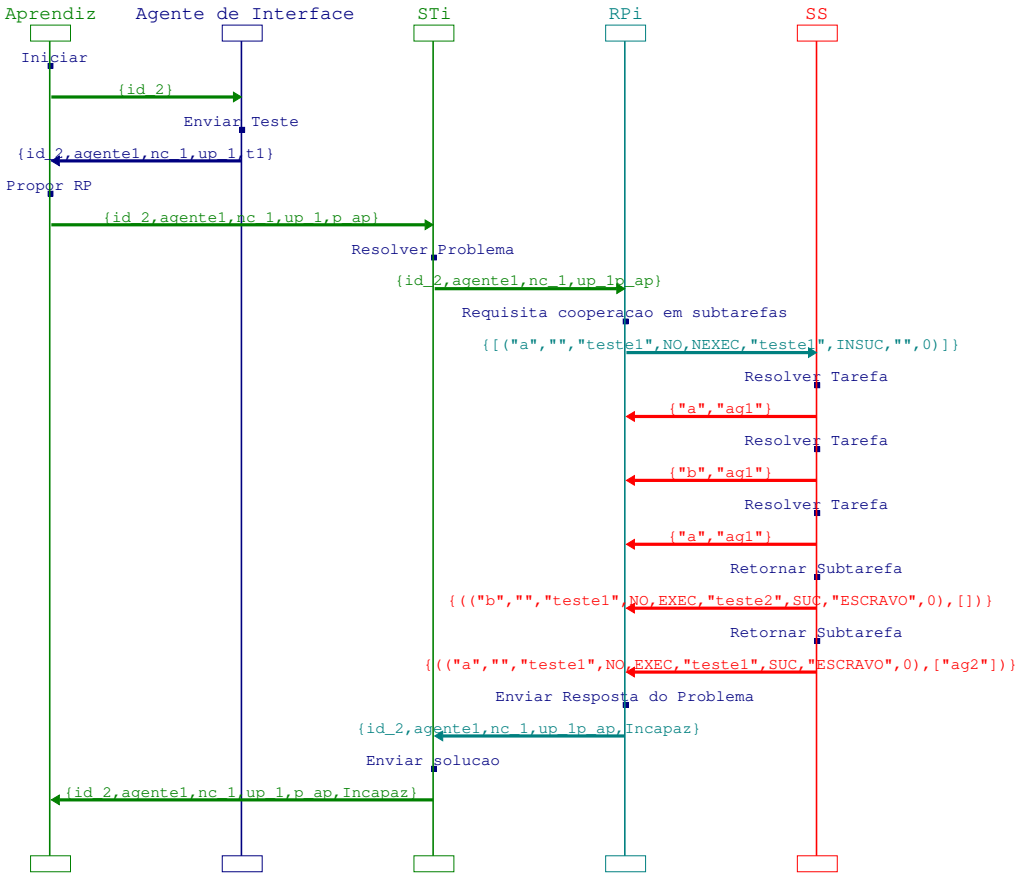


Figura 4.16: Inicialização e Resolução de Problemas do Aprendiz (4.3.2)

Para cada cenário descrito anteriormente, foi gerado o grafo de ocorrência, e, pu-

demos verificar que as marcações finais correspondiam às especificações do sistema.

Capítulo 5

Conclusão

Os objetivos deste trabalho foram a definição, modelagem e análise de um Sistema Tutor Multi-Agentes no domínio de redes de Petri.

Como metodologia, inicialmente foi realizada uma revisão bibliográfica sobre os assuntos tratados neste trabalho: Inteligência Artificial Distribuída, Sistemas Tutores Inteligentes, Sistemas Multi-Agentes e Informática na Educação. Além da pesquisa sobre estes temas, foi realizado um estudo mais aprofundado sobre redes de Petri e o ambiente MATHEMA.

A partir disto, buscamos aplicar o modelo do conhecimento descrito na Seção 3.1 para o domínio de redes de Petri, obtendo o modelo mostrado na Seção 3.1. A obtenção deste modelo não foi um processo direto, mas sim, um processo cíclico, onde a partir de uma definição do domínio, voltávamos e verificávamos se este modelo obedecia aos requisitos definidos no MATHEMA, e também, se expressava corretamente o domínio em questão, no caso redes de Petri.

Em seguida, buscamos construir os modelos de redes de Petri colorida do STMA-RP a partir da definição dos sistemas que compõem o MATHEMA, e, estudando as possíveis interações que poderiam ocorrer entre estes sistemas. Da mesma forma, este processo caracterizou-se como um processo cíclico, onde, durante a construção destes modelos, voltávamos para a definição dos sistemas e verificávamos se os modelos atendiam às definições do MATHEMA. Para isto, utilizamos a ferramenta Design/CPN para realizar a simulação dos modelos, conforme está descrito na Seção 4.1.

Por fim, realizamos a análise dos modelos de redes de Petri coloridas hierárquicas,

através da ferramenta Design/CPN, cujos resultados forma mostrados na Seção 4.2.

Procuramos construir os modelos de forma abrangente, de forma que as principais características do sistema fossem preservadas, como por exemplo: adaptabilidade do sistema ao aprendiz, facilidade de inserção de agentes no sistema, bem como, facilidade da inserção/atualização dos níveis de conhecimento pelos quais cada agente é responsável.

Como principais contribuições deste trabalho, temos:

1. Concluímos que a utilização do MATHEMA como arcabouço conceitual para a construção de sistemas tutores multi-agentes é válida, e, além disso, revela-se bastante promissora, principalmente pelo modelo de conhecimento e sua arquitetura.

2. Definição de um protótipo que servirá para a implementação de um Sistema Tutor Multi-Agentes em redes de Petri, auxiliando o ensino desta ferramenta de modelagem.

Como trabalhos futuros, propomos:

1. Incremento do modelo de conhecimento definido para redes de Petri, de forma que eles possam abranger outras extensões de redes de Petri.

2. Implementação de um sistema tutor em redes de Petri.

Bibliografia

[1] R. Baecker. *Groupware and Computer-Supported Cooperative Work*. Morgan Kaufmann Publishers, USA, 1993.

[2] A. H. Band and L. Gaesser. *Reading in Distributed Artificial Intelligence*. Morgan Kaufman, 1998.

[3] A. Barr and E. A. Feigenbaum. Applications-oriented ai research. In A. Barr and E. A. Feigenbaum, editors, *The Handbook of Artificial Intelligence*, volume 2, pages 283–290. Addison Wesley, USA, 1982.

[4] Tomaz C. Barros. Uma técnica de modelagem por redes de petri voltada à automação da manufatura. Dissertação de mestrado, Departamento de Eletrônica e Sistemas, Universidade Federal de Pernambuco, 1990.

[5] F. Belli and K.-E. Grosspietsch. Specification of fault-tolerant system issues by predicate/transition nets and regular expressions-approach and case study. volume 17, pages 513 – 526, June 1991.

[6] J. S. BROWN, R. R. BURTON, and A. G. BELL. Sophie: A step towards a reactive learning environment. In *International Journal of Man-Machine Studies*, pages 675–696, 1975.

[7] J. R. Carbonnel. Ai in cai: An artificial intelligence approach to computer assisted instruction. In *Proc. of IEEE Transactions on Man-Machine Systems*, volume 11, pages 190–202, 1970.

[8] E. B. Costa. *Um Ambiente Interativo de Ensino e Aprendizagem*. PhD thesis,

Departamento de Engenharia Elétrica, Universidade Federal da Paraíba, Campina Grande, Brasil, 1997.

