CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DO PARANÁ

Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial

---

Tese

apresentada ao Centro Federal de Educação Tecnológica do Paraná
como requisito parcial para a obtenção do grau de

DOUTOR EM CIÊNCIAS

por

FABIANO SILVA

REDE DE PLANOS: UMA PROPOSTA PARA A SOLUÇÃO DE PROBLEMAS DE PLANEJAMENTO EM INTELIGÊNCIA ARTIFICIAL USANDO REDES DE PETRI

Orientador:

Prof. Dr. LUIS ALLAN KÜNZLE	CPGEI, CEFET/PR - BRASIL

Co-orientador:

Prof. Dr. MARCOS CASTILHO	DINF, UFPR - BRASIL

Committee members:

Prof. Dr. JOSÉ REINALDO SILVA	POLI, USP - BRASIL
Prof. Dr. SILVIA BOTELHO	EE, FURG - BRASIL
Prof. Dr. ALEXANDRE DIRENE	DINF, UFPR - BRASIL
Prof. Dr. RICARDO LUDERS	CPGEI, CEFET/PR - BRASIL

Fevereiro, 2005

Fabiano Silva

Rede de Planos: Uma Proposta para a Solução de Problemas de Planejamento em Inteligência Artificial usando Redes de Petri

Tese submetida ao Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial do Centro Federal de Educação Tecnológica do Paraná como requisito parcial para a obtenção do grau de Doutor em Ciências.

Orientador: Prof. Dr. Luis Allan Künzle
Co-orientador: Prof. Dr. Marcos Castilho

Curitiba - PR, Brasil
Fevereiro 2005

Agradecimentos

Agradeço aos orientadores e amigos Luis e Marcos pela grande ajuda e por tornarem este trabalho possível; aos membros da banca examinadora pelas contribuições; aos amigos do LSIP-CefetPR e LIC-UFPR pelo trabalho em grupo que tanto enriqueceu esta pesquisa; aos professores Alexandre, André, Jair e Renato da UFPR pelas excelentes discussões técnicas e ao professor Hector Geffner por me receber em sua equipe na UPF durante o estágio em Barcelona.

Agradeço especialmente à minha esposa Letícia, que foi a grande responsável pela conclusão de mais esta etapa em nossas vidas; à toda minha família e à da Letícia pelo suporte e apoio incondicionais e aos amigos Bona, Betinha, Rosa, Alexandre, Sunye, Ligia, André, Laura, Huei, Wu, Shiro, Juan, Habib, Angela, Marcelo, Marquinhos, Andrea, João, Celina, Malga e amigos de Barcelona pelo apoio.

Dedico este trabalho à Leticia.

Sumário

Lista de Figuras

Lista de Tabelas

Resumo

Este documento apresenta uma investigação sobre os relacionamentos entre os problemas de planejamento em inteligência artificial e de alcançabilidade em redes de Petri. O trabalho trata da análise de algumas maneiras de se representar um problema de planejamento como uma rede de Petri e da comparação da rede obtida com o grafo de planos.

São destacadas as principais vantagens e desvantagens do uso das redes de Petri em comparação com o grafo de planos. Procura-se argumentar em favor da primeira, pois ela permite representar de maneira ao mesmo tempo precisa e econômica os mesmos relacionamentos contidos na segunda estrutura. Um dos focos da pesquisa é encontrar a melhor maneira de substituir as redundâncias presentes no grafo de planos pela dinâmica da rede de Petri. Em particular, na rede, consegue-se uma melhor representação para as relações de inconsistência e de exclusão mútua entre ações.

Abstract

This thesis dissertation reports on the investigation of the relationships between the problems of planning, in the sense of Artificial Intelligence, and that of reachability, in the sense of Petri nets. The research approaches different ways to represent a planning problem as a Petri net, as well as the comparison of the given net with the plan graph.

The main advantages and disadvantages in applying Petri nets compared to the plan graph method. We claim that, the use of Petri nets allows more precise and compact representation of action relationships than those obtained with the counter part method. One of the main research aims is to eliminate representational redundancies of the plan graph by projecting them against the dynamic aspects of the net. Examples of the comparative improvements are shown in the text, particularly for the relationships of inconsistency and the mutual exclusion of actions.

1 Introdução

O problema de planejamento em inteligência artificial pode ser definido informalmente como o processo de se estabelecer uma seqüência de ações para transformar o mundo atual em algum outro estado desejado. Apesar de ser um problema de alta complexidade computacional [Byl94, ENS91], o que inviabiliza o tratamento de grandes instâncias, a pesquisa e o desenvolvimento de sistemas computacionais de alto desempenho se justifica pela gama de situações do mundo real que podem ser modeladas e resolvidas como problemas de planejamento.

Suponha o cenário onde alguém tenha que transportar vários objetos entre o seu escritório e a sua casa. No entanto se este agente não sabe qual o conjunto de ações que deve tomar e nem a ordem correta para realizá-las, ele tem em mãos um *problema de planejamento*.

Um *planejador* é o sistema que estabelece o plano que será executado pelo agente. Por exemplo, pegar a chave do carro, colocar os objetos no porta-malas, dirigir do escritório até a casa e finalmente descarregar os objetos em casa. O planejador recebe como entrada um estado inicial do mundo, um objetivo e um conjunto de ações que podem ser aplicadas e gera como saída uma seqüência de ações. Por exemplo, recebe o estado inicial "Estar no escritório com os objetos", o objetivo "Ter os objetos em casa", diversas descrições de ações como por exemplo "dirigir" ou "pegar os objetos". A saída de um planejador é uma seqüência de ações como por exemplo: "pega os objetos", "abre o porta-malas", "coloca os objetos no porta-malas", "fecha o porta-malas", "entra no carro", "dirige até a casa", "sai do carro", "abre o porta-malas" e "descarrega os objetos". Esta seqüência, ou qualquer outra que termine com o objetivo atingido, é chamada de *plano*.

Formalmente, um algoritmo que resolve um problema de planejamento possui três entradas, codificadas em alguma linguagem formal:

1. uma descrição do estado inicial do mundo;

2. uma descrição do objetivo do agente; e

3. uma descrição das possíveis ações a serem executadas pelo agente, chamada de *teoria do domínio*.

A saída de um planejador é uma seqüência de ações que, quando executada em algum mundo satisfazendo a descrição do estado inicial, alcançará o objetivo. Note que esta formulação do problema de planejamento é muito abstrata. De fato, ela realmente especifica uma *classe* de problemas de planejamento parametrizados por linguagens usadas para representar o mundo, objetivos e ações.

Em geral, existem diversas maneiras de se representar um problema de planejamento. Evidentemente a tarefa de escrever um algoritmo de planejamento é mais difícil para linguagens de representação mais expressivas e o tempo de resposta do algoritmo resultante aumenta proporcionalmente. Vale notar que a classe de complexidade do problema e até mesmo sua decidibilidade pode variar de acordo com as características da linguagem de representação utilizada. O fato é que é difícil encontrar um bom compromisso entre o poder de representação e a eficiência do algoritmo.

As primeiras abordagens para o problema de planejamento, que datam dos anos 60, adotam a lógica de primeira ordem como representação. Achar um plano consistia em provar um teorema e resgatar os passos para a obtenção da prova, o que é conhecido como o problema de "explicação de respostas" [Gre69]. Infelizmente esta abordagem tem um sério impedimento, que é a necessidade de se tratar o problema da persistência [MH69], inerente ao tratamento de ações usando lógica. Por ser um problema complexo e implicar em métodos de prova extremamente ineficientes e incompletos, encontrar um formalismo alternativo foi o caminho escolhido pela comunidade desde então.

A primeira solução para o problema que propôs uma estrutura formal alternativa à lógica de primeira ordem foi apresentada por Fikes e Nilsson em 1971: o sistema *STRIPS*[1] [FN71]. Esta solução agregava uma linguagem de representação simples e um algoritmo clássico de busca no espaço de estados do mundo. Em outras palavras, o algoritmo procurava por uma solução utilizando busca em uma árvora onde os nós representavam os estados do mundo e os arcos as ações do plano. Apesar de simples, o algoritmo tratava na prática um número pequeno de problemas devido à explosão combinatória na árvore de busca.

[1] *STanford Research Institute Problem Solver*.

Apesar do algoritmo ineficiente, o sistema formal *STRIPS* permite representar ações e estados de maneira simplificada, fornecendo uma independência entre a linguagem e o algoritmo que resolve o problema. Por este motivo foi adotado como sistema padrão pela comunidade de planejamento e é até hoje a base dos principais planejadores modernos.

Até a primeira metade dos anos 90 os planejadores ainda tratavam o problema com procedimentos clássicos de busca em representações explícitas do espaço de estados ou do espaço de planos. Neste último os nós da árvore de busca são planos parciais e as transições entre os nós são transformações sobre os planos. No espaço de planos a solução do problema é dada pelo nó, enquanto no espaço de estados é dada pelo caminho percorrido na busca. O principal representante das abordagens baseadas no espaço de planos é o planejador *Ucpop* [PW92].

Kaltz e Selman, em 1992, propuseram uma tradução do problema de planejamento representado em *STRIPS* para o cálculo proposicional. O planejador proposto, *Satplan* [KS92, KS96], traduz o problema de planejamento em um problema de satisfabilidade (SAT) e tira proveito dos então recentes métodos para SAT [SLM92, SKC94]. Esta abordagem resultou em um planejador extremamente rápido quando comparado aos trabalhos anteriores.

Três anos depois Blum e Furst apresentaram um novo algoritmo, o *Graphplan* [BF95]. O *Graphplan* inova ao mostrar que, a partir de uma descrição *STRIPS*, pode-se construir um grafo que reduz sensivelmente a representação do espaço de busca do problema. Esta redução se deve ao tratamento de conflitos entre as ações do domínio. O grafo proposto é chamado *grafo de planos*.

A partir do *Satplan* e do *Graphplan*, houve uma grande motivação por novas pesquisas na área. Com o crescente número de novos planejadores e a melhoria de desempenho na solução de problemas, a comunidade decidiu, em 1998, realizar a primeira competição bienal de planejadores [McD98a]. O objetivo da competição era comparar o desempenho dos planejadores. Para tanto foi necessário unificar a linguagem de descrição dos problemas [McD98b].

A idéia de se construir um grafo para reduzir o espaço de busca foi bem aproveitada por Kautz e Selman, que mostraram que este pode ser traduzido para uma instância SAT muito menor que a gerada pelo *Satplan*. O planejador resultante, o *Blackbox* [KS99], usa um algoritmo de duas fases: na primeira o grafo de planos é construído e então traduzido

para uma instância SAT; na segunda fase um resolvedor SAT é aplicado para obter o plano que resolve o problema.

O tratamento dos conflitos entre ações presentes no grafo de planos é semelhante à função heurística proposta por Bonet e Geffner [BG98] no planejador *HSP*. Esta função é usada para guiar um procedimento de busca de subida de encosta neste planejador. Funções heurísticas eficientes para guiar o procedimento de busca do planejador também foram utilizadas no *FF* [HN01] e no *LPG* [GSS03], que obtiveram respectivamente os melhores desempenhos nas competições de 2000 e 2002. Vale notar que estes dois planejadores usam, como base para o cálculo de suas heurísticas, estruturas derivadas do grafo de planos.

A tradução do problema de planejamento também foi usada no sentido de transformá-lo em problemas de satisfação de restrições sobre coleções de variáveis derivadas da descrição do problema. Bockmayr e Dimopoulos [BD98] usaram variáveis inteiras 0-1 de modo similar às abordagens SAT e examinaram o efeito de adicionar restrições redundantes ao problema de programação inteira. Vossen e colegas [VBLN99, VBLN01] discutiram a importância de encontrar a representação correta do problema de planejamento em termos de um problema de programação inteira. Eles mostram várias formulações possíveis e suas vantagens. Kautz e Walser [KW99] trataram da solução dos problemas de programação inteira resultantes da tradução através de algoritmos de busca local. Van Beek e Chen [BC99] propuseram tratar ambas abordagens SAT e programação inteira como um caso particular do clássico problema de satisfação de restrições, mostrando como a sua abordagem é melhor em termos de tempo de processamento e utilização de memória.

A utilização de redes de Petri como ferramenta para resolver problemas de planejamento foi proposta inicialmente por Drummond em 1985 [Dru85] estendendo as redes de Petri procedurais para o tratamento de ciclos e ações condicionais. Murata e Nelson em 1991 apresentaram uma abordagem usando redes predicado-transição para modelar o problema de planejamento [MN91], entretanto sua proposta não resultou na implementação de um planejador eficiente. Em 2000, Meiller e Fabiani [MF00] usaram redes coloridas para modelar o problema.

Ainda em 2000, Silva e colegas [SCK00] propuseram a transformação do grafo de planos em uma rede de Petri acíclica e trataram o problema de planejamento como um problema de determinar uma seqüência de disparos de transições que obtém uma determinada marcação nesta rede. A rede acíclica permite o uso de técnicas de programação inteira na solução do problema de alcançabilidade, se assemelhando às abordagens baseadas em restrições.

O planejador resultante, o *Petriplan*, é semelhante ao *Blackbox* no sentido de usar o grafo de planos como uma fase de pré-processamento do problema de planejamento e em seguida transformá-lo em outro problema. Este trabalho é a base dos modelos e resultados apresentados neste documento.

O desempenho dos planejadores baseados em redes de Petri não é, em geral, melhor do que as outras abordagens; entretanto o uso das redes de Petri ou outras estruturas formais equivalentes constituem um rico cenário para investigação, como por exemplo o uso da lógica linear proposto por Küngas [Kün02].

Em suma, planejamento é uma área de pesquisa extremamente complexa, que integra diversas áreas do conhecimento, indo desde tradicionais problemas de IA, como SAT, CSP e buscas heurísticas, até teoria dos grafos, programação inteira e redes de Petri. A figura 1 mostra o interrelacionamento entre estas áreas e os trabalhos que definiram o relacionamento.

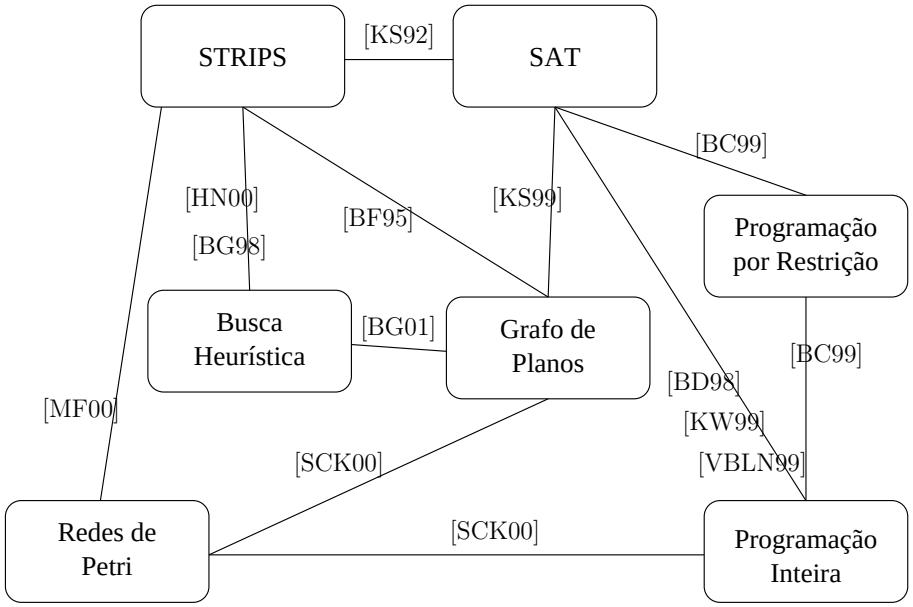


Figura 1: Relacionamento entre áreas do conhecimento aplicadas em planejamento.

Este documento trata dos relacionamentos do problema de planejamento em inteligência artificial com o problema de alcançabilidade em redes de Petri. O problema de planejamento é revisto de forma aprofundada no próximo capítulo. No capítulo 3 são descritas as redes de Petri bem como o problema de alcançabilidade.

No capítulo 4 é resgatado o modelo proposto por Silva e colegas [SCK00] que traduz

o grafo de planos numa rede de Petri. Também são apresentadas algumas simplificações sobre este modelo.

O capítulo 5 apresenta o principal resultado do trabalho, onde as relações de inconsistência entre ações são estendidas e usadas na construção de um modelo em redes de Petri para o problema de planejamento, chamado *rede de planos*. Este modelo explora em maior grau o poder de representação das redes de Petri.

No capítulo final é apresentada a conclusão do trabalho, os trabalhos derivados, e propostas para futuras linhas de pesquisa.

2 Planejamento

Este capítulo apresenta a linguagem *STRIPS*, a complexidade teórica do problema de planejamento baseado nesta linguagem, e o grafo de planos, que é reconhecidamente a melhor estrutura de representação do espaço de busca.

2.1 A representação *STRIPS*

Um dos primeiros sistemas computacionais desenvolvidos para o problema de planejamento foi o *STRIPS* em 1971 [FN71]. Este sistema é composto por um modelo formal para representar as ações e os estados do mundo e um mecanismo de busca no espaço de estados.

A principal característica da representação *STRIPS* é que ela não define apenas uma forma de descrever as ações e os literais de um problema de planejamento, mas também um conjunto de regras de transformação dos estados do mundo. Assim, a representação *STRIPS* pode ser considerada um sistema baseado em regras de produção.

Os sistemas baseados em regras de produção são uma antiga técnica de inteligência artificial que permite descrever transformações em bases de conhecimento de forma simples e com um custo computacional relativamente pequeno quando comparado a sistemas que usam lógica de primeira ordem como estrutura de representação.

A representação *STRIPS*, ou simplesmente *STRIPS*, é derivada do cálculo proposicional e usa uma representação de primeira ordem. A descrição das ações e dos literais do problema é feita a partir de esquemas parametrizados por variáveis que são instanciadas pelo procedimento de busca usando as constantes presentes no problema. Os estados do mundo são representados por conjunções de literais instanciados e as ações instanciadas definem as regras de transformação entre os estados do mundo.

Um problema de planejamento escrito em *STRIPS* é dividido em duas partes: a des-

crição do domínio e a descrição do problema. O domínio contém o conjunto de literais e a descrição das ações que são dadas por:

- *identificação da ação*, que define o nome e os parâmetros da ação;
- *pré-condições*, que é uma conjunção de literais positivos que devem estar presentes na descrição do estado para que a ação possa ser aplicada sobre este; e
- *efeitos*, que é uma conjunção que pode incluir literais positivos ou negativos. Eles descrevem as alterações que devem ser realizadas sobre um estado do mundo, sendo que os literais positivos são incluídos no estado e os negativos removidos.

Os termos *pré-condição* e *efeito* são usados para tratar de apenas um literal das conjunções que representam respectivamente as pré-condições e os efeitos da ação. Quando o literal é positivo chamamos de *pré-condição positiva* ou *efeito positivo*. No caso de um literal negativo temos *pré-condição negativa* ou *efeito negativo*.

Na formulação original de Fikes e Nilsson [FN71] a conjunção efeito é representada por duas listas de literais: a lista de inclusão contendo os literais positivos e a lista de remoção contendo os literais negativos.

A descrição do problema é dada por:

- O conjunto de constantes presentes no problema;
- O estado inicial do mundo, dado por uma conjunção de literais positivos instanciados;
- O objetivo do problema, também dado por uma conjunção de literais positivos instanciados.

Tomando como exemplo o problema de transportar alguns objetos do escritório para casa usando um carro, podemos descrever o estado inicial do problema de "levar os objetos para casa" como:

$$\mathsf{EstarEm(você,\ escritório)} \wedge \mathsf{EstarEm(objetos,\ escritório)} \wedge \mathsf{EstarEm(carro,\ escritório)} \wedge \cdots$$

Considere que o literal $\mathsf{EstarEm}(x, y)$ é o esquema presente na descrição do domínio e foi instanciado para as constantes $\mathsf{você}$, $\mathsf{objetos}$, carro e $\mathsf{escritório}$.

Em *STRIPS* é necessária uma descrição completa dos estados do mundo. Tudo que não é explicitamente descrito é considerado falso, pois se baseia na hipótese do mundo

fechado. Assim considere os "$\cdots$" na descrição dos estado inicial anterior como todos os outros fatos verdadeiros no estado inicial do mundo. Nenhum literal negativo é incluído nas descrições dos estados, pois se um literal não está presente é automaticamente considerado falso. Como conseqüência as ações têm apenas pré-condições positivas. Entretanto o uso de literais negativos como pré-condições, por exemplo $\neg P$, não aumenta a complexidade do tratamento das ações. Basta observar que estas ações só podem ser aplicadas sobre estados que não contenham o literal P.

O objetivo do problema pode ser descrito por:

$$\mathsf{EstarEm}(\mathsf{objetos}, \mathsf{casa})$$

A ação "dirigir" parametrizada por motorista (m), veículo (v), origem (o) e destino (d) pode ser descrita por:

Ação	:	$\mathsf{Dirigir}(m, v, o, d)$
Pré-Condição	:	$\mathsf{EstarEm}(m, o) \wedge \mathsf{EstarEm}(v, o) \wedge \mathsf{SaberDirigir}(m, v)$
Efeito	:	$\mathsf{EstarEm}(m, d) \wedge \mathsf{EstarEm}(v, d) \wedge \neg\mathsf{EstarEm}(m, o) \wedge$ $\neg\mathsf{EstarEm}(v, o)$

Vale notar que esta é uma descrição genérica da ação, que pode ser instanciada de acordo com as constantes presentes no problema, por exemplo "dirigir do escritório até a casa" pode ser dada pela instanciação da ação com as constantes $\mathsf{você}$, carro, $\mathsf{escritório}$ e casa:

Ação	:	$\mathsf{Dirigir}(\mathsf{você}, \mathsf{carro}, \mathsf{escritório}, \mathsf{casa})$
Pré-Condição	:	$\mathsf{EstarEm}(\mathsf{você}, \mathsf{escritório}) \wedge \mathsf{EstarEm}(\mathsf{carro}, \mathsf{escritório}) \wedge$ $\mathsf{SaberDirigir}(\mathsf{você}, \mathsf{carro})$
Efeito	:	$\mathsf{EstarEm}(\mathsf{você}, \mathsf{casa}) \wedge \mathsf{EstarEm}(\mathsf{carro}, \mathsf{casa}) \wedge$ $\neg\mathsf{EstarEm}(\mathsf{você}, \mathsf{escritório}) \wedge \neg\mathsf{EstarEm}(\mathsf{carro}, \mathsf{escritório})$

Esta ação pode ser aplicada sobre qualquer estado que contenha os fatos:

EstarEm(você, escritório), EstarEm(carro, escritório), SaberDirigir(você, carro).

Após a aplicação da ação os literais

EstarEm(você, casa), EstarEm(carro, casa)

são incluídos na descrição do estado e os literais

EstarEm(você, escritório), EstarEm(carro, escritório)

são removidos da descrição do estado. Os literais que estavam no estado antes da aplicação da ação e não foram removidos são copiados para o estado resultante, ou seja, as ações tratam apenas do que muda de um estado para outro.

A entrada para um planejador é dada pela descrição do domínio contendo os literais e as ações e pela descrição do problema composta por constantes, pelo estado inicial e por uma descrição do objetivo. Quando chamado com esta entrada, um planejador deve retornar uma seqüência de ações que transforme o estado inicial em um estado que contenha o objetivo.

O uso de *STRIPS* para descrever problemas de planejamento restringe em muito o número de problemas que podem ser modelados como problemas de planejamento. Por ser uma linguagem baseada no cálculo proposicional as estruturas de representação se limitam à proposições e operadores lógicos clássicos, o que inviabiliza a modelagem de cenários onde é necessário representar tempo ou quantidades numéricas. As principais vantagens deste formalismo são tornar o problema decidível [ENS91] e facilitar a construção de planejadores.

Uma maneira óbvia de se construir um planejador usando *STRIPS* é tratá-lo como um problema de busca no espaço de estados possíveis do mundo. Foi desta maneira que o problema foi originamente considerado e até os anos 90 os planejadores eram variantes mais ou menos sofisticadas de procedimentos clássicos de busca. Entretanto, o problema de planejamento baseado na representação *STRIPS* é em geral PSPACE-Completo, inviabilizando qualquer abordagem exaustiva. Este fato é o principal motivador de pesquisas que tentam reduzir o espaço de busca ou propor novas estratégias para o tratamento do problema.

A próxima seção trata da complexidade teórica do problema de planejamento em

STRIPS.

2.2 Complexidade do problema

As classes de complexidade computacional, em geral, podem ser definidas a partir de linguagens e máquinas de Turing que reconhecem estas linguagens. As características das máquinas, o número de passos e o espaço ocupado na fita da máquina durante o reconhecimento da linguagem definem as diferentes classes de complexidade.

Um problema pertence a uma classe de complexidade quando a linguagem que o descreve é aceita por alguma máquina de Turing que atende as restrições desta classe.

Por exemplo, a classe NP é a classe dos problemas que são aceitos por máquinas de Turing não-determinísticas em um número polinomial de passos em relação ao tamanho da instância do problema.

Um problema P é dito *Completo* em relação a uma classe quando é possível transformar, em um número polinomial de passos, qualquer problema da classe em uma instância de P. Assim P pode ser considerado tão difícil quanto qualquer problema da classe. Por exemplo, SAT é um problema NP-Completo.

A equivalência entre as máquinas de Turing e os modelos de computação atuais permite estabelecer relações entre os problemas e a implementação real de algoritmos para estes problemas. Um algoritmo para um problema NP-Completo usa, em geral, um número exponencial de operações para encontrar uma solução.

Uma descrição mais detalhada das classes de complexidade e das máquinas de Turing pode ser encontrada em [Pap94].

No caso do problema de planejamento, as características da linguagem de representação influenciam diretamente a complexidade computacional do problema, implicando desde tempo constante até EXPTIME-Completo [ENS91, Byl94]. Considerando apenas a representação *STRIPS*, a complexidade do problema de planejamento é PSPACE-Completo, que é a classe dos problemas mais difíceis da classe PSPACE.

A classe PSPACE é a classe dos problemas que são aceitos por uma máquina de Turing determinística usando apenas um número polinomial de células da fita da máquina durante a computação, em relação ao tamanho da entrada. Na prática, os algoritmos para problemas desta classe tratam espaços de estados exponenciais.

A análise da complexidade do problema de planejamento é feita sobre uma versão do problema chamada de problema de decisão, ou seja, o problema de verificar a existência de um plano que resolve o problema. Ao final da computação a máquina de Turing correspondente deve parar em um estado que indique a existência ou não de um plano para o problema.

A seguir é apresentada a prova da complexidade do problema decisão correspondente ao problema de planejamento em *STRIPS*, seguindo os mesmos moldes da prova proposta por Bylander [Byl94]. Inicialmente é mostrado que o problema pertence à classe PSPACE e em seguida mostramos que qualquer problema desta classe pode ser reduzido a um problema de planejamento em um número polinomial de passos.

2.2.1 Planejamento pertence à PSPACE

Para mostrar que o problema de planejamento pertence à PSPACE assumimos a equivalência entre as classes PSPACE e NPSPACE [Sav70] e apresentamos um algoritmo não-determinístico que verifica a existência de um plano dado o conjunto A de ações, o conjunto F de literais, o estado inicial s_0 e o objetivo g do problema:

ExistePlano(A, F, s_0, g)
1 $m \leftarrow |F|$;
2 $k \leftarrow 0$;
3 $s \leftarrow s_0$;
4 repita
5 $k \leftarrow k + 1$;
6 escolhe uma ação $a \in A$;
7 $s \leftarrow next(s, a)$;
8 até que $g \subseteq s$ ou $k = 2^m - 1$;
9 se $g \subseteq s$ então o plano é válido.

