MARCOS ANTONIO SCHREINER

# PLANEJAMENTO POR SATISFATIBILIDADE CLAUSAL E NÃO-CLAUSAL BASEADO NA REDE DE PLANOS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Marcos Alexandre Castilho

CURITIBA

2012

MARCOS ANTONIO SCHREINER

# PLANEJAMENTO POR SATISFATIBILIDADE CLAUSAL E NÃO-CLAUSAL BASEADO NA REDE DE PLANOS

Dissertação apresentada como requisito parcial à obtenção do grau de Mestre. Programa de Pós-Graduação em Informática, Setor de Ciências Exatas, Universidade Federal do Paraná.
Orientador: Prof. Dr. Marcos Alexandre Castilho

CURITIBA

2012

Schreiner, Marcos Antonio
Planejamento por satisfatibilidade clausal e não-clausal baseado na rede de planos / Marcos Antonio Schreiner. - Curitiba, 2012.
74 f. : il., tabs.

Dissertação (Mestrado) – Universidade Federal do Paraná, Setor de Ciências Exatas, Programa de Pós-Graduação em Informática.
Orientador: Marcos Alexandre Castilho

1. Redes de Petri. 2. Algoritmos. 3. Teoria dos grafos. I. Castilho, Marcos Alexandre. II. Título.

CDD 511.35

Ministério da Educação
Universidade Federal do Paraná
Programa de Pós-Graduação em Informática

# PARECER

Nós, abaixo assinados, membros da Banca Examinadora da defesa de Dissertação de Mestrado em Informática, do aluno Marcos Antonio Schreiner, avaliamos o trabalho intitulado, *"Planejamento por satisfatibilidade clausal e não-clausal baseado na rede de planos"*, cuja defesa foi realizada no dia 03 de julho de 2012, às 14:00 horas, no Departamento de Informática do Setor de Ciências Exatas da Universidade Federal do Paraná. Após a avaliação, decidimos pela aprovação do candidato.

Curitiba, 03 de julho de 2012.

Prof. Dr. Marcos Alexandre Castilho
**DINF/UFPR** – Orientador

Prof. Dr. Flavio Tonidandel
**FEI** – Membro Externo

Prof. Dr. Fabiano Silva
**DINF/UFPR** – Membro Interno

# AGRADECIMENTOS

A Deus, pois é fonte de sabedoria.

Ao meu orientador, professor Dr. Marcos A. Castilho pelo apoio, condução dos trabalhos, discussões e amizade. Em vários momentos o Castilho se mostrou presente e decisivo no desenvolvimento deste trabalho. Os conhecimentos adquiridos com sua orientação foram fundamentais para obtenção do título de mestre.

Ao professor Dr. Fabiano Silva pelo apoio e amizade. As reuniões e conversas realizadas ao longo do mestrado foram decisivas e fundamentais para conclusão deste mestrado.

Aos meus pais, Mario José Schreiner e Lourdes Schreiner, que além do incentivo e motivação, ensinaram-me a base para encarar desafios em busca da felicidade. Grande parte do que sou hoje devo a eles. Os seus conselhos, a sua compreensão e confiança, garantiram a obtenção desta vitória. É fato que em momentos cuja solução parecia impossível, vislumbrei a figura destes sábios.

As minhas irmãs, Lucie Maria Schreiner e Luciane Terezinha Schreiner e ao meu sobrinho Marcos Vinícios Braun, cujas amizades ultrapassaram distâncias, garantindo a determinação para a conclusão deste trabalho. Em meio aos dias de estudo existia um telefonema, um *email*, uma mensagem, que me fizeram lembrar os motivos do curso deste caminho;

Aos amigos Bruno César Ribas, Razer Anthom Nizer Rojas Montaño, Franck Carlos Velez Benito, Willian Zalewski, João Eugenio Marynowki, Luiz Antonio Rodrigues, Guilherme Galante, Neimar Neitzel, Rafaella Aline Lopes, Diogo César Teixeira Batista, entre outros bons amigos que propiciaram momentos de trabalho, troca de experiências, lazer e alegria.

# SUMÁRIO

# LISTA DE FIGURAS

# LISTA DE TABELAS

# RESUMO

Neste trabalho o problema de planejamento clássico é formalizado em redes de Petri. Com base no fluxo da rede de Petri foi feita uma revisão nas relações de exclusão mútua e nas ações de manutenção definidas no Graphplan. Os *mutex* foram classificados em pares de ações conflitantes definidos em quatro estruturas de controle diferentes, que são usadas na construção da rede do planos. Neste contexto, o problema de planejamento representado na rede de planos é traduzido em uma instância de SAT e resolvido usando um moderno resolvedor SAT. As vantagens do planejador definido neste trabalho são evidenciadas por meio de experimentos comparativos com o PETRIPLAN e o SATPLAN.

**Palavras-chave**: problema de planejamento, rede de Petri, rede de planos, satisfatibilidade

# ABSTRACT

In this work the classical planning problem is formalized as a Petri net. We review the Graphplan notions of mutex relation and maintenance actions based on the Petri net flow. We classify pairs of conflicting actions in terms of four different control structures, which are used to build the Plan Net. The planning problem represented by the net is translated into a SAT instance and then solved using a modern SAT solver. We show the advantages of our method comparing our new planner with PETRIPLAN e o SATPLAN in classical planning domains.

**Palavras-chave**: planning problems, Petri net, planning net, satifiability

# CAPÍTULO 1

# INTRODUÇÃO

O ser humano realiza a ação de planejar quando almeja atingir uma meta. Logo, o resultado do ato de planejar é uma sequência de ações que possibilita obter uma situação objetivo a partir de fatos iniciais. O conjunto de fatos iniciais, assim como a situação objetivo, representam dois estados distintos.

Um estado representa uma situação do mundo, fatos descritos por um conjunto de proposições. A mudança de estado é realizada pela aplicação de uma ação. Uma sequência de ações que possibilita obter um estado objetivo a partir de um estado inicial é um plano.

As ações de um plano representam um subconjunto contido no conjunto de ações disponíveis no ato de planejar. Neste conjunto de ações disponíveis, pode existir mais de um plano, ou até mesmo nenhum plano válido.

No conjunto de ações disponíveis para o planejamento, também pode existir subconjuntos de ações paralelas entre si, ou seja, ações que podem ser realizadas no mesmo instante de tempo. Então um plano pode ser classificado como:

- plano parcialmente ordenado: quando o plano contém ações paralelas entre si;
- plano ordenado: quando o plano não contém ações paralelas entre si.

Em IA, o problema de planejamento tem como entrada um conjunto de ações, um estado inicial e um estado objetivo. O resultado é um plano, ordenado ou parcialmente ordenado, o qual contém as ações necessárias para obter o estado objetivo a partir do estado inicial.

Por exemplo, o problema de logística de um comerciante que possui três lojas $l_1, l_2, l_3$, sendo que existe um caminho direto entre elas. A loja $l_2$ contém o pacote $p_1$, a loja $l_1$ contém o pacote $p_2$ e o comerciante tem a sua disposição um caminhão $c$ que está na loja $l_3$ (estado inicial).