[9] E. B. Costa, M.A. Lopes, and E. Ferneda. Mathema: A Learning Environment Based on a Multi-Agent Architecture. In J Wainer and A. Carvalho, editors, *Proc. of 12th Brazilian Symposium on Artificial Intelligence*, volume 991 of *Lecture Notes in Artificial Intelligence*, pages 141–150. Springer-Verlag, Campinas, Brazil, October 1995.

[10] E.B. Costa, G.M. Góis, J.C.A. de Figueiredo, and A. Perkusich. Towards a multi-agent interactive learning environment oriented to the petri net domain. In *Proc. of IEEE Int. Conf. on Systems Man and Cybernetics*, pages 250–261, San Diego, USA, October 1998.

[11] E.B. Costa and A. Perkusich. Modeling the cooperative interactions in a teaching/learning situation. In *Proc. of The Intelligent Tutoring Systems, ITS, 96, Lecture Notes in Artificial Intelligence*, Montreal, Canada, June 1996.

[12] E.B. Costa, A. Perkusich, and J.C.A. de Figueiredo. A multi-agent environment to aid in the design of petri nets based software systems. In *Proc. of The Eighth International Conference on Software Engineering and Knowledge Engineering, SEKE'96*, pages 253–259, Lake Tahoe, USA, June 1996.

[13] E.B. Costa, A. Perkusich, and A.A. Jatobá. Petri net based modelling of the cooperative interaction in multi-agent based learning environment. In *Proc. of IEEE Computational Engineering in Systems Application, CESA'96*, Lille, France, July 1996.

[14] Josenildo Costa da Silva. Aquisição de conhecimentos e manutenção para uma sociedade de agentes tutores artificiais. Dissertação de mestrado, Coordenação de Pós-Graduação em Informática, Centro de Ciências e Tecnologia, Universidade Federal da Paraíba, 1999.

[15] J. Ezpeleta and J.M. Colom. Automatic synthesis of colored petri nets for the control of fms. volume 13, pages 327 – 337, June 1997.

[16] J. Ferber. Sl'intelligence artificielle distribuée. In *La Recherche*, pages 750–758, 1991.

[17] S. Franklin and A. Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages*, Springer-Verlag, 1996.

[18] L. Gomes and J.-C. Garção. Towards the implementation of conflit resolution on a non autonomous highlevel- petri net model. pages 143 – 160, June 1996.

[19] Caroline C. Hayes. Agents in a nutshell - a very brief introduction. In *IEEE Transactions on Knowledge and Data Engineering*, volume 11, Lille, France, Jan 1999.

[20] H.P. Huang and P.C. Chang. Specification, modeling and control of a flexible manufacturing cell. volume 30, pages 2515 – 2543, 1992.

[21] F.J. Jaimes-Romero, D. Minoz-Rodriguez, and H. Tawfik. Modeling resource management in cellular systems using petri nets. volume 46, pages 298 – 312, May 1997.

[22] K. Jensen. Springer - Verlag.

[23] K. Jensen and et. al. *Design/CPN 4.0.* Meta Software Corporation and Department of Computer Science, University of Aarhus, Denmark. On-line version:http://www.daimi.aau.dk/designCPN/.

[24] K. Jensen and et. al. *Design/CPN Manuals.* Meta Software Corporation and Department of Computer Science, University of Aarhus, Denmark. On-line version:http://www.daimi.aau.dk/designCPN/.

[25] D. H. Jonassen and S. Wang. The physics tutor: Integrating hypertext and expert systems. In *Journal of Educational Technology Systems*, volume 22(1), pages 19–28, 1993.

[26] G. kearsley. *Artificial Intelligence and Instruction: applications and methods.* Addison-Wesley, 1987.

[27] M.V.C. Miranda and A. Perkusich. Modeling and analysis of a multi-agent system using colored petri nets. In *Proc. of Workshop on Applications of Petri Nets to Intelligent System Development*, pages 87–99, June 1999.

[28] T. Murata. Petri nets: Properties, analysis and applications. volume 77, pages 541–580, apr 1989.

[29] M. A. Orey and W. A. Nelson. Development principles for intelligent tutoring systems: Integrating cognitive theory into the development of computer-based instruction. In *Educational Technology Research and Development*, volume 41, pages 59–72, 1993.

[30] O. Park. Functional characteristics of intelligent computer-assisted instruction: Intelligent features. In *Educational Technology*, pages 7–14, 1988.

[31] P. Peleties and R. DeCarlo. Analysis of a hybrid system using symbolic dynamics and petri nets. volume 30, page 1421, September 1994.

[32] A. Perkusich and J.C.A de Figueiredo. Object oriented design of a track-vehicle system. In *Proc. of The Seventh International Conference on Software Engineering and Knowledge Engineering, SEKE'95*, pages 283–290, Washington, USA, June 1995.

[33] J. Piaget. *Fazer e Compreender*. Melhoramentos: Editora da Universidade de São Paulo, São Paulo, 1978.

[34] J. Prock. A new technique for fault detection using petri nets. volume 27, pages 239 – 245, March 1991.

[35] J. W. Rickel. Intelligent computer-aided instruction: A survey organized around system components. In *Proc. of IEEE Transactions on Systems, Man, and Cybernetics*, volume 19, pages 40–57, 1989.

[36] S. Russel and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, New York, 1995.

[37] R.M. Shapiro. Validation of a vlsi chip using hierarchical colored petri nets. volume 31, pages 607 – 625, 1991.

[38] B. Shneiderman. *Designing The User Interfaces: Strategies for Effective Human Computer Interaction*. Addisson-Wesley, 1992.

[39] J. S. Sichman, Y. Demazeau, and O. Baissier. When can knowledge-based systems be called agents? In *Anais do Simpósio Brasileiro de Inteligência Artificial*, pages 172–185, Rio de Janeiro, Brasil.

[40] J.F.N. Tchako, B. Beldjilali, and C. Tahon. Modelling with coloured timed petri nets and simulation of a dynamic and distributed management system for a manufacturing cell. volume 7, pages 79 – 80, November 1994.

[41] Jeffrey Ullman. *Elements of ML Programming*. Prentice-Hall, 1993.

[42] Jeffrey Ullman. *Elements of ML Programming, ML97 Edition*. Prentice-Hall, 1998.

[43] J. A. Valente. Diferentes usos do computador na educação. In *Núcleo de Informática Aplicada à Educação (NIED)*, pages 1–23, Gráfica Central da Unicamp, São Paulo, 1993.

[44] L. S. Vygotsky. *A Formação Social da Mente*. Livraria Martins Fontes Editora LTDA., 1991.

[45] E. Wenger. *Artificial Intelligence and Tutoring Systems*. Morgan Kaufman, 1987.

[46] C.-H. Wu and X.-L. Xie. Enhanced high-level petri nets with multiple colors for knowledge verification/validation of rule-based expert systems. volume 27, pages 760 – 773, October 1997.

[47] R. Zurawski. Systematic construction of functional abstractions of petri net models of flexible manufacturing systems. volume 41, pages 584 – 592, December 1994.

[48] R. Zurawski and M.C. Zhou. Petri nets and industrial applications: A tutorial. volume 41, pages 567 – 583, December 1994.