O algoritmo escolhe não-deterministicamente, na linha 6, k ações que levam do estado inicial do problema s_0 até um estado que contenha o objetivo g. Na linha 7 a ação escolhida é aplicada sobre o estado atual s do mundo e o estado resultante passa a ser o novo estado atual. O algoritmo termina quando algum estado que contenha o objetivo g do problema é alcançado ou quando todos os $2^m - 1$ estados possíveis do mundo são visitados pelo algoritmo, onde m é o número de literais do problema.

O espaço utilizado pelo algoritmo corresponde ao estado atual do mundo, o estado anterior e o contador de ações do plano k. Como o espaço necessário para guardar um estado é no máximo m, temos que o algoritmo ocupa no máximo $2m+1$, que é polinomial em relação ao tamanho da entrada. Logo ExistePlano $\in$ NPSPACE, que é a classe dos problemas aceitos por uma máquina de Turing não-determinística usando espaço polinomial.

Dado que NPSPACE = PSPACE [Sav70] temos que ExistePlano $\in$ PSPACE.

2.2.2 Planejamento é PSPACE-Difícil

Dado que o problema de planejamento pertence à classe PSPACE, conforme apresentado na seção anterior, mostrar que o problema é PSPACE-Completo consiste de provar que ele é PSPACE-Difícil. A prova apresentada a seguir segue a mesma linha de [Byl94].

Um problema PSPACE-Difícil pode ser entendido como um problema tão difícil quanto qualquer problema da classe PSPACE, em outras palavras, existe uma transformação em tempo polinomial de qualquer problema da classe para o problema PSPACE-Difícil. Entretanto mostrar a transformação de todos os problemas da classe para o problema de planejamento não é um caminho viável.

A opção é mostrar que qualquer máquina de Turing PSPACE pode ser transformada em um problema de planejamento em tempo polinomial e que uma solução para o problema de planejamento indica que a máquina aceita a entrada. Desta forma qualquer problema que é aceito por uma máquina de Turing em espaço polinomial pode ser mapeado para um problema de planejamento em tempo polinomial através da transformação da sua máquina de Turing em um problema de planejamento.

Considere uma máquina de Turing determinística dada por:

- um conjunto de estados: $S = \{q_1, \cdots, q_n\}$, sendo que $yes \in S$ e é um estado de aceitação;
- um alfabeto da entrada: $\Sigma = \{\sigma_1, \cdots, \sigma_{m-1}\}$;
- um alfabeto da fita: $\Gamma = \Sigma \cup \{\#\}$, sendo # o símbolo que indica que uma determinada posição da fita está vazia. Escrevemos $\Gamma = \{\gamma_1, \cdots, \gamma_m\}$;
- uma função parcial de transição: $\delta(q, \gamma) = \langle\, q', \gamma', \phi \,\rangle$ onde ϕ é a direção do movimento da cabeça da máquina dado por $\leftarrow$ ou $\rightarrow$;

- uma cabeça de leitura da fita posicionada na primeira posição ocupada pela entrada na fita;

- uma fita t com k posições, sendo l posições são ocupadas pela entrada da máquina as demais posições ocupadas pelo símbolo #, indicando que estas posições estão vazias e serão utilizadas durante o processamento.

Dado que a máquina em questão ocupa um espaço polinomial durante o processamento, temos que o número máximo de células utilizadas na fita t durante o processamento é $k = P(l)$, onde P é um polinômio sobre o tamanho da entrada l.

O primeiro passo na construção de um problema de planejamento a partir da máquina de Turing é a definição de algumas proposições para representar a estrutura da máquina:

- $tape(\gamma, i)$: indica que o símbolo na posição i da fita é γ;

- $head(i)$: indica que a cabeça da máquina está na posição i da fita;

- $state(q)$: indica que o estado da máquina é q;

- $accept$: indica que a máquina está em um estado de aceitação yes.

O estado inicial do problema de planejamento é dado pelo estado inicial da máquina e pode ser definido por:

$$\begin{aligned} s_0 \quad = \quad & state(q_0) \wedge head(0) \wedge tape(x_1, 0) \wedge \cdots \wedge tape(x_l, l-1) \wedge \\ & tape(\#, l) \wedge \cdots \wedge tape(\#, k-1) \end{aligned}$$

onde # é um símbolo especial que indica que a posição da fita não está acupada. O estado inicial do problema codifica o estado inicial da máquina e o conteúdo da fita, onde x_i são os símbolos da entrada.

O objetivo do problema de planejamento é dado por:

$$g = accept$$

este objetivo só é alcançado quando a máquina chega em um estado de aceitação.

As ações do problema são separadas em três grupos:

- As ações que representam as transições $\delta(q, \gamma) = \langle\ q', \gamma', \leftarrow\ \rangle$ que movem a cabeça da máquina para a esquerda, dadas por $\alpha_{\leftarrow}(i, q, \gamma)$ com a pré-condição:

$$state(q) \wedge head(i) \wedge tape(\gamma, i)$$

e o efeito:

$$state(q') \wedge \neg state(q)\ \wedge$$
$$tape(\gamma', i) \wedge \neg tape(\gamma, i)\ \wedge$$
$$head(i-1) \wedge \neg head(i)$$

Vale notar que esta ação não é instanciada para $i < 1$.

- As ações que representam as transições $\delta(q, \gamma) = \langle\ q', \gamma', \rightarrow\ \rangle$ que movem a cabeça da máquina para a direita, dadas por $\alpha_{\rightarrow}(i, q, \gamma)$ com a pré-condição:

$$state(q) \wedge head(i) \wedge tape(\gamma, i)$$

e o efeito:

$$state(q') \wedge \neg state(q)\ \wedge$$
$$tape(\gamma', i) \wedge \neg tape(\gamma, i)\ \wedge$$
$$head(i+1) \wedge \neg head(i)$$

vale notar que esta ação não é instanciada para $i > (k-2)$.

- A ação que indica que a máquina chegou a um estado de aceitação, dada por $\alpha_{ac}(i, yes, \gamma)$ com a pré-condição:

$$state(yes) \wedge head(i) \wedge tape(\gamma, i)$$

e o efeito:

$$accept$$

Assim temos a máquina de Turing de espaço polinomial transformada em um problema de planejamento. Esta transformação é polinomial pois:

- o tamanho do estado inicial do problema é $k + 2$, onde k dado por um polinômio sobre l e,

- o número de ações é polinomial em relação à entrada pois o número de combinações (i, q, x) é dado pela função de transição da máquina de Turing $\delta(q, x)$ aplicada sobre as k posições da fita.

Portanto qualquer máquina de Turing PSPACE com sua entrada pode ser reduzida, em um número polinomial de passos, para uma instância do problema de planejamento usando a representação *STRIPS*. Logo o problema de planejamento usando *STRIPS* é PSPACE-Difícil.

Dado que o problema é PSPACE-Completo e que o espaço de estados é invariavelmente intratável para problemas relevantes, torna-se necessário o uso de alguma estrutura eficiente para representar este espaço. A próxima seção trata da estrutura que melhor representa o espaço de busca para o problema, o grafo de planos.

2.3 Grafo de planos

O tratamento do problema de planejamento como um problema de busca no espaço de estados pode ser visto como uma busca em uma árvore, onde os nós são estados do mundo e as arestas são ações que fazem a transição entre os estados. Neste modelo a solução do problema é dada pelo caminho na árvore que leva desde a raiz até algum estado que contém o objetivo do problema, ou seja, é uma seqüência completamente ordenada de ações.

Restringir a solução do problema a planos completamente ordenados limita a qualidade da solução em domínios que permitem que ações sejam executadas em paralelo. Nestes domínios as soluções que ordenam parcialmente as ações do plano são, em geral, melhores que aquelas que ordenam completamente. Entretanto, o uso de uma árvore de busca no espaço de estado torna extremamente complexa a obtenção de soluções parcialmente ordenadas para o problema.

Outro ponto problemático deste modelo são as informações redundantes mantidas em nós semelhantes da árvore de busca. O armazenamento destas redundâncias limita em muito o tamanho dos problemas que podem ser tratados por um planejador. É facil notar que estados filhos de um mesmo estado na árvore compartilham grande parte de suas descrições, diferenciando apenas no que diz respeito aos efeitos das ações que os geraram. A figura 2 mostra um fragmento de uma árvore de estados, onde os nós apresentam redundâncias.

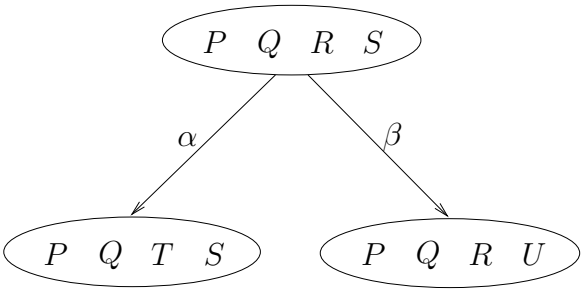


Figura 2: Árvore de estados onde o estado inicial é dado por $P \wedge Q \wedge R \wedge S$. A ação α que tem como pré-condição $P \wedge R$ e como efeito $T \wedge \neg R$. A ação β que tem como pré-condição $Q \wedge R \wedge S$ e como efeito $U \wedge \neg S$. Note que devido ao uso da representação *STRIPS* as proposições negativas podem ser suprimidas dos estados.

Uma estrutura de representação do espaço de estados que permite a obtenção de soluções parcialmente ordenadas para o problema e que reduz significativamente as redundâncias presentes na árvore de busca foi proposta por Blum e Furst em 1995 [BF95]: o *grafo de planos*. A figura 3 mostra o grafo de planos correspondente à árvore da figura 2.

O grafo de planos é um grafo dirigido constituído de dois tipos de vértices, os que representam as proposições e os que representam as ações de um problema de planejamento. Estes vértices são distribuídos em camadas de proposições e de ações intercaladas. Vale notar que a inclusão ou não das proposições negativas no grafo não afeta a representação do problema.

A construção do grafo pode ser vista como uma simplificação do caminhamento em largura na árvore de busca. As camadas de proposições representam a união dos estados da árvore até a profundidade correspondente no grafo. Por exemplo, a terceira camada de proposições do grafo contém todas as proposições dos estados da árvore que têm profundidade menor que 3. As camadas de ações contém todas as ações da árvore até a profundidade correspondente. Uma camada de ações do grafo pode ser vista como uma transição entre conjuntos de estados do mundo.

As arestas do grafo são de três tipos: as que ligam proposições a ações, indicando as pré-condições da ação; as que ligam ações a proposições, indicando os efeitos da ação e as arestas não dirigidas que ligam duas ações ou duas proposições e indicam um conflito entre elas.

O grafo inicia com uma camada de proposições representando o estado inicial do problema. As camadas do grafo são numeradas iniciando em 0. As camadas pares contêm

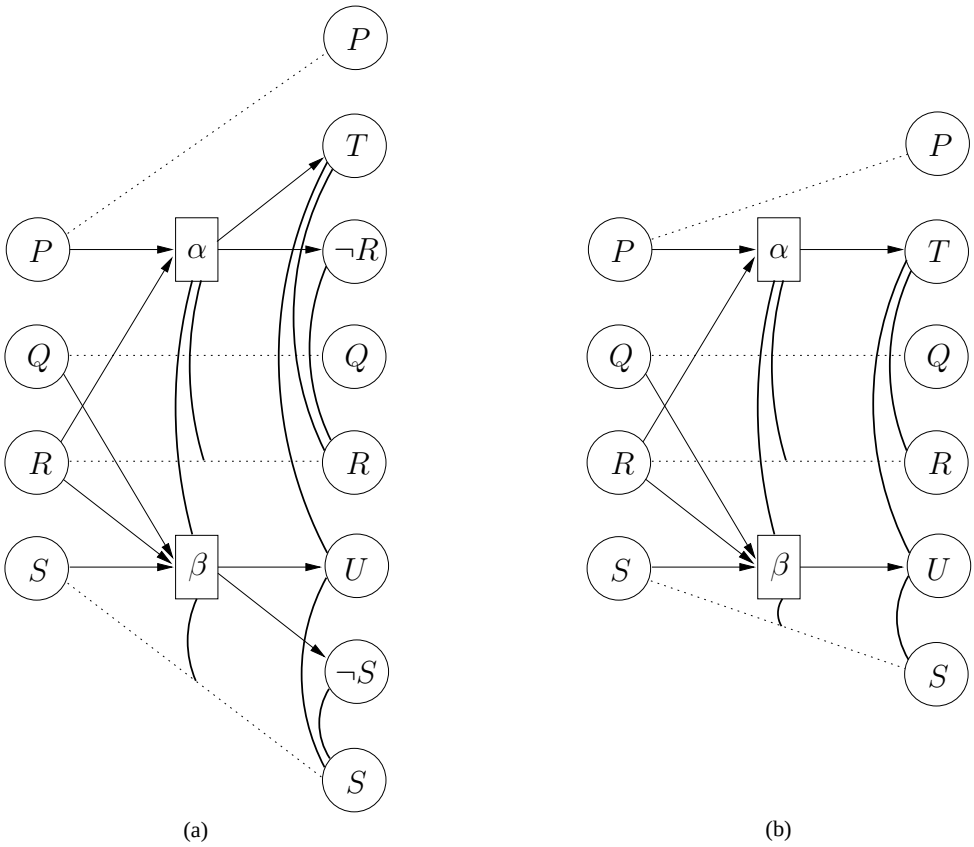


Figura 3: (a) Grafo de plano correspondente à árvore de estados da figura 2. Os círculos representam as proposições e os retângulos representam as ações. As arestas dirigidas indicam as pré-condições e os efeitos das ações. Os arcos não-dirigidos representam as relações de exclusão mútua entre as ações e entre as proposições. As linhas pontilhadas representam ações de manutenção. (b) O mesmo grafo sem a representação das proposições negativas.

os *nós proposições* que representam as proposição para o problema a ser resolvido. As camadas ímpares contêm os *nós ações*, que são as instâncias de ações cujas pré-condições são satisfeitas pelas proposições da camada anterior. Para facilitar a distinção entre proposições e ações, utilizaremos círculos para representar os nós proposições e retângulos para representar os nós ações.

As arestas do grafo conectam os nós proposições aos nós ações da camada posterior, ligando as pré-condições das ações às proposições correspondentes. Da mesma forma, as arestas ligam os efeitos de ações de uma camada às proposições correspondentes da camada seguinte.

Uma ação especial é incluída no conjunto de ações do problema, chamada de *ação de manutenção*. Esta ação tem como pré-condição uma proposição qualquer e como efeito

a mesma proposição, ou seja, ela copia uma proposição da camada atual para a próxima camada de proposições do grafo. As linhas pontilhadas da figura 3 representam este tipo de ação.

Para cada proposição existente na camada i é aplicada a referida ação de manutenção ocasionando novamente a existência dessas proposições na camada $i + 2$. Esse processo garante que uma ação executada na camada i seja novamente válida na próxima camada, ou seja, é possível adiar a execução de uma ação para um próximo instante de tempo se consideramos as camadas de ações do grafo como uma seqüência temporal.

O procedimento de construção do grafo é iniciado com uma camada de proposições, que representa o estado inicial do problema. A *expansão* do grafo é o processo de incluir uma nova camada de ações seguida de um nova camada de proposições. Na primeira expansão, as ações que possuem todas as suas pré-condições disponíveis na camada anterior são adicionadas no grafo. Após a camada de ações, uma nova camada de proposições é incluída, esta contém as proposições efeito das ações que foram adicionadas.

A expansão ocorre até que uma camada de proposições contenha as proposições do objetivo do problema ou até que uma camada de proposições obtida seja idêntica à camada de proposições anterior. Este último caso é a condição de parada do processo de expansão, indicando que o problema não tem solução.

A obtenção de uma camada com as proposições do objetivo indica que pode existir um plano parcialmente ordenado que é solução para o problema. Entretanto as camadas do grafo representam conjuntos de estados do mundo e conjunto de ações que podem conter inconsistências entre si, impossibilitando a obtenção de uma solução. Assim é necessário marcar estas inconsistências com uma relação de exclusão mútua entre as estruturas. Estas inconsistências são indicadas pelos arcos não dirigidas do grafo.

A relação de exclusão mútua entre ações *mutex* é representada por uma aresta que liga ações presentes em uma mesma camada. Esta relação é definida como segue:

Definição: Duas instâncias de ações a e b numa camada de ações i são mutuamente exclusivas se:

- um efeito de uma ação é a negação de um dos efeitos da outra (figura 4);
- um efeito de uma ação é a negação de uma pré-condição de outra (figura 5);
- as ações a e b têm pré-condições que são mutuamente exclusivas na camada $i-1$

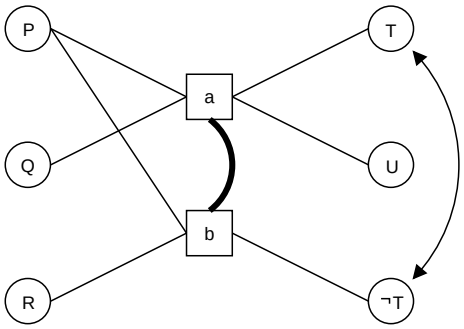


Figura 4: A ação b tem como efeito a proposição $\neg T$ que é a negação de um efeito da ação a, portanto as ações a e b são mutuamente exclusivas na camada.

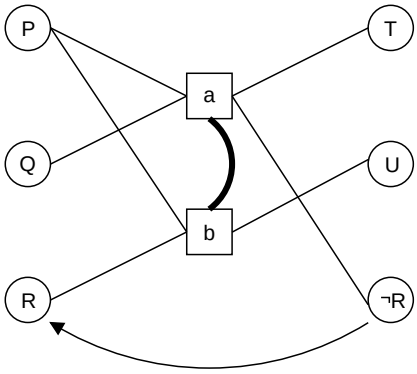


Figura 5: A ação a tem como efeito a proposição $\neg R$ que é a negação de uma pré-condição da ação b, portanto as ações a e b são mutuamente exclusivas.

(figura 6). Este caso é chamado de *competição de necessidades*.

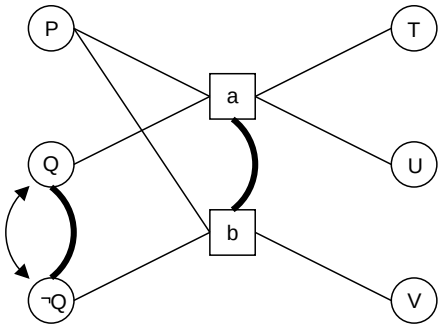


Figura 6: As ações a e b têm pré-condições que são mutuamente exclusivas na camada anterior Q e $\neg Q$, portanto também são mutuamente exclusivas.

A relação de exclusão mútua também é definida para as proposições. Ela indica que as proposições assim relacionadas não podem ser obtidas simultaneamente na mesma camada,

ou seja, somente uma delas pode ser utilizada. Essa relação é apresentada também como uma aresta que liga duas proposições em uma mesma camada, definida como segue:

Definição: Duas proposições P e Q numa camada de proposições i são mutuamente exclusivas se todas as maneiras de obter as proposições são mutuamente exclusivas entre si, chamado *suporte inconsistente*. Ou seja, as ações da camada $i-1$ que têm como efeito estas proposições são duas a duas mutuamente exclusivas (figura 7).

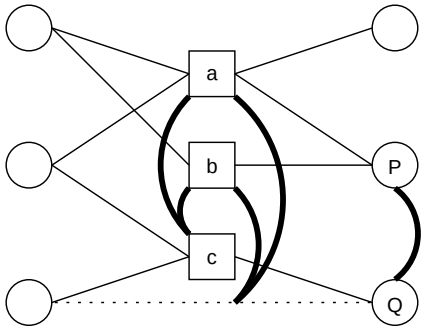


Figura 7: As proposições P e Q são mutuamente exclusivas devido às exclusões mútuas entre as ações que obtêm P (a e b) e as que obtêm Q (c e a ação de manutenção indicada pela linha pontilhada).

Observação: A definição de relação de exclusão mútua é recursiva, como no terceiro caso de exclusão entre ações: "as ações têm pré-condições que são mutuamente exclusivas na camada $i-1$". Portanto uma exclusão mútua numa determinada camada pode ter sido gerada por ações ou proposições de uma camada anterior. Um novo processo de expansão se faz necessário e é aqui que se nota a importância das ações de manutenção.

A figura 8 mostra o grafo de planos para o problema de transportar um pacote (pacote) entre duas lojas (loja_1 e loja_2) usando um caminhão (caminhão), uma versão simplificada do clássico problema de logística. As ações do problema são:

- $\mathsf{Dirigir}(v, l_1, l_2)$: que leva o veículo v da localidade l_1 para l_2 e tem como pré-condição $\mathsf{EstarEm}(v, l_1)$ e como efeito $\mathsf{EstarEm}(v, l_2) \wedge \neg\mathsf{EstarEm}(v, l_1)$;

- $\mathsf{Carregar}(p, v, l)$: que carrega o pacote p no veículo v na localidade l e tem como pré-condição $\mathsf{EstarEm}(p, l) \wedge \mathsf{EstarEm}(v, l)$ e como efeito $\mathsf{Dentro}(p, v) \wedge \neg\mathsf{EstarEm}(p, l)$;

- $\mathsf{Descarregar}(p, v, l)$: que descarrega o pacote p do veículo v na localidade l e tem como pré-condição $\mathsf{Dentro}(p, v) \wedge \mathsf{EstarEm}(v, l)$ e como efeito $\mathsf{EstarEm}(p, l) \wedge \neg\mathsf{Dentro}(p, v)$.

O estado inicial do problema é dado por:

$$\mathsf{EstarEm}(\mathsf{caminhão}, \mathsf{loja}_2) \wedge \mathsf{EstarEm}(\mathsf{pacote}, \mathsf{loja}_1)$$

e o objetivo por: $\mathsf{EstarEm}(\mathsf{pacote}, \mathsf{loja}_2)$.

A busca pela solução no grafo de planos consiste de encontrar em cada camada de ações do grafo um conjunto de ações que não apresentem mutex entre si. Sendo que estas ações devem constituir um fluxo neste grafo que inicia na primeira camada do grafo e leva até as proposições do objetivo na última camada. Este fluxo é um plano parcialmente ordenado para o problema. Caso não exista tal fluxo uma nova expansão acontece e em seguida um nova busca, até que um solução seja obtida ou a condição de parada seja alcançada. A figura 9 mostra a solução encontrada no grafo da figura 8.

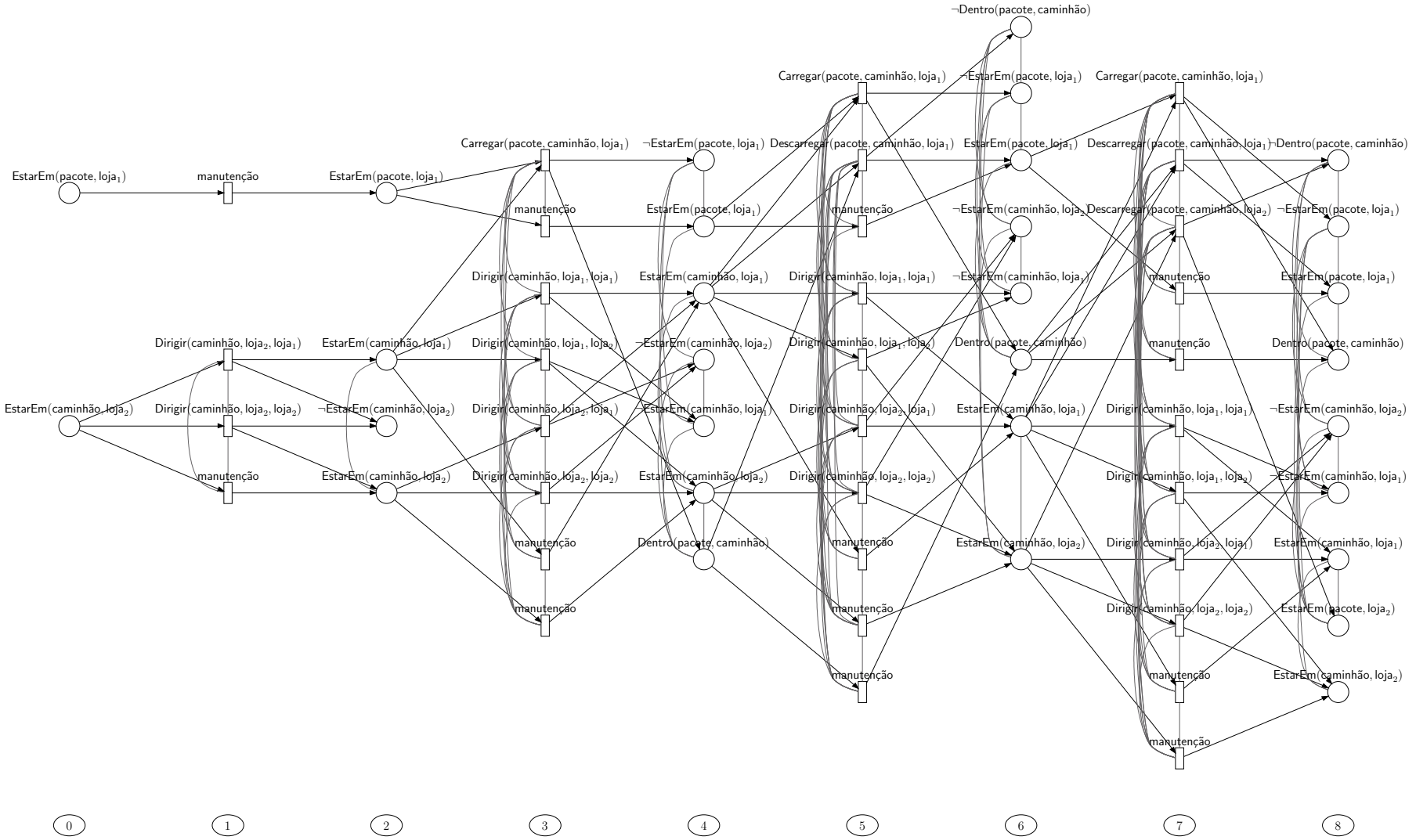


Figura 8: O grafo de planos para o problema do transporte do pacote.