Se o comerciante precisa que $p_1$ esteja em $l_1$ e $p_2$ esteja em $l_2$ (estado objetivo), o plano solução para este problema seria: dirigir $c$ da $l_3$ para $l_1$; carregar $p_2$ em $c$; dirigir da $l_1$ para $l_2$; descarregar $p_2$ em $l_2$ e carregar $p_1$ em $c$; dirigir $c$ da $l_2$ para $l_1$; descarregar $p_1$ em $l_1$.

Os primeiros conceitos do problema de planejamento em IA foram definidos no final dos anos 60, quando a lógica era utilizada como método de representação. Porém, este método apresentou alguns problemas, dentre eles o problema da persistência (*frame problem*) [16].

O problema da persistência consiste da necessidade de representar, na mudança de estado, as proposições que não são alteradas pela ação, pois não é possível realizar a inferência sobre um estado, sem considerar estas proposições.

Na aplicação de uma ação, devem ser incluídas expressões que adicionem ao estado resultante os fatos inalterados por esta ação. Estas expressões são denominadas de axiomas de persistência.

Entretanto, o número de axiomas de persistência pode ser intratável. De acordo com McCarthy, para um conjunto de $n$ ações e $m$ proposições, o número de axiomas de persistência é aproximadamente $n \times m$ [16, 22].

Fikes e Nilsson [8] em 1971, propuseram então o planejador STRIPS (*STanford Research Institute Problem Solver*), composto de uma nova linguagem de representação, e do algoritmo de busca heurística no espaço de estados, definindo assim uma alternativa ao uso da lógica como técnica de solução para o problema de planejamento.

A estrutura de dados do STRIPS é uma árvore, onde cada nó da árvore é um conjunto de proposições que representam um estado. As arestas da árvore, deste modo, são ações que ao serem aplicadas, realizam a mudança de estado.

O algoritmo STRIPS tratava na prática um número pequeno de problemas, pois utilizava uma heurística ineficiente e até mesmo ineficaz para alguns domínios, como o *mundo dos blocos* [8].

Com base nos resultados do STRIPS e sabendo que o problema de planejamento é PSPACE-Completo [6], novas pesquisas em torno do problema de planejamento foram empreendidas, pois ele é um problema de interesse para a sociedade, indústria, entre

outros.

Em 1992, os estudos de Kautz e Selman [13] sobre resolvedores SAT impulsionaram também avanços científicos sobre métodos para solução do problema de planejamento.

A partir de então as pesquisas fundamentadas em STRIPS tiveram seu enfoque direcionado na melhoria da eficiência dos planejadores. Estes estudos resultaram em alguns métodos para resolver o problema de planejamento, dentre os quais destacam-se:

- busca heurística no espaço de estados [5, 9];
- grafo de planos [4];
- planejamento com redes de Petri [25];
- rede de planos [24];
- estruturas convertidas para uma instância SAT [13, 11, 18, 19].

Esses métodos possibilitaram a solução de problemas de planejamento que o planejador STRIPS não conseguia resolver, como por exemplo, a anomalia de Sussman no domínio do mundo dos blocos.

Os avanços da busca heurística no espaço de estados, de Hoffmann e Nebel, se destacaram com a utilização de heurísticas baseadas na definição do problema de planejamento sem considerar os efeitos negativos, o qual é definido como problema de planejamento relaxado [5, 9].

Ao contrário das pesquisas realizadas na área de busca heurística, outros estudos foram voltadas para a estrutura de dados do planejador, dando origem ao grafo de planos e a rede de planos.

O grafo de planos, de Blum e Furst [4], é uma estrutura de dados que otimiza a representação de informações do problema, e possibilita a representação de ações paralelas entre si, e consequentemente a obtenção de planos parcialmente ordenados [4, 29].

No grafo de planos, as ações não paralelas entre si são tratadas como ações mutuamente exclusivas ou *mutex*, ou seja, se uma ação $a$ for aplicada em um instante de tempo, as outras ações não paralelas com $a$, não podem ser aplicadas no mesmo instante de tempo.

As ações mutuamente exclusivas também são representadas na primeira versão do método de planejamento com redes de Petri, pois este método também possibilita que ações paralelas sejam aplicadas.

A rede de Petri é um modelo que permite representar a estrutura e o comportamento de um sistema, pois possibilita representar estados, eventos e fluxo de informações, de um sistema baseado em pré-condições e efeitos.

Uma versão do método de planejamento com redes de Petri permite uma representação similar ao grafo de planos [23]. Contudo, como nela é possível realizar simplificações significativas [24].

A rede de planos é um modelo que utiliza o poder de representação da rede de Petri para simplificar as estruturas de dados para solução do problema de planejamento. Na definição deste modelo, Silva [23] propôs um novo método de construção da rede, e um método para o tratamento de conflitos. Enquanto o grafo de planos representa conflitos entre ações de uma camada, a rede de planos representa conflitos e ordenações parciais entre ações.

Além das estruturas de dados, outro campo promissor para a pesquisa em torno do problema de planejamento foi a conversão destas estruturas para uma fórmula proposicional. Assim, o plano pode ser encontrado por meio da valoração que torna esta fórmula satisfatível.

Este método foi explorado por Kautz e Selman, que propuseram representar o problema de planejamento definido em STRIPS em uma fórmula proposicional em CNF (Forma Normal Clausal)[13]. Eles também propuseram obter uma fórmula proposicional em CNF a partir do grafo de planos, pois a fórmula gerada seria teoricamente menor [11].

Montaño, por sua vez, propôs converter a primeira versão do método de planejamento com redes de Petri, para fórmulas proposicionais não-clausais. A ideia era aproveitar a estrutura original do problema para facilitar a busca por uma valoração satisfatível. A valoração obtida para a fórmula, em conjunto com a rede de Petri, são utilizadas para encontrar um plano [18].

Montaño percebeu que a primeira versão do método de planejamento com redes de

Petri podia ser facilmente representada em NNF (Forma Normal Negativa). Percebeu também, que a valoração para as variáveis da fórmula em NNF, poderiam ser obtidas por meio de uma fórmula em FNNF (*Factored Negation Normal Form*) equivalente, já que é possível encontrar a valoração SAT para uma fórmula em FNNF em tempo polinomial. A complexidade deste método de planejamento está em converter uma fórmula em NNF para FNNF [18].

Outro padrão que também pode ser utilizado nos métodos de solução do problema de planejamento é formato ISCAS, pois existem resolvedores SAT eficientes, que utilizam este padrão como entrada.

O formato ISCAS é um padrão utilizado para codificar circuitos, sendo que nele é possível descrever variáveis e operadores. As variáveis são os dados de entrada do circuito, enquanto os operadores são atributos que recebem o resultado de operações lógicas entre uma ou mais variáveis e/ou um ou mais operadores.

Montaño propôs então converter a primeira versão do método de planejamento com redes de Petri para fórmulas no formato ISCAS. Este método de solução apresentou resultados promissores [19].

Assim, a expectativa volta-se para a rede de planos, pois ela aperfeiçoa a estrutura de solução do problema de planejamento em redes de Petri.

O objetivo deste trabalho é desenvolver um planejador eficiente baseado em redes de Petri. Para isso, serão realizadas correções e simplificações na rede de planos, cuja estrutura será codificada em fórmulas proposicionais ISCAS e CNF. A solução do problema de planejamento será obtida por meio de um resolvedor SAT.