Apêndice A

Redes de Petri

Um modelo é uma representação, geralmente em termos matemáticos, das principais características de um objeto ou sistema. Através da análise do modelo, um sistema real pode ser estudado sem o perigo, custo ou inconveniência da manipulação de seus elementos. Dentre as técnicas formais para modelar e analisar sistemas, podemos citar: modelos de fila, álgebra de processos, lógica temporal e redes de Petri.

Sistemas podem geralmente ser vistos como constituídos de vários subsistemas interagindo entre si. Há cada vez mais a necessidade ou conveniência de um funcionamento paralelo de subsistemas de um sistema complexo. Maior poder de processamento, compartilhamento de informações em banco de dados distribuídos, execução paralela de tarefas em um sistema de produção industrial e tolerância a falhas são exemplos de justificativas para o paralelismo. O paralelismo, por outro lado, complica a concepção e a análise de um sistema devido à explosão de estados inerente, aspectos temporais e necessidade de sincronização, entre outras causas.

Redes de Petri [28] é uma ferramenta matemática e gráfica de modelagem que pode ser aplicada em diversos tipos de sistemas. Aplicam-se apropriadamente a sistemas assíncronos e com alto índice de concorrência ou paralelismo. São, portanto, áreas privilegiadas as redes de computadores e protocolos de comunicações, sistemas operacionais, bancos de dados distribuídos, entre muitas outras. Qualquer área em que concorrência seja um fator preponderante é passível de ter vantajosamente aplicadas as redes de Petri. A análise de redes de Petri pode revelar informações importantes sobre a estrutura e o comportamento dinâmico do sistema modelado.

Na seção A.1 apresentamos as definições das redes de Petri tradicionais. A seguir, na seção A.2 daremos uma breve introdução de análises em redes de Petri. Nas seções A.3 e A.4, apresentaremos 2 métodos de análise para as redes de Petri: Enumeração do Espaço de Estados e Invariantes, respectivamente. Na Seção A.4 detalharemos a técnica de análise através de invariantes, bem como as técnicas de composição fusão de lugar e fusão de transição. Nesta Seção mostraremos que o sistema composto pelos subsistemas preservam as mesmas características dos subsistemas [4]. Por fim, na seção A.5, apresentamos a rede de Petri de alto nível denominada redes de Petri colorida.

A.1 Conceitos de redes de Petri

Redes de Petri é uma estrutura de rede mais uma marcação inicial. Definimos uma estrutura de rede como sendo uma tupla $N = \langle P, T, F, W \rangle$, onde:

- P é um conjunto finito de lugares,
- T é um conjunto finito de transições,
- $F \subseteq (P \times T) \cup (T \times P)$ é um conjunto finito de arcos,
- $W : F \to I\!N^*$ é uma função peso,
- $P \cap T = \emptyset$ e $P \cup T \neq \emptyset$.

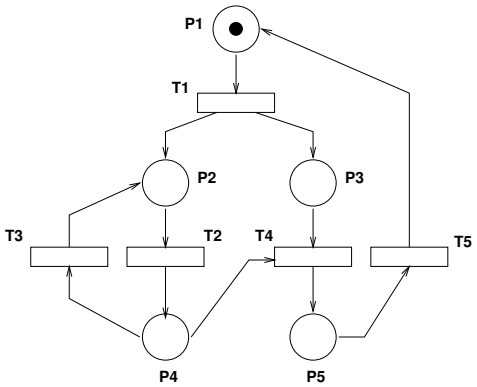


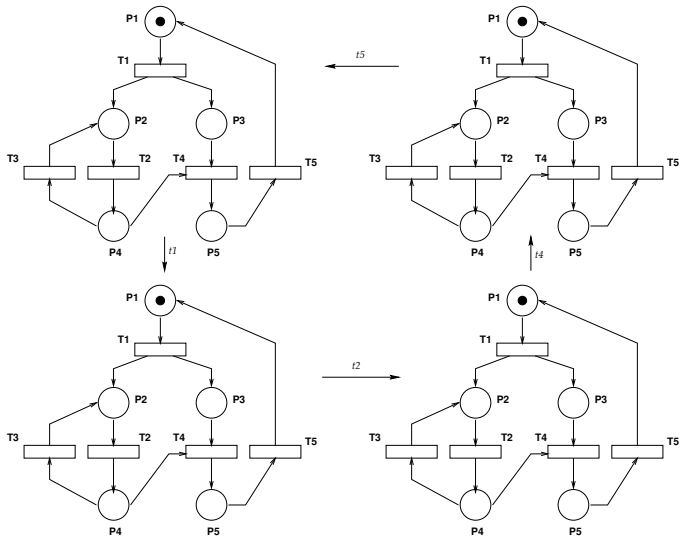
Figura A.1: Representação Gráfica de uma Rede de Petri

A marcação de uma rede de Petri é uma função $M : P \to I\!N$ e sua evolução simula o comportamento dinâmico do sistema modelado. Assim, podemos definir uma rede

de Petri como sendo uma tupla $PN = \langle P, T, F, W, M_0 \rangle$. Podemos vê-la como um grafo mais uma marcação inicial (que representa um estado inicial). Na sua representação gráfica, os lugares são representados por círculos e as transições por retângulos ou barras. Os arcos, que representam uma relação de fluxo, e vão de um lugar para uma transição ou de uma transição para um lugar, são rotulados com seus pesos. O valor 1 é omitido e um arco com peso k pode ser interpretado como um conjunto de k arcos paralelos. As fichas são representadas por pontos no interior dos lugares. Informalmente, costuma-se referir à representação gráfica como se fosse a própria rede de Petri. Na Figura A.1 temos a representação gráfica de uma rede de Petri.

Para simular a dinâmica de um sistema, a marcação (ou estado) de uma rede de Petri evolui de acordo com a regra de disparo enunciada a seguir:

- (Pré-condição): Uma transição está habilitada se cada lugar de entrada p de t contém pelo menos *w(p,t)* fichas, onde *w(p,t)* é o peso do arco de p para t.
- Uma transição habilitada pode ou não disparar.
- (Pós-condição) Ao disparar t, *w(p,t)* fichas são removidas de cada entrada p de t e *w(p,t)* fichas são acrescentadas a cada saída p de t, onde *w(p,t)* é o peso do arco de t para p.



A Figura A.1 ilustra, na forma de um grafo, todos os acionamentos possíveis das transições da rede da Figura A.1.

As redes de Petri têm sido utilizadas em diversas áreas. Por exemplo, por meio das redes de Petri podemos modelar atividades paralelas ou concorrentes , protocolos de comunicação [21], o controle de sincronização em sistemas distribuídos [20], sistemas híbridos [31], sistemas tolerantes a faltas [34], a base de conhecimento de sistemas especialistas [46], conflitos [18], etc. Por exemplo, ver Figura A.2(a), as atividades paralelas ou concorrentes representadas pelas transições t_2 e t_3 começam após a ocorrência da transição t_1 e terminam após a ocorrência da transição t_4. Portanto, as transições t_1 e t_4 sincronizam o início e o fim das atividades paralelas ou concorrentes representadas pelas transições t_2 e t_3, respectivamente. Duas transições são ditas concorrentes se uma transição pode ocorrer antes, em paralelo ou após a outra transição. Note que, cada lugar da rede da Figura A.2(a) possui apenas uma transição de entrada e uma de saída. Redes de Petri com esta propriedade são denominadas grafos marcados. Esta classe de rede permite modelar apenas a sincronização e a concorrência de atividades paralelas, mas não conflitos ou tomadas de decisões [28]. Dois eventos estão em conflito se a ocorrência de um evento desabilita a ocorrência do outro. Esta situação pode ser modelada naturalmente por redes de Petri. Por exemplo, conforme mostra-se na Figura A.2(b), as transições t_1 e t_2 estão conflitos. Isto porque, a ocorrência da transição $t_1(t_2)$ desabilita a ocorrência da transição $t_2(t_1)$. A causa deste conflito pode ser o compartilhamento de um recurso. Suponha que um robô é utilizado para carregar duas máquinas m_1 e m_2. Note que, o robô é um recurso compartilhado pelas máquinas e portanto este somente pode carregar uma máquina de cada vez. Neste caso, as transições t_1 e t_2, ilustradas na Figura A.2(b), podem modelar a ocorrência dos eventos *iniciar carregamento da máquina* m_1 e *iniciar carregamento da máquina* m_2, respectivamente.