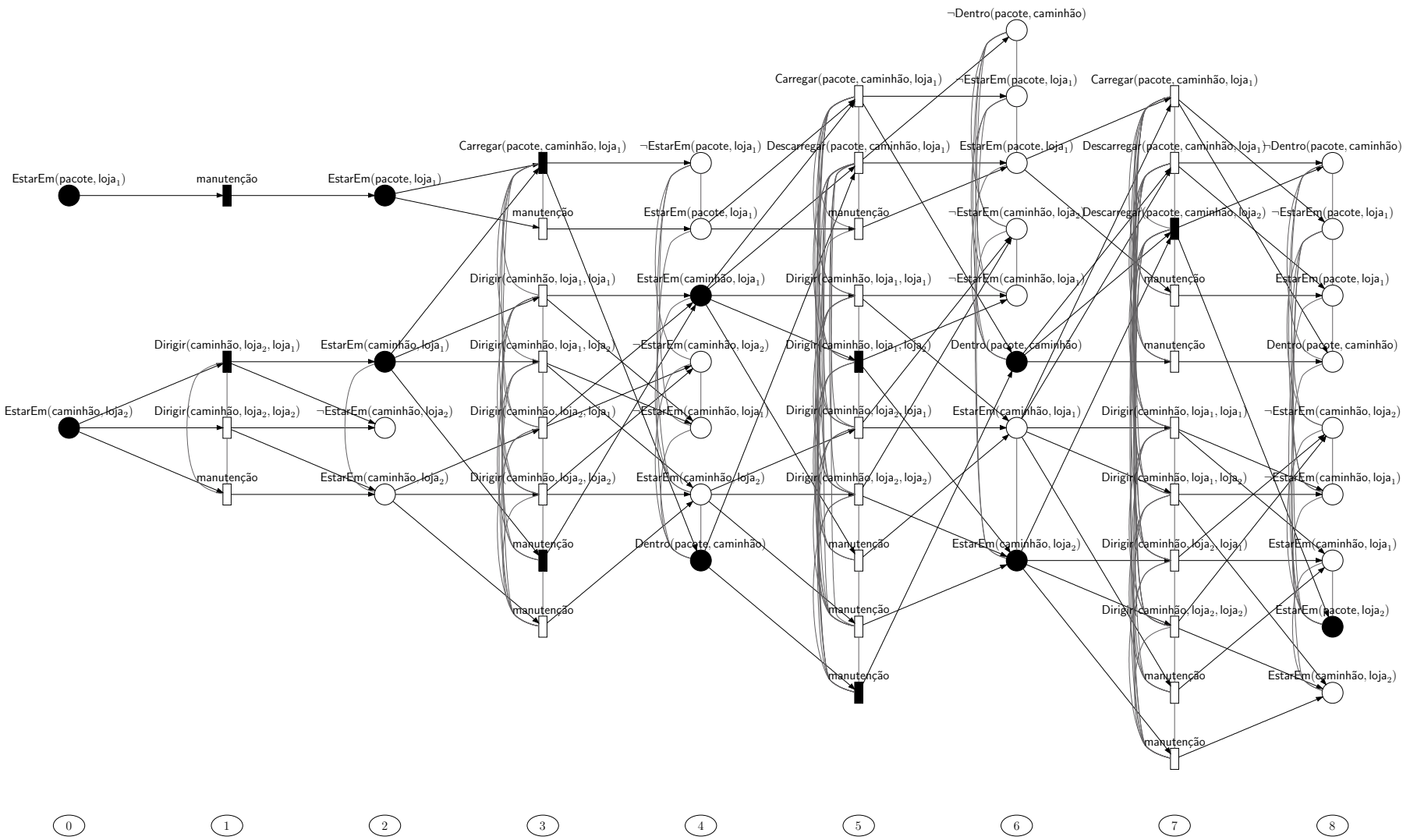


Figura 9: A solução para o problema do transporte do pacote.

2.4 Considerações

Este capítulo tratou da representação *STRIPS* para problemas de planejamento, da complexidade do problema e do grafo de planos como estrutura de representação do espaço de busca do problema.

A representação *STRIPS* é um modelo formal para o tratamento de transformações nos estados do mundo a partir de ações. A capacidade de representação do modelo se limita a uma linguagem derivada do cálculo proposicional, o que reduz o número de problemas que podem ser modelados como problemas de planejamento. Entretanto esta representação também restringe a complexidade computacional do problema, facilitando o desenvolvimento de planejadores.

Em geral, a principal questão no desenvolvimento de um planejador é como representar o espaço de busca do problema, que é invariavelmente exponencial. Como o grafo de planos representa vários estados do espaço de busca em uma mesma camada e, de forma similar, representa várias ações acontecendo concorrentemente em uma mesma camada, torna-se uma representação eficiente do espaço de busca. Esse foi um grande incentivo para o desenvolvimento de vários planejadores que utilizaram o grafo de planos como representação do espaço de busca.

3 Redes de Petri

As redes de Petri se constituem em uma ferramenta formal que é particularmente utilizada para representar paralelismo, concorrência, conflito e relações causais em sistemas dinâmicos de eventos discretos.

Este capítulo possui diferentes objetivos relacionados às redes de Petri. Inicialmente será apresentado o formalismo, sua representação gráfica e sua descrição matricial. Serão descritos também os principais conceitos utilizados ao longo do trabalho. Em seguida o conceito da alcançabilidade entre marcações será analisado com profundidade, com maior ênfase à demonstração da classe de complexidade computacional de redes acíclicas e redes k-limitadas. Finalmente, estes resultados serão utilizados no estudo da equivalência entre problemas de planejamento representados em *STRIPS* e problemas de alcançabilidade em redes k-limitadas.

3.1 Representação

Segundo [CV97], as redes de Petri podem ser definidas como coleções de matrizes e vetores de números naturais associados a procedimentos dinâmicos de atualização.

Uma rede de Petri é uma quádrupla:

$$R = (L, T, I_{pre}, I_{pos}),$$

onde:

$L = \{l_1, l_2, \ldots, l_n\}$	é um conjunto finito de lugares;
$T = \{t_1, t_2, \ldots, t_m\}$	é um conjunto finito de transições;
$I_{pre} : L \times T \to \mathbb{N}$	é a função de incidência de entrada;
$I_{pos} : L \times T \to \mathbb{N}$	é a função de incidência de saída.

A definição apresentada contempla a estrutura estática do modelo. As matrizes I_{pre} e I_{pos} definem esta estrutura estática ligando os lugares e as transições da rede. Um lugar pode ser visto como uma estrutura que mantém um fragmento do estado do sistema modelado. Uma transição pode ser vista como uma regra de produção ou como uma relação de causalidade entre os lugares da rede.

O estado do sistema modelado é dado pela associação de uma quantidade inteira a cada lugar da rede, esta associação é chamada *marcação*. A marcação da rede define o estado corrente do sistema modelado.

Uma rede de Petri com uma dada marcação inicial é denotada por (R, M_0), onde:

$$M_0 : L \rightarrow \mathbb{N} \quad \text{é a marcação inicial.}$$

Podemos visualizar uma rede de Petri usando uma representação gráfica, onde círculos representam lugares e retângulos representam transições. Pequenos círculos pretos no interior dos lugares representam a marcação da rede, chamados de marcas. Os arcos orientados representam as funções de incidência de entrada e saída de uma transição e os números neles rotulados, denominados pesos destes arcos, são os valores das funções de incidência. Assim, na figura 10(a) temos três lugares (a, b e c), duas transições (x e y), quatro arcos com peso um e um arco com peso três, uma marca no lugar a e duas no lugar b.

Lugares são os nós descrevendo os estados (um lugar é uma representação parcial de um estado) e transições descrevem possíveis mudanças parciais de estado. Por exemplo, ao dispararmos a transição x da rede representada pela figura 10(a), uma marca é removida do lugar a e outra do lugar b, e três são inseridas no lugar c. Estas quantidades são dadas pelo valor dos arcos associados à transição x. A marcação resultante é dada pela figura 10(b). A definição formal do disparo de uma transição e das transformações ocorridas na marcação da rede serão apresentadas a seguir.

A função de incidência I_{pre} descreve arcos orientados que ligam lugares a transições. Ela representa, para cada transição t, o fragmento do estado em que o sistema deve estar antes da mudança de estado correspondente à ocorrência de t. $I_{pre}(l, t)$ é o peso do arco (l, t). $I_{pre}(l, t) = 0$ representa a ausência de um arco entre o lugar l e a transição t.

A função de incidência I_{pos} descreve arcos orientados que ligam transições a lugares. Ela representa, para cada transição t, o fragmento do estado em que o sistema deve estar após a mudança de estado correspondente à ocorrência de t. $I_{pos}(l, t)$ é o peso do arco (t, l).

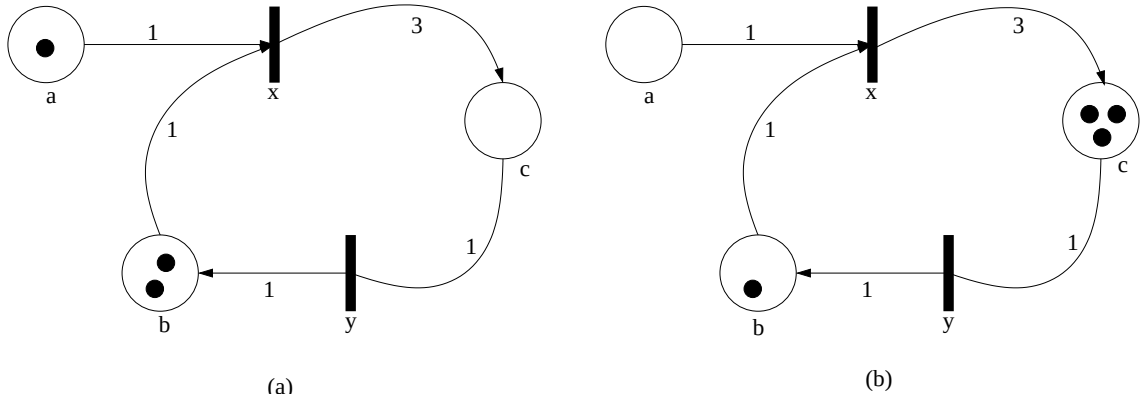


Figura 10: Representação gráfica de uma rede de Petri.

$I_{pos}(l,t) = 0$ representa a ausência de um arco entre a transição t e o lugar l.

O vetor $I_{pre}(.,t)$ denota todos os arcos de entrada da transição t com seus pesos. O vetor $I_{pos}(.,t)$ denota todos os arcos de saída da transição t com seus pesos.

Na representação matricial de uma rede de Petri, I_{pre} e I_{pos} são matrizes de n linhas (os lugares) e m colunas (as transições) e seus elementos pertencem a $\mathbb{N}$.

3.2 Dinâmica da rede

A dinâmica da rede de Petri é dada pelo *disparo* das transições habilitadas, cuja ocorrência corresponde a uma mudança de estado do sistema modelado pela rede. Uma transição t de uma rede de Petri R está *habilitada* para uma marcação M se e somente se:

$$M \geq I_{pre}(.,t).$$

Esta condição de habilitação, expressa sob a forma de uma desigualdade entre dois vetores, é equivalente a:

$$\forall\ l \in L,\ M(l) \geq I_{pre}(l,t).$$

Na marcação da figura 10(a) apenas a transição x está habilitada e na figura 10(b) apenas a transição y. Apenas transições habilitadas podem ser disparadas. Se M é uma marcação de R habilitando uma transição t, e M' a marcação derivada pelo disparo de t a partir de M, então:

$$M' = M + I_{pos}(.,t) - I_{pre}(.,t).$$

Note que o disparo da transição t a partir da marcação M deriva a marcação M', repre-

sentada por:

$$M' : M \xrightarrow{t} M'.$$

Podemos generalizar esta fórmula para calcular uma nova marcação após o disparo de uma seqüência s de transições. Vamos considerar a matriz C dada por:

$$C = I_{pos} - I_{pre},$$

chamada *matriz de incidência*, e o vetor $\overline{s}$, chamado *vetor característico* do disparo de uma seqüência s, dado por $\overline{s} : T \to \mathbb{N}$, tal que $\overline{s}(t)$ é o número de vezes que a transição t aparece na seqüência s. O número de transições em T define a dimensão do vetor $\overline{s}$. Então, uma nova marcação M_g de uma marcação M, após o disparo da seqüência s de transições, é dada por:

$$M_g = M + C.\overline{s}. \tag{3.1}$$

Esta equação é chamada *equação fundamental* de R.

Podemos usar a equação fundamental para determinar um vetor $\overline{s}$ para uma dada rede R e duas marcações M e M_g. A solução que satisfaz a equação deve ser um vetor inteiro não-negativo. Entretanto, sua existência é apenas uma condição necessária para que M_g seja alcançada a partir de M, uma vez que é possível obter vetores $\overline{s}$ que não correspondam a seqüências factíveis de disparos na rede.

3.3 Alcançabilidade

O disparo de uma transição t sobre uma marcação M obtendo uma marcação M' define uma relação entre as marcações M e M'. Esta relação entre as marcações da rede é chamada de *relação de alcançabilidade*, M' é *alcançável* a partir de M.

A relação de alcançabilidade entre marcações de uma transição disparável pode ser estendida, por transitividade, para a alcançabilidade do disparo de uma seqüência de transições. Assim, em uma rede de Petri R, é dito que a marcação M_g é alcançável a partir da marcação M se e somente se existe uma seqüência de transições s tal que:

$$M \xrightarrow{s} M_g.$$

O *conjunto alcançabilidade* de uma rede de Petri marcada (R, M_0) é o conjunto $\mathcal{R}(R, M_0)$

tal que

$$(M \in \mathcal{R}(R, M_0)) \Leftrightarrow (\exists s \; M_0 \xrightarrow{s} M).$$

A partir da relação de alcançabilidade entre as marcações da rede podemos definir o *problema de alcançabilidade* para redes de Petri como o problema de verificar se uma dada marcação M_g é alcançável a partir de uma marcação inicial M_0, ou seja, se $M_g \in \mathcal{R}(R, M_0)$.

Uma *sub-marcação* M_s de uma rede $R = (L, T, I_{pre}, I_{pos})$ é dada pela função

$$M_s : L_s \rightarrow \mathbb{N}$$

onde $L_s \subset L$.

O *problema de alcançabilidade de sub-marcação* consiste em verificar se uma determinada sub-marcação M_s é alcançável a partir da marcação inicial M_0. Este problema pode ser definido a partir do conjunto alcançabilidade da rede como: $M_g \in \mathcal{R}(R, M_0)$, onde $M_s \subset M_g$. Ou seja, é o problema de verificar se alguma marcação que contenha a sub-marcação desejada é alcançável a partir da marcação inicial da rede. Vale notar que o problema de alcançabilidade de sub-marcação é teoricamente equivalente ao problema de alcançabilidade [Hac76].

O problema de alcançabilidade é um problema decidível usando espaço exponencial [Lip76, ST77] e sua complexidade computacional, no caso geral, permanece aberta. Entretanto, para algumas classes de redes de Petri a complexidade do problema é conhecida. Esparza e Nielsen apresentam em [EN94] um apanhado de alguns resultados teóricos para o problema. Em [Rau90], Rauhamaa apresenta um análise comparativa de várias técnicas para o tratamento do problema de alcançabilidade.

Nas próximas seções apresentamos o problema de alcançabilidade para duas classes de redes de Petri, as redes acíclicas e as redes limitadas.

3.4 Alcançabilidade em redes acíclicas

Para redes de Petri acíclicas, ou seja, redes cuja estrutura não possui ciclos dirigidos, o problema de alcançabilidade é NP-Completo [Ste92]. O problema pode ser visto como um problema de programação inteira, reconhecidamente um problema desta mesma classe de complexidade [Pap81].

O problema de alcançar uma marcação M_g a partir da marcação inicial M_0 em uma

rede $R = (L, T, I_{pre}, I_{pos})$ pode ser mapeado para o problema de encontrar um vetor inteiro não-negativo $\overline{s}$ que atende à restrição dada pela equação:

$$M_g = M + (I_{pos} - I_{pre}) \ . \ \overline{s}$$

A marcação M_g é alcançável se e somente se existe algum vetor $\overline{s} > 0$.

Esta modelagem como um problema de programação inteira é possível devido ao fato da equação fundamental das redes de Petri 3.1 ser uma condição necessária e suficiente para a alcançabilidade em redes acíclicas [Mur89].

3.5 Alcançabilidade em redes limitadas

Uma rede de Petri R com uma marcação inicial M_0 é dita *limitada* se o conjunto de marcações alcançáveis $\mathcal{R}(R, M_0)$ é finito. Como conseqüência temos que o número de marcas em cada lugar da rede é finito, ou seja, é limitado por algum número natural.

O problema de alcançabilidade para as redes limitadas é PSPACE-Completo [How91], a mesma classe de complexidade do problema de planejamento em *STRIPS*. Esta complexidade será demonstrada a seguir em duas partes. Inicialmente mostramos que o problema pertence à classe PSPACE e em seguida mostramos que qualquer problema desta classe pode ser transformado em um problema de alcançabilidade em uma rede limitada. As próximas duas seções apresentam as duas partes da demonstração.

Uma rede R com uma marcação inicial M_0 é dita *k-limitada* se nenhuma marcação de $\mathcal{R}(R, M_0)$ têm mais que k marcas em qualquer lugar da rede. A complexidade do problema é a mesma para as redes k-limitadas [How91].

A seção 3.5.3 apresenta uma transformação do problema de alcançabilidade em redes k-limitadas para um problema de planejamento em *STRIPS*.

3.5.1 Alcançabilidade em redes limitadas pertence à PSPACE

Para mostrar que o problema de alcançabilidade pertence à PSPACE assumimos a equivalência entre as classes PSPACE e NPSPACE [Sav70] e apresentamos um algoritmo não-determinístico que verifica se uma marcação M_g é alcançável a partir de uma marcação M_0 em uma rede $R = (L, T, I_{pre}, I_{pos})$:

Alcançabilidade(R, M_0, M_g)

1 $k \leftarrow 0$;
2 $M \leftarrow M_0$;
3 repita
4 escolhe uma transição $t \in T$;
5 se $M \geq I_{pre}(., t)$ então
6 $k \leftarrow k + 1$;
7 $M \leftarrow M + I_{pos}(., t) - I_{pre}(., t)$;
8 até que $M = M_g$ ou $M < I_{pre}(., t)$;
9 se $M = M_g$ então M_g é alcançável.

O algoritmo escolhe não-deterministicamente, na linha 4, k disparos de transições que levam da marcação inicial M_0 até a marcação M_g. A linha 5 verifica se a transição t escolhida está habilitada para a marcação corrente M, caso não existam mais transições habilitadas o algoritmo pára e não é possível alcançar a marcação M_g.

O espaço utilizado pelo algoritmo corresponde à marcação corrente da rede, a marcação anterior onde t foi disparado e o contador de ações do plano k. Como uma marcação pode ser codificada em uma representação binária que ocupa espaço polinomial [How91] o espaço ocupado pelo algoritmo é polinomial em relação ao tamanho da entrada. Logo Alcançabilidade $\in$ NPSPACE, que é a classe dos problemas aceitos por uma máquina de Turing não-determinística usando espaço polinomial.

Dado que NPSPACE = PSPACE [Sav70] temos que Alcançabilidade $\in$ PSPACE.

3.5.2 Alcançabilidade em redes limitadas é PSPACE-Difícil

Para mostrar que o problema de alcançabilidade em redes de Petri limitadas é PSPACE-Difícil vamos apresentar uma transformação de uma máquina de Turing PSPACE para uma rede de Petri limitada, onde a máquina de Turing aceita a entrada se, e somente se, o problema de alcançabilidade associado tem solução.

Considere uma máquina de Turing determinística D dada por:

- um conjunto de estados: $S = \{q_1, \cdots, q_n\}$, sendo que $yes \in S$ e é um estado de aceitação;
- um alfabeto da entrada: $\Sigma = \{\sigma_1, \cdots, \sigma_{m-1}\}$;

- um alfabeto da fita: $\Gamma = \Sigma \cup \{\#\}$, sendo # o símbolo que indica que uma determinada posição da fita está vazia. Escrevemos $\Gamma = \{\gamma_1, \cdots, \gamma_m\}$;

- uma função parcial de transição: $\delta(q, \gamma) = \langle\ q', \gamma', \phi\ \rangle$ onde ϕ é a direção do movimento da cabeça da máquina dado por $\leftarrow$ ou $\rightarrow$;

- uma cabeça de leitura da fita posicionada na primeira posição ocupada pela entrada na fita;

- uma fita t com k posições, sendo l posições são ocupadas pela entrada da máquina as demais posições ocupadas pelo símbolo #, indicando que estas posições estão vazias e serão utilizadas durante o processamento.

Dado que a máquina em questão ocupa um espaço polinomial durante o processamento, temos que o número máximo de células utilizadas na fita t durante o processamento é $k = P(l)$, onde P é um polinômio sobre o tamanho da entrada l.

Seja $R = (L, T, I_{pre}, I_{pos})$ a rede de Petri correspondente à máquina de Turing D. O conjunto L de lugares é constituído por:

- um lugar para cada estado da máquina: $q_1, \cdots, q_n$;

- um lugar para cada posição possível da cabeça de leitura: $p_1, \cdots, p_k$;

- um lugar para cada símbolo do alfabeto indicando que o símbolo γ_i foi lido pela cabeça da máquina: $r_{\gamma_1}, \cdots, r_{\gamma_m}$;

- $k.m$ lugares para representar o conteúdo da fita, m lugares para cada uma das k posições da fita: $f_{\gamma_1,1}, \cdots, f_{\gamma_m,1}, \cdots, f_{\gamma_1,k}, \cdots, f_{\gamma_m,k}$;

- d lugares indicando que símbolo será escrito na fita e qual será o movimento da cabeça, onde d é o número de pares diferentes (γ^i, ϕ^i) presentes na função de transição $\delta(q, \gamma) = \langle\ q', \gamma^i, \phi^i\ \rangle$, dados por: $w_{\gamma^1,\phi^1}, \cdots, w_{\gamma^d,\phi^d}$.

O conjunto T de transições é constituído por:

- $k.m$ transições para representar as leituras feitas pela cabeça, m transições para cada uma das k posições da fita: $tr_{\gamma_1,1}, \cdots, tr_{\gamma_m,1}, \cdots, tr_{\gamma_1,k}, \cdots, tr_{\gamma_m,k}$;

- uma transições para representar cada elemento da função de transição $\delta(q,\gamma)$: $t\delta_{q_1,\gamma_1}$, $\cdots$, $t\delta_{q_1,\gamma_m}$, $\cdots$, $t\delta_{q_n,\gamma_1}$, $\cdots$, $t\delta_{q_n,\gamma_m}$, vale notar que o número máximo de elementos da função de transição é $m.n$;

- $k.d$ transições para representar as escritas e os movimentos da cabeça, onde d é o número de pares diferentes (γ',ϕ) presentes na função de transição $\delta(q,\gamma) = \langle\ q',\gamma',\phi\ \rangle$, dados por: $tw_{\gamma^1,\phi^1,1}$, $\cdots$, $tw_{\gamma^d,\phi^d,1}$ $\cdots$, $tw_{\gamma^1,\phi^1,k}$, $\cdots$, $tw_{\gamma^d,\phi^d,k}$.

As funções de incidência I_{pre} e I_{pos} são diferentes de zero para:

- $I_{pre}(f_{\gamma_i,j}, tr_{\gamma_i,j}) = 1$, para $1 \leq i \leq m$ e $1 \leq j \leq k$;
- $I_{pre}(p_j, tr_{\gamma_i,j}) = 1$, para $1 \leq i \leq m$ e $1 \leq j \leq k$;
- $I_{pos}(r_{\gamma_i}, tr_{\gamma_i,j}) = 1$, para $1 \leq i \leq m$ e $1 \leq j \leq k$;
- $I_{pos}(p_j, tr_{\gamma_i,j}) = 1$, para $1 \leq i \leq m$ e $1 \leq j \leq k$;
- $I_{pre}(q_j, t\delta_{q_j,\gamma_i}) = 1$, para $1 \leq i \leq m$ e $1 \leq j \leq n$;
- $I_{pre}(r_{\gamma_i}, t\delta_{q_j,\gamma_i}) = 1$, para $1 \leq i \leq m$ e $1 \leq j \leq n$;
- $I_{pos}(q_l, t\delta_{q_j,\gamma_i}) = 1$, para $1 \leq i \leq m$, $1 \leq j \leq n$ e $\delta(q_j,\gamma_i) = \langle\ q_l,\gamma',\phi\ \rangle$;
- $I_{pos}(w_{\gamma_l,\phi}, t\delta_{q_j,\gamma_i}) = 1$, para $1 \leq i \leq m$ e $1 \leq j \leq n$ e $\delta(q_j,\gamma_i) = \langle\ q',\gamma_l,\phi\ \rangle$;
- $I_{pre}(w_{\gamma_i,\phi}, tw_{\gamma_i,\phi,j}) = 1$, $\forall tw_{\gamma_i,\phi,j} \in T$;
- $I_{pre}(p_j, tw_{\gamma_i,\phi,j}) = 1$, $\forall tw_{\gamma_i,\phi,j} \in T$;
- $I_{pos}(f_{\gamma_i,j}, tw_{\gamma_i,\phi,j}) = 1$, $\forall tw_{\gamma_i,\phi,j} \in T$;
- $I_{pos}(p_l, tw_{\gamma_i,\phi,j}) = 1$, $\forall tw_{\gamma_i,\phi,j} \in T$ e se $\phi =\leftarrow$ então $l = j-1$ senão $l = j+1$, caso $p_l \notin L$ então $l = j$.

A marcação inicial M_0 para a rede R é diferente de zero para os lugares:

- $M_0(f_{\gamma_i,j}) = 1$, se γ_i é o símbolo que está na posição j da fita;
- $M_0(p_i) = 1$, se a cabeça de leitura da máquina está na posição i da fita;
- $M_0(q_i) = 1$, se o estado inicial da máquina é o estado q_i.

Considere como exemplo a máquina de Turing dada por:

$$\begin{aligned} S &= \{q_1, q_2, yes\} \\ \Sigma &= \{a, b, c\} \\ \Gamma &= \{a, b, c, \#\} \\ \delta(q_1, a) &= \langle\ q_1, c, \rightarrow\rangle \\ \delta(q_1, b) &= \langle\ q_1, c, \rightarrow\rangle \\ \delta(q_1, c) &= \langle\ q_1, c, \rightarrow\rangle \\ \delta(q_1, \#) &= \langle\ q_2, \#, \leftarrow\rangle \\ \delta(q_2, c) &= \langle\ q_2, a, \leftarrow\rangle \\ \delta(q_2, \#) &= \langle\ yes, \#, \rightarrow\rangle \end{aligned}$$

com a entrada "ab" na fita, a cabeça de leitura na primeira posição da entrada, o estado inicial da máquina é q_1 e yes é o estado de aceitação. A figura 11 mostra a rede de Petri obtida a partir do esquema de construção apresentado e sua marcação inicial.

A construção da rede R requer um número polinomial de passos em relação a k, pois o tamanho do conjunto de lugares é dado por $|L| = (n + k + m + k.m + d)$ e o tamanho do conjunto de transições é no máximo $|T| = (k.m + n.m + k.d)$.

O problema de alcançabilidade pode ser formulado como um problema de verificar se alguma marcação M_g, com $M_g(yes) = 1$, é alcançável. Ou seja, é um problema de alcançabilidade de sub-marcação para o lugar que representa o estado de aceitação yes da maquina D. Dada a equivalência teórica dos dois problemas [Hac76] temos que qualquer máquina de Turing PSPACE pode ser transformada em um problema de alcançabilidade em uma rede de Petri limitada usando um número polinomial de passos. Logo, alcançabilidade em redes de Petri limitadas é PSPACE-Difícil.