O objetivo deste trabalho foi definido com base nas contribuições dos métodos de solução do problema de planejamento descritos no capítulo 2. As abordagens da rede de planos, do grafo de planos e das estruturas convertidas para uma instância SAT influenciaram diretamente o método de solução do problema de planejamento defendido nesta dissertação.

No capítulo 3 são apresentados métodos para o mapeamento da rede de planos para o fórmulas clausais e não-clausais (ISCAS). No capítulo 4 são descritos os experimentos

realizados e os resultados obtidos.

No capítulo 5 são apresentadas as considerações finais desta dissertação, bem como os trabalhos futuros.

# CAPÍTULO 2

# MÉTODOS PARA RESOLVER O PROBLEMA DE PLANEJAMENTO EM IA

Para desenvolver um método de solução do problema de planejamento consideram-se os seguintes aspectos: a linguagem de representação; a estrutura de dados; e o algoritmo de busca da solução do problema.

No planejador STRIPS a estrutura de dados utilizada foi uma árvore de estados, a qual é base para o algoritmo de busca heurística no espaço de estados [8]. A ineficiência e ineficácia do STRIPS são justificadas pela heurística inapropriada ao problema, visto que existem planejadores de alto desempenho baseados em busca heurística no espaço de estados.

Apesar de o STRIPS apresentar um algoritmo ineficaz, os conceitos da linguagem de representação foram mantidos pela maioria dos planejadores atuais.

Na linguagem STRIPS a representação do domínio – classe geral de um problema composta de predicados e ações – é feita por meio de variáveis que são instanciadas com as constantes do problema, gerando um conjunto de proposições $P$ e um conjunto de ações $O$.

Um subconjunto de $P$ representa um estado do mundo. Um determinado mundo possui diversos estados, entre eles o estado $I$ (estado inicial) e $G$ (estado objetivo). As ações instanciadas $O$ representam o mecanismo de transformação entre estes estados.

O conjunto $O$ pode ser representado pela tripla $<p, o, e>$, onde $o$ é o nome da ação; $p$ são pré-condições da ação e $p \subseteq P$; e $e$ são os efeitos da ação, onde $e = <del, add>$, tal que, $del \subseteq P$ e $add \subseteq P$.

Uma ação só pode ser aplicada se $p$ está contido no conjunto de proposições de um estado. O estado resultante da aplicação da ação conterá as proposições do estado causa,

somado com as proposições de *add*, subtraído das proposições de *del*.

Por exemplo, sejam as proposições $em(c, l_3)$, $em(p_1, l_2)$, $em(p_2, l_1)$, $em(c, l_1)$, $em(c, l_2)$ e as ações $dirigir(c, l_3, l_1)$, $dirigir(c, l_3, l_2)$ da figura 2.1. As ações $dirigir(c, l_3, l_1)$, $dirigir(c, l_3, l_2)$ possuem como pré-condição a proposição $em(c, l_3)$. A ação $dirigir(c, l_3, l_1)$ gera o estado $em(c, l_1)$, $em(p_1, l_2)$, $em(p_2, l_1)$, pois remove $em(c, l_3)$ e adiciona $em(c, l_1)$; e a ação $dirigir(c, l_3, l_2)$ gera o estado $em(c, l_3)$, $em(p_1, l_2)$, $em(p_2, l_1)$, porque remove $em(c, l_3)$ e adiciona $em(c, l_2)$.

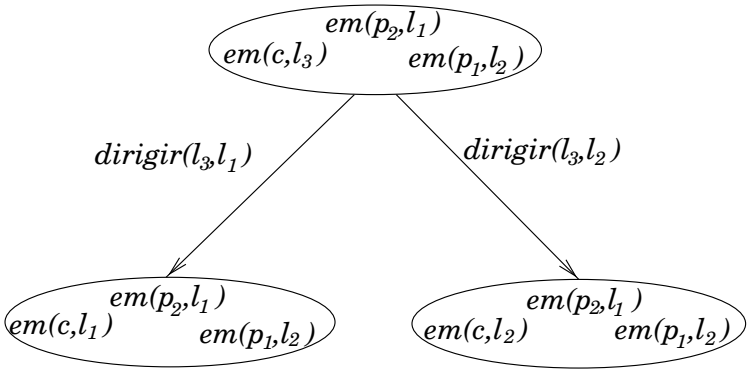



Figura 2.1: Representação de estados e ações em STRIPS

A linguagem STRIPS, juntamente com a linguagem ADL (*Action Descripition Language)*, foram utilizadas como base para a definição da linguagem PDDL (*Planning Domain Definition Language*), desenvolvida por McDermott e colegas em 1998, com o objetivo de unificar a representação de problemas de planejamento, para facilitar a comparação entre planejadores diferentes [17].

Um problema de planejamento descrito em PDDL deve conter a descrição do domínio, o qual, assim como STRIPS, é a definição da classe geral de um problema. Também deve conter a definição do problema, ou seja, a representação dos objetos, do estado inicial e do estado final.

Em PDDL as palavras precedidas por - (hífen) são tipos; palavras precedidas por : (dois pontos) são palavras reservadas; e as precedidas por ? (interrogação) são variáveis.

Por exemplo, para problema logística descrito no capítulo 1, o domínio em PDDL pode ser definido conforme a figuras 2.2 e 2.3.

Conforme representado na figura 2.2, o arquivo do domínio deve conter a definição do

```
(define (domain logística)
(:types caminhao - veiculo
        pacote veiculo - objeto
        localidade - local
        objeto local - geral)

(:predicates
(em ?obj - objeto ?loc - localidade)
(dentro ?pkg - pacote ?veh - veiculo))
```

Figura 2.2: Domínio do problema de logística do capítulo 1 em PDDL (parte 1)

nome do domínio (logística), e o conjunto do tipos existentes, caso a representação for tipada.

No domínio também deve ser representado o conjunto de predicados presentes no domínio, os quais ao serem instanciados, com os objetos do problema, representarão as proposições do problema de planejamento. Assim os predicados são utilizados na descrição das ações e na formação das proposições no processo de instanciação.

```
(:action carregar
:parameters (?pkg - pacote ?truck - caminhao ?loc - localidade)
:precondition (and (em ?truck ?loc) (em ?pkg ?loc))
:effect (and (not (em ?pkg ?loc)) (dentro ?pkg ?truck)))

(:action descarregar
:parameters (?pkg - pacote ?truck - caminhao ?loc - localidade)
:precondition (and (em ?truck ?loc) (dentro ?pkg ?truck))
:effect (and (not (dentro ?pkg ?truck)) (em ?pkg ?loc)))

(:action dirigir
:parameters (?truck - caminhao ?loc-from - localidade ?loc-to - localidade)
:precondition (and (em ?truck ?loc-from) )
:effect (and (not (em ?truck ?loc-from)) (em ?truck ?loc-to)))
)
```

Figura 2.3: Domínio do problema de logística do capítulo 1 em PDDL (parte 2)

Na figura 2.3 são descritas as ações do problema de logística do capítulo 1. Nota-se que em PDDL deve-se descrever o nome da ação (carregar, descarregar, dirigir); os parâmetros passados para a ação, os quais são utilizados nos predicados das pré-condições ou efeitos; o conjunto das pré-condições da ação; e conjunto dos efeitos da ação.