A.2 Análise de Modelos de Redes de Petri

Analisando o modelo de rede de Petri de um sistema, podemos determinar se o sistema modelado apresenta ou não as propriedades de *alcançabilidade, limitação/segurança, conservação, vivacidade ou reversibilidade/recorrência* [48]. Tais propriedades são descritas a seguir:

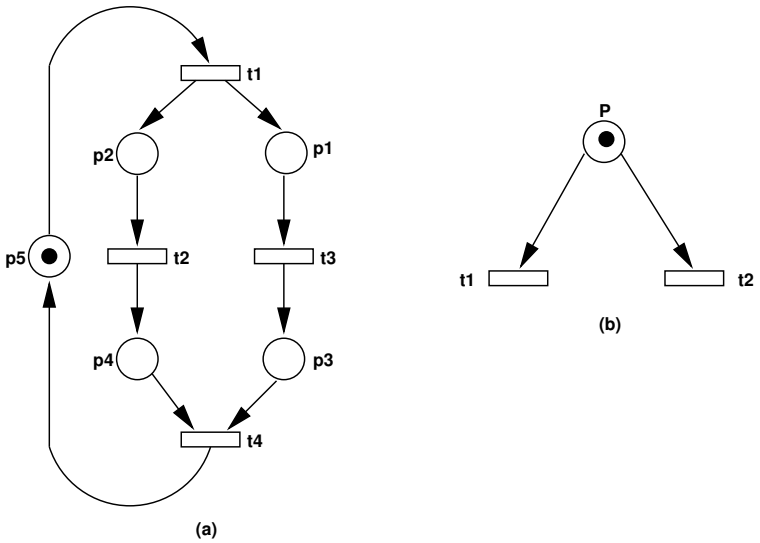


Figura A.2: (a) Modelagem de atividades paralelas (b) Conflito ou decisão

1. *Alcançabilidade:* Uma marcação M_i é dita alcançável a partir da marcação inicial M_0 se existir uma seqüência de ocorrências de transições σ que transforma M_0 em M_i, denotado por $M_0[\sigma\rangle M_i$. Denota-se por $R(M_0)$ o conjunto de todas as marcações alcançáveis a partir de M_0 e por $L(M_0)$ ao conjunto de todas as seqüências de ocorrências de transições. Então o problema de determinar a existência de uma marcação específica M_i, consiste em determinar se $M_i \in R(M_0)$.

2. *Limitação/Segurança:* A propriedade de limitação está relacionada com a capaciade de armazenamento de um sistema. Uma rede de Petri é dita *k-limitada* se o número de fichas em qualquer lugar $p \in P$ é sempre menor ou igual a k para cada marcação M_i alcançável a partir de M_0, onde $M_i \in R(M_0)$ e k é um inteiro positivo. Uma rede de Petri é dita *segura* se for 1- limitada.

3. *Conservação:* Uma rede de Petri é dita conservativa se, para qualquer marcação alcançável, a soma das fichas de cada lugar da rede, ponderada pelas componentes de um vetor $W = (w_1, w_2, \cdots, w_j, \cdots, w_m)$, é constante, onde m é o número de lugares da rede e $w_j > 0, \forall\, j = 1, 2, \cdots, m$. Uma rede de Petri é dita *estritamente conservativa* se cada componente do vetor W é a unidade.

4. *Vivacidade:* A propriedade de vivacidade está relacionada com a ocorrência de bloqueios em um sistema. Uma transição $t \in T$ é dita viva se para qualquer

marcação $M_i \in R(M_0)$ existir, a partir de M_i, uma seqüência de ocorrência σ contendo t. Uma rede de Petri é dita viva se todas as suas transições são vivas. Entretanto, na prática, esta propriedade é considerada ideal. Desta forma, diferentes níveis de vivacidade são definidos, com o objetivo de tornar este conceito o mais abrangente possível. Uma transição t de uma rede de Petri é dita:

- L_0-viva: se t não puder ocorrer em qualquer seqüência de ocorrência $\sigma \in L(M_0)$.
- L_1-viva: se t puder ocorrer pelo menos uma vez em alguma seqüência de ocorrência $\sigma \in L(M_0)$.
- L_2-viva: se, dado um inteiro positivo k, t puder ocorrer pelo menos k vezes em alguma seqüência de ocorrência $\sigma \in L(M_0)$.
- L_3-viva: se t puder ocorrer infinitamente em alguma seqüência de ocorrência $\sigma \in L(M_0)$.
- L_4-viva: se t for L_1-viva para cada marcação $M \in R(M_0)$.

Uma rede de Petri é L_k-viva se cada transição da rede for L_k-viva, isto para, *k=0,1,2,3,4*. Note que, conforme o conceito de vivacidade, uma rede de Petri é viva se for considerada L_4-viva. Por exemplo, a rede mostrada na Figura A.3(a) é estritamente L_1-viva pois cada transição somente pode ocorrer, nesta seqüência $t_2t_4t_5t_1t_3$, apenas uma vez. Note que, as transições t_0, t_1, t_2, t_3 na Figura A.3(b) são estritamente L_0-viva (morta), L_1-viva, L_2-viva e L_3-viva, respectivamente.

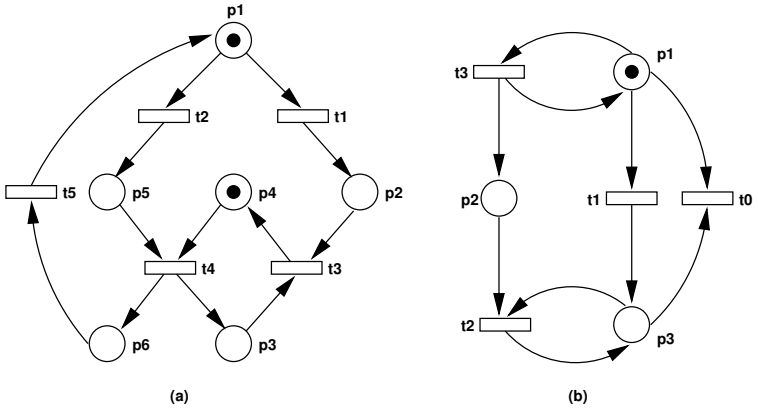


Figura A.3: Redes utilizadas para exemplificar os níveis de vivacidade

5. *Reversibilidade/Recorrência:* Uma rede de Petri, para uma dada marcação inicial M_0, é dita reversível se para cada marcação $M_i \in R(M_0)$, M_0 pode ser alcançada a partir de M_i. Uma marcação M_j é um *estado recorrente* se para cada marcação $M_i \in R(M_0)$, M_j é alcançável a partir de M_i.