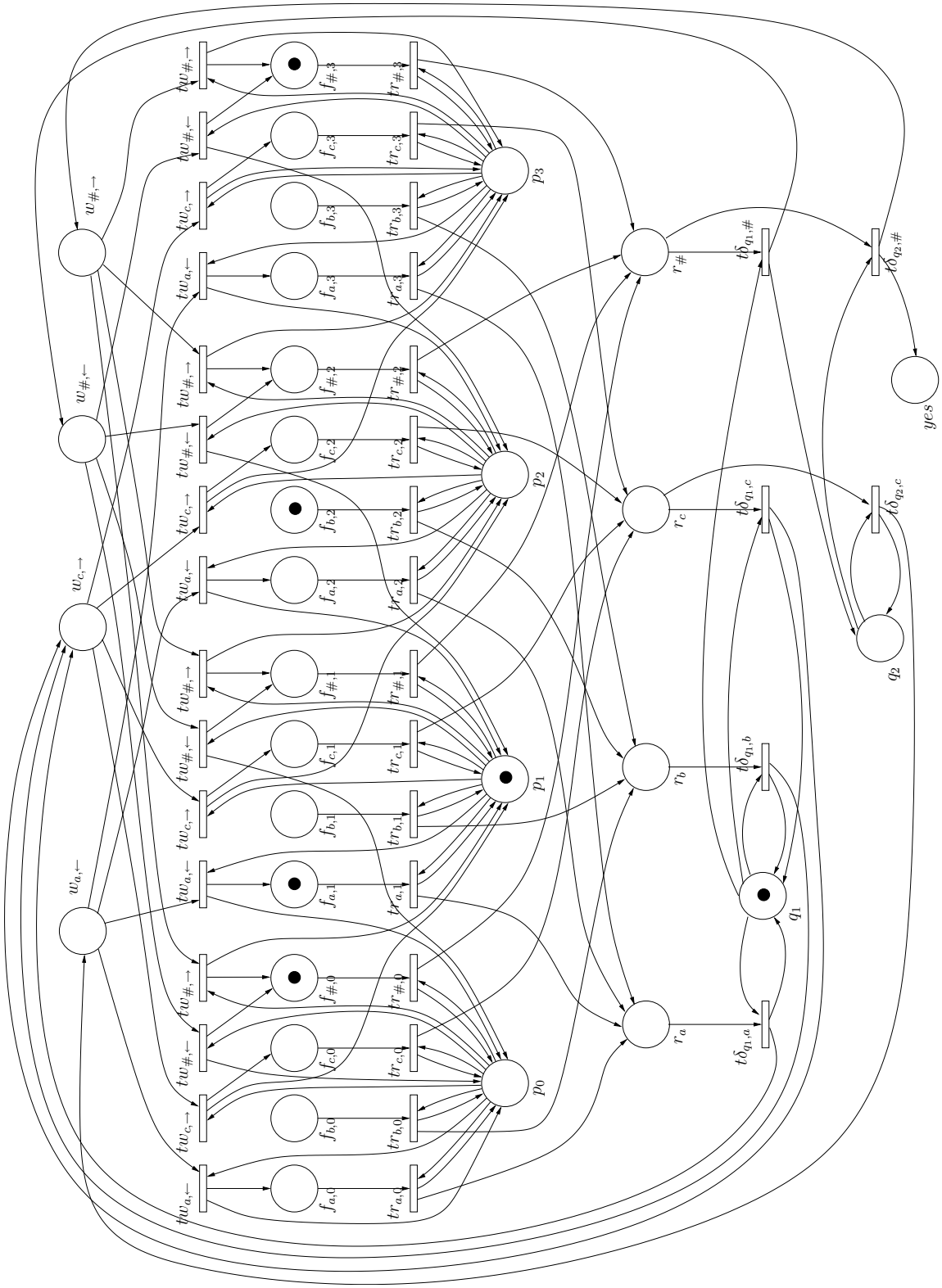


Figura 11: Rede de Petri correspondente à máquina de Turing dada como exemplo.

3.5.3 Alcançabilidade como planejamento

O problema de alcançabilidade em redes k-limitadas pode ser modelado como um problema de planejamento em *STRIPS*. A seguir apresentamos um esquema de tradução da rede de Petri em um domínio de planejamento e o problema de alcançabilidade em um problema de planejamento para este domínio.

Em geral, as redes de Petri são representadas por coleções de vetores e matrizes de números naturais. A simulação ou a modelagem destas redes em outras estruturas de representação requer que estas estruturas suportem números naturais e operações aritméticas básicas. No caso de *STRIPS*, a representação de números e operações sobre eles não é direta.

No caso da modelagem do problema de alcançabilidade em redes k-limitadas é suficiente representar operações de incremento e decremento sobre números naturais do intervalo $[0, k]$ e as relações "$\geq$" e "$=$".

Os números naturais deste intervalo podem ser representados em *STRIPS* pelo conjunto de constantes:

$$\{\mathsf{n}_0, \cdots, \mathsf{n}_k\},$$

onde n_i representa o natural i.

A relação de ordem "$\geq$" é representada em *STRIPS* pelas proposições:

$$\mathsf{maiorigual}(\mathsf{n}_i, \mathsf{n}_j),$$

onde i e $j \in \mathbb{N}$, $i \geq j$, $0 \leq i \leq k$ e $0 \leq j \leq k$.

A relação de igualdade "$=$" é representada em *STRIPS* pelas proposições:

$$\mathsf{igual}(\mathsf{n}_i, \mathsf{n}_i),$$

onde $i \in \mathbb{N}$, $0 \leq i \leq k$.

A operação de soma é definida por um conjunto de relações de incremento sobre os números naturais representadas pelas proposições:

$$\mathsf{inc}_s(\mathsf{n}_i, \mathsf{n}_j),$$

onde s, i e $j \in \mathbb{N}$, $j = (i + s)$, $1 \leq s \leq k$ e $0 \leq i \leq (k - s)$. Assim $\mathsf{inc}_3(\mathsf{n}_1, \mathsf{n}_4)$ indica que 1 incrementado de 3 é igual a 4.

De forma semelhante a operação de subtração é definida por um conjunto de relações de decremento representadas pelas proposições:

$$\mathsf{dec}_s(\mathsf{n}_i, \mathsf{n}_j),$$

onde s, i e $j \in \mathbb{N}$, $i = (j + s)$, $1 \leq s \leq k$ e $0 \leq j \leq (k - s)$.

Os lugares da rede de Petri são modelados como conjuntos de proposições que indicam o número de marcas presentes no lugar:

$$\mathsf{lugar}_l(\mathsf{n}_i),$$

indica que o lugar l da rede contém i marcas.

As marcações da rede são representadas por conjuntos de proposições $\mathsf{lugar}_l(\mathsf{n}_i)$ para todos os lugares l da rede com suas respectivas marcações i.

As transições são modeladas como conjuntos de ações em *STRIPS*. Considere uma transição t e seja $P = \{\ l \mid (I_{pre}(i,t) > 0) \vee (I_{pos}(i,t) > 0)\}$ o conjunto dos lugares ligados à transição t. A ação α_t que modela a transição t é dada por:

- parâmetros: $(m_1, m'_1, \cdots, m_l, m'_l)$. Estas variáveis representam as marcações dos l lugares ligados à transição t, m_i indica a marcação do lugar i antes do disparo de t e m'_i a marcação de i após o disparo, para $i \in P$;

- pré-condição: $\mathsf{lugar}_i(m_i) \wedge \mathsf{maiorigual}(m_i, \mathsf{n}_{I_{pre}(i,t)}) \wedge \Delta(m_i, m'_i)$, para todo lugar $i \in P$. Onde $\Delta(m_i, m'_i)$ é dado por:

  - se $I_{pre}(i,t) > I_{pos}(i,t)$: $\mathsf{dec}_s(m_i, m'_i)$, sendo $s = I_{pre}(i,t) - I_{pos}(i,t)$;
  - se $I_{pre}(i,t) < I_{pos}(i,t)$: $\mathsf{inc}_s(m_i, m'_i)$, sendo $s = I_{pos}(i,t) - I_{pre}(i,t)$;
  - se $I_{pre}(i,t) = I_{pos}(i,t)$: $\mathsf{igual}(m_i, m'_i)$;

- efeito: $\mathsf{lugar}_i(m'_i) \wedge \neg\mathsf{lugar}_i(m_i)$, para todo lugar $i \in P$, tal que $I_{pre}(i,t) \neq I_{pos}(i,t)$.

O conjunto de ações é dado pela instanciação das variáveis m e m' com as constantes $\mathsf{n_i}$, que representam os números naturais, para cada transição da rede. Considerando a rede da figura 12, a ação que modela o disparo desta transição para a marcação dada é:

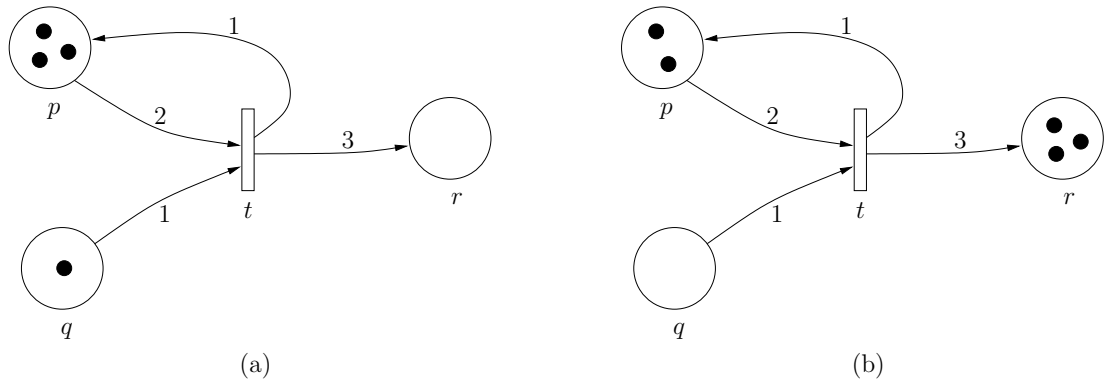


Figura 12: Exemplo de rede de Petri, (a) antes do disparo de t e (b) depois do disparo de t.

Ação : $\alpha_t(\mathsf{n}_3, \mathsf{n}_2, \mathsf{n}_1, \mathsf{n}_0, \mathsf{n}_0, \mathsf{n}_3)$

Pré-Condição : $\mathsf{lugar}_p(\mathsf{n}_3) \wedge \mathsf{maiorigual}(\mathsf{n}_3, \mathsf{n}_2) \wedge \mathsf{dec}_1(\mathsf{n}_3, \mathsf{n}_2) \wedge$
$\mathsf{lugar}_q(\mathsf{n}_1) \wedge \mathsf{maiorigual}(\mathsf{n}_1, \mathsf{n}_1) \wedge \mathsf{dec}_1(\mathsf{n}_1, \mathsf{n}_0) \wedge$
$\mathsf{lugar}_r(\mathsf{n}_0) \wedge \mathsf{maiorigual}(\mathsf{n}_0, \mathsf{n}_0) \wedge \mathsf{inc}_3(\mathsf{n}_0, \mathsf{n}_3)$

Efeito : $\mathsf{lugar}_p(\mathsf{n}_2) \wedge \mathsf{lugar}_q(\mathsf{n}_0) \wedge \mathsf{lugar}_r(\mathsf{n}_3) \wedge$
$\neg\mathsf{lugar}_p(\mathsf{n}_3) \wedge \neg\mathsf{lugar}_q(\mathsf{n}_1) \wedge \neg\mathsf{lugar}_r(\mathsf{n}_0)$

Um problema de alcançabilidade é dado por uma rede de Petri e duas marcações, a marcação inicial M_0 e a marcação que se deseja alcançar M_g, ou a sub-marcação S_g. O problema pode ser modelado como um problema de planejamento seguindo a descrição apresentada. O problema de planejamento resultante é constituído por:

- as ações α_t obtidas a partir das transições, instanciadas com as constantes n_i;
- o estado inicial do problema dado pela conjunção das proposições maiorigual, igual, inc e dec, instanciadas com as constantes n_i e ainda as proposições lugar que representam M_0;
- o objetivo do problema é dado pela conjunção das proposições lugar que representam M_g ou S_g.

A existência de um plano que resolve o problema de planejamento obtido indica que a

marcação M_g, ou a sub-marcação S_g, é alcançável a partir de M_0. O plano corresponde à seqüência de disparos das transições da rede que levam até a marcação desejada.

3.6 Considerações

Foi apresentado neste capítulo o formalismo das redes de Petri e algumas características do problema de alcançabilidade entre marcações para as sub-classes das redes acíclicas e das redes limitadas.

No caso das redes acíclicas o problema de alcançabilidade da marcação M_g pode ser visto como um problema de programação inteira onde as transições da rede são associadas às variáveis inteiras do problema e os lugares às restrições. Assim, existe uma solução que atende às restrições do problema se e somente se a marcação M_g é alcançável. Vale notar que esta solução, além de decidir a alcançabilidade, indica o número de vezes que cada transição da rede é disparada para obter M_g. Uma simples análise da estrutura da rede acíclica permite determinar restrições de ordenação entre as transições e assim estabelecer seqüências de disparos que levam da marcação inicial até a marcação M_g. Esta abordagem é usada na construção da família de planejadores apresentada nos próximos capítulos.

Para as redes limitadas a complexidade computacional do problema de alcançabilidade é apresentada de maneira informal na seção 3.5. Inicialmente é mostrado que o problema pode ser tratado por um algoritmo não-determinístico que usa espaço polinomial, o que corresponde a uma máquina de Turing PSPACE. A outra parte da prova apresenta a transformação destas máquinas em problemas de alcançabilidade em redes de Petri limitadas. Portanto, temos a equivalência entre máquinas de Turing PSPACE e alcançabilidade em redes limitadas.

O mesmo tipo de equivalência é apresentada na seção 2.2 para o problema de planejamento. Estas duas equivalências mostram que os dois problemas pertencerem à mesma classe de complexidade e que as máquinas de Turing PSPACE podem ser reduzidas a estes problemas em um número polinomial de passos. Ou seja, os problemas de planejamento em *STRIPS* e de alcançabilidade em redes limitadas são equivalentes em termos de complexidade.

Esta equivalência motiva a pesquisa por transformações diretas entre os dois problemas. A seção 3.5.3 apresenta um modelo de tradução de problemas de alcançabilidade em problemas de planejamento, permitindo assim aplicar planejadores na análise de alcançabilidade

em redes de Petri. Entretanto, vale notar que apesar dos problemas terem a mesma complexidade computacional, não é trivial estabelecer transformações eficientes. No caso do modelo proposto na seção 3.5.3 a instância do problema de planejamento gerada tem, no pior caso, tamanho exponencial em relação ao tamanho da rede. A tradução de cada transição da rede pode gerar k^n ações diferentes, onde n é o número de lugares associados à transição e k é o número máximo de marcas em qualquer lugar da rede k-limitada.

O mapeamento de problemas de planejamento em problemas de alcançabilidade é o tema central do trabalho e a base dos planejadores apresentados nos próximos capítulos. Neste caso os problemas de planejamento são transformados em redes acíclicas. O tamanho das redes obtidas é proporcional ao tamanho do grafo de planos.

4 Petriplan

Este capítulo apresenta o algoritmo *Petriplan* [SCK00], que é uma abordagem para o problema de planejamento usando redes de Petri como ferramenta de modelagem. A seção 4.1 apresenta os resultados obtidos na dissertação de mestrado [Sil00], as seções 4.2 e 4.3 tratam de variações sobre o modelo de representação do *Petriplan*. As seções 4.4 e 4.5 apresentam alguns experimentos realizados e considerações finais sobre o capítulo.

4.1 O algoritmo *Petriplan*

O *Petriplan* é um algoritmo de três fases que usa uma estrutura semelhante à dos planejadores baseados no *Graphplan* [BF95], notadamente o *Blackbox* [KS99].

A primeira fase consiste da construção do grafo de planos seguindo as mesmas regras do *Graphplan*. Esta fase pode ser vista como um pré-processamento, buscando simplificar o problema para a próxima fase.

Na segunda fase, o grafo de planos gerado é transformado em uma rede de Petri equivalente, onde o problema de planejamento original pode ser visto como um problema de encontrar uma seqüência de disparos de transições que resolve um problema de alcançabilidade de sub-marcação.

Na terceira fase do algoritmo o problema de alcançabilidade definido na fase anterior é convertido em um problema de programação inteira, que então é resolvido por sistemas específicos para otimização de problemas de programação inteira.

Se nenhuma solução for encontrada para o problema de programação inteira na terceira fase, o algoritmo volta para a primeira fase e uma nova expansão do grafo é realizada, este laço continua até que uma solução seja encontrada na fase 3 ou a condição de falha do *Graphplan* seja atendida na fase 1. Quando uma solução é encontrada na fase 3 ela é convertida para um plano que resolve o problema original de planejamento.

Nas próximas seções são apresentados, usando um exemplo, o grafo de planos usado na fase 1, a tradução para redes de Petri da fase 2, a solução do problema de alcançabilidade na fase 3 usando programação inteira e a recuperação do plano a partir da solução do problema de programação inteira. O método completo pode ser visto em [Sil00].

4.1.1 Tradução para rede de Petri

A construção do grafo de planos, que é a estrutura de representação do modelo usada no *Graphplan* e na primeira fase do *Petriplan*, é feita seguindo a descrição apresentada na seção 2.3. O algoritmo original é de [BF95].

Como exemplo será usado um problema apresentado em [Wel99]. O problema é preparar um jantar surpresa para alguém que está dormindo. Os objetivos são: remover o lixo, preparar o jantar e embrulhar um presente. As quatro ações possíveis são: *cook*, *wrap*, *carry* e *dolly*. A ação *cook* requer mãos limpas (*clean*) e obtém o jantar (*dinner*). A ação *wrap* tem que ser realizada em silêncio (*quiet*) e obtém o presente embrulhado (*present*). *Dolly* elimina o lixo (*garbage*) usando um carrinho de mão e produz barulho (negando *quiet*), *Carry* também elimina o lixo mas suja as mãos (negando *clean*).

No estado inicial o agente está com as mãos limpas, tem lixo na casa e está em silêncio, todas as outras proposições são falsas. Este estado é representado pela primeira camada do grafo de planos da figura 13. Após a inclusão de todas as ações aplicáveis sobre o estado inicial (segunda camada), uma nova camada de proposições é construída (terceira camada) com os efeitos da camada de ações anterior. Este procedimento é chamado de expansão do grafo.

Os arcos em negrito no grafo representam relações de exclusão mútua entre duas ações do problema. Assim, durante o processo de busca pela solução estas relações impedem que duas ações que apresentam inconsistências entre si estejam ambas presentes no plano.

A figura 14 mostra o grafo de planos para o problema do jantar com uma das soluções possíveis (um plano) marcada.

A algoritmo *Graphplan* [BF95] consiste de duas fases, a expansão do grafo e a busca pela solução. Caso a busca não encontre uma solução para o problema uma nova expansão é realizada e uma nova busca acontece. O algoritmo continua até que uma solução seja encontrada ou a expansão resulte em uma repetição da camada anterior, o que indica que nenhuma solução para o problema será encontrada e o algoritmo para.

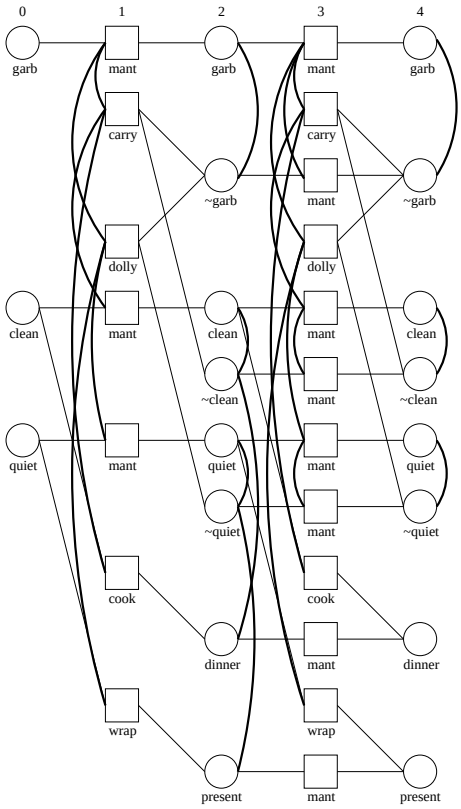


Figura 13: O grafo de planos para o problema do jantar.

Usando os conceitos apresentados no capítulo 3 mostra-se que o problema de encontrar um plano em um grafo de planos é equivalente ao problema de encontrar uma seqüência de disparos de transições para o problema de alcançabilidade de sub-marcação numa rede de Petri. A construção desta rede a partir do grafo de planos é apresentada a seguir.

Um grafo de planos pode ser fragmentado em cinco estruturas: nós-ação, nós-proposição, arestas de nós-ação para nós-proposição (*arestas-efeito*), arestas de nós-proposição para nós-ação (*arestas-pré-condição*) e relações de exclusão mútua.

Cada uma destas estruturas é traduzida diretamente em uma estrutura de redes de Petri equivalente. A seguir são apresentadas estas traduções.

Nós-ação: um nó ação do grafo é traduzido em uma única transição na rede de Petri. A figura 15 mostra à esquerda o nó do grafo e à direita a transição correspondente.

Os nós-ação do grafo são traduzidos diretamente para uma transição da rede de Petri, que só estará habilitada para disparo se todas as suas pré-condições forem verdadeiras, ou seja, se existirem marcas em todos os lugares que representam as arestas-pré-condição

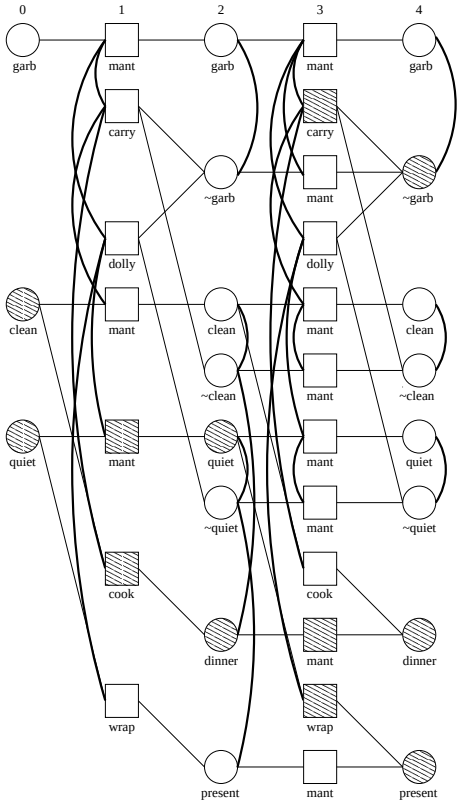


Figura 14: O grafo de planos para o problema do jantar com uma de suas soluções.

Figura 15: Tradução dos nós-ação.

ligadas à ação representada pela transição. Após o disparo de uma transição que representa uma ação, uma marca é removida de cada lugar representando uma aresta-pré-condição e outra é inserida em cada lugar representando uma proposição efeito.

Nós-proposição: um nó proposição é traduzido em um lugar e uma transição, com um arco do lugar para a transição. A figura 16 mostra esta tradução.

Figura 16: Tradução dos nós-proposição.

Os nós-proposição são representados por um lugar ligado a uma transição. Esta transição é necessária para que todos os lugares que representam arestas-pré-condição recebam marcas após o seu disparo. O disparo só ocorrerá se existir uma marca no lugar que representa a proposição, ou seja, a proposição é verdadeira.

Arestas-efeito: uma aresta-efeito é traduzida em um arco que vai da transição que representa o nó ação para o lugar representando o nó proposição. A figura 17 mostra esta tradução.

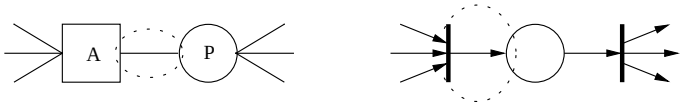


Figura 17: Tradução das arestas-efeito.

Arestas-pré-condição: uma aresta-pré-condição é traduzida em um lugar com dois arcos: um vindo da transição que representa o nó proposição e outro que vai para a transição representando o nó ação. A figura 18 mostra esta tradução.

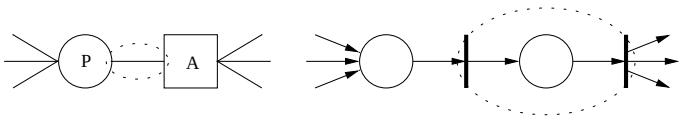


Figura 18: Tradução das arestas-pré-condição.

As arestas-pré-condição são representadas por um lugar uma vez que um nó proposição do grafo pode ser pré-condição de mais de uma ação. Assim, cada transição que representa ações tem um conjunto de arestas-pré-condição independente. Caso esta estrutura não fosse utilizada, o disparo de uma transição ação desabilitaria outra transição ação que tivesse uma pré-condição em comum, pois o disparo da primeira removeria as marcas de todos os lugares que representam pré-condições de outras ações.

Exclusão mútua: a relação binária de exclusão mútua é traduzida em um lugar com dois arcos saindo, um para cada transição que representa cada nó ação da relação, e uma marca neste lugar. A figura 19 mostra esta tradução.

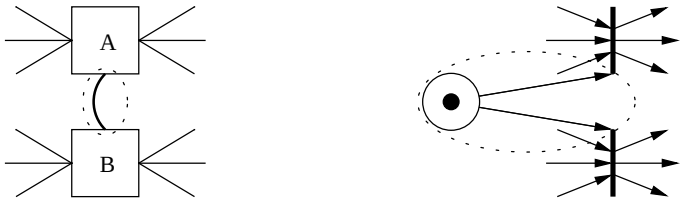


Figura 19: Tradução das relações de exclusão mútua.

Note que na figura 19 há uma marca no lugar que representa a relação de exclusão mútua porque apenas uma das duas ações pode ser executada, e esta marca habilita as duas transições para disparo, mas após o disparo de uma delas a outra deixa de estar habilitada, impossibilitando o seu disparo.

As relações de exclusão mútua entre proposições não são representadas na rede, pois estas são utilizadas somente durante a construção do grafo de planos para determinar as exclusões mútuas entre as ações do grafo.

Todos os arcos da rede têm peso igual a um, pois não há necessidade de se colocar mais de uma marca em cada lugar da rede. Quando existe uma marca em um determinado lugar da rede isso pode indicar que uma proposição é válida ou que uma pré-condição é válida ou ainda que existe uma relação de exclusão mútua entre duas ações. Observe que após o disparo de um conjunto qualquer de transições é possível que existam lugares com mais de uma marca na rede, indicando que existe mais de um caminho na rede que valida a estrutura representada.

Para representar o estado inicial do problema de planejamento usa-se uma marca em cada lugar que representa a camada zero do grafo de planos.

As marcas nos lugares do estado inicial e as marcas nos lugares que controlam as relações de exclusão mútua definem a marcação inicial da rede. A figura 20 mostra a rede de Petri obtida pela tradução do problema do jantar apresentado na figura 13.

O estado meta do problema de planejamento é representado pela sub-marcação da rede que contém marcas nos lugares que representam os nós-proposições do estado meta. A figura 21 mostra uma rede de Petri com uma marcação que contém o estado meta para o problema do jantar.