Por exemplo, a pré-condição da ação *dirigir* é que um determinado caminhão esteja em uma localidade. O efeito desta ação é que o caminhão esteja em outra localidade e por isso sua definição remove o predicado pré-condição desta ação.

Em PDDL os arquivos de descrição do problema são instâncias de um domínio. Por exemplo, conforme problema logística descrito no capítulo 1, o arquivo problema deve conter os itens apresentados na figura 2.4.

```
(define (problem comerciante)
(:domain logística)
(:objects p2 p1 - pacote
        c - caminhao
        l1 l2 l3 - localidade)
(:init (em c l3)
(em p2 l1)
(em p1 l2))
(:goal (and (em p2 l2)
(em p1 l1)))
)
```

Figura 2.4: Definição do problema de logística do capítulo 1 em PDDL

O arquivo de definição do problema de planejamento deve conter o nome do problema (comerciante); a qual domínio o problema faz referência (logística); o conjunto de objetos existentes no problema e o tipo dos quais estes objetos são instancias; a definição do estado inicial; e do estado final.

O processo de instanciação tem como entrada o domínio e o problema definido em PDDL. O resultado deste processo é um conjunto de proposições e ações que são utilizadas para resolver o problema de planejamento. Por exemplo, com as definições da figura 2.2, 2.3 e 2.4, o processo de instanciação gera proposições e ações, como as apresentadas na figura 2.5.

As proposições e ações apresentadas na figura 2.5 são um subconjunto de todas as proposições e ações resultantes do processo de instanciação. Não foi utilizado um padrão específico para descrevê-las, mas buscou-se apresentá-las de forma clara quais são as proposições e ações, bem como as pré-condições e efeitos das ações.

Por exemplo, a ação $dirigir(c, l_3, l_1)$ tem como pré-condição que o caminhão esteja na

Proposições:
$em(p_1, l_2)$
$em(p_2, l_1)$
$em(c, l_3)$
$dentro(p_1, c)$
...
Ações:
$dirigir(c, l_3, l_1)$
pré-condição : $em(c, l_3)$
$efeitos : not(em(c, l_3)), em(c, l_1)$
$dirigir(c, l_3, l_2)$
pré-condição : $em(c, l_3)$
$efeitos : not(em(c, l_3)), em(c, l_2)$
$carregar(p_1, c, l_2)$
pré-condição : $em(c, l_2), em(p_1, l_2)$
$efeitos : not(em(p_1, l_2)), dentro(p_1, c)$
$descarregar(p_1, c, l_2)$
pré-condição : $em(c, l_2), dentro(p_1, c)$
$efeitos : not(dentro(p_1, c)), em(c, l_2)$

Figura 2.5: Instanciação do problema denfinido nas figuras 2.2, 2.3 e 2.4

loja 3. O efeito da ação $dirigir(c, l_3, l_1)$ é que o caminhão não está mais na loja 3, mas sim na loja 1.

Com a definição do PDDL, o enfoque do desenvolvimento de métodos para resolver o problema de planejamento é direcionado para a estrutura de dados e o algoritmo de busca da solução problema. Nas próximas seções serão apresentadas algumas destas estruturas. O grafo de planos é de particular interesse neste trabalho.

## 2.1 Grafo de planos

Ao contrário da árvore de estados, o grafo de planos é uma estrutura de dados que aperfeiçoa a representação de informações e possibilita a obtenção de planos parcialmente ordenados [4].

No grafo de planos não existe a informação de estado, existe o conceito de camada. Uma camada de proposições, que está contida nas camadas pares do grafo, é a união de

todos os estado possíveis para um determinado instante de tempo na árvore de estados. Uma camada de ações, que está contida nas camadas ímpares do grafo, é a união das ações cujas pré-condições estão contidas no conjunto de proposições da camada par anterior.

Seguindo o conceito de camada, por exemplo, a árvore de estados da figura 2.1 poderia ser representada pelo grafo da figura 2.6.

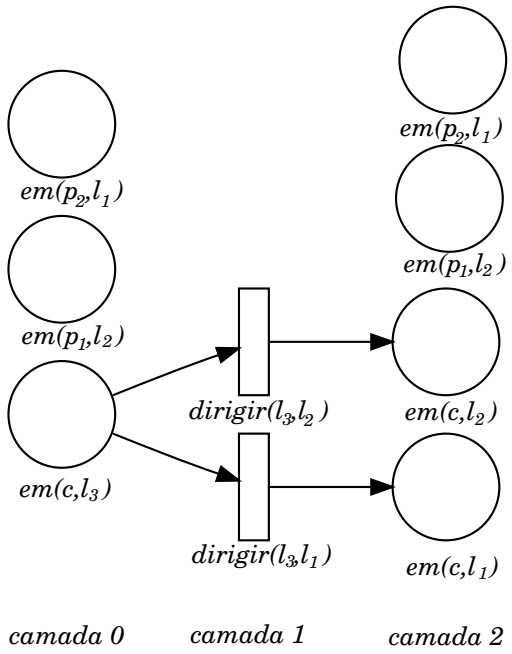



Figura 2.6: Grafo de planos – conceito de camada

Na figura 2.6, as proposições $em(p_1, l_2)$ e $em(p_2, l_1)$ pertencem a camada 2, mas não são efeitos de nenhum ação da camada 1. As proposições $em(p_1, l_2)$ e $em(p_2, l_1)$ pertencem a camada 2, porque pertenciam aos estados possíveis no espaço de estados representado na figura 2.1.

Além disso, uma ação que tem como pré-condição $em(p_1, l_2)$ só pode ser inserida camada 3, se $em(p_1, l_2)$ pertencer a camada 2.

No grafo de planos foram então definidas as ações de manutenção, ações cuja pré-condição e efeito são a mesma proposição. Para cada proposição existente numa camada $i$, existe uma ação de manutenção na camada $i + 1$, o que resulta na ocorrência de todas as proposições da camada $i$ na camada $i + 2$. Esse processo também garante que uma ação presente em $i + 1$ também estará presente nas próximas camadas ímpares.

Com a inclusão das ações de manutenção o grafo da figura 2.6 é representado pelo grafo de planos da figura 2.7.

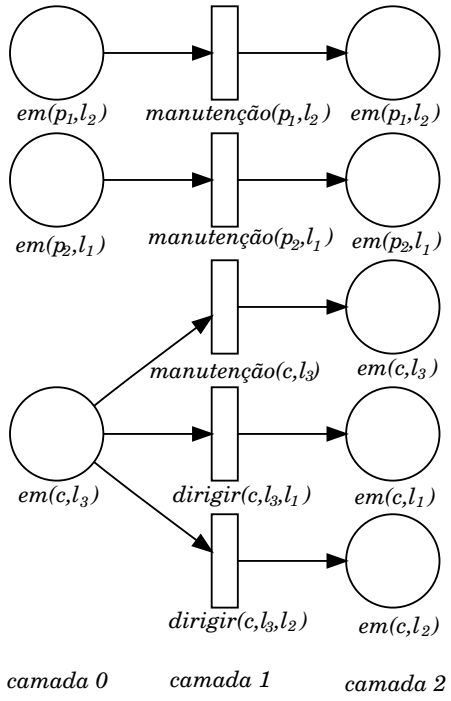



Figura 2.7: Grafo de planos – ações de manutenção

O grafo de planos da figura 2.7 é um exemplo de um grafo de planos relaxado. Nele as ações $dirigir(c, l_3, l_1)$ e $dirigir(c, l_3, l_2)$ poderiam ser aplicadas em paralelo em um plano, o que geraria um plano inconsistente devido a contradição entre as pré-condições e efeitos destas ações.