Os métodos de análise estão baseados na enumeração do espaço de estado do sistema ou na utilização de equações algébricas [47], fato mostrado a seguir.

A.3 Enumeração do Espaço de Estados

No que se refere a enumeração do espaço de estado, o conjunto de marcações $R(M_0)$ de uma rede de Petri pode ser infinito ou finito. Denomina-se *árvore de cobertura*, a árvore obtida para o caso infinito e por *árvore de alcançabilidade*, a árvore obtida para o caso finito. O *grafo de cobertura* de uma rede de Petri é o grafo orientado $G = (V, E)$, onde o conjunto de nós V é o conjunto de todas as marcações distintas obtidas da árvore de cobertura e E é o conjunto de arcos. Cada arco é rotulado por uma transição t_k cuja ocorrência transforma uma marcação M_i na marcação M_j onde, M_i e $M_j \in V$. No caso de uma rede de Petri limitada, ou seja, aquela cujo conjunto de marcações $R(M_0)$ é finito, o grafo de cobertura é chamado de *grafo de alcançabilidade*, pois o conjunto de nós V possui todos os elementos de $R(M_0)$. Desde que a árvore de cobertura representa um conjunto infinito de marcações, devemos sistematizar a construção deste tipo de árvore por meio do seguinte algoritmo:

Algoritmo de Construção da Árvore de Cobertura

Passo 1) Denominar de *nova* a raiz da árvore M_0.

Passo 2) Enquanto existir marcações do tipo *nova* fazer o seguinte:

Passo 2.1) Selecionar uma marcação *nova* M.

Passo 2.2) Se M é idêntica a outra marcação na árvore, então denominar *velha* a marcação M e selecionar outra marcação.

Passo 2.3) Se nenhuma transição está habilitada em M, denominar esta marcação de *terminal*.

Passo 2.4) Para cada transição t habilitada na marcação M fazer o seguinte:

Passo 2.4.1) Obter a marcação M' que resulta da ocorrência de t em M.

Passo 2.4.2) Se no caminho da raiz até M, existir uma marcação M" tal que $M'(p) \geq M$"(p) para cada lugar $p \in P$, e $M'(p) \neq M$"(p), então substitua $M'(p)$ por ω para cada lugar p onde $M'(p) > M$"(p).

Passo 2.4.3) Introduzir a marcação M' na árvore, desenhando um arco com rótulo t de M para M' e denominar *nova* a marcação M'.

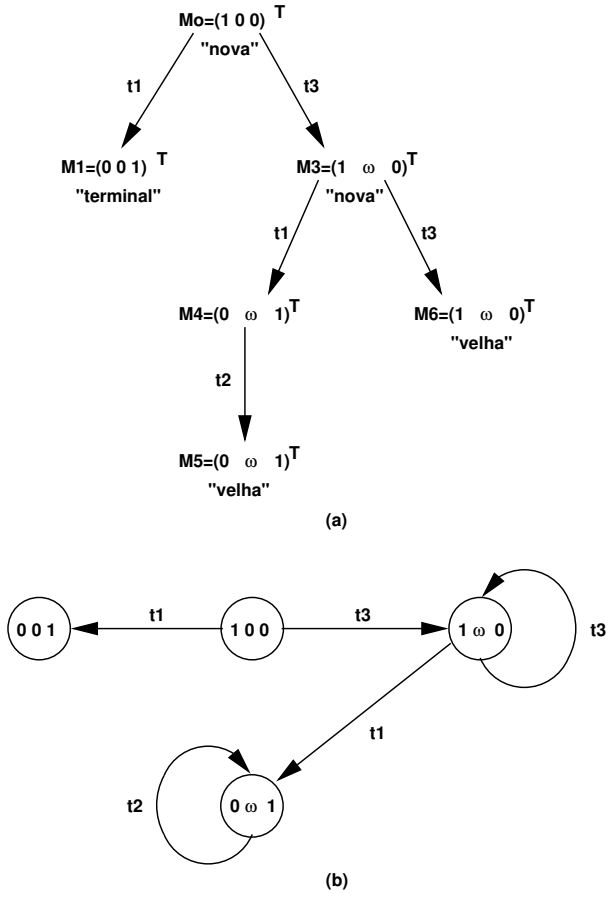


Figura A.4: (a) Árvore de cobertura (b) Grafo de cobertura

As propriedades podem ser provadas enumerando-se o espaço de estados e analisando estados alcançáveis e seqüências de eventos (ocorrências de transições). Por exemplo, aparecendo o símbolo ω em qualquer nó da árvore, a rede é ilimitada, visto que ω representa um inteiro positivo arbitrariamente grande. Caso contrário, a rede é limitada. Se cada nó da árvore apresentar somente zeros e uns, então a rede é segura.

Uma transição é dita morta caso ela não apareça como rótulo em nenhum arco da árvore. Note que, para uma rede limitada, a árvore contém todas as possíveis marcações alcançáveis a partir da marcação inicial. Para exemplificar o uso do algoritmo acima (ver Figura A.4(a)), considere a rede mostrada na Figura A.3(b). Na marcação inicial $M_0 = (1,0,0)^T$, as transições t_1 e t_3 estão habilitadas. A ocorrência de t_1 transforma $M_0 = (1,0,0)^T$ em $M_1 = (0,0,1)^T$ que é na verdade um nó terminal da árvore, pois, nenhuma transição está habilitada nesta marcação. Ocorrendo t_3 em M_0 resulta em $M_3' = (1,1,0)^T$. Note que, esta marcação cobre M_0. Portanto, a nova marcação introduzida é $M_3 = (1,\omega,0)^T$, onde, as transições t_1 e t_3 estão novamente habilitadas. Ocorrendo t_1 transforma M_3 em $M_4 = (0,\omega,1)$, onde, t_2 pode ocorrer, resultando na marcação *velha* $M_5 = M_4$. Ocorrendo t_3 em M_3 resulta na marcação *velha* $M_6 = M_4$. O grafo de cobertura é mostrado na Figura A.4(b).

A.4 Invariantes de Redes de Petri

A seguir, introduziremos uma série de conceitos necessários ao entendimento do que seja os invariantes de uma rede de Petri.

A.4.1 Matriz de Incidência

A matriz de incidência de uma rede de Petri com N transições e M lugares é uma matriz de inteiros, $N \times M$, $C = [C_{ij}]$, com: $C_{ij} = {C_{ij}}^+ - {C_{ij}}^-$, onde:

- ${C_{ij}}^+ = w(i,j)$, é o peso do arco da transição i para o lugar de saída j; representa, portanto, o número de fichas acrescentadas ao lugar j quando a transição i dispara uma vez.

- ${C_{ij}}^- = w(i,j)$, é o peso do arco que vai do lugar de entrada j para a transição i; representa, o número de fichas removidas do lugar j quando a transição i dispara uma vez.

Desta forma, C_{ij} representa o número de fichas alteradas no lugar j quando a transição i dispara uma vez. A matriz de incidência da rede mostrada na Figura A.1 é:

$$C = \begin{vmatrix} -1 & 1 & 2 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & -2 & -1 & 1 \\ 1 & 0 & 0 & 0 & -1 \end{vmatrix}$$

A.4.2 Equação de Estado

Consideremos uma marcação M_k representada por um vetor coluna $M \times 1$. O *j-ésimo* termo de M_k denota o número de marcas no lugar j imediatamente após o *k-ésimo* disparo em alguma seqüência de disparos. O vetor do *k-ésimo* disparo ou vetor de controle, U_k, é um vetor coluna, N, de N-1 0´s e um 1 na *i-ésima* linha, indicando que a transição i dispara no *k-ésimo* disparo. Define-se a equação de estados para uma rede de Petri da seguinte forma:

- $M_k = M_{k-1} + C.U_k$, onde C representa a matriz de incidência e U_k representa o vetor de controle.