Seja N a rede obtida pela tradução do grafo de planos, M_0 a marcação inicial da rede

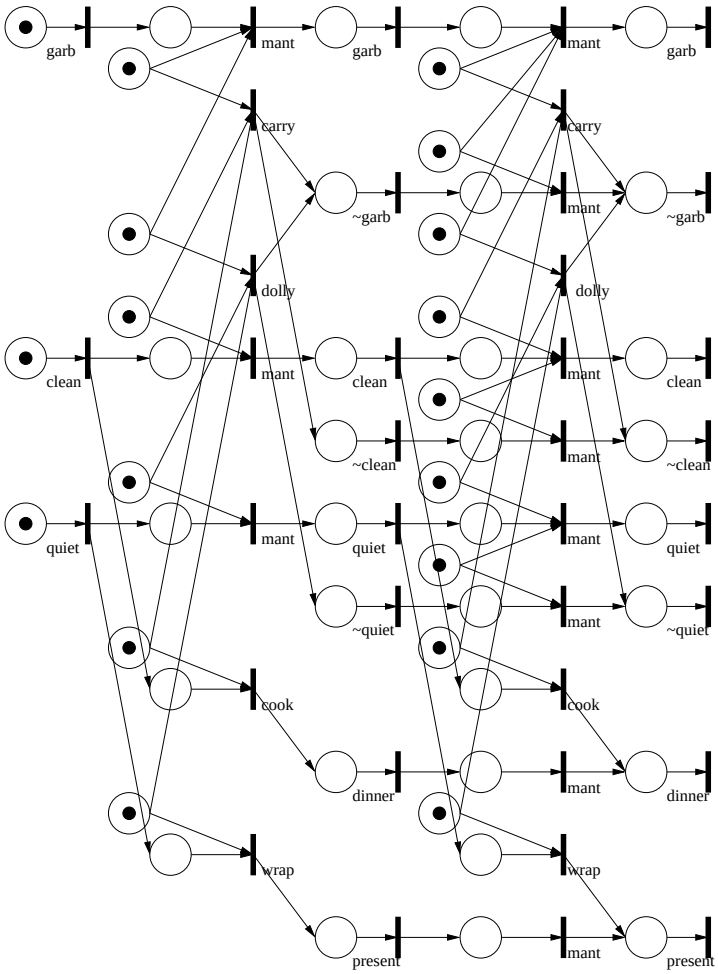


Figura 20: A rede de Petri com marcação inicial para o problema do jantar.

e M_g a marcação que contém marcas nos lugares que representam as proposições do estado meta. Pode-se definir o problema relacionado à alcançabilidade da marcação M_g como: "encontre uma seqüência s de transições de N que quando disparada transforma M_0 em M_g".

O problema aqui é que M_g é uma marcação completa da rede que contém os lugares representando os nós-proposições do estado meta e não se conhece a marcação completa desejada. Apenas o subconjunto $S_g \subseteq M_g$ que contém o estado meta é conhecido. Portanto, o problema relacionado à alcançabilidade da marcação M_g não é bem formado, pois M_g não é conhecida.

A alternativa é usar o problema de encontrar uma seqüência de transições s que alcance alguma sub-marcação S_g; isto é, resolver o problema de alcançabilidade da sub-marcação S_g a partir de M_0. A solução deste problema será tratada na próxima seção.

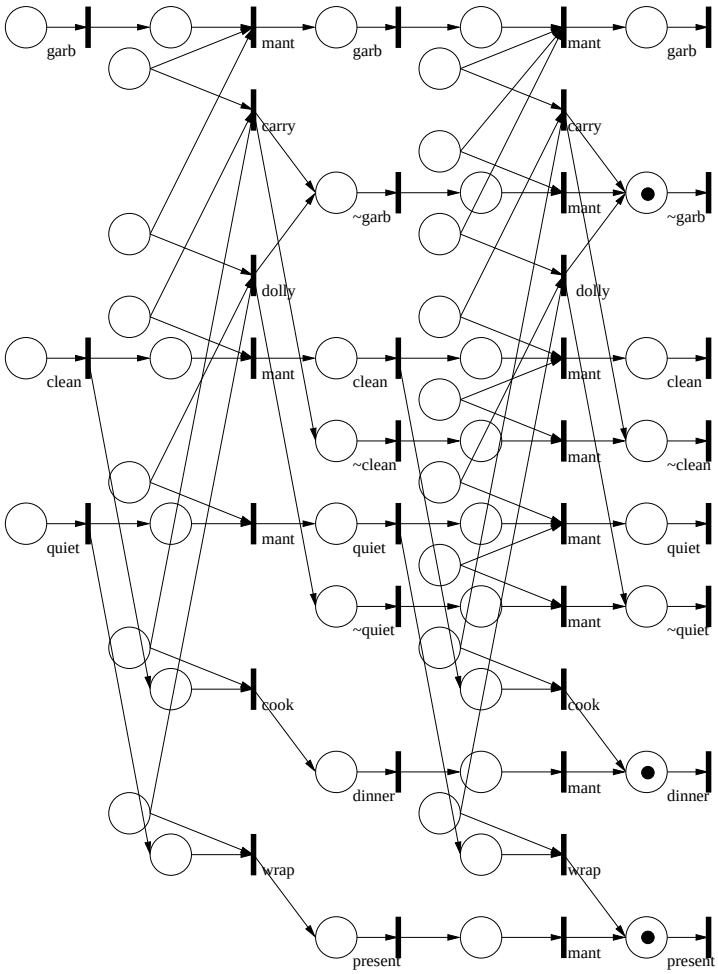


Figura 21: A rede de Petri com a marcação final desejada para o problema do jantar.

A seqüência de transições s define um conjunto de caminhos na rede de Petri que sai dos lugares que representam o estado inicial e chegam aos lugares que representam o estado meta. Considerando apenas as transições que representam ações nesta seqüência, tem-se uma estrutura semelhante à de um plano do problema original. A recuperação deste plano será tratada na seção 4.1.3.

4.1.2 Alcançabilidade e programação inteira

Usando a rede de Petri obtida na seção anterior mostra-se como usar métodos para problemas de programação inteira para resolver o problema de alcançabilidade de submarcação nesta rede e, assim, resolver o problema original de planejamento.

Vale notar que a rede resultante do processo de tradução é acíclica e portanto a equação fundamental (3.1) das redes de Petri pode ser usada para definir o problema de programação

inteira associado ao problema de alcançabilidade.

Seja N a rede resultante da tradução, M_0 a marcação inicial para o problema e S_g a sub-marcação que representa o objetivo do problema, o problema de programação inteira correspondente é definido pela restrição vetorial:

$$C.\overline{s} \geq S_g - M_0. \tag{4.1}$$

Qualquer vetor $\overline{s}$ que é solução para o problema de programação inteira (4.1) representa quantas vezes cada transição da rede é disparada para obter M_g a partir de M_0, onde $M_g(p) \geq S_g(p)$, para todo lugar p na rede N.

A partir do exemplo do problema do jantar das seções anteriores (figura 20), considere o sub-problema de embrulhar o presente e remover o lixo da casa, dado pela rede N_p, apresentada nas figuras 22 e 23. Esta rede pode ser representada matricialmente pela matriz de incidência C_p dada por[1]:

$$C_p = \begin{bmatrix}
-&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.\\
.&-&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.\\
.&.&-&-&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.\\
.&.&-&.&-&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.\\
.&.&.&.&-&-&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.\\
.&.&.&.&-&.&-&.&.&.&.&.&.&.&.&.&.&.&.&.&.\\
+&.&-&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.\\
.&+&.&.&.&-&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.\\
.&+&.&.&.&.&-&.&.&.&.&.&.&.&.&.&.&.&.&.&.\\
.&.&+&.&.&.&.&-&.&.&.&.&.&.&.&.&.&.&.&.&.\\
.&.&.&+&+&.&.&.&-&.&.&.&.&.&.&.&.&.&.&.&.\\
.&.&.&.&.&+&.&.&.&-&.&.&.&.&.&.&.&.&.&.&.\\
.&.&.&.&+&.&.&.&.&.&-&.&.&.&.&.&.&.&.&.&.\\
.&.&.&.&.&.&+&.&.&.&.&-&.&.&.&.&.&.&.&.&.\\
.&.&.&.&.&.&.&.&.&.&.&.&-&-&.&.&.&.&.&.&.\\
.&.&.&.&.&.&.&.&.&.&.&.&-&.&.&-&.&.&.&.&.\\
.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&-&-&.&.&.&.\\
.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&-&-&.&.&.\\
.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&-&.&.&-&.&.\\
.&.&.&.&.&.&.&+&.&.&.&.&-&.&.&.&.&.&.&.&.\\
.&.&.&.&.&.&.&.&+&.&.&.&.&.&-&.&.&.&.&.&.\\
.&.&.&.&.&.&.&.&.&+&.&.&.&.&.&.&-&.&.&.&.\\
.&.&.&.&.&.&.&.&.&.&+&.&.&.&.&.&.&-&.&.&.\\
.&.&.&.&.&.&.&.&.&+&.&.&.&.&.&.&.&.&-&.&.\\
.&.&.&.&.&.&.&.&.&.&.&+&.&.&.&.&.&.&.&.&-\\
.&.&.&.&.&.&.&.&.&.&.&.&+&.&.&.&.&.&.&.&.\\
.&.&.&.&.&.&.&.&.&.&.&.&.&+&+&+&.&.&.&.&.\\
.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&+&.&.&.&.\\
.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&+&.&+&.&.\\
.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&.&+&+
\end{bmatrix}$$

Os valores desta matriz podem ser interpretados como:

- um "+" em uma posição (x, y) da matriz indica que existe um arco entrando no lugar x e saindo da transição y;

- um "−" em uma posição (x, y) da matriz indica que existe um arco saindo do lugar

[1]Para facilitar a visualização os valores da matriz foram substituídos da seguinte forma, "0" por ".", "1" por "+" e "−1" por "−".

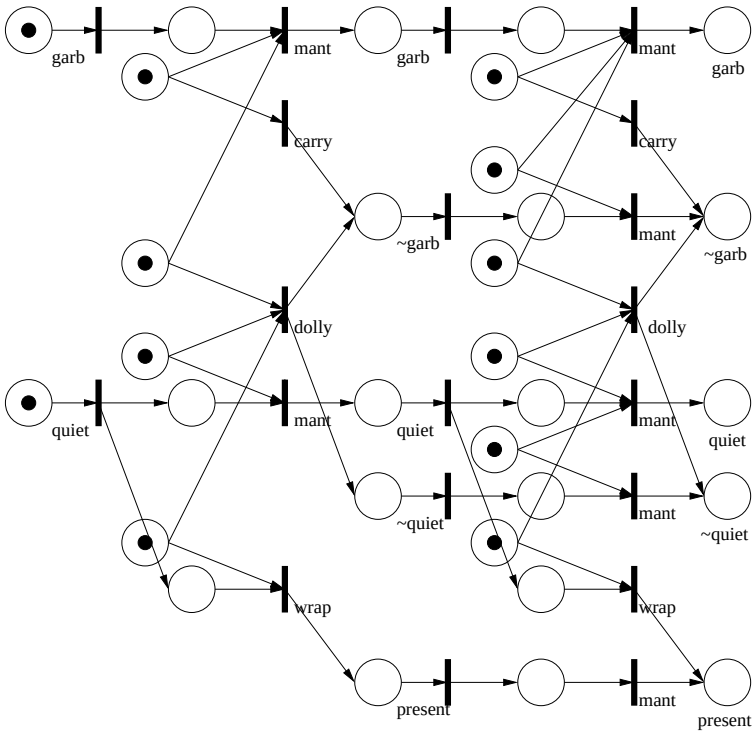


Figura 22: A rede de Petri com a marcação inicial para o problema do presente e do lixo.

x e entrando na transição y;

- um "." em uma posição (x, y) da matriz indica que não existem arcos entre o lugar x e a transição y,

onde x é uma linha da matriz e y uma coluna. Estes índices, x para os lugares e y para as transições, são representados na figura 23 por números acima e a direita de cada estrutura da rede.

Note que as transições que representam os nós-proposição da última camada do grafo, ou seja, a última camada da rede, foram removidas da representação, pois nenhuma ação tem como pré-condição estas proposições. Esta remoção não afeta a representação do problema e diminui o número de colunas da matriz C_p e portanto diminui o problema de programação inteira correspondente.

A marcação inicial M_{p0} apresentada na figura 22 é dada por:

$$M_{p0} = [\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1\ 1\ 1\ 1\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ ]^T.$$

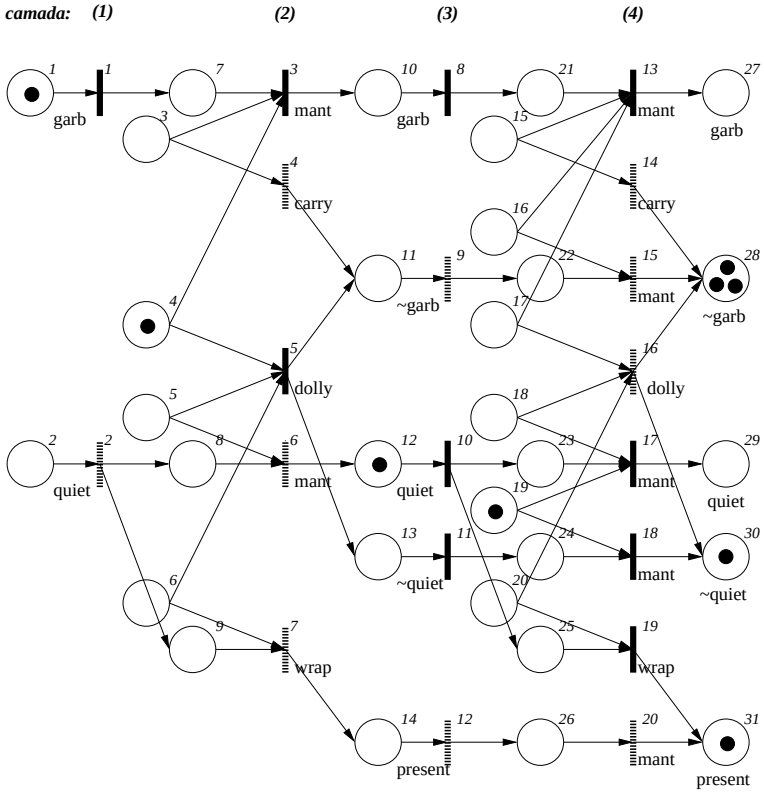


Figura 23: A rede de Petri com uma marcação final para o problema do presente e do lixo.

A marcação final M_{p_g} apresentadas na figura 23 é dada por:

$$M_{p_g} = [\ 1\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 3\ 0\ 1\ 1\ ]^T.$$

A sub-marcação $S_{p_g} < M_{p_g}$ que contém o estado meta para o problema do presente e do lixo da figura 22 é dada pelo vetor abaixo, onde "·" indica que este lugar não pertence à sub-marcação:

$$S_{p_g} = [\ \cdot\ 1\ \cdot\ \cdot\ 1\ ]^T.$$

Aplicando a inequação 4.1, que define o problema de programação inteira, sobre a rede de Petri N_p, dada pela matriz C_p, a marcação inicial M_{p_0} e a sub-marcação final S_{p_g}, tem-se:

$$C_p.\overline{s_p} \geq S_{p_g} - M_{p_0} \quad , \tag{4.2}$$

onde $\overline{s_p}$ é o vetor característico da rede N_p e representa o número de vezes que cada transição desta rede é disparada.

A inequação 4.2 é um problema de programação inteira para o problema do presente e do lixo. Uma solução para o problema de programação inteira, que obtém a marcação M_{p_g} da figura 23, é dada pelo vetor:

$$\overline{s_p} = [\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ 1\ ]^T.$$

As transições disparadas do vetor $\overline{s_p}$ estão representadas na figura 23 por barras verticais pontilhadas.

Observe que, apesar das transições t_1 e t_{10} estarem habilitadas, elas não foram disparadas em $\overline{s_p}$, pois não influenciam na obtenção da sub-marcação S_{p_g}. Este fato ocorre devido à desigualdade ($\geq$) da inequação 4.2, permitindo que marcas fiquem espalhadas pela rede caso estas não influenciem na obtenção de S_{p_g}.

O problema que se tem agora é que o vetor $\overline{s_p}$ não é uma seqüência ordenada de transições que transforma a marcação M_{p_0} na marcação M_{p_g}, ele apenas nos diz quantas vezes cada ação desta seqüência foi disparada. Este problema é resolvido tomando-se as transições da esquerda para a direita de acordo com as camadas verticais indicadas na figura 23, pois, a estrutura da rede deriva diretamente das camadas do grafo de planos. Assim tem-se a seqüência parcialmente ordenada s_p dada por:

$$s_p = (T_1, T_2, T_3, T_4),$$

onde T_i é o conjunto das transições t_j da camada i da figura 23, tal que $\overline{s_p}(t_j) \geq 1$, dado por:

$$\begin{aligned} T_1 &= \{t_2\}, \\ T_2 &= \{t_4, t_6, t_7\}, \\ T_3 &= \{t_9, t_{12}\}, \\ T_4 &= \{t_{14}, t_{15}, t_{16}, t_{20}\}. \end{aligned}$$

A partir da seqüência parcialmente ordenada s_p obtém-se seqüências totalmente ordenadas pela escolha de uma ordem para os elementos dos conjuntos T_i, como por exemplo as seqüências abaixo:

$$\begin{aligned} s_{p_1} &= (t_2, t_4, t_6, t_7, t_9, t_{12}, t_{14}, t_{15}, t_{16}, t_{20}), \\ s_{p_2} &= (t_2, t_7, t_4, t_6, t_{12}, t_9, t_{16}, t_{20}, t_{14}, t_{15}). \end{aligned}$$

Portanto as seqüências de transições s_{p_1} e s_{p_2}, derivadas da solução $\overline{s_p}$ do problema de programação inteira (4.2), são soluções para o problema de alcançabilidade da sub-marcação S_{p_g} da rede de Petri N_p.

4.1.3 Encontrando o plano

A solução do problema de alcançabilidade pode conter, além da solução do problema de planejamento, disparos de transições que não representam ações no problema de planejamento, como transições de manutenção e transições que representam proposições.

A questão é: como recuperar o plano que resolve o problema de planejamento a partir da seqüência de transições parcialmente ordenada que resolve o problema de alcançabilidade?

Para encontrar um plano que resolve o problema de planejamento, inicialmente usa-se o seguinte procedimento para simplificar a rede de Petri com a marcação final para o problema de alcançabilidade:

1. remova da rede todos os lugares que representam relações de exclusão mútua;

2. remova da rede todas as transições t para as quais $\overline{s}(t) = 0$;

3. a partir dos lugares l, tal que $l \in S_g$, percorra a rede no sentido contrário ao dos arcos marcando os lugares e as transições por onde passar;

4. remova todos os lugares e transições que não foram marcados no passo anterior.

A rede resultante contém apenas os caminhos que representam planos que são solução para o problema original. Considerando o problema do presente e do lixo da seção anterior (figura 23), após a aplicação do procedimento acima tem-se a rede da figura 24 como resultado.

Para se obter um plano que seja solução para o problema de planejamento original basta escolher na rede um caminho para cada lugar de S_g, percorrendo os arcos em sentido contrário. As seqüências de transições que representam ações nestes caminhos constituem o plano.

Por exemplo, para o problema do presente e do lixo, pode-se escolher as seguintes seqüencias $garb_i$ de transições para obter o lugar 28 da figura 24, que representa a proposição

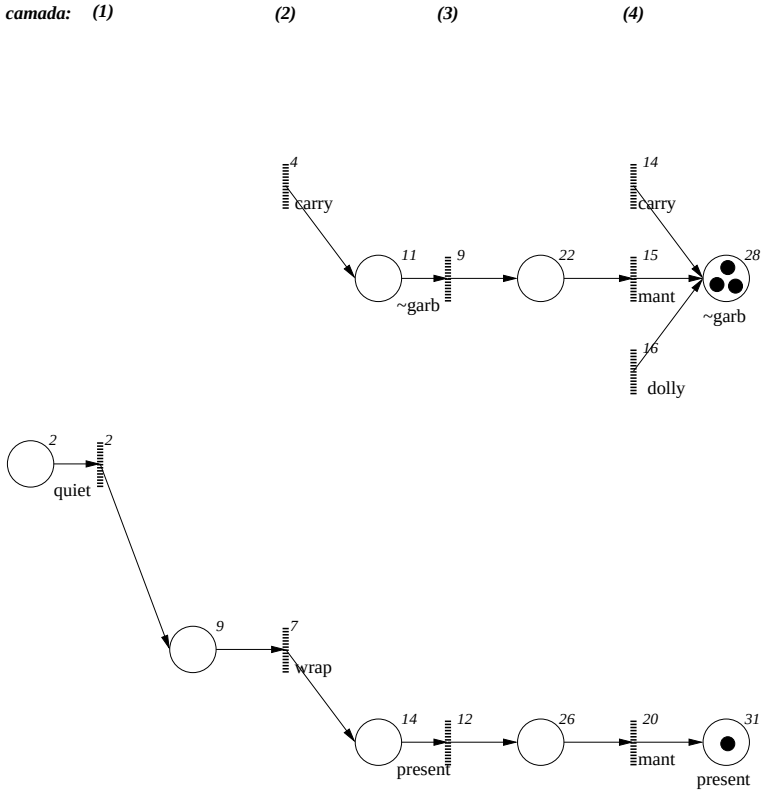


Figura 24: A parte relevante da rede de Petri para o problema do presente e do lixo.

$\sim garb$:

$$garb_1 = (t_{14}),$$
$$garb_2 = (t_4, t_9, t_{15}),$$
$$garb_3 = (t_{16}).$$

Para o lugar 31 que representa a proposição *present* temos apenas uma seqüência $pres_i$:

$$pres_1 = (t_2, t_7, t_{12}, t_{20}).$$

Intercalando estas seqüências, duas a duas, de acordo com as camadas onde as transições aparecem, tem-se as seguintes seqüências s_i parcialmente ordenadas:

$$s_1 = (\ t_2,\ t_7,\ t_{12},\ \{\ t_{14},\ t_{20}\ \}\ ),$$
$$s_2 = (\ t_2,\ \{\ t_4,\ t_7\ \},\ \{\ t_9,\ t_{12}\ \},\ \{\ t_{15},\ t_{20}\ \}\ ),$$
$$s_3 = (\ t_2,\ t_7,\ t_{12},\ \{\ t_{16},\ t_{20}\ \}\ ),$$

onde s_1 foi obtida a partir de $garb_1$ e $pres_1$, s_2 a partir de $garb_2$ e $pres_1$, e s_3 a partir de $garb_3$ e $pres_1$. Retirando destas seqüências as transições que representam proposições e as que representam ações de manutenção tem-se:

$$s'_1 = (\ t_7,\ t_{14}\ ),$$
$$s'_2 = (\ \{\ t_4,\ t_7\ \}\ ),$$
$$s'_3 = (\ t_7,\ t_{16}\ ).$$

A partir das seqüências s'_1, s'_2 e s'_3 obtem-se os seguintes planos para o problema de planejamento:

$$plano_1 = (\ wrap,\ carry\ ),$$
$$plano_2 = (\ \{\ carry,\ wrap\ \}\ ),$$
$$plano_3 = (\ wrap,\ dolly\ ),$$

onde os planos $plano_1$ e $plano_3$ são completamente ordenados, em que uma ação deve ser executada após a outra. O plano $plano_2$ é um plano paralelo, ou parcialmente ordenado, indicando que as ações podem ser executadas em qualquer ordem, ou ainda, de forma concorrente.

Os três planos obtidos a partir da seqüência parcialmente ordenada s_p e da simplificação da rede N_p são soluções para o problema de planejamento: "embrulhar o presente e remover o lixo da casa", apresentado anteriormente.

O número de transições presentes na rede obtida pela tradução apresentada é proporcional ao número de vértices do grafo de planos. De fato, para cada proposição ou ação do grafo se tem uma transição na rede resultante. O número de transições está diretamente ligado à dificuldade em se encontrar uma seqüência de disparos de transições que resolva o problema de alcançabilidade.

É fácil notar que ao tratar o problema de alcançabilidade como um problema de programação inteira se tem uma variável associada a cada transição da rede. Quanto menor o número de variáveis de um problema de programação inteira menor o espaço de busca a ser explorado durante a solução do problema. Na próxima seção é apresentada uma tradução dos nós-proposição que reduz significativamente o número de transições da rede resultante.

Os lugares da rede que representam as relações de mutex do grafo atuam como pontos

de ramificação da árvore de busca do problema, pois para cada um destes lugares temos duas transições habilitadas, entretanto, apenas uma delas pode ser disparada. Caso o procedimento de busca opte pelo disparo de uma transição que não leva a uma solução é necessário revisar esta decisão e explorar o disparo da outra transição. A redução do número de lugares que representam conflitos na rede diminui o fator de ramificação no espaço de busca. A seção 4.3 apresenta algumas estruturas redundantes na rede que podem ser simplificadas, entre elas algumas relações de mutex.

4.2 Tradução alternativa

Esta seção apresenta um esquema alternativo de tradução do grafo de plano que resulta em uma rede de Petri acíclica mais compacta que a tradução já apresentada na seção 4.1.1.

4.2.1 Estrutura de tradução

Assim como apresentado na seção 4.1.1 o grafo de planos é separado em cinco estruturas básicas: nós-ação, nós proposição, arestas-efeito, arestas-pré-condição e relações de exclusão mútua.

Os esquemas de tradução já apresentados para nós-ação, arestas-efeito e relações de exclusão mútua não sofrem alterações, a diferença está na tradução dos nós-proposição e das arestas-pré-condição.

Os nós-proposições do grafo são representados diretamente por lugares na rede Petri. Entretanto, se uma proposição é pré-condição para mais de uma ação no grafo, o lugar que representa esta proposição será um conflito na rede, pois apenas uma das transições que representam as ações poderá ser disparada. A solução para este conflito é criar uma cópia do lugar para cada transição que tem esse lugar como pré-condição. A figura 25 mostra a tradução da proposição P, que é efeito das ações A, B e C e pré-condição das ações D e E. A rede resultante apresenta dois lugares que representam a proposição, sendo P' uma cópia do lugar P.

No caso da tradução de nós-proposição que pertencem ao estado inicial, uma marca é colocada em cada lugar que a representa. A figura 26 mostra a tradução do grafo de planos do problema do jantar apresentado na seção 2.3 usando este esquema alternativo de tradução.

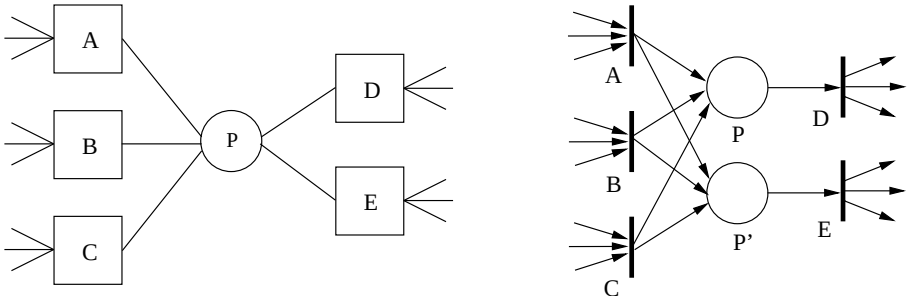


Figura 25: Tradução alternativa dos nós-proposição e das arestas-pré-condição.