Ações que não podem ser executadas em paralelo são ações mutuamente exclusivas ou *mutex*. Duas ações são mutuamente exclusivas se o efeito de uma ação é a negação da pré-condição ou do efeito de outra ação. No grafo de planos este *mutex* é denominado interferência (*interference*) e é representado por uma aresta que conecta duas ações em uma mesma camada.

Por exemplo, o grafo de planos relaxado representado na figura 2.7 é apresentado na figura 2.8 com a inclusão dos mutex de interferência. As arestas ($dirigir(c, l_3, l_1)$, $dirigir(c, l_3, l_2)$), ($manutenção(c, l_3)$, $dirigir(c, l_3, l_2)$) e ($dirigir(c, l_3, l_1)$, $manutenção(c, l_3)$), são exemplos do *mutex* de interferência. O cálculo deste *mutex* tem por base os conflitos

entre pré-condições e efeitos das ações intanciadas da figura 2.5

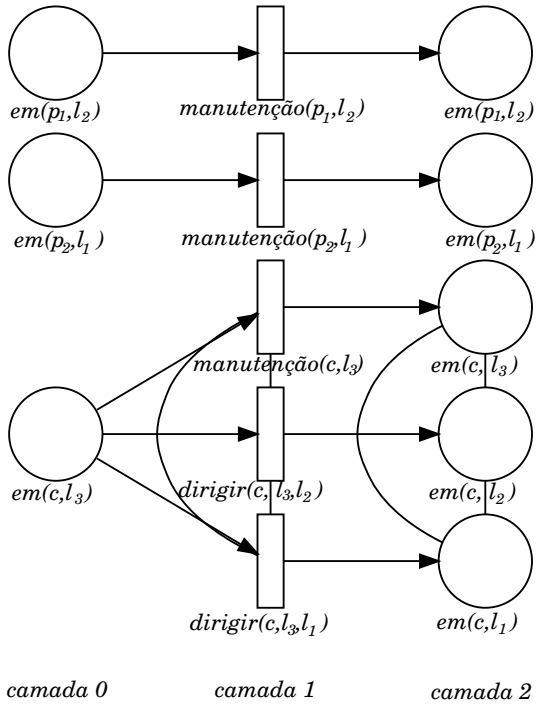



Figura 2.8: Grafo de planos – mutex (interferência)

Também foram incluídos na figura 2.8 os *mutex* de proposições. No grafo de planos, duas proposições são mutuamente exclusivas se todas as ações que tem como efeito estas proposições também forem mutuamente exclusivas. Essa relação também é representada com uma aresta que conecta duas proposições em uma mesma camada, como por exemplo a aresta $(em(c, l_1), em(c, l_2))$.

As proposições mutuamente exclusivas também são um critério para identificação de ações mutuamente exclusivas no grafo de planos. Duas ações são mutuamente exclusivas se a pré-condição de uma ação é mutuamente exclusiva com a pré-condição da outra ação. Este *mutex* é denominado competição de necessidades (*competing needs*) e também é representado por uma aresta que liga duas ações.

A competição de necessidade possibilita a propagação de *mutex* ao longo do grafo de planos. Com a união da definição de exclusão mútua das proposições com a definição da relação de competição de necessidade, é possível concluir que um *mutex*, identificado pela regra de competição de necessidade em uma determinada camada, é gerado por conflitos

de camadas anteriores.

O grafo de planos da figura 2.9 exemplifica o efeito da inclusão do *mutex* de competição de necessidade. Esta figura é o grafo de planos da figura 2.8 após uma expansão, ou seja, após a inclusão de mais uma camada de ações e proposições. A camada 3 não contém todas as ações que a camada 2 possibilita inserir devido a complexidade que seria inserida à figura.

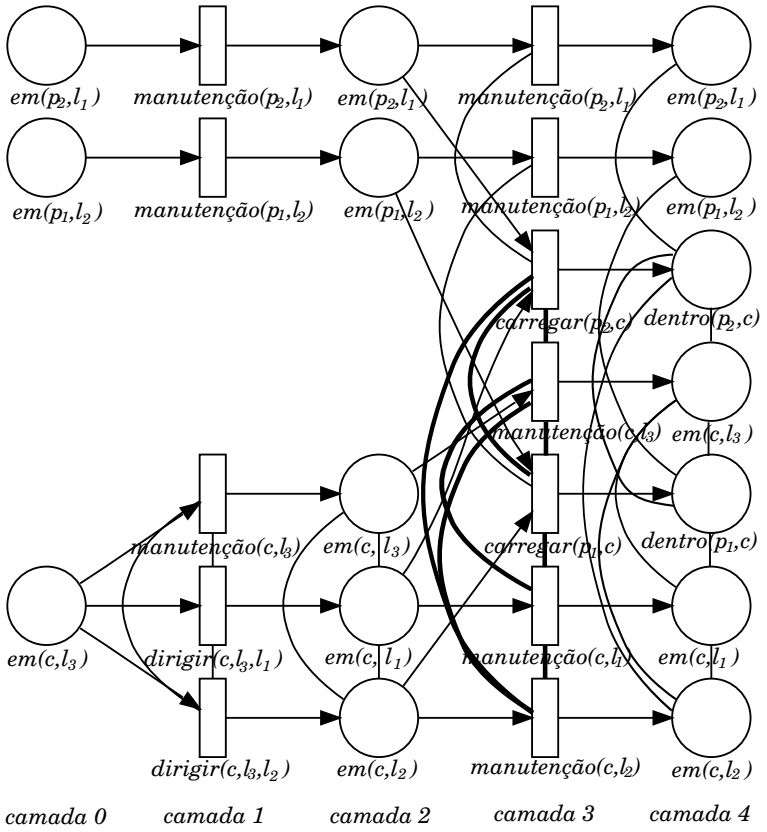



Figura 2.9: Grafo de planos – mutex

Na figura 2.9 as arestas em negrito representam os *mutex* de competição de necessidade. Estes *mutex* foram identificados porque as proposições $em(c, l_1)$, $em(c, l_2)$ e $em(c, l_3)$ são duas a duas mutuamente exclusivas. Por sua vez, estas proposições são mutuamente exclusivas devido a contradição lógica entre as pré-condições e efeitos das ações $dirigir(c, l_3, l_1)$, $dirigir(c, l_3, l_2)$ e $manutenção(c, l_3)$.

Com a definição da relação de competição de necessidade, duas proposições podem ser mutuamente exclusivas em apenas algumas camadas do grafo de planos, porque ações no-

vas podem ser inseridas no grafo. Para tanto estas ações devem ter relações de paralelismo que eliminem o *mutex* entre estas proposições.

Por exemplo, na figura 2.10, as proposições $dentro(p_1, c)$ e $dentro(p_2, c)$ são *mutex* na camada 4, mas na camada 8 estas proposições não são mais mutuamente exclusivas. Vale destacar que o grafo da figura 2.10 é apenas um subgrafo do grafo de planos original.