A equação de estado informa qual o estado que a rede de Petri irá atingir, após uma seqüência de disparos das transições. O vetor de controle carrega esta seqüência de disparos, porém, ele não informa a ordem que cada transição é disparada em relação às outras transições.

Os conceitos apresentados até agora, servirão para a compreensão dos conceitos de invariantes de lugar e de transição no contexto de redes de Petri. O objetivo é, além de demonstrar os conceitos de invariantes, mostrar que os invariantes de uma rede de Petri R, formada por composição a partir de duas sub-redes $R^{'}$ e $R^{''}$, podem ser obtidos diretamente dos invariantes destas sub-redes, por justaposição e/ou concatenação, isto sem que seja necessário a resolução de sistemas de equações quase sempre de dimensionalidades elevadas. A técnica de composição é baseada na fusão de elementos do tipo lugar ou na fusão de elementos do tipo transição. Esta técnica permite tanto conservar certos invariantes quanto obter novos invariantes na rede de Petri global, como será demonstrado a seguir.

A.4.3 Definição de Invariantes

Seja T o conjunto de transições de uma rede de Petri, podemos definir formalmente os invariantes de transição desta rede da seguinte forma:

Definição A.1 *[Invariante de Transição] Um vetor transição $I_t : T \rightarrow Z$ é dito um invariante de transição de R se, e somente se:*

- $C.I_t^T = 0$.

Em outras palavras, um invariante de transição é definido por um vetor de dimensão igual ao número de transições da rede, onde cada componente indica o número de vezes que a transição correspondente deve ser disparada. Cada invariante associa um conjunto de transições cuja seqüência de disparos não modifica a marcação da rede.

Definição A.2 *[Invariante de Lugar] Um vetor lugar $I_p : P \rightarrow Z$ é dito um invariante de lugar R se, e somente se:*

- $C.I_p = 0$.

Em outras palavras, um invariante de lugar é definido por um vetor de dimensão igual ao número de lugares da rede, onde cada componente associa um peso a um lugar. Estes pesos definem a ponderação a ser feita para que a soma das fichas contidas nos lugares seja uma constante, isto independentemente dos disparos das transições. Desta forma, tem-se:

- $I_p = [\alpha_1, \ldots, \alpha_i, \ldots, \alpha_m]$, onde I_p é um invariante de lugar de R. Então,
- $\alpha_1.M(p_1) + \ldots + \alpha_i.M(p_i) + \ldots + \alpha_m.M(p_m) = K$, onde K é uma constante que é determinada pela marcação inicial da rede, e $M(p)$ representa a marcação do lugar p.

A.4.4 Fusão de elementos

Um elemento de uma rede de Petri será considerado aqui como um lugar ou uma transição. A técnica de fundir elementos do mesmo tipo, quer sejam lugares quer sejam transições, possibilita, de forma sistemática e estruturada, a construção de redes

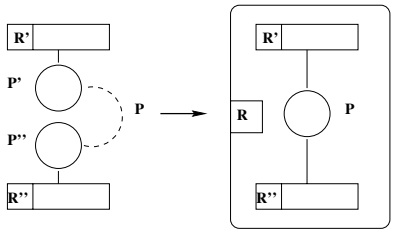


Figura A.5: Rede R formada pela fusão de lugares

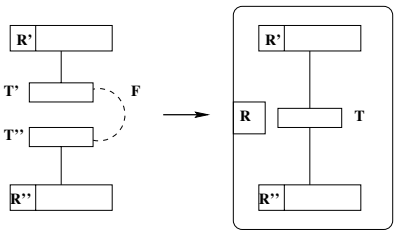


Figura A.6: Rede R formada pela fusão de Transições

de grande dimensão. As Figuras A.5 e A.6 ilustram esta técnica, quando aplicada a duas sub-redes R' e R''.

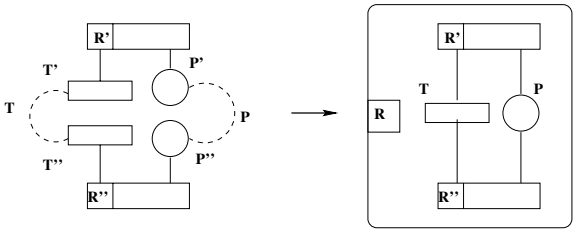


Figura A.7: Rede R formada pela fusão de Lugares e Transições

Os arcos tracejados nas Figuras A.5 e A.6 indicam quais os elementos do mesmo tipo, pertencentes às sub-redes R' e R'' que estão envolvidos no processo de fusão. Considerando esta notação, a Figura A.5 mostra a fusão de elementos do tipo lugar, enquanto que a Figura A.6 mostra a fusão de elementos do tipo transição. Nota-se que os elementos do mesmo tipo desaparecem na fusão, dando origem a um novo elemento na rede R. Combinando as Figuras A.5 e A.6 obtém-se uma rede que é formada simultaneamente pela fusão de lugares e transições, conforme mostra a Figura A.7.

Estas técnicas de fusão de lugar e/ou fusão de transição permite tanto conservar certos invariantes, quanto obter outros por concatenação e/ou justaposição como será visto a seguir.

t'1	...	*t'n'*	*t"1*	...	*t"n"*		
C'(1,1)	...	C'(1,n')	0	0	0	*P'1*	
⋮		⋮	0	0	0	⋮	
C'(m'-k,1)	...	C'(m'-k,n')	0	0	0	⋮	
C'(m'-k+1,1)	...	C'(m'-k+1,n')	C"(1,1)	...	C"(1,n")	*P'm'-k+1*	*P"1*
⋮		⋮	⋮		⋮	⋮	⋮
C'(m',1)	...	C'(m',n')	C"(k,1)	...	C"(k,n")	*P'm'*	*P"k*
0	0	0	C"(k+1,1)	...	C"(k+1,n")		
0	0	0	⋮		⋮		
0	0	0	C"(m",1)	...	C"(m",n")		*P"m"*

Figura A.8: Matriz C obtida pela fusão de k lugares de R' e R"

Considere duas sub-redes R' e R'', representadas por suas respectivas matrizes de incidência C' e C''. Seja R uma rede de Petri obtida pela composição de R' e R'' através da fusão dos k últimos lugares de R' com os k primeiros lugares de R''. Então a matriz de incidência C da rede global R toma a forma mostrada pela Figura A.4.4.