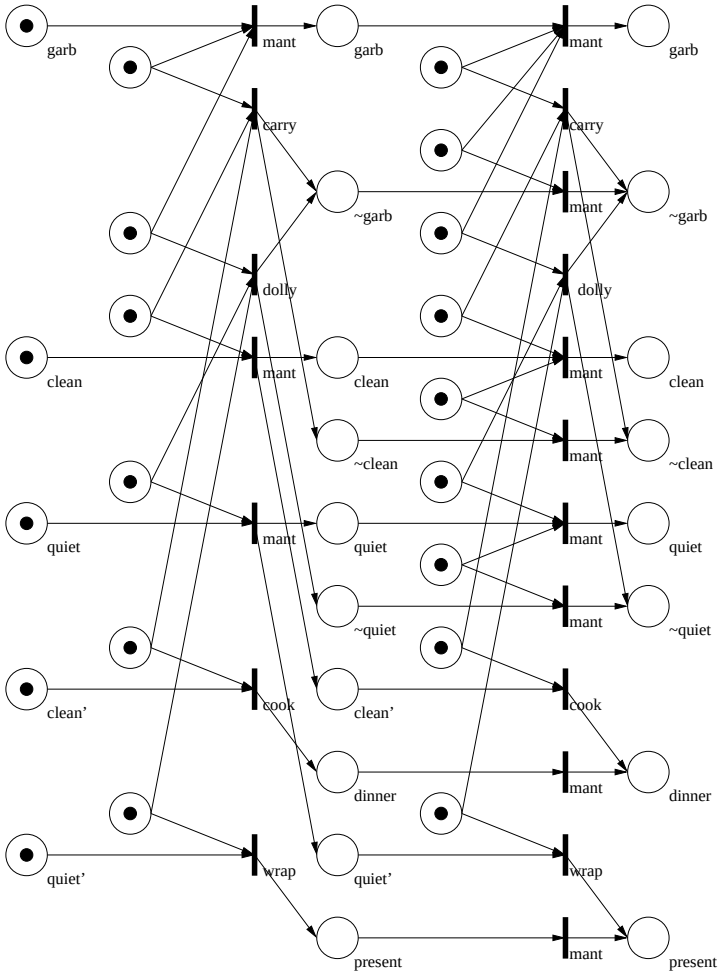


Figura 26: A tradução alternativa da rede da figura 20.

4.2.2 Análise

O esquema alternativo de tradução das proposições e pré-condições do grafo pode ser usado diretamente no algoritmo *Petriplan* (seção 4.1), ou seja, o problema de programação inteira resultante (seção 4.1.2) é resolvido da mesma forma e o plano é obtido segundo o

mesmo procedimento (seção 4.1.3).

O tamanho do problema de programação inteira obtido a partir da rede Petri é dado pelo número de restrições e variáveis deste problema. Baseado no processo de tradução do grafo de planos para rede Petri apresentado na seção 4.1.1, o número de restrições do problema de programação inteira r é igual ao número de lugares da rede, dado por:

$$r = n_p + a_p + m,$$

onde n_p é o número de nós-proposição, a_p é o número de arestas-pré-condição e m o número de relações de exclusão mútua. O número de variáveis do problema de programação inteira v é igual ao número de transições da rede, dado por:

$$v = n_p + n_a,$$

onde n_p é o número de nós-proposição e n_a é o número de nós-ação.

No caso da tradução alternativa apresentada na seção 4.2.1 o número de restrições do problema de programação inteira é dado por:

$$r = a_p + m,$$

e o número de variáveis é dado por:

$$v = n_a.$$

Nos dois casos o tamanho do problema de programação inteira é linearmente proporcional ao tamanho do grafo de planos do problema original de planejamento. Vale notar que o esquema alternativo de tradução resulta em um problema de programação inteira equivalente mas de tamanho menor, onde apenas as pré-condições são representadas como lugares e apenas as ações como transições.

4.3 Simplificações do modelo

Esta seção trata da simplificação da rede resultante do processo de tradução através da remoção de estruturas, com o objetivo de obter uma rede menor e conseqüentemente um problema de programação inteira menor.

4.3.1 Estruturas equivalentes

É fácil verificar que o esquema de tradução apresentado na seção 4.2 gera redes que possuem algumas estruturas que podem ser simplificadas.

A figura 27 mostra um caso onde uma transição tem apenas um lugar como efeito e este lugar é a única pré-condição de apenas uma transição. No caso da figura, as transições a e b e o lugar que as liga podem ser ser substituídos por um única transição $a;b$, que representa o disparo de a seguido do disparo de b.

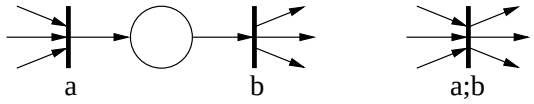


Figura 27: Simplificação da rede.

Numa rede de Petri qualquer esta simplificação implicaria perda de informação, pois, o estado intermediário entre o disparo de a e b não é mais representado. Entretanto, no caso do problema de alcançabilidade resultante este tipo de simplificação não causa problemas.

Este tipo de simplificação reduz a rede de um lugar e uma transição. Assim o problema de programação inteira é reduzido de uma restrição (lugar) e uma variável (transição).

4.3.2 Remoção de mutex

Devido à dinâmica do fluxo de marcas da rede de Petri, nem todas as relações de exclusão mútua são necessárias, ou seja, mesmo removendo alguns lugares que representam estas relações é possível obter o mesmo fluxo na rede.

Um dos casos de mutex entre ações no grafo de planos é quando alguma pré-condição p de uma ação é mutuamente exclusiva a alguma pré-condição q da outra ação. Isto ocorre quando todas as ações que tem p como efeito são mutuamente exclusivas às que tem q como efeito. Os lugares que representam este tipo de mutex podem ser removidos da rede. Entretanto, os outros tipos de mutex do grafo de planos não podem ser removidos da rede, pois isto permitiria o disparo de transições conflitantes.

A figura 28 mostra um exemplo deste tipo de simplificação. Note que os caminhos que levam aos lugares p e q são mutuamente exclusivos, tornando desnecessário o mutex entre a e b. Os caminhos que levam às ações a e b já são mutuamente exclusivos e o controle é

Tabela 1: Número de transições da rede de Petri.

Problema	Original	OrigMutex	Alternativa	AlterMutex
Gripper-1	1071	1071	766	766
Gripper-2	1667	1667	1204	1204
Gripper-3	2221	2221	1612	1612
Logistics	2057	2057	1408	1408
Rovers-1	4961	4961	3208	3208
Rovers-2	8825	8825	6151	6151
Satellite-1	5506	5506	4906	4906
Satellite-2	6478	6478	5864	5864

feito pela dinâmica da rede.

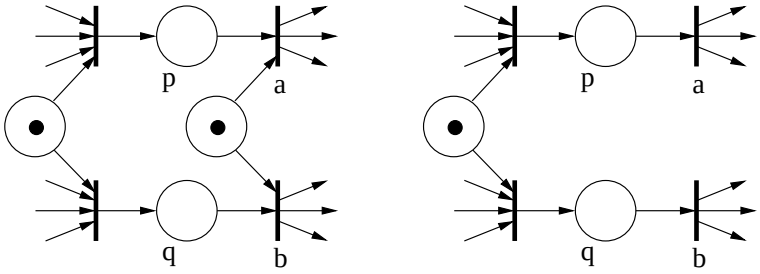


Figura 28: Simplifição de mutex na rede.

4.4 Experimentos realizados

Foram realizados alguns experimentos para avaliar o tamanho das representações em redes de Petri para problemas de planejamento, considerando algumas variações do *Petriplan*. Quatro variações do algoritmo foram comparadas: a tradução original, a tradução original com remoção de mutex, a tradução alternativa e a tradução alternativa com remoção de mutex.

As implementações foram realizadas sobre a plataforma IPE, que é uma ferramenta para a construção e avaliação de planejadores. A plataforma IPE e uma comparação mais completa das versões citadas do *Petriplan* podem ser encontradas em [Mar04].

A tabelas 1, 2, 3 apresentam o tamanho das redes obtidas pelas quatro variações implementadas para alguns problemas clássicos de planejamento usados nas competições de planejadores de 1998 e 2000 [McD98a, FL00].

O número de transições da rede é reduzido com o esquema de tradução alternativo, o

Tabela 2: Número de lugares da rede de Petri.

Problema	Original	OrigMutex	Alternativa	AlterMutex
Gripper-1	16228	11846	15923	11541
Gripper-2	32124	26020	31661	25557
Gripper-3	51650	40803	51041	40194
Logistics	127614	108471	126965	107822
Rovers-1	120388	96310	118635	94557
Rovers-2	439286	351428	436612	348754
Satellite-1	88609	71773	88009	71173
Satellite-2	134739	109138	134125	108524

Tabela 3: Número de mutex da rede de Petri.

Problema	Original	OrigMutex	Alternativa	AlterMutex
Gripper-1	14485	10103	14485	10103
Gripper-2	29369	23265	29369	23265
Gripper-3	51925	41078	51925	41078
Logistics	124381	105238	124381	105238
Rovers-1	109112	85034	109112	85034
Rovers-2	416736	328878	416736	328878
Satellite-1	85107	68271	85107	68271
Satellite-2	131448	105847	131448	105847

que resulta em problema de programação inteira com um número menor de variáveis, em média uma redução de 25%. Entretanto, o número de lugares da rede não apresenta uma redução significativa, pois o principal componente deste número é dado pelo número de mutex do modelo.

O número de relações de exclusões mútua em ambas as traduções é equivalente, ou seja, a tradução alternativa não afeta o número de mutex. A redução de mutex não afeta o número de transições do modelo. Vale notar que na tabela 3 a redução chega a aproximadamente 20% do total de mutex da rede.

A aplicação da codificação alternativa em conjunto com a redução de mutex resulta em um modelo mais compacto para o problema.

4.5 Considerações

Neste capítulo foi apresentada uma abordagem para o problema de planejamento em *STRIPS* que une o formalismo de redes de Petri a problemas de programação inteira.

Usando a teoria de redes de Petri mostrou-se como definir o problema de planejamento como um problema de encontrar uma seqüência de disparos que resolve o problema de alcançabilidade de sub-marcação em uma rede de Petri. Técnicas de programação inteira foram usadas para resolver o problema de alcançabilidade, obtendo a partir desta solução o plano que resolve o problema original de planejamento.

O algoritmo proposto é baseado na mesma estratégia do planejador *Blackbox*, onde o problema de planejamento é traduzido para um problema de satisfabilidade para então ser tratado por um resolvedor SAT. No caso do *Petriplan*, o problema é traduzido para um problema de alcançabilidade em uma rede de Petri.

Os procedimentos de tradução e recuperação da solução apresentados são bastante simples, ficando a complexidade do algoritmo diretamente ligada à criação do grafo de planos e à solução do problema de programação inteira.

As redes obtidas pelos dois esquemas de tradução apresentados podem ser simplificadas com a remoção de estruturas redundantes presentes no grafo de planos. Desta forma, a representação do problema de planejamento por redes de Petri, em termos de espaço, é mais eficiente que o grafo de planos. A principal vantagem desta representação é que as relações de exclusão mútua fazem parte da estrutura do modelo e não meta-informação como no caso do grafo de planos.

Os experimentos realizados mostram uma redução significativa no tamanho da rede quando a tradução alternativa proposta é combinada com a remoção de relações de exclusão mútua redundantes. Esta redução tem como conseqüência direta um problema de programação inteira mais compacto.

Para a implementação do algoritmo foi usada a plataforma IPE. O grafo foi traduzido para uma representação matricial da rede de Petri, armazenada em uma matriz esparsa, que foi usada na definição do problema de programação inteira da terceira fase. O problema de programação inteira foi resolvido usando-se a biblioteca *CPLEX* [ILO00] para programação inteira. A recuperação da solução foi implementada a partir de um simples caminhamento na matriz de incidência da rede e no grafo de planos.

5 Rede de Planos

Os diferentes tipos de modelagem do problema de planejamento através de redes de Petri apresentados no capítulo 4 derivam diretamente do grafo de planos. Entretanto, os modelos obtidos apresentam em geral estruturas redundantes com um grande potencial de simplificação.

Este capítulo trata de um modelo para o problema de planejamento que explora melhor o potencial das redes de Petri, pela construção da rede diretamente da descrição do problema em PDDL [McD98b]. O processo de construção resulta numa rede acíclica, como nos modelos apresentados no capítulo 4, mantendo-se dessa forma as mesmas estratégias para o processo de busca da solução do problema. A rede obtida é chamada *rede de planos*.

A seguir são apresentadas as estruturas básicas usadas na construção deste novo modelo, um exemplo da sua utilização e uma análise do modelo obtido.

5.1 Estrutura básica do modelo

A partir da descrição do problema de planejamento são definidos dois conjuntos $\mathcal{P}$ e $\mathcal{A}$ contendo, respectivamente, todas as proposições instanciadas e todas as ações instanciadas do problema. A instanciação é feita a partir da aplicação das constantes do problema sobre as descrições das ações e proposições do domínio. O conjunto das ações pode ser visto como um relacionamento entre dois conjuntos de proposições:

$$\mathcal{A} = \{\ (C, E)\ \},$$

onde, $C, E \subseteq \mathcal{P}$. Cada ação do conjunto $\mathcal{A}$ é dada pelo par ordenado de dois subconjuntos de proposições, o das suas pré-condições C e o dos seus efeitos E.

Considere a ação a, "$move(r_1, r_2)$", dada por $(\{c\}, \{e', e''\}) \in \mathcal{A}$, onde a pré-condição $c \in C_a$, é dada por "$at(r_1)$", e os efeitos $e', e'' \in E_a$, são dados por "$at(r_2)$" e "$\neg at(r_1)$"

respectivamente.

A figura 29 apresenta a estrutura de representação da ação e das proposições na rede de Petri. Com esta estrutura tem-se as transições da rede representando as ações de $\mathcal{A}$ e os lugares representando as pré-condições e os efeitos das ações, que são as proposições presentes em $\mathcal{P}$. Os lugares que representam as pré-condições da ação serão chamados de *pré-condições da transição* e os que representam efeitos, *efeitos da transição*. Note que o efeito $\neg at(r_1)$ na figura está tracejado pois efeitos negativos não serão representados no modelo. Os efeitos negativos são úteis apenas para o cálculo das relações de conflito entre as ações. A opção pela não representação das proposições negativas é uma prática comum nos planejadores baseados na linguagem *STRIPS*.

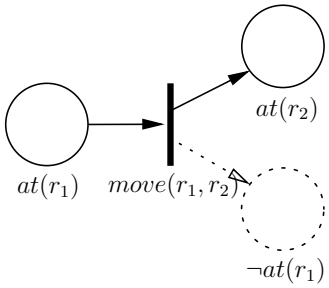


Figura 29: Estrutura de representação da ação "$move(r_1, r_2)$".

A construção da rede de planos inicia com a inclusão de um lugar para cada proposição do estado inicial do problema.

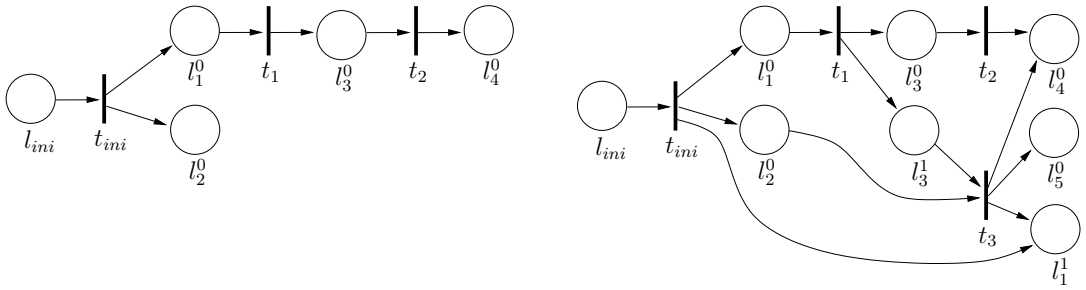
Seja $R = (L, T, I_{pre}, I_{pos})$ a rede de planos e $S_0 \subseteq \mathcal{P}$ o estado inicial do problema. Inicialmente a rede de planos R é dada por um lugar inicial l_{ini}, por uma transição inicial t_{ini} e pelos lugares representando as proposições do estado inicial:

$$\begin{aligned} L &= \{l_{ini}\} \cup \{l_i^0 \mid i \in S_0\} \\ T &= \{t_{ini}\} \\ I_{pre} &= \{(l_{ini}, t_{ini}) \mapsto 1\} \\ I_{pos} &= \{(l_i^0, t_{ini}) \mapsto 1 \mid i \in S_0\} \end{aligned} \tag{5.1}$$

onde os índices i e k do lugar l_i^k são, respectivamente, a proposição associada e o índice que identifica cada cópia de l_i, sendo l_i^0 a primeira ocorrência de l_i.

Assim como no grafo de planos, o próximo passo na construção do modelo consiste da inclusão das ações que têm suas pré-condições presentes no modelo. Considere, como

exemplo, a rede da figura 30(a) e uma transição t_3, que tem como pré-condições l_2 e l_3 e como efeitos l_1, l_4 e l_5. A inclusão de t_3 na rede da figura 30(a) resulta na rede da figura 30(b).



(a) (b)

Figura 30: Inclusão da transição t_3 e dos lugares l_3^1, l_1^1 e l_5^0.

Observe que as duas pré-condições da transição estão presentes na rede, o que permite a inclusão de t_3. O lugar l_2 não é utilizado por nenhuma outra transição da rede como pré-condição, já o lugar l_3 é pré-condição para t_2. Portanto, é necessária a cópia do lugar l_3 para que não haja conflito entre o disparo de t_2 e t_3. Note que a função I_{pos} para l_3 também é copiada.

Os efeitos que não estão presentes na rede são incluídos, como o caso de l_5. Para os efeitos que estão presentes e não são usados como pré-condição, a função I_{pos} da transição é atualizada, ligando t_3 à l_4^0. Para efeitos que já são usados como pré-condição é necessária a cópia do lugar e de sua função I_{pos}, como no caso de l_1^1.

Seja t_a uma transição que representa a ação $a = (C_a, E_a)$, a inclusão de t_a no modelo pode requerer que alguns lugares sejam copiados para evitar conflitos na rede R. L_c é o conjunto de cópias de lugares da rede que são usados como pré-condição:

$$L_c \;=\; \{l_i^{\mathsf{m}+1} \mid \exists t_b \;(\; I_{pre}(l_i^{\mathsf{m}}, t_b) > 0 \;)\; \wedge \; l_i^{\mathsf{m}} \in L \; \wedge \; i \in C_a\} \tag{5.2}$$

onde:

$$\mathsf{m} = \begin{cases} -1 & \text{se} \quad l_i^0 \notin L, \\ k & \text{se} \quad l_i^k \in L \;\wedge\; \forall l_i^x \;(x \leq k \;\wedge\; l_i^x \in L) \end{cases} \tag{5.3}$$

Ou seja, m é o maior índice das cópias de $l_i \in L$ ou -1 no caso de l_i não pertencer à rede. O conjunto L_c é o conjunto dos lugares que são pré-condição para a transição t_a e que já são pré-condição para alguma transição t_b da rede.

O conjunto L_{ec} contém os lugares associados aos efeitos E_a que serão incluídos como cópias de outros lugares da rede R que já são usados como pré-condição por alguma transição. L_{en} é o conjunto dos lugares que são efeito da transição t_a e que ainda não estão presentes na rede R. L_{ec} e L_{en} são dados por:

$$L_{ec} = \{l_j^{\mathsf{m}+1} \mid \exists t_b \ (\ I_{pre}(l_j^{\mathsf{m}}, t_b) > 0\ ) \ \wedge \ l_j^{\mathsf{m}} \in L \ \wedge \ j \in E_a\} \tag{5.4}$$

$$L_{en} = \{l_j^0 \mid l_j^0 \notin L \ \wedge \ j \in E_a\}, \tag{5.5}$$

onde m é o maior índice das cópias de l_i, dado pela equação 5.3.

A inclusão da transição t_a na rede R, resulta na rede de planos $R' = (L', T', I'_{pre}, I'_{pos})$ dada por:

$$\begin{aligned}
L' &= L \ \cup \ L_c \ \cup \ L_{ec} \ \cup \ L_{en} \qquad (5.6)\\
T' &= T \ \cup \ \{t_a\}\\
I'_{pre} &= I_{pre} \ \cup \ \{(l_i^x, t_a) \mapsto 1 \mid l_i^x \in L_c \ \wedge \ i \in C_a\}\\
&\quad \cup \{(l_i^{\mathsf{m}}, t_a) \mapsto 1 \mid l_i^{\mathsf{m}} \in L \ \wedge \ l_i^x \notin L_c \ \wedge \ x > \mathsf{m} \ \wedge \ i \in C_a\}\\
I'_{pos} &= I_{pos} \ \cup \ \{(l_i^x, t_a) \mapsto 1 \mid l_i^x \in (L_{ec} \cup L_{en}) \ \wedge \ i \in E_a\}\\
&\quad \cup \{(l_i^{\mathsf{m}}, t_a) \mapsto 1 \mid l_i^{\mathsf{m}} \in L \ \wedge \ l_i^x \notin L_{ec} \ \wedge \ x > \mathsf{m} \ \wedge \ i \in E_a\}\\
&\quad \cup \{(l_i^x, t_b) \mapsto 1 \mid I_{pos}(l_i^{\mathsf{m}}, t_b) > 0 \ \wedge \ l_i^{\mathsf{m}} \in L \ \wedge \ l_i^x \in L_c \ \wedge \ i \in C_a \ \wedge \ t_b \in T\}\\
&\quad \cup \{(l_i^x, t_b) \mapsto 1 \mid I_{pos}(l_i^{\mathsf{m}}, t_b) > 0 \ \wedge \ l_i^{\mathsf{m}} \in L \ \wedge \ l_i^x \in L_{ec} \ \wedge \ i \in E_a \ \wedge \ t_b \in T\}
\end{aligned}$$

sendo m o maior índice das cópias de l_i presentes em L (equação 5.3) e t_b uma transição da rede original.

O conjunto de lugares da rede resultante L' é dado pelos seguintes elementos: os lugares da rede original; as cópias dos lugares necessários para a inclusão da transição (L_c e L_{ec}); e os novos efeitos gerados pela transição (L_{en}). O conjunto de transições T' é dado pela inclusão da transição t_a na rede original. A função de incidência I'_{pre} é dada por: a função original I_{pre}; os pares que ligam t_a aos lugares copiados que estão em L_c; e os pares que ligam t_a aos lugares da rede original que são pré-condição de t_a e não estão em L_c. A função de incidência I'_{pos} é dada por: a função original I_{pos}; os pares que ligam t_a aos seus efeitos novos (L_{en}) ou copiados (L_{ec}); os pares que ligam t_a aos lugares da rede original que são efeitos de t_a e não foram copiados; e os pares que ligam os lugares copiados de L_c e L_{ec} a todas as transições t_b que têm como efeito algum destes lugares na rede original.

A inclusão da transição t_3 na rede da figura 30(a) pode ser descrita por:

$$\begin{aligned}
L' &= \{l_{ini},\ l_1^0,\ l_2^0,\ l_3^0,\ l_4^0\} \cup \{l_3^1\} \cup \{l_1^1\} \cup \{l_5^0\} \\
T' &= \{t_{ini},\ t_1,\ t_2\} \cup \{t_3\} \\
I'_{pre} &= \{(l_{ini}, t_{ini}) \mapsto 1,\ (l_1^0, t_1) \mapsto 1,\ (l_3^0, t_2) \mapsto 1\} \\
&\quad \cup \{(l_2^0, t_3) \mapsto 1\} \\
&\quad \cup \{(l_3^1, t_3) \mapsto 1\} \\
I'_{pos} &= \{(l_1^0, t_{ini}) \mapsto 1,\ (l_2^0, t_{ini}) \mapsto 1,\ (l_3^0, t_1) \mapsto 1,\ (l_4^0, t_2) \mapsto 1\} \\
&\quad \cup \{(l_1^1, t_3) \mapsto 1,\ (l_5^0, t_3) \mapsto 1\} \\
&\quad \cup \{(l_4^0, t_3) \mapsto 1\} \\
&\quad \cup \{(l_3^1, t_1) \mapsto 1\} \\
&\quad \cup \{(l_1^1, t_{ini}) \mapsto 1\}
\end{aligned}$$

onde $L_c = \{l_3^1\}$, $L_{ec} = \{l_1^1\}$ e $L_{en} = \{l_5^0\}$.

A construção da rede que representa o problema de planejamento a partir da configuração inicial, dada pela equação 5.1, ocorre de maneira semelhante à construção do grafo de planos. Após a representação do estado inicial do problema, são incluídas na rede todas as transições que têm suas pré-condições presentes na rede com as cópias necessárias dos lugares pré-condição. Em seguida, os efeitos das transições são incluídos, sejam novos ou cópias de outros lugares da rede original. As equações 5.2 e 5.4 definem as cópias de lugares necessárias para evitar conflitos na inclusão de uma nova transição na rede. As equações 5.6 definem a inclusão da transição, dos lugares copiados e a atualização das funções de incidência.

As ações incluídas na rede, as cópias dos lugares e os novos lugares incluídos como efeito das ações constituem uma *camada* da rede de planos. Assim como no grafo de planos, todas as transições de uma mesma camada da rede de planos são concorrentes. Entretanto, as ações podem apresentar inconsistências entre si que devem ser representadas no modelo para evitar que planos com ações conflitantes sejam gerados.

A próxima seção trata das inconsistências entre as ações de uma mesma camada e apresenta estruturas da rede capazes de controlar o disparo das transições evitando as inconsistências.

5.2 Relações de ordenação

A inclusão de uma nova transição pode resultar em inconsistências na rede, por exemplo, quando uma ação invalida a pré-condição de outra. Assim é necessário o estabelecimento de relações de ordem ou exclusão mútua entre transições da rede.

Duas transições t_a e t_b podem apresentar os seguintes tipos de inconsistências:

Tipo 1: t_a tem como efeito a negação de algum efeito de t_b.

Tipo 2: t_b tem como efeito a negação de alguma pré-condição de t_a;

Tipo 3: t_b tem como efeito a negação de alguma pré-condição de t_a que por sua vez tem como efeito a negação de alguma pré-condição de t_b;

A partir destas inconsistências pode-se definir as seguintes relações de ordenação entre os disparos de duas transições t_a e t_b:

- $t_a \parallel t_b$ (t_a e t_b são concorrentes): quando não existem inconsistências entre as transições, todas as seqüências resultantes da concorrência entre t_a e t_b são válidas:
  - o disparo concorrente t_a e t_b;
  - apenas o disparo de t_a;
  - apenas o disparo de t_b;
  - o disparo de t_a seguido do disparo de t_b;
  - o disparo de t_b seguido do disparo de t_a;
  - nenhum disparo.
- $t_a \nparallel t_b$ (t_a e t_b são não-concorrentes): quando apenas o disparo concorrente das transições gera inconsistência (inconsistência tipo 1). Neste caso os seguintes disparos são válidos:
  - apenas o disparo de t_a;
  - apenas o disparo de t_b;
  - o disparo de t_a seguido do disparo de t_b;
  - o disparo de t_b seguido do disparo de t_a;

  - nenhum disparo.

- $t_a \triangleleft t_b$ (t_a precede t_b): quando as transições apresentam inconsistência do tipo 2, esta inconsistência pode ser resolvida somente com a antecipação do disparo da transição t_a. Portanto, os disparos possíveis são:
  - apenas o disparo de t_a;
  - apenas o disparo de t_b;
  - o disparo de t_a seguido do disparo de t_b;
  - nenhum disparo.

- $t_a \diamond t_b$ (t_a e t_b são mutuamente exclusivos, ou mutex): quando as transições apresentam inconsistência do tipo 3, neste caso os disparos são mutuamente exclusivos, sendo possíveis:
  - apenas o disparo de t_a;
  - apenas o disparo de t_b;
  - nenhum disparo.

As relações de ordenação indicam como os conflitos entre as ações podem ser resolvidos. As seqüências de disparo permitidas por cada relação são a base para a construção de estruturas na rede planos que controlam os disparos das transições, evitando assim os conflitos entre as ações.