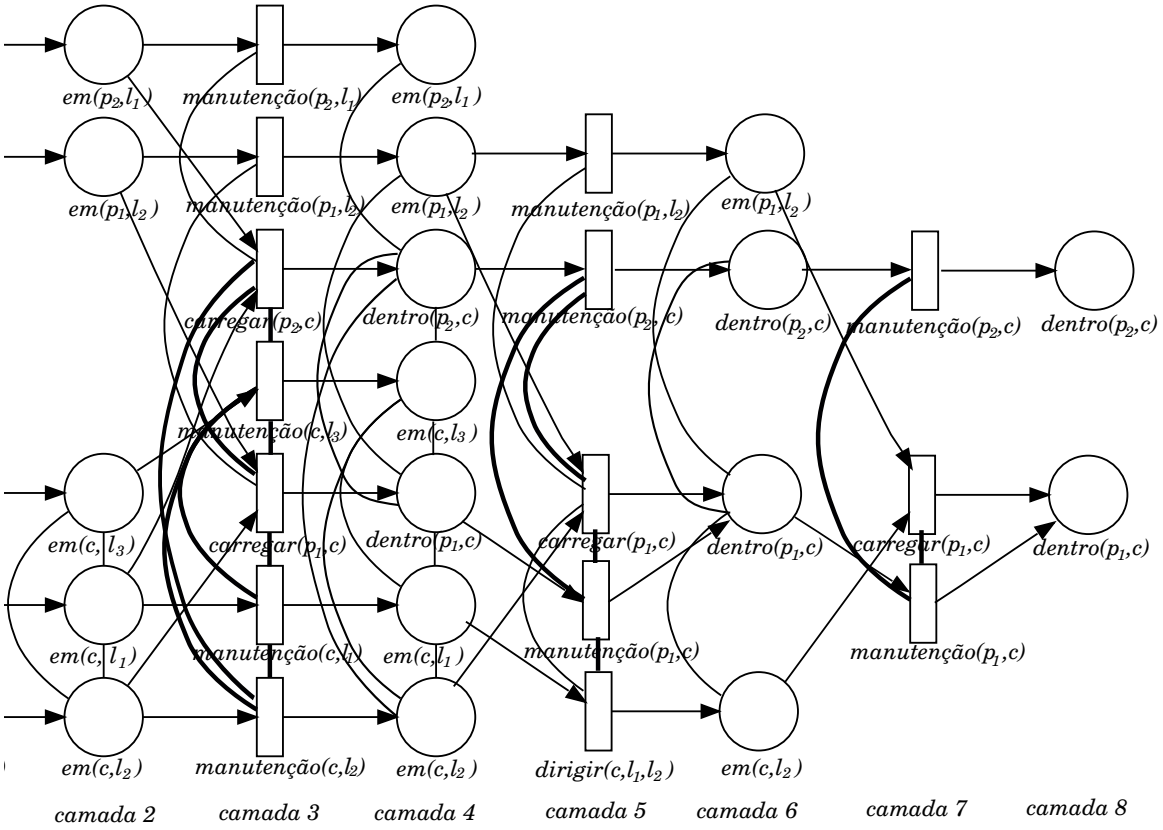



Figura 2.10: Grafo de planos – mutex de proposições

A eliminação do *mutex* entre as proposições $dentro(p_1, c)$ e $dentro(p_2, c)$ ocorre porque a ação $dirigir(c, l_1, l_2)$ não é *mutex* com a ação de $manutenção(p_2, c)$ na camada 5, eliminando a relação mutuamente exclusiva entre as proposições $dentro(p_2, c)$ e $em(c, l_2)$ na camada 6. Se $dentro(p_2, c)$ e $em(c, l_2)$ não são mais mutuamente exclusivas, então as ações $carregar(p_1, c)$ e $manutenção(p_2, c)$ também não são *mutex* na camada 7, fazendo com que $dentro(p_1, c)$ e $dentro(p_2, c)$ também não sejem *mutex* na camada 8.

As regras de *mutex* do grafo de planos preservam a ideia de que um objeto não pode estar em dois lugares ao mesmo tempo, se o estado inicial e a ação de movimento forem

definidos deste modo.

Por exemplo, se no estado inicial é definido que um caminhão está em apenas uma loja e a ação dirigir tem como pré-condição que o caminhão esteja em um loja $x$ e tem como efeito a remoção de $x$ e a adição da loja $y$, então as regras de *mutex* do grafo de planos garantirão que o caminhão nunca esteja duas lojas ao mesmo tempo.

Enfim, o uso do grafo de planos como estrutura de solução de um problema de planejamento ocorre em duas etapas. Uma é o processo de construção do grafo, a outra é a busca pelo plano.

O grafo de planos é construído a partir do estado inicial do problema, seguido de um processo de expansão, no qual é incluída uma nova camada $i$ de ações cujas suas pré-condições estejam presentes na camada $i-1$ do grafo, e não sejam mutuamente exclusivas. Para cada proposição de $i-1$ é criada uma ação de manutenção.

Em seguida criam-se as relações de exclusão mútua entre as ações de acordo com as contradições entre suas pré-condições e efeitos.

Por fim é adicionada nova camada de proposições, que é composta pela união dos efeitos das ações inseridas na camada $i$. As proposições mutuamente exclusivas são então conectadas entre si.

O processo expansão do grafo ocorre até que a camada de proposições contenha os objetivos do problema, pois isso representa a possibilidade de existência de plano no espaço de busca do grafo de planos.

Se o as proposições objetivo do problema não forem encontradas a expansão do grafo ocorre até que uma camada seja idêntica a camada par anterior, inclusive com relação aos *mutex*. Neste caso não existe solução para o problema.

O algoritmo de busca do plano para o problema utiliza a técnica de busca regressiva. A partir do conjunto de proposições objetivo $G$ contido na camada $i$, obtêm-se um conjunto de ações não *mutex* em $i-1$ que tem $G$ como efeito. As pré-condições desse conjunto de ações em $i-2$ formam um conjunto de proposições $m$. Se não for encontrado um conjunto de ações não *mutex* em $i-3$ que tem $m$ como efeito, o algoritmo realiza retrocesso (*backtracking*) e encontra um conjunto diferente de ações em $i-1$.

Esse processo se repete para as camadas anteriores até a camada inicial do grafo de planos, sendo que neste caso um plano for encontrado. Caso contrário, uma nova expansão do grafo é necessária.

Uma das principais contribuições do grafo de planos é a simplificação da estrutura, a qual elimina a redundância da representação das proposições em cada instante de tempo, devido à ausência da informação de estado. Esta estrutura também possibilita a obtenção de planos parcialmente ordenados.

Entre as desvantagens do uso do grafo de planos pode-se apontar a perda da informação dos estados e consequentemente o uso adicional de memória para a representação dos *mutex*.

Outros estudos foram realizados com redes de Petri, pois elas permitem uma representação semelhante ao grafo de planos, e possibilitam a simplificação do modelo, pois permitem a representação do fluxo de informação.

## 2.2 Rede de planos

As redes de Petri também são um modelo utilizado para resolver problema de planejamento. Este modelo permite representar a estrutura e o comportamento de um sistema [20].

A primeira estrutura para resolver o problema de planejamento baseada em redes de Petri era semelhante ao grafo de planos. Nesta estrutura não eram realizadas as simplificações possibilitadas pela rede de Petri [23].