Como os invariantes de transição da rede R são dados pela solução das equações $C.I_t^T = 0$, logo este sistema pode ser analisado da seguinte forma:

$$C^1(j,1).\beta_1^{'} + \ldots + C^1(j,n^1).\beta_{n'} = 0, \text{ onde } j = 1,2,\ldots,m' - k \tag{A.1}$$

$$\begin{aligned} &[C'(m^{'} - k + j, 1).\beta_1' + \ldots + C^1(m^1 - k + j, n^1).\beta_{n'}'] + \\ &[C''(j,1).\beta_1'' + \ldots + C''(j,n^{''}), \beta_{n''}''] = 0, \text{ onde } j = 1,2,\ldots,k \end{aligned} \tag{A.2}$$

$$C''(j,1).\beta_1'' + \ldots + C''(j,n^{''}).\beta_n''] = 0, \text{ onde } j = (k+1),\ldots,m'' \tag{A.3}$$

É importante notar que os invariantes de transição de R', são soluções das equações A.1 e A.2, bastando para isso que os valores das incógnitas $(\beta_1'', \ldots, \beta_{n''}'')$ sejam todos nulos. Da mesma forma, os invariantes de transição de R'', são soluções das

equações A.2 e A.3 caso os valores das incógnitas $(\beta'_1, \ldots, \beta'_{n'})$ sejam todos nulos. Isto quer dizer que, todos os invariantes de transição de R' e de R'' estão presentes, respectivamente, nos invariantes de transição de R da forma:

$(\beta'_1, \beta'_2, \ldots, \beta'_{n'}, \underbrace{0, \ldots, 0}_{n''zeros})$ e $(\underbrace{0, \ldots, 0}_{n'zeros}, \beta''_1, \beta''_2, \ldots, \beta''_{n'})$

Estes são obtidos pela concatenação dos invariantes de R' e R'' com vetores nulos com dimensão e posicionamento adequados à construção de R, implícita em C da Figura A.4.4. Eventualmente outros invariantes de transição existem para R que tem a forma $(\beta_1, \beta_2, \ldots, \beta_{n'+n''})$ e são soluções das equações A.1, A.2, A.3. Considerando o caso dos invariantes de lugar que são obtidos pela solução da equação $Ip.C = 0$, tem-se:

$$C'(1,j).\alpha'_1 + \ldots + C'(m',j).\alpha'_{m'} = 0, \text{ onde } j = 1, 2, \ldots, n' \tag{A.4}$$

$$C''(1,j).\alpha''_1 + \ldots + C''(m'',j).\alpha''_{n''} = 0, \text{ onde } j = 1, 2, \ldots, n'' \tag{A.5}$$

$$\alpha'_{m'-k-j}, \text{ onde } j = 1, 2, \ldots, k \tag{A.6}$$

Nestes três grupos de equações, o primeiro (eq. A.4) contém um número de equações igual ao número de transições de R'' e o terceiro (eq. A.6) às $2k$ incógnitas correspondendo aqueles lugares fundidos que intervêm em todas as equações. Observa-se que as equações A.4 e A.5 são verificadas se, e somente se, as condições impostas pelas equações A.6 forem verdadeiras, isto é, se todos os valores das $2k$ incógnitas associados aos lugares fundidos forem idênticos dois a dois. De acordo com os valores destas incógnitas tem-se dois casos a considerar:

- $\alpha'_{m'-k-j} = 0$ ou α''_j, para todo $j = 1, 2, \ldots, k$

Neste caso, todos os invariantes de lugar de R' e de R" são respectivamente da forma:

$\alpha'_1, \alpha'_2, \ldots, \alpha'_{m'-k}, \underbrace{0, \ldots, 0}_{kzeros})$ e $(\underbrace{0, \ldots, 0}_{nzeros}, \alpha''_{k+1}, \alpha''_{k+2}, \ldots, \alpha''_{m''})$

Assim os invariantes de lugar de R serão da forma:

$(\alpha'_1, \alpha'_2, \ldots, \alpha'_{m'-k}, \underbrace{0, \ldots, 0}_{kzeros}, \alpha''_{k+1}, \alpha''_{k+2}, \ldots, \alpha''_{m''})$

obtidos pela justaposição dos invariantes de lugar de R' e R".

- $\alpha'_{m'-k-j} = \alpha''_j \neq 0$, para todo $j = 1, 2, ..., k$

Se o valor de pelo menos uma dessas incógnitas não é nulo, então é necessário encontrar pelo menos um invariante de lugar de R' e pelo menos um outro de R" para os quais estas $2k$ incógnitas sejam idênticas duas a duas, para se construir um invariante de lugar de R por justaposição dos invariantes de lugar de R' e R'.

Em ambos os casos, o sistema de equações fornece os invariantes de lugar da rede R, obtidos por justaposição dos invariantes de R' com os invariantes de R". Se não for encontrado um invariante de R' e outro de R" que satisfaçam à equação A.6, então estes invariantes desaparecem na rede global R.

Devido à dualidade existente entre os elementos de tipos diferentes de uma rede de Petri, ou seja, entre lugares e transições, o ato de fundir elementos do mesmo tipo de duas redes R' e R", resultando na rede R, conduz às seguintes considerações e propriedades abaixo:

1. Considerações

   (a) O elemento dual, na fusão de elementos do tipo transição é considerado lugar da rede;

   (b) O elemento dual, na fusão de elementos do tipo lugar é considerado transição da rede;

2. Propriedades

   (a) Todos os invariantes de elemento dual de $R´$ e de $R´´$ são invariantes de elemento dual de R, estes são obtidos por concatenação. Eventualmente, novos invariantes de elementos dual podem aparecer para R.

   (b) Todos os invariantes de elemento de $R´$ e $R´´$, para os quais os valores de k incógnitas associadas aos elementos fundidos são zero, serão invariantes de elemento de R. Estes são obtidos por justaposição.

(c) Se existe um invariante de elemento de R' e um invariante de elemento de R'' tal que os pesos associados aos elementos fundidos sejam idênticos dois a dois, então a justaposição destes invariantes é um invariante de R (a propriedade 2b é um caso particular).

(d) Todo invariante de elemento de R resulta da justaposição de um invariante de elemento de R' com um invariante de elemento de R'' pela propriedade 2b ou pela propriedade 2c . Eventualmente invariantes podem desaparecer na rede global R.

Estas propriedades são válidas sempre que uma rede de Petri R é formada por composição, a partir de duas sub-redes R' e R'', somente pela fusão de lugares ou de transições. Entretanto, se R é formada, simultaneamente pela fusão de lugar e transição, pode-se avaliar sua matriz de incidência C e aplicar as definições A.1 e A.2 para obter todos os seus invariantes.

A.5 Redes de Petri Coloridas

No mundo real os sistemas tendem a ser complexos, e as redes de Petri tradicionais não são adequadas para modelá-los. Neste caso, o modelo de rede de Petri de um sistema do mundo real apresenta estruturas similares e que podem ser representadas por uma única estrutura de alto nível. Desta forma, é possível representar modelos de grandes dimensões utilizando-se estruturas de alto nível relativamente menores. O desenvolvimento e aplicação das redes de Petri de alto nível, tais como as redes de Petri Coloridas (*CPNs*) [22] e as redes de Petri Predicado/Transição [5], constituiu um avanço significativo nesta direção. As redes de Petri de alto nível, mais especificamente as CPNs, têm o mesmo poder de descrição e análise das redes de Petri de baixo nível. Assim, todos os sistemas modelados por redes de Petri podem ser modelados por CPNs, mas de forma muito mais compacta. A representação mais compacta das CPNs é conseguida através de informações associadas às fichas, ou seja, numa CPN cada ficha possui uma cor que a diferencia das demais, indicando assim sua identidade. Formalmente, definimos uma CPN não hierárquica como sendo [22] uma a tupla $\langle \Sigma, P, T, A, N, C, G, E, I \rangle$, onde,