5.2.1 Estruturas de controle

A figura 31 mostra a estrutura de controle de disparo para uma transição na rede de planos. O disparo da transição t numa camada da rede de planos representa a execução da ação correspondente em determinado ponto do plano. Entretanto, se esta ação não faz parte da solução ou é executada em outro ponto do plano, o disparo de t deve ser impedido, ou seja, t não pode estar habilitada. A transição θ_t indica o início do controle do disparo de t, os lugares I e F representam as fronteiras da camada, a marca presente em I indica o início da execução das ações nesta camada. O lugar l_1 é um conflito na rede que permite apenas o disparo de uma das transições: t ou λ_t, ou seja, ou a ação representada por t está presente no plano ou não está. A transição ϕ_t representa o fim do controle do disparo da transição t e o lugar F indica o fim da execução das ações da camada.

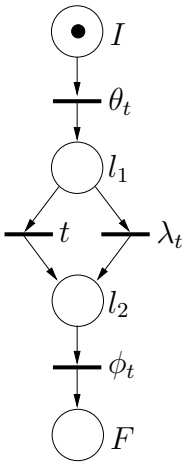


Figura 31: Estrutura de controle de disparo para uma transição da rede.

A estrutura de controle apresentada para uma única ação pode ser extendida para controlar o disparo de transições concorrentes ou que apresentem algum tipo de inconsistência. As figuras 32, 33, 34 e 35 mostram, respectivamente, as estruturas de controle para as relações de ordenação $a \shortparallel b$, $a \nparallel b$, $a \lhd b$ e $a \diamond b$. As regiões tracejadas destas figuras mostram a estrutura de controle da ação a sendo usada como um componente na composição das estruturas de controle para as relações de ordenação.

Vale observar que os lugares F das figuras 32, 33, 34 e 35 indicam, quando marcados, que mais nenhuma transição controlada por estas estruturas poderá ser disparada. Assim, caso uma determinada transição que representa uma ação não seja disparada, obrigatoriamente a transição λ correspondente é disparada, indicando que a ação não faz parte do plano.

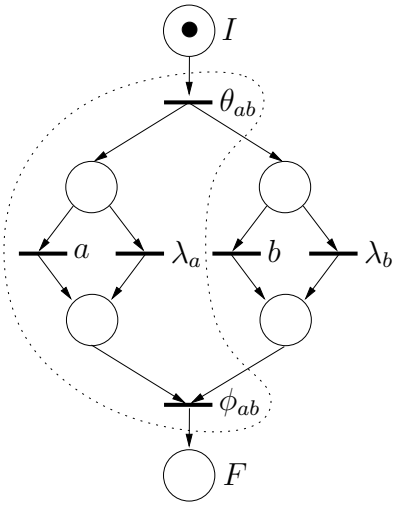


Figura 32: Estrutura de controle de disparo para $a \parallel b$.

A estrutura de controle da figura 32 permite a execução dos seguintes sub-planos no problema de planejamento: λ, (a), (b), $(a|b)$, $(a;b)$ e $(b;a)$. Onde λ indica um plano vazio, (x) indica que o plano é composto pela ação x, $(x;y)$ indica que o plano é dado pela ação x seguida da ação y e $(x|y)$ indica que as ações x e y são concorrentes no plano. As transições θ_{ab} e ϕ_{ab} representam, respectivamente, o início e o fim da concorrência entre as ações a e b no plano.

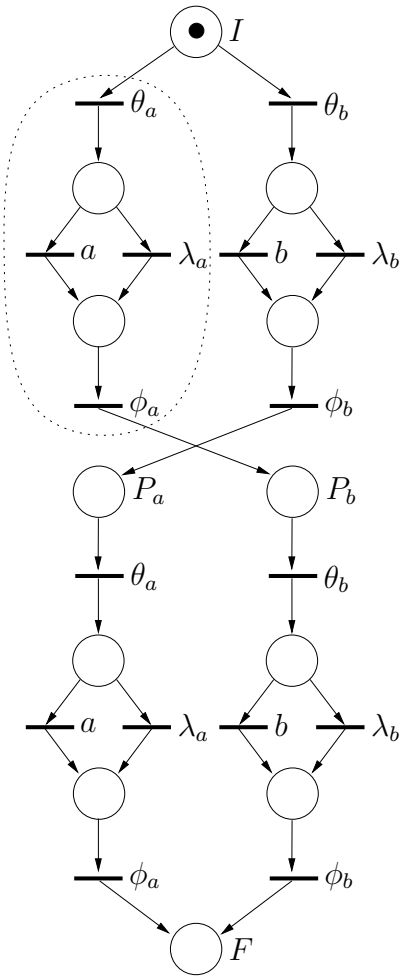


Figura 33: Estrutura de controle de disparo para $a \curlywedge b$.

A estrutura de controle da figura 33 permite a execução dos seguintes sub-planos no problema de planejamento: λ, (a), (b), $(a; b)$ e $(b; a)$. Vale notar que neste caso as estruturas de controle das ações a e b aparecem repetidas, o que força a serialização das ações. Os lugares P_a e P_b são responsáveis pelo encadeamento e ordenação das ações na estrutura de controle.

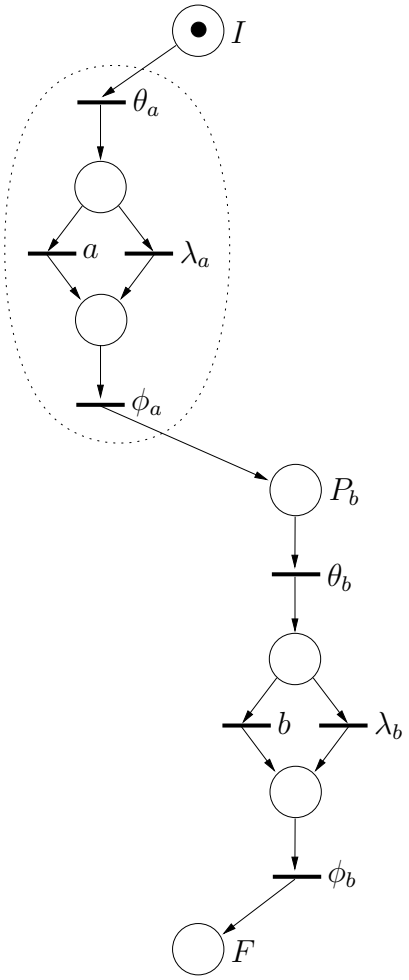


Figura 34: Estrutura de controle de disparo para $a \triangleleft b$.

A estrutura de controle da figura 34 permite a execução dos seguintes sub-planos no problema de planejamento: λ, (a), (b) e $(a; b)$. O lugar P_b é o responsável pela ordenação da ação.

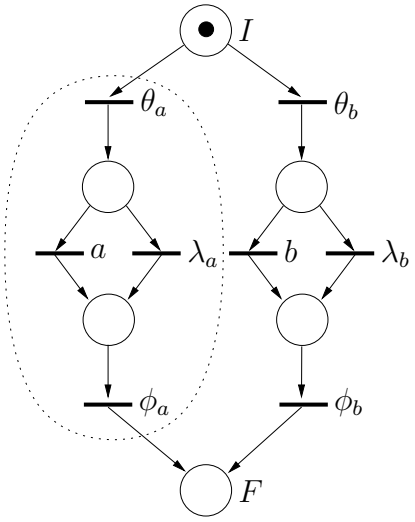


Figura 35: Estrutura de controle de disparo para $a \diamond b$.

A estrutura de controle da figura 35 permite a execução dos seguintes sub-planos no problema de planejamento: λ, (a) e (b).

As estruturas básicas de controle apresentadas nas figuras 32, 33, 34 e 35 podem ser compostas para controlar mais de uma relação de ordenação sobre um conjunto de ações. O objetivo é construir, para o conjunto de ações de cada camada do modelo, uma única estrutura de controle.

Suponha uma camada do modelo com quatro ações: a, b, c e d. E as sequintes relações de ordenação entre estas ações: $a \nmid b$, $a \nmid c$, $a \diamond d$, $b \diamond c$, $c \triangleleft d$ e $d \triangleleft b$. A figura 36 apresenta a composição das estruturas de controle para cada uma das relações em uma única rede.

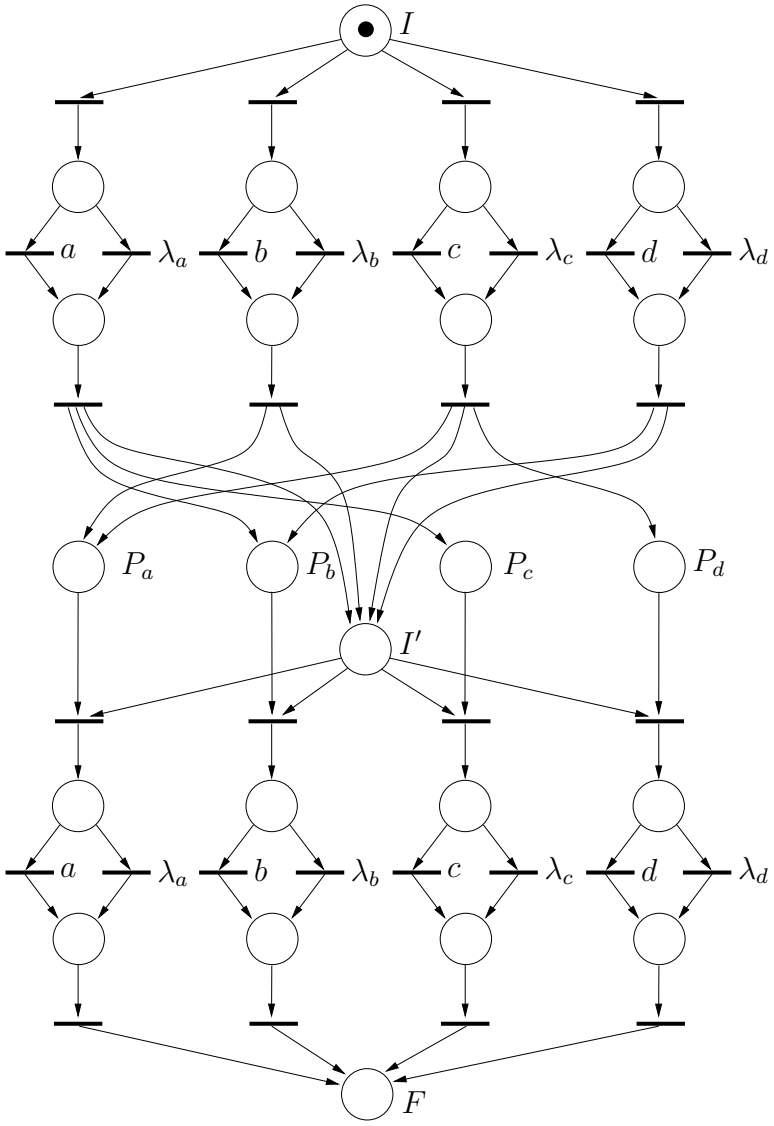


Figura 36: Estrutura de controle de disparo para a composição das relações $a \nmid b$, $a \nmid c$, $a \diamond d$, $b \diamond c$, $c \triangleleft d$ e $d \triangleleft b$.

A composição das estruturas de controle na figura 36 pode resultar em mais de um caminho disparável na rede. Para evitar este problema é incluído na estrutura um lugar de sincronização I' que impede que mais de uma sub estrutura seja executada na parte inferior da rede.

Esta estrutura de controle permite as seguintes seqüências de disparos: λ, (a), (b), (c), (d), $(a;b)$, $(a;c)$, $(b;a)$, $(c;a)$, $(c;d)$ e $(d;b)$.

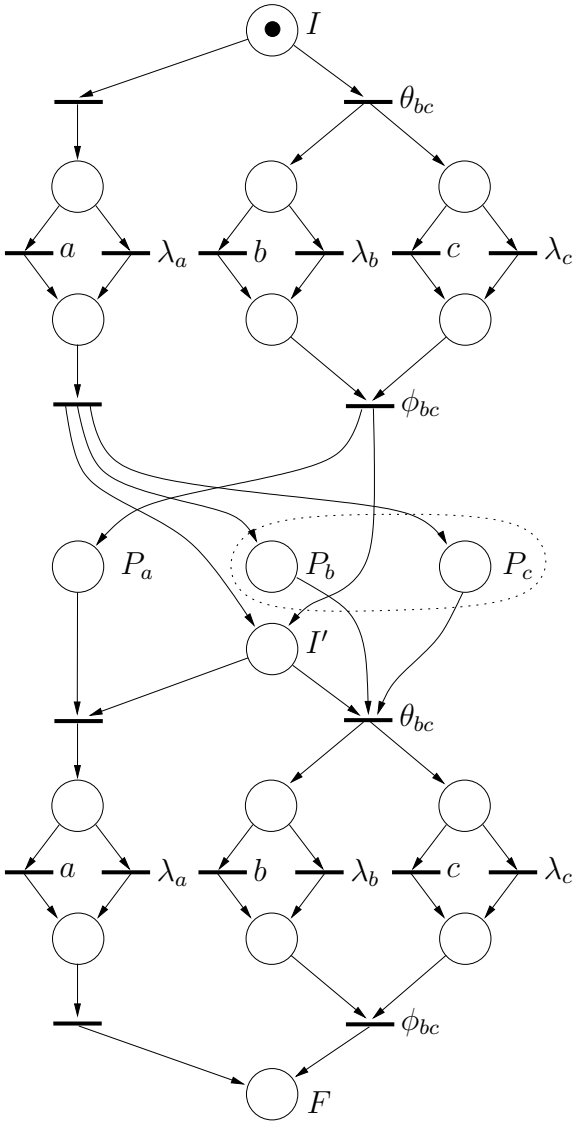


Figura 37: Estrutura de controle de disparo para a composição das relações $a \curlywedge b$, $a \curlywedge c$ e $b \shortparallel c$.

A figura 37 mostra um exemplo de composição com um par de ações concorrentes. Os sub-planos permitidos nesta estrutura de controle são: λ, (a), (b), (c), $(a;b)$, $(a;c)$, $(a;(b|c))$, $(b;a)$, $(b;c)$, $(c;a)$, $(c;b)$ e $((b|c);a)$. A região tracejada da figura mostra uma redundância do modelo para o caso de ações concorrentes. No caso deste exemplo esta redundância pode ser simplificada sem comprometer a estrutura de controle. Entretanto, no exemplo da figura 38 este tipo de simplificação não é possível.

Considere, por exemplo, um plano dado por $(a;((b;c)|d);e;(f|g|h))$. O número de ações deste plano é oito. O número de *passos* deste plano é cinco, pois são necessários no mínimo

cinco eventos temporais distintos para a execução do plano. Por exemplo, este plano pode ser executado nestes cinco passos: a, $(b|d)$, c, e e $(f|g|h)$.

No grafo de planos e nos modelos apresentados no capítulo 4 os conflitos entre as ações de uma mesma camada são resolvidos por relações de exclusão mútua que permitem apenas a execução de ações concorrentes na camada. Desta forma, cada camada representa um passo no plano que resolve o problema. Já as estruturas de controle apresentadas permitem que cada camada da rede de planos represente até dois passos no plano. Apesar das relações de ordenação da figura 37 permitirem as seqüências $(b; a; c)$ e $(c; a; b)$, elas não são possíveis na estrutura de controle, pois são planos com três passos.

A composição de relações do tipo $\shortparallel$ com outras relações pode resultar em estruturas de controle inconsistentes, além daquelas controladas pela inclusão do lugar I', que permitem o disparo de seqüências de ações que apresentam inconsistências. A figura 38(a) mostra uma estrutura de controle onde é possível obter uma marca no lugar P_a partindo do disparo de θ_{bc}. Ou seja, é possível iniciar o controle da relação $b \shortparallel c$ em θ_{bc} e terminá-lo em ϕ_b, o que pode resultar em sub-planos inconsistentes, pois ϕ_b pertence neste caso ao controle da relação $a \nvdash b$. O mesmo pode ocorrer entre as estruturas de $b \shortparallel c$ e $c \triangleleft d$. Este problema pode ser resolvido com a inclusão de mais um lugar de controle nas estruturas tracejadas, o que impede que o controle de uma relação termine na estrutura de controle de outra. A figura 38(b) mostra a inclusão do lugar Q que resolve este tipo de inconsistência.

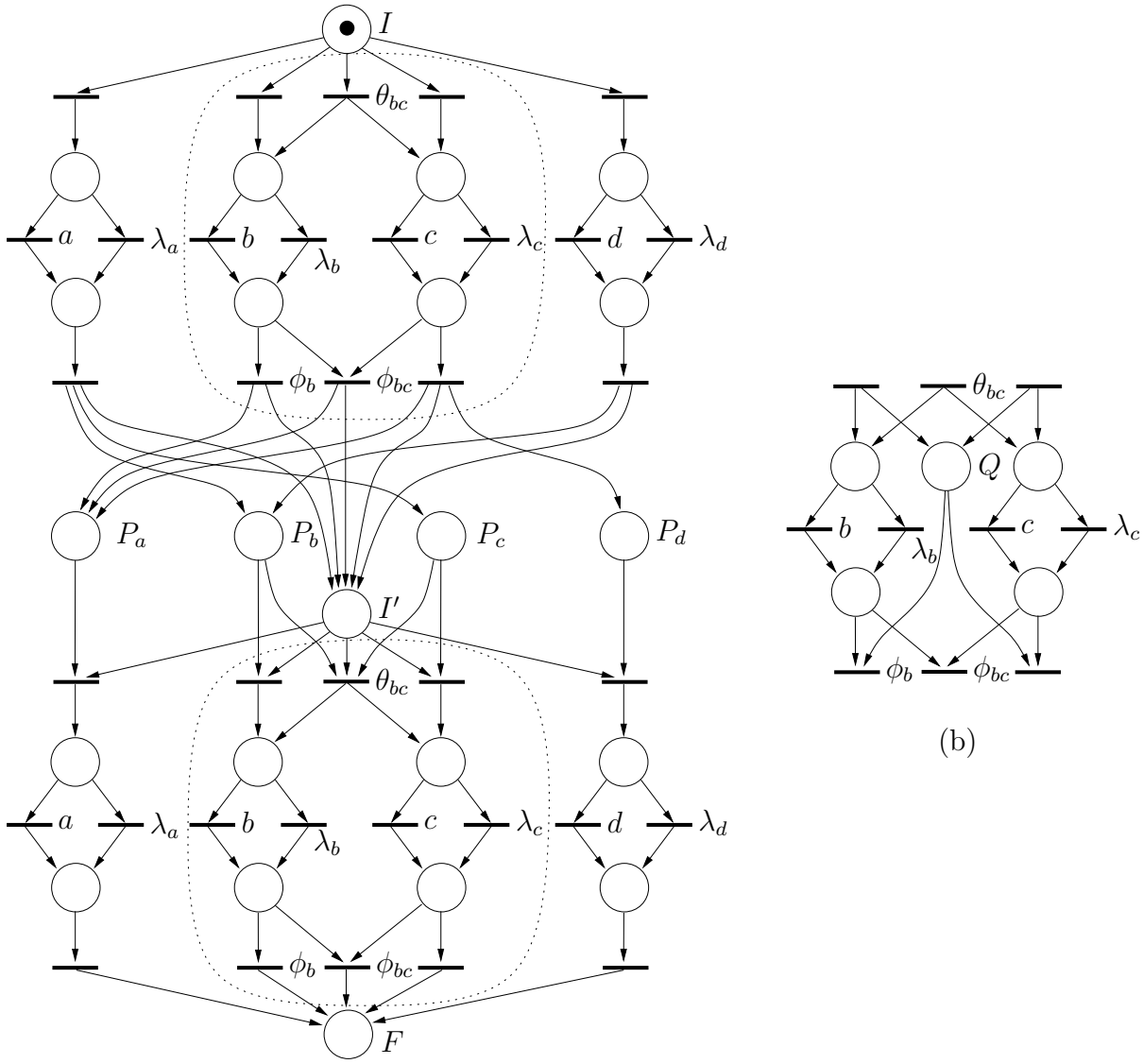


Figura 38: Estrutura de controle de disparo para a composição das relações $a \rightleftharpoons b$, $a \rightleftharpoons c$, $a \diamond d$, $b \shortparallel c$, $c \triangleleft d$ e $d \triangleleft b$.

A figura 38(a) apresenta um exemplo completo de estrutura de controle para as relações $a \rightleftharpoons b$, $a \rightleftharpoons c$, $a \diamond d$, $b \shortparallel c$, $c \triangleleft d$ e $d \triangleleft b$. Neste exemplo as seqüências possíveis de disparos são: λ, (a), (b), (c), (d), $(a;b)$, $(a;c)$, $(a;(b|c))$, $(b;a)$, $(c;a)$, $(c;d)$, $(d;b)$ e $((b|c);a)$.

Os lugares presentes nas estruturas são apenas para controle do fluxo de disparos e não representam pré-condições ou efeitos das ações. Entretanto as transições que representam ações no modelo estão ligadas as suas respectivas pré-condições e efeitos e não foram repre-

sentadas nas figuras apresentadas para facilitar a visualização das estruturas de controle. As transições que representam ações e aparecem mais de uma vez na mesma camada são cópias que compartilham as mesmas pré-condições e efeitos.

As quatro relações apresentadas ($\parallel$, $\nparallel$, $\lhd$, $\diamond$) definem as ordenações possíveis entre duas transições quaisquer de uma camada. Nos modelos de representação derivados diretamente do grafo de planos apenas as relações $\parallel$ e $\diamond$ são calculadas durante a construção da representação, ou seja, duas ações no modelo podem ser ou totalmente independentes ou mutuamente exclusivas. As demais relações são obtidas pela expansão do grafo com camadas adicionais e pela aplicação das relações de exclusão mútua sobre as ações que apresentam inconsistências.

A relação $\parallel$ é a menos restritiva quanto à ordenação dos disparos das transições. No outro extremo está a relação de exclusão mútua, a mais restritiva, que permite o disparo de apenas uma das transições. Ao contrário do grafo de planos, o modelo proposto neste capítulo usa as outras três relações no processo de construção, gerando um modelo que representa de forma mais completa o problema de planejamento.

As relações $\lhd$ e $\diamond$ impõem ordenações entre as ações no modelo. A inclusão das sub-redes que representam estas relações no modelo tem efeito semelhante ao processo de expansão do grafo de planos. As cópias das transições que são atrasadas ou adiantadas no fluxo da rede representam as ações que são copiadas entre as camadas do grafo de planos na fase de expansão.

Na próxima seção é apresentado o algoritmo para a construção de um modelo de representação para um problema de planejamento usando as relações apresentadas e a estrutura básica de construção da seção 5.1.

5.3 Construção do modelo

O algoritmo de construção da uma rede de planos R com k camadas, que recebe como parâmetros o conjunto de ações do problema $\mathcal{A}$ e o estado inicial S_0, é dado por:

RedeDePlanos$(k, \mathcal{A}, S_0)$
1 Inclui S_0 na rede de planos R;
2 $i \leftarrow 1$;
3 $S \leftarrow S_0$;
4 enquanto $i \leq k$ faça
5 $A \leftarrow$ AçõesVálidas$(\mathcal{A}, S)$;
6 Inclui A na rede de planos R;
7 $I \leftarrow$ CalculaInconsistências(A);
8 $C \leftarrow$ EstruturaDeControle(I);
9 Inclui C na rede de planos R;
10 $S \leftarrow S \cup$ EfeitosDe(A);
11 $i \leftarrow i + 1$;
12 retorne R.

A linha 1 do algoritmo constrói a estrutura inicial da rede de planos R a partir do estado inicial do problema de planejamento S_0. Esta estrutura inicial é dada pelas equações 5.1 da seção 5.1. Na linha 2 o contador de camadas i é iniciado em 1, pois a estrutura inicial da rede representando o estado inicial é considerada como sendo a primeira camada do modelo. Na linha 3, o conjunto S, que representa o conjunto das proposições presentes na rede de planos, é iniciado com o estado inicial do problema. Após a inclusão da primeira camada, o algoritmo de construção repete o laço que compreende as linhas 5 a 11, inserindo na rede mais $k - 1$ camadas. O conjunto A é o sub-conjunto das ações de $\mathcal{A}$ que têm suas pré-condições presentes na rede R. Cada ação do conjunto A é incluída em R seguindo as equações 5.6 apresentadas na seção 5.1. Na linha 7 são calculadas as inconsistências entre as ações incluídas na camada atual da rede R. O conjunto I contém estas inconsistências representadas pelas relações de ordenação apresentadas na seção 5.2. Na linha 8 a estrutura de controle C que implementa as relações de ordenação do conjunto I é construída seguindo os modelos apresentados na seção 5.2.1. Esta estrutura é incluída na rede de planos na linha 9. O conjunto de proposições da rede S é atualizado a partir dos efeitos das ações incluídas na camada i e a inclusão de uma nova camada é iniciada. Ao final a rede de

planos R contém k camadas com suas respectivas estruturas de controle.

As estruturas de controle são constituídas de cópias das transições que representam ações conflitantes e de lugares que controlam o fluxo de disparo das transições. Cada cópia de transição que é incluída no modelo final se liga às pré-condições e aos efeitos da transição original. Desta forma, as pré-condições e os efeitos de uma ação são compartilhados por todas as suas cópias numa mesma camada. As instâncias originais das transições que representam as ações da camada são aquelas que aparecem ligadas ao lugar raiz I das estruturas de controle.

A seguir é apresentada, como exemplo, a construção da rede de planos para o problema do jantar surpresa usado no capítulo 4. O problema é preparar um jantar surpresa para alguém que está dormindo. Os objetivos são: remover o lixo, preparar o jantar e embrulhar um presente. As quatro ações possíveis são: *cook*, *wrap*, *carry* e *dolly*. A ação *cook* requer mãos limpas (*clean*) e obtém o jantar (*dinner*). A ação *wrap* tem que ser realizada em silêncio (*quiet*) e obtém o presente embrulhado (*present*). *Dolly* elimina o lixo (*garbage*) usando um carrinho de mão e produz barulho (negando *quiet*), *Carry* também elimina o lixo, mas suja as mãos (negando *clean*).

Considere a chamada do algoritmo RedeDePlanos com os seguintes parâmetros: $k = 2$, $\mathcal{A} = \{dolly, carry, cook, wrap\}$ e $S_0 = \{clean, quiet\}$. A inclusão do estado inicial na rede de planos resulta na rede da figura 39.

Na linha 5 do algoritmo o conjunto de ações válidas A contém todas as ações do problema. A figura 40 mostra a rede de planos após a inclusão destas ações. Vale notar que as equações que definem a inclusão das ações na rede (5.6) também incluem no modelo os efeitos das ações: $\{\neg garb, dinner, present\}$.

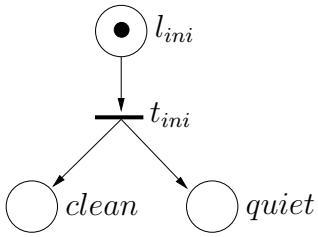


Figura 39: Primeira camada da rede de planos para o problema do jantar.

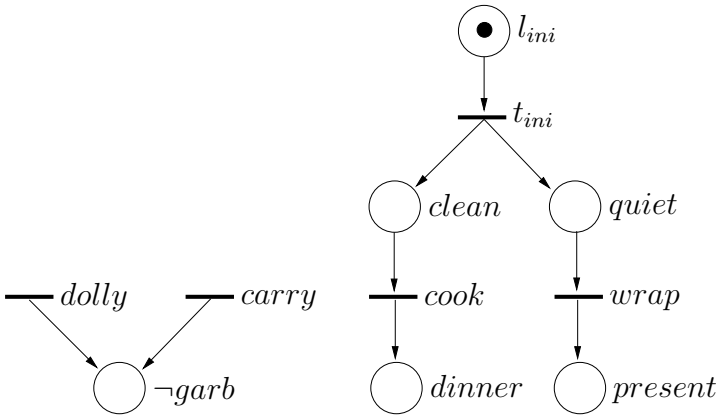


Figura 40: Inclusão da segunda camada na rede de planos para o problema do jantar.