Silva [24] propôs então um novo método de construção da rede, o qual permite a simplificação do modelo.

Nas próximas subseções será descrita a rede de Petri; o modelo de solução do problema de planejamento baseado em redes de Petri e definido por Silva (rede de planos); e soluções de inconsistências na definição da rede de planos.

### 2.2.1 Rede de Petri

Uma rede de Petri é um modelo baseado em estados e eventos que geram a mudança de estado no sistema. Para haver a mudança de estado um conjunto de pré-condições deve ser satisfeito. O novo estado pode ocasionar novos eventos.

As redes de Petri podem ser representadas por meio de um grafo dirigido com dois tipos de nós e um comportamento dinâmico. Este grafo é formado pelos componentes apresentados nas figuras 2.2.1, os quais são descritos a seguir [7, 20]:

- lugares (representados por um círculo): nós que podem ser interpretados como uma condição, um efeito, um elemento de um estado. São os componentes passivos do modelo que em geral podem ter um predicado associado, como por exemplo $em(c, l_1)$. Neles são acrescidas ou removidas marcas de acordo com a dinâmica da rede;

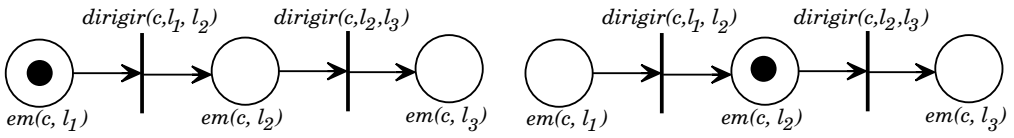
- marcas (representado por um ponto num lugar): é um indicador de que a condição associada a um lugar é verificada, como por exemplo, se houver uma marca no lugar $em(c, l_1)$, significa que o caminhão $c$ está na loja 1. São os componentes dinâmicos do modelo, pois seu fluxo determina o comportamento do sistema ao longo do tempo, sendo que o número de marcas e sua localização determinam um estado do sistema;

- transições (representada por barra ou retângulo): são associadas a um evento do sistema, portanto são os componentes ativos do modelo. As transições removem um conjunto de marcas dos lugares pré-condição e criam outro número de marcas nos lugares efeito;

- arcos: conectam as pré-condições com as transições e as transições com os efeitos. Eles podem conter pesos que indicam quantas marcas devem ser consumidas pela transição ou quantas marcas são criadas nos lugares efeito.

Uma rede de Petri pode ser definida por uma quádrupla $R =< P, T, Pre, Post >$, onde:

- $P$ é um conjunto de lugares de tamanho $n$;




(a) Rede de Petri antes do disparo de $dirigir(c, l_1, l_2)$

(b) Rede de de Petri após o disparo de $dirigir(c, l_1, l_2)$

Figura 2.11: Disparo de uma transição em rede de Petri

- $T$ é um conjunto de transições de tamanho $m$;
- $Pre$: é a aplicação dos lugares precedentes ou a incidência anterior. É uma matriz de números naturais $P \times T$;
- $Post$: é a aplicação dos lugares seguintes ou a incidência posterior. Também é uma matriz de números naturais $P \times T$;

Uma rede de Petri marcada é definida formalmente por uma tupla $N =< R, M >$, onde $R$ é uma rede de Petri e $M$ é um vetor de tamanho $n$ que contém a marcação inicial, sendo que $M(p)$ contém o número de marcas do lugar $p$.

Duas transições $t_1$ e $t_2$ estão em conflito estrutural se $t_1$ e $t_2$ são duas possibilidades excludentes para o fluxo de marcas na rede, ou seja, $t_1$ e $t_2$ possuem o mesmo lugar $p$ como pré-condição. O conflito estrutural é definido pela equação 2.1 e representado pela figura 2.12 [7].

$$\exists p \in P, Pre(p, t_1) Pre(p, t_2) \neq 0 \tag{2.1}$$

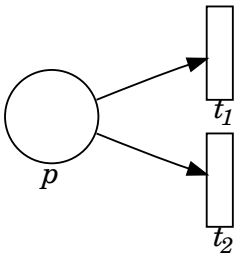



Figura 2.12: Rede de Petri - conflito estrutural

Duas transições $t_1$ e $t_2$ são paralelas estruturalmente se $t_1$ e $t_2$ não possuem nenhum

lugar $p$ como pré-condição. O paralelismo estrutural é definido pela equação 2.2 e representado pela figura 2.13 [7].

$$\exists p \in P, Pre(p, t_1)Pre(p, t_2) = 0 \tag{2.2}$$

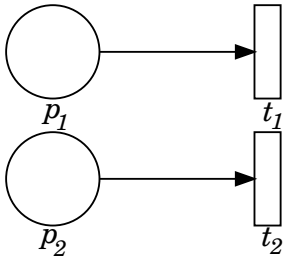



Figura 2.13: Rede de Petri - paralelismo estrutural

O número de marcas em um lugar $p$ pode sensibilizar (habilitar) uma transição $t$, quando o número de marcas em $p$ for maior ou igual ao peso do arco entre $p$ e $t$. Formalmente este conceito pode ser definido de dois modos, como apresentado nas equações 2.3 [7].

$$\forall p \in P, \quad M(p) \geq Pre(p, t) \tag{2.3}$$

Se $t$ é sensibilizada por $M$, uma nova marcação $M'$ pode ser obtida com o disparo de $t$. A marcação $M'$ é dada pela equação 2.4 [7].

$$\forall p \in P, M'(p) = M(p) - Pre(p, t) + Post(p, t) \tag{2.4}$$

Dada uma marcação inicial $M$ e uma marcação objetivo $M_g$, verificar se existe uma sequência de disparos, que obtém $M_g$ a partir de $M$, é a definição do problema de alcançabilidade em redes de Petri.

Dada a equação 2.5, onde $A(R, M)$ é o conjunto marcações alcançáveis por uma sequência de disparos $s$, a partir da marcação $M$, verficar se a marcação $M_g$ pertence a $A(R, M)$ é a definição do problema de alcançabilidade em rede de Petri [7].

$$A(R, M) = \{M_i, \exists s M \xrightarrow{s} M_i\} \tag{2.5}$$

O problema de alcançabilidade de sub-marcação consiste em verificar se uma sub-marcação $M_s$ é alcançavel a partir de $M$. Este problema consiste então em verificar se $M_s \subseteq M_g$.

O problema de alcançabilidade em redes de Petri é PSPACE-completo [21, 15]. Contudo para as rede de Petri acíclicas o problema de alcançabilidade é NP-Completo [26].

O problema de alcançar uma marcação $M_g$, a partir de uma marcação $M_i$ em uma rede de Petri acíclica $R =< P, T, Pre, Post >$, pode ser visto como o problema de programação inteira de encontrar um vetor inteiro não negativo $\bar{s}$, que respeite as restrições da equação 2.6.

$$M_g = M_i + (Post - Pre) \cdot \bar{s} \tag{2.6}$$

O problema de planejamento pode ser traduzido para um problema de alcançabilidade em rede de Petri. A solução encontrada para o problema de alcançabilidade representa a solução para o problema de planejamento. Uma das representações para o problema de planejamento em redes de petri é a rede de planos.