1. Σ é um conjunto finito e não vazio de *cores*,
2. P é um conjunto finito de *lugares*,
3. T é um conjunto finito de *transições*,
4. A é um conjunto finito de *arcos* tal que $P \cap T = P \cap A = T \cap A = \emptyset$,
5. $N : A \rightarrow P \times T \cup T \times P$ é a *função nó*,
6. $C : P \rightarrow \Sigma$ é a *função cor*,
7. $G : T \rightarrow EXPR$ é a *função de guarda*,
8. $E : A \rightarrow EXPR$ é a *função de arco*,
9. $I : P \rightarrow EXPR$ é a *função de inicialização*.

onde, $EXPR$ é um conjunto de expressões definidas no domínio do conjunto das cores. Cada componente de uma CPN tem o seguinte o significado:

(1) O conjunto de cores determina os tipos, operações e funções que podem ser usadas nas inscrições das rede, ou seja, nas expressões de arco, guardas e expressões de inicialização,

(2)+(3)+(4) *Lugares*, *transições* e *arcos* são representados pelos conjuntos P,T e A. As redes CPNs podem apresentar estrutura vazia, ou seja, $P \cup T \cup A = \emptyset$. Isto permite ao usuário definir e verificar sintaticamente o conjunto de cores sem que seja necessário desenhar a rede,

(5) A *função nó* associa cada arco a um par, onde o primeiro elemento do par é o nó fonte (lugar/transição) e o segundo o nó destino (transição/lugar),

(6) A *função de cor* C associa cada lugar da rede a uma cor pertecente ao conjunto de cores. Isto significa dizer que, as fichas dos lugares devem possuir tipos pertencentes ao conjunto de cores,

(7) A *função de guarda* G associa cada transição da rede a uma expressão do tipo booleana, onde, as variáveis das expressões devem possuir tipos pertencentes ao conjunto de cores. A guarda pode ser omitida se for o caso,

(8) A *função de arco E* associa cada arco da rede a uma expressão, onde, a avaliação de uma expressão de arco é um multi conjunto sobre o conjunto de cores,

(9) A *função de inicialização I* mapeia cada lugar da rede a uma expressão onde, cada expressão é um multi conjunto sobre o conjunto de cores.

Desta forma, uma CPN é composta essencialmente por uma estrutura, um conjunto de inscrições e um conjunto de declarações. Como as redes de Petri, as CPNs são também grafos direcionados e bipartidos. Entretanto, ao invés de pesos inteiros, aos arcos são associadas inscrições que determinam dinamicamente quantas e quais fichas devem ser removidas ou adicionadas aos lugares associados, na ocorrência de uma transição. Inscrições, denominadas guardas, podem ser também associadas às transições. Guardas restringem a ocorrência de transições a determinadas condições. O estado inicial de uma CPN também é determinado por inscrições associadas aos lugares. Cada inscrição é, em geral, uma expressão construída a partir de constantes, variáveis e operadores previamente definidos. Uma CPN também possui um conjunto de declarações para indicar a natureza dos elementos citados nas diversas inscrições, à semelhança de uma área de declarações de uma linguagem de programação qualquer.

As inscrições e declarações de uma CPN podem, *a priori*, ser escritas em praticamente qualquer linguagem com sintaxe e semântica bem definidas. Em geral, as CPNs têem sido utilizadas em associação com uma linguagem denominada CPN-ML [24], derivada da linguagem funcional *Standard ML* [41], cuja sintaxe é bastante semelhante à usada por linguagens de programação convencionais. Atualmente as CPNs utilizam a linguagem *Standard ML'97* [42; 23].

Por razões históricas, na teoria das CPNs usa-se a expressão *conjunto de cores (colour set)* em substituição a tipos de dados e, por conseqüência, cada valor é denominado cor (*colour*), que pode ser de um tipo arbitrário de dados (inteiro, real, lista, etc.). Desta forma, cada lugar na estrutura interna é associado a um conjunto de cores, que indica o tipo de fichas que o lugar pode conter, i.e., para um dado lugar, todas as fichas devem ter cores que pertencem a um mesmo tipo. A linguagem CPN-ML dispõe de mecanismos que permitem a definição de conjuntos de cores relativamente complexos.

Variáveis de Transição referem-se ao conjunto de variáveis presentes nas inscrições dos arcos e na guarda da referida transição. Uma *ligação (binding)* é a substituição de cada variável da transição por uma cor (valor). É requerido, entretanto, que as cores pertençam aos conjuntos de cores apropriados e que impliquem a avaliação da guarda como verdadeira .

Em cada marcação, a ocorrência de uma transição sob uma determinada ligação é dita habilitada se todos os seus lugares de entrada tiverem fichas suficientes para satisfazer às expressões dos arcos. Cada expressão deve ser devidamente avaliada segundo as substituições determinadas pela ligação, a fim de determinar quantas e quais fichas são requeridas nos lugares de entrada. Caso a transição ocorra, então são retiradas fichas dos lugares de entrada e depositadas novas fichas nos lugares de saída. A quantidade de fichas é determinada também pela avaliação das expressões dos arcos segundo as substituições implicadas pela ligação.

Exemplos do uso de CPNs podem ser encontrados em [37; 15; 40]. Outros detalhes referentes a semântica, técnicas de análise, bem como aplicações adicionais podem ser encontrados em [22].

A.5.1 Redes de Petri Coloridas Hierárquicas

A idéia básica das redes de Petri Coloridas Hierárquicas (CPNH) é possibilitar a construção de um modelo através da combinação de um conjunto de redes relativamente menores denominadas *páginas*, de forma análoga à construção de um programa a partir de um conjunto de módulos e funções [22]. O poder de modelagem de uma rede de Petri, de uma CPN e de uma CPNH são equivalentes. É sempre possível traduzir uma CPNH para uma não hierárquica, que por sua vez pode ser traduzida para uma rede de Petri. Em termos de linguagem de programação, podemos comparar as redes de Petri com as linguagens de máquina, as CPNs com a introdução de elementos de dados estruturados na programação e as CPNHs com o uso de módulos e funções.

As CPNHs são construídas utilizando-se o conceito de *lugares de fusão* e *transições de substituição*. Lugares de fusão são estruturas que permitem especificar um conjunto de lugares como funcionalmente um único lugar, isto é, se uma ficha é removida ou adicionada de um dos lugares, uma ficha idêntica é adicionada ou removida de todos os

outros lugares pertences ao conjunto. Um conjunto de lugares de fusão é denominado conjunto de fusão (*fusion set*). Uma transição de substituição pode ser vista como uma transição de mais alto nível que se relaciona a uma rede mais complexa e que fornece maiores detalhes das atividades representadas pela transição de substituição. A página que contém a transição de substituição é denominada *superpágina* e a que contém uma visão mais detalhada é denominada *subpágina*. Cada transição de substituição é denominada *supernó* da subpágina correspondente. Uma transição de substituição se relaciona com sua subpágina através da utilização de um tipo de conjunto de fusão de dois membros denominados portas e *sockets*. Estas estruturas descrevem a interface entre a transição de substituição e a subpágina. *Sockets* são atribuídos aos lugares conectados à transição de substituição, e portas são associadas a determinados lugares na subpágina tal que um par *socket*/porta forma um conjunto de fusão. Dessa forma, quando uma ficha é depositada num *socket*, ela aparece também na porta associada aquele *socket*, permitindo assim a conexão entre a superpágina e a subpágina. É sempre possível traduzir uma CPNH para sua correspondente não hierárquica. Para isso, basta substituir cada transição de substituição e arcos conectados, por sua respectiva subpágina "colando" cada *socket* com sua respectiva porta. A definição formal de CPNH pode ser encontrada em [22] . Exemplo do uso de CPNHs pode ser visto em [37].