O conjunto das relações de ordenação, calculado na linha 7 para o conjunto de ações $\{dolly, carry, cook, wrap\}$, é dado por $I = \{(dolly \shortparallel carry), (dolly \shortparallel wrap), (carry \shortparallel cook), (cook \triangleleft dolly), (cook \shortparallel wrap), (wrap \triangleleft carry)\}$. A inclusão na rede de planos da estrutura de controle que correnponde à composição destas relações é apresentada na figura 41.

Observe que após a inclusão desta camada, o modelo já contém uma solução para o problema de planejamento proposto. Ou seja, existe uma seqüência de disparos de transições que alcança os lugares $\neg garb$, *dinner* e *present* partindo do disparo da transição t_{ini}. Este mesmo problema requer pelo menos duas expansões no grafo de planos e nos modelos derivados deste, como se pode ver no exemplo apresentado no capítulo 4.

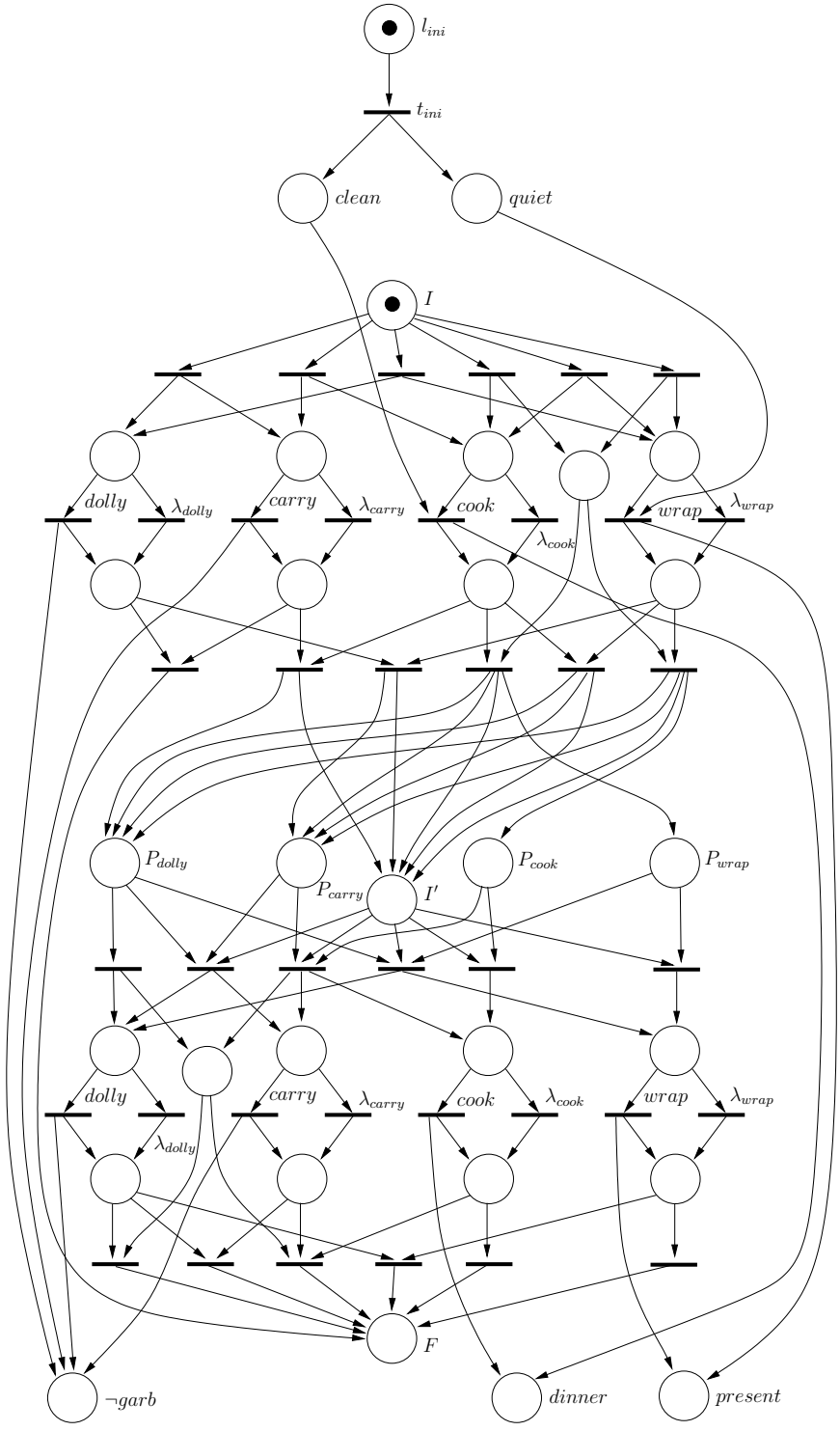


Figura 41: Inclusão da estrutura de controle para a segunda camada da rede de planos para o problema do jantar.

5.4 Considerações

Este capítulo apresentou uma nova estrutura de representação para o espaço de busca de um problema de planejamento usando um modelo em redes de Petri. Esta rede, ao contrário daquelas apresentadas no capítulo 4, não é construída a partir do grafo de planos mas sim diretamente da descrição do problema. Vale notar que este novo processo de construção obtém uma rede acíclica, como nos modelos anteriores, valendo assim as mesmas considerações quanto à solução dos problemas de alcançabilidade diretamente pela equação fundamental das redes de Petri (equação 3.1).

A estrutura da rede e o processo de cópia dos lugares durante a inclusão das transições na rede permite eliminar a noção de ação de manutenção, reduzindo sensivelmente o número de transições, quando comparada com as redes do capítulo 4 e com o grafo de planos.

Foram definidas novas relações de inconsistência entre ações num problema de planejamento, que permitem uma modelagem do espaço de busca mais eficiente do que aquela proposta pelo grafo de planos. Estas novas relações eliminam os caminhos inconsistentes presentes no grafo de planos. O modelo resultante é mais preciso que o grafo de planos, permitindo um processo mais eficiente de busca pela solução.

As estruturas de controle, que resolvem as inconsistências entre as ações, substituem o grande número de mutex dos modelos de representação do capítulo 4. Apesar do tamanho de tais estruturas de controle não ser pequeno, nos casos estudados elas são menores que aquela geradas a partir do grafo de planos.

6 Conclusão

Este documento contém os resultados obtidos pela investigação dos relacionamentos entre problemas de planejamento clássico em inteligência artificial e problemas de alcançabilidade em redes de Petri acíclicas e limitadas.

A relação entre os problemas constitui um cenário de pesquisa rico em uma fronteira nebulosa e promissora entre duas áreas da ciência da computação. A primeira, a de planejamento em inteligência artificial, é uma área extremamente dinâmica voltada para o desenvolvimento de soluções de alto desempenho com um grande apelo para a aplicação prática em problemas do mundo real. A outra, a de redes de Petri, é uma área consolidada que historicamente esteve associada à modelagem de sistemas reais e que apresenta um vasto ferramental teórico.

Inicialmente são apresentadas reduções entre cada um dos problemas e as máquinas de Turing PSPACE. Estes resultados teóricos dão fundamento à tese de que existe uma equivalência direta entre os problemas e que esta pode ser utilizada em termos práticos no desenvolvimento de mapeamentos eficientes entre eles, o que permite o uso de técnicas de planejamento na análise de alcançabilidade em redes de Petri e o uso dos métodos relacionados à alcançabilidade na solução de problemas de planejamento.

A modelagem de problemas de planejamento usando redes de Petri já foi explorada em outros trabalhos, como por exemplo em [MN91] e [MF00]. Entretanto, estes trabalhos utilizam redes de Petri de alto nível, redes predicado-transição e coloridas respectivamente, o que resulta em modelos cuja complexidade de análise é maior do que a das redes convencionais. Apesar disto, os resultados apresentados neste documento são inéditos, pois relacionam numa mesma classe de complexidade os problemas de planejamento às redes de Petri convencionais.

O enfoque principal deste trabalho diz respeito ao tratamento de problemas de planejamento usando redes de Petri. O caminho inverso é brevemente explorado no capítulo

3, onde é proposta uma tradução de problemas de alcançabilidade em redes k-limitadas para problemas de planejamento em *STRIPS*. O problema de planejamento resultante tem tamanho exponencial em relação a estrutura da rede. Entretanto, como os problemas pertencem à mesma classe de complexidade, PSPACE-Completo, acredita-se que codificações mais eficientes sejam possíveis.

Os principais resultados obtidos são aqueles apresentados nos capítulos 4 e 5, nos quais vários modelos de representação para problemas de planejamento são reduzidos a problemas de alcançabilidade em redes acíclicas. A primeira versão foi apresentada como dissertação de mestrado [Sil00] e consistia basicamente de uma simples tradução da estrutura do grafo de planos em uma rede de Petri. O modelo evoluiu para outros onde cada vez mais se explora o poder representacional das redes.

Os algoritmos apresentados no capítulo 4 constróem modelos de representação para o problema de planejamento em redes de Petri acíclicas e tenta resolver os problemas de alcançabilidade associados. Mais especificamente, a tradução objetiva encontrar, na rede de Petri resultante, uma seqüência de disparos que resolve o problema de alcançabilidade, ou seja, o problema de otimização associado ao problema de alcançabilidade. No entanto, vale notar que em termos de implementação real não há diferença entre resolver o problema de alcançabilidade e encontrar a seqüência de disparos associada.

Na hipótese de não haver solução para o problema de alcançabilidade, o modelo de representação é expandido e um novo modelo em redes de Petri é obtido a partir da incorporação de novos lugares e transições. Desta forma o problema de planejamento, que é PSPACE-Completo, é mapeado em uma coleção de instâncias do problema de alcançabilidade em redes acíclicas, que é um problema NP-Completo. A estrutura dos algoritmos é semelhante à dos planejadores baseados no grafo de planos, o que permite utilizar diferentes máquinas de busca para os problemas de alcançabilidade de forma independente do modelo de representação em redes de Petri.

As simplificações apontadas sobre o modelo inicial visam reduzir o tamanho da rede, o que potencialmente reduz o espaço de busca do problema de alcançabilidade. Um dos fatores da explosão combinatorial inerente a este tipo de problema de busca está associado diretamente ao número de lugares da rede. Experimentos realizados em [Mar04] apontam neste sentido. Vale notar ainda que estas simplificações são possíveis devido ao grande volume de redundâncias presentes no grafo de planos que são carregadas para os modelos em redes de Petri pelo processo de tradução. O modelo proposto no capítulo 5, a rede de

planos, visa reduzir as redundâncias estruturais da rede.

A rede de planos [CKS04], é uma rede de Petri acíclica que busca representar o espaço de estados do problema de uma forma mais eficiente que o grafo de planos e os modelos apresentados no capítulo 4. Sua principal diferença está em abandonar o grafo de planos, traduzindo problemas de planejamento diretamente em uma rede de Petri acíclica, tratando os conflitos entre ações de uma forma mais detalhada.

As relações de ordenação apresentadas na seção 5.2 indicam quais são as reordenações que resolvem os conflitos associados a duas ações do modelo. A partir destas relações são construídas estruturas de controle do disparo das transições associadas às ações que substituem os mutex dos modelos anteriores. Estas estruturas limitam as reordenações possíveis dos disparos na rede facilitando o processo de busca pela solução do problema de alcançabilidade correspondente. Vale notar que a dinâmica das redes de Petri tem papel fundamental nas simplificações dos modelos do capítulo 4 e nas estruturas de controle do capítulo 5, fazendo com que não sejam triviais os mapeamentos de tais abordagens para o grafo de planos original.

Um diferencial da rede de planos com relação ao grafo de planos é que, neste e nos modelos em redes de Petri dele derivados diretamente, existem apenas dois tipos de relacionamentos entre duas ações: elas podem ser independentes ou mutuamente exclusivas. A sucessão de camadas destes modelos permite que os conflitos sejam resolvidos por reordenações destas ações durante o processo de busca pela solução. Entretanto, esta estratégia ainda carrega muitas redundâncias e faz com que grande parte das reordenações não levem a uma solução do problema, tornando caro o processo de busca.

Uma outra diferença dos modelos propostos sobre o grafo de planos está na representação das inconsistências entre as ações. No grafo, estas inconsistências são tratadas por relações de exclusão mútua usadas como meta informação no procedimento de busca, ou seja, não são explicitamente um componente estrutural do grafo. Já nos modelos em redes de Petri as inconsistências entre as ações são tratadas por estruturas no próprio modelo, o que faz com que o processo de busca por uma solução não dependa de nenhuma informação adicional além da estrutura da rede.

Foram implementadas cinco versões do planejador *Petriplan* [SCK02], usando os modelos de representação propostos e suas simplificações. Uma destas implementações participou da Competição Internacional de Planejadores de 2004. O planejador inscrito na

competição [CGL+04] foi desenvolvido usando o modelo proposto no capítulo 4 como estrutura de representação e um procedimento de busca clássico para resolver o problema de alcançabilidade. O desempenho do planejador não foi satisfatório e a equipe decidiu retirar o planejador antes do final da competição.

Na verdade, apesar dos modelos propostos serem, em geral, mais expressivos que o grafo de planos, o desempenho dos planejadores resultantes se mostrou muito baixo quando comparado ao dos planejadores recentes. A principal razão está na ineficiência dos atuais algoritmos para alcançabilidade em redes de Petri. Em geral, os métodos disponíveis [Rau90] tratam redes cíclicas genéricas com um número pequeno de lugares e transições, o oposto dos modelos apresentados que podem chegar a dezenas de milhares de lugares e transições.

Neste trabalho optou-se por não explorar as questões de desempenho relacionadas ao problema de alcançabilidade. A pesquisa foi direcionada para os relacionamentos entre os problemas e para os modelos de representação, pois acredita-se que estes temas constituam uma base consistente e relevante para o desenvolvimento de pesquisas futuras.

No que se refere ao problema de métodos para alcançabilidade, dois trabalhos de mestrado estão andamento. Um deles visa integrar o sistema TINA [Ber04] de análise de redes de Petri aos planejadores implementados pela equipe, possibilitando assim o uso de outros métodos para a solução de problemas de alcançabilidade [Ben05]. Outro trabalho visa aplicar técnicas de inteligência artificial para melhorar os desempenho dos métodos para alcançabilidade. O objetivo é construir uma máquina de busca heurística para redes semelhantes aos modelos apresentados e avaliar o desempenho de funções heurísticas usadas em planejamento nos problemas de alcançabilidade [Mon05].

De fato, este trabalho derivou vários outros projetos e por isso teve papel importante na consolidação do grupo de pesquisa em planejamento que envolve o Laboratório de Sistemas Inteligentes de Produção do Cefet-PR e o Laboratório de Inteligência Computacional da UFPR.

Um dos trabalhos teve como objetivo facilitar a implementação e análise dos planejadores desenvolvidos pelo grupo de pesquisa. Para isso foi definida uma plataforma didática para a construção de planejadores, o IPE. Este ambiente de desenvolvimento é constituído por uma coleção de objetos que vão desde modelos de representação para os problemas de planejamento, como redes de Petri e grafo de planos, até máquinas de busca para pro-

blemas relacionados, como SAT, alcançabilidade, buscas heurísticas. A construção de um planejador neste ambiente se resume à encaixar blocos já existentes e construir novos blocos com novas estruturas ou funcionalidades. Os planejadores apresentados neste documento foram implementados nesta plataforma. O trabalho já está concluído e pode ser visto em detalhes na dissertação de mestrado concluída [Mar04].

O grupo também explorou planejadores baseados em algoritmos genéticos, onde foi avaliado o impacto do uso do grafo de planos como uma estrutura de refinamento do espaço de busca do problema. Estes trabalhos mostraram um grande potencial de aplicação dos algoritmos genéticos na busca por caminhos em modelos como o grafo de planos. Os resultados obtidos motivam trabalhos futuros no uso destes algoritmos para a busca em redes ricas em conflitos. Os trabalhos podem ser encontrados no trabalho de graduação [Pal03] e na dissertação de mestrado [Lec04] já concluídos. Os principais resultados destes trabalhos geraram uma publicação na Conferência Ibero-Americana de Inteligência Artificial de 2004 [CLPS04].

Também está em andamento outro trabalho de mestrado que visa adaptar e simplificar os modelos baseados em redes de Petri para problemas de planejamento em domínios de aplicações reais. O objetivo é verificar se o modelo de representação em redes de Petri é mais adequado a uma determinada classe de problemas de planejamento e/ou otimizar o processo de busca da alcançabilidade a domínios específicos [Nov05].

Por outro lado, o uso da representação *STRIPS* como base para a definição dos domínios de planejamento restringe em muito a gama de problemas que podem ser modelados. Os modelos em redes de Petri propostos apresentam potencial para o tratamento de estruturas que podem enriquecer a linguagem de representação e atingir classes de problemas de planejamento ditos não clássicos, como a inclusão de tempo, o uso de recursos e até mesmo o tratamento de incerteza. A substituição, nos modelos propostos, das redes convencionais por extensões de redes de Petri que incorporam estas características viabiliza o tratamento de problemas de planejamento não clássicos, o que foi considerado como fator motivador desta pesquisa.

Neste sentido, está em andamento um trabalho de mestrado que busca, através da utilização de redes de Petri temporizadas [Wan98], tratar problemas de planejamento cujas ações possuem duração, conforme as versões mais recentes da linguagem PDDL [FL02]. Neste trabalho, além da busca da seqüência de disparos de transições que resolve o problema de alcançabilidade, a análise temporal da solução é realizada através de métodos

desenvolvidos para as redes de Petri temporais [dC05].

Finalmente, uma linha de pesquisa promissora para a realização de trabalhos futuros é o uso de planejadores na análise de propriedades de redes de Petri associadas à alcançabilidade. Neste sentido a busca por traduções mais eficientes que aquela apresentada no capítulo 3 aliada ao desempenho dos planejadores atuais representam um grande potencial para o desenvolvimento de ferramentas mais rápidas para a análise de redes de Petri.

Referências

[BC99] P. Beek and X. Chen. CPlan: A constraint programming approach to planning. In *Proc. of the Sixteenth National Conference on AI (AAAI-99)*, pages 585–590, 1999.

[BD98] A. Bockmayr and Y. Dimopoulos. Mixed integer programming models for planning problems. In *Proc. of the Workshop on Constraint Problem Reformulation (CP-98)*, pages 1–6, Pisa, Italy, 1998.

[Ben05] Daniel Richartz Benke. Integração de métodos para alcançabilidade em redes de Petri ao ambiente IPE. Master's thesis, DInf - UFPR, 2005. Em andamento.

[Ber04] B. Berthomieu. The tool TINA: Construction of abstract state spaces for petri nets and time Petri nets. *International Journal of Production Research*, 42(4), 2004.

[BF95] A. Blum and M. Furst. Fast planning through planning graph analysis. In *Proceedings of IJCAI-95*, pages 1636–1642, Montreal, August 1995.

[BG98] B. Bonet and H. Geffner. HSP: Planning as heuristic search. In *Entry at the AIPS-98 Planning Competition*, Pittsburgh, June 1998.

[BG01] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, 2001.

[Byl94] T. Bylander. The computational complexity of propositional strips planning. *Artificial Intelligence*, 69:165–205, 1994.

[CGL+04] Marcos Castilho, André Guedes, Tiago Lima, João Marynowski, Razer Montaño, Luis Künzle, and Fabiano Silva. A petri net based representation for planning problems. In *Booklet of International Planning Competition - IPC'04*, pages 27–29, Whistler, British Columbia, Canada, June 2004. ICAPS'04.

[CKS04] Marcos Castilho, Luis Künzle, and Fabiano Silva. A petri net based representation for planning problems. In *Proceedings of V International Conference on Knowledge Based Computer Systems - KBCS'04*, Hyderabad, India, December 2004. C-DAC Mumbai (formerly NCST). Accepted paper.

[CLPS04] Marcos Castilho, Edson Lecheta, Viviane Palodeto, and Fabiano Silva. An invesigation on genetic algorithms for generic strips planning. In *Procceedings of IX Ibero-American Conference on Artificial Intelligence - IBERAMIA'04*, Puebla, Mexico, November 2004. Instituto Nacional de Astrofísica Óptica y Electrónica. Accepted paper.

[CV97] Janette Cardoso and Robert Valette. *Redes de Petri*. Editora da UFSC, 1997.

[dC05] Malgarete Rodrigues da Costa. Planejamento temporal usando redes de Petri. Master's thesis, DInf - UFPR, 2005. Em andamento.

[Dru85] Mark Drummond. Refining and extending the procedural net. In *IJCAI*, pages 1010–1012, 1985.

[EN94] J. Esparza and M. Nielsen. Decidability issues for Petri nets - a survey. *J. Inform. Process. Cybernet. - EIK*, 30(3):143–160, 1994.

[ENS91] K. Erol, D.S. Nau, and V.S. Subrahmanian. Complexity, decidability and undecidability results for domain-independent planning: A detailed analysis. Technical report, University of Maryland, 1991. CS-TR-2797, UMIACS-TR-91-154, SRC-TR-91-96.

[FL00] Maria Fox and Derek Long, editors. *AIPS-2000 Planning Competition*, 2000.

[FL02] M. Fox and D. Long. Pddl 2.1. In *AIPS2002*, Toulouse, France, 2002.

[FN71] R. Fikes and N. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Journal of Artificial Intelligence*, 2(3-4), 1971.

[Gre69] C. Green. Application of theorem proving to problem solving. In *Proceedings First International Joint Conference on AI*, pages 219–239, 1969.

[GSS03] A. Gerevini, A. Saetti, and I. Serina. Planning through stochastic local search and temporal action graphs. *Journal of Artificial Intelligence Research*, 20:291–341, 2003.

[Hac76] M. H. T. Hack. *Decidability questions for Petri nets*. PhD thesis, MIT, 1976.

[HN00] J. Hoffmann and B. Nebel. The ff planning system: Fast plan generation through heuristic search. In *AIPS Planning Competition*, 2000.

[HN01] J. Hoffmann and B. Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[How91] Rodney R. Howell. The complexity of problems involving structurally bounded and conservative Petri nets. *Information Processing Letters*, (39):309–315, 1991.

[ILO00] ILOG Company. *CPLEX Manual*, oct 2000. version 7.

[KS92] H. Kautz and B. Selman. Planning as satisfiability. In *Proc. 10th Eur. Conf. AI*, pages 359–363, Vienna, Austria, 1992. Wiley.

[KS96] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 1194–1201, Portland, OR, 1996. MIT Press.

[KS99] H. Kautz and B. Selman. Unifying sat-based and graph-based planning. In *Proceedings of IJCAI*, pages 318–325, Stockholm, Sweden, 1999. Morgan Kaufmann.

[Kün02] Peep Küngas. Linear Logic Programming for AI Planning. Master's thesis, Institute of Cybernetics at Tallinn Technical University, Estonia, May 2002.

[KW99] H. Kautz and J.P. Walser. State-space planning by integer optimization. In *Proceedings Fifteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 526–533, Menlo Park, CA, 1999. AAAI Press.

[Lec04] Edson Lecheta. Algoritmos genéticos para planejamento em inteligência artificial. Master's thesis, Universidade Federal do Paraná, Curitiba PR Brasil, fevereiro 2004.

[Lip76] R. Lipton. The reachability problem requires exponential space. Technical Report 62, Yale University, 1976.

[Mar04] João E. Marynoswki. Ambiente de Planejamento Ipê. Master's thesis, Dinf-UFPR, Curitiba, Brasil, Dez 2004.

[McD98a] D. McDermott, editor. *AIPS-98 Planning Competition Results*, 1998.

[McD98b] D. McDermott. *PDDL - The Planning Domain Definition Language*, 1998.

[MF00] Y. Mieller and P. Fabiani. Planning with Petri nets. In *Proc. of RJCIA-2000*, Lyon, France, 2000.

[MH69] J. McCarthy and P. Hayes. Some philosophical problems from the standpoint of artificial intelligence. *Machine Intelligence*, 4:463–502, 1969.

[MN91] T. Murata and P.C. Nelson. A predicate-transition net model for multiple agent planning. *Information Sciences*, 57-58:361–384, 1991.

[Mon05] Razer Anthom Nizer Rojas Montaño. Busca heurística para alcançabilidade em redes de Petri. Master's thesis, DInf - UFPR, 2005. Em andamento.

[Mur89] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.

[Nov05] Paulo José Dantas Novaes. Planejamento dependente de domínio aplicado à processos industriais. Master's thesis, DInf - UFPR, 2005. Em andamento.

[Pal03] Viviane Palodeto. Geração de população inicial para algoritmos genéticos para planificação. Technical report, Dinf-UFPR, 2003.

[Pap81] Christos H. Papadimitriou. On the complexity of integer programming. *J. ACM*, 28(2):765–769, 1981.

[Pap94] Christos H. Papadimitriou. *Computational Complexity*. Addison Wesley, 1994.

[PW92] J. S. Penberthy and D. S. Weld. UCPOP: A sound, complete, partial order planner for ADL. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *KR'92. Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*, pages 103–114, San Mateo, California, 1992. Morgan Kaufmann.

[Rau90] Marko Rauhamaa. A comparative study of methods for efficient reachability analysis. Technical Report 14, Helsinki University of Technology - Digital System Laboratory, Finland, Sep 1990.

[Sav70] W. J. Savitch. Relationship between nondeterministic and deterministic tape complexities. *J. Computer and System Sciences*, (4):177–192, 1970.

[SCK00] F. Silva, M. Castilho, and L. Künzle. Petriplan: a new algorithm for plan generation (preliminary report). In *Proc. of IBERAMIA/SBIA-2000*, pages 86–95. Springer-Verlag, 2000.

[SCK02] Fabiano Silva, Marcos Castilho, and Luis Künzle. Petriplan: Um novo algoritmo para geração de planos. In *Anais do I Workshop de Teses e Dissertações em Inteligência Artificial - WTDIA'02*, Porto de Galinhas-PE, Brasil, Novembro 2002. SBIA'02.

[Sil00] Fabiano Silva. Algoritmos para planificação baseada em *STRIPS*. Master's thesis, Departamento de Informática - UFPR, Curitiba, Brasil, Oct 2000.

[SKC94] B. Selman, H.A. Kautz, and B. Cohen. Noise strategies for improving local search. In *Proceedings of the AAAI-94*, pages 337–343, Seattle, WA, July 1994. AAAI Press.

[SLM92] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 440–446, San Jose, CA, July 1992.

[ST77] G. S. Sacerdote and R. L. Tenney. The decidability of the reachability problem for vector addition systems. In *9th Annual Symposium on Theory of Computing*, pages 61–76, Boulder, 1977.

[Ste92] I. A. Stewart. On the reachability problem for some classes of Petri nets. Technical report, University of Newcastle upon Tyne, 1992.

[VBLN99] T. Vossen, M. Ball, A. Lotem, and D. Nau. On the use of integer programming models in ai planning. In *Proceedings of the IJCAI 99*, pages 304–309, 1999.

[VBLN01] T. Vossen, M. Ball, A. Lotem, and D. Nau. Applying integer programming to AI planning. *Knowledge Engineering Review*, 16:85–100, 2001.

[Wan98] J. Wang. *Timed Petri Nets, Theory and Application*. Kluwer Academic Publishers, 1998.

[Wel99] D. Weld. Recent advances in ai planning. *AI Magazine*, 20(2):93–123, 1999.