### 2.2.2 Definições da rede de planos

A marcação inicial da rede de Petri pode representar o estado inicial do problema de planejamento. A marcação final pode representar o estado objetivo. A sequência de disparos das transições, deste modo, é o conjunto de ações que possibilitam encontrar o estado objetivo, a partir do estado inicial [25].

As redes de Petri utilizadas para resolver o problema de planejamento podem ser obtidas por meio de um processo de conversão do grafo de planos [23]. Entretanto este modelo apresenta estruturas redundantes e com potencial de simplificação, pois herdam as ações de manutenção, que aumentam consideravelmente o número de transições [24].

Em 2005, Silva [24] definiu a rede de planos, um modelo de construção de redes de Petri acíclica que não possui ações de manutenção e aproveita melhor a dinâmica da rede.

Para definição da rede de planos foi criada uma estrutura básica e foram definidas novas regras de inconsistências entre ações, para identificar relações de ordenação como o *mutex*. Estas relações de ordenação possibilitaram a definição de novas estruturas de controle para redes de Petri.

A estrutura básica é composta de transições (ações) que podem estar conectadas com lugares pré-condição e com lugares efeito. Os efeitos negativos não são representados, mas são úteis para o cálculo das relações de ordenação.

A rede de planos contém um lugar inicial $L_{ini}$, ligado a uma transição inicial $T_{ini}$, a qual está ligada a um lugar para cada proposição do estado inicial $l_i^k$, onde $i$ é o indice da proposição e $k$ o número do lugar $l$ que representa a proposição $i$, conforme apresentado na figura 2.14.

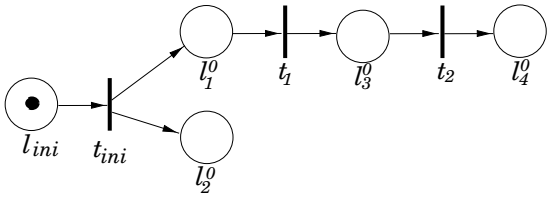



Figura 2.14: Rede de planos básica

Com o objetivo de evitar conflitos estruturais na rede e garantir que a rede de planos seja uma rede de Petri acíclica, é utilizado mais de um lugar para representar uma proposição. Este conjunto de lugares é criado durante a inclusão de uma transição na rede.

Uma transição é incluída na rede de planos se todas as suas pré-condições estão contidas na rede. A inclusão de uma transição deve estar de acordo com as seguintes restrições [24]:

1. se a pré-condição da transição não for usada como pré-condição por nenhuma outra transição, conecta-se a pré-condição a nova transição. Por exemplo, na figura 2.15 a pré-condição $\mathrm{l}_2^0$;

2. se a pré-condição da nova transição for um lugar utilizado como pré-condição por alguma outra transição, deve ser feita uma cópia deste lugar, mantendo-se as ligações

dos efeitos das transições anteriores. Por exemplo, na figura 2.15 a pré-condição $l_3^1$;

3. os efeitos da nova transição, não presentes na rede, devem ser incluídos. Por exemplo, na figura 2.15 o efeito $l_5^0$;

4. os efeitos da nova transição presentes na rede, não utilizados como pré-condição, são ligados a esta nova transição. Por exemplo, na figura 2.15 o efeito $l_4^0$;

5. os efeitos da nova transição usados como pré-condição são copiados mantendo-se as ligações dos efeitos das transições anteriores. Por exemplo, na figura 2.15 o efeito $l_1^1$.

Se não fosse realizada a criação do lugar $l_1^1$ na figura 2.15, o efeito da transição $t_3$ geraria um ciclo na rede, o que é inconsistente com a definição inicial de que a rede de planos é acíclica.

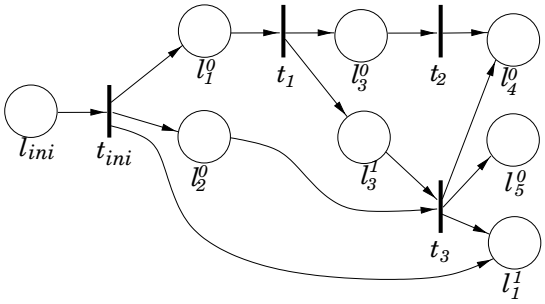



Figura 2.15: Inclusão da transição $t_3$ e dos lugares $l_3^1$, $l_1^1$ e $l_5^0$ na rede de planos da figura 2.14

Com a inclusão de duas ou mais transições na rede de planos é necessária a criação de estruturas de controle que garantam que transições não sejam disparadas de modo a obter planos inconsistentes.

As contradições entre as pré-condições e efeitos de duas transições $t_a$ e $t_b$ que podem gerar planos inconsistentes na rede de planos são:

- 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 das inconsistências definidas para a rede de planos é possível descrever as relações de ordenação entre $t_a$ e $t_b$ como segue:

- $t_a$ e $t_b$ são concorrentes ($t_a \shortparallel t_b$): o disparo de $t_a$ com relação ao disparo de $t_b$ não resulta em planos inconsistentes, portanto, $t_a$ pode preceder $t_b$; $t_b$ pode preceder $t_a$; e $t_a$ pode disparar em paralelo com $t_b$;

- $t_a$ e $t_b$ são não-concorrentes ($t_a \nparallel t_b$): relação de ordenação gerada pela inconsistência do tipo 1. Apenas os disparos concorrentes entre $t_a$ e $t_b$ geram planos inconsistentes;

- $t_a$ precede $t_b$ ($t_a \triangleleft t_b$): gerada pela inconsistência do tipo 2. O disparo de $t_b$ antes de $t_a$ e o disparo concorrente de $t_a$ e $t_b$ geram planos inconsistentes;

- $t_a$ e $t_b$ são mutuamente exclusivos ($t_a \diamondsuit t_b$): quando as transições apresentam inconsistência do tipo 3. Neste caso os disparos são mutuamente exclusivos, ou seja, se ocorrer o disparo de $t_a$, $t_b$ não pode ser disparada na camada.

As relações de ordenação definidas na rede de planos são uma alternativa ao *mutex* interferência do grafo de planos. A dinâmica da rede de Petri dispensa a representação do *mutex* de competição de necessidade, porque o fluxo de marcas da rede evita o disparo simultâneo de transições do conflito de competição de necessidade.

Para adaptar a estrutura da rede de planos às novas relações de ordenação, o conceito de camada foi redefinido. Uma camada na rede de planos é composta por dois passos, sendo que o primeiro contém o conjunto de transições que precedem outra transição na relação de ordenação e o segundo passo contém o conjunto de transições que sucedem outra transição na relação de ordenação. Uma camada também é composta pelas cópias e novos lugares inseridos como efeito.

Para cada relação de ordenação foi definida uma estrutura de controle que possibilita o disparo de transições válidas, evitando inconsistências na rede.

A estrutura de controle do disparo de uma transição é apresentada na figura 2.16. Esta estrutura contém o lugar $I$ e o lugar $F$, sendo que $I$ indica o início do controle da camada e $F$ o fim do controle. Também apresenta as transições $\theta t$ e $\Phi t$, que respectivamente, representam o início do controle e o fim do controle do disparo da transição $t$.

Na estrutura de controle do disparo de uma transição também exite um lugar $l_1$ que representa um conflito: o disparo de $t$, que indica que a ação $t$ pertence ao plano, ou $\lambda t$ quando $t$ não pertence ao plano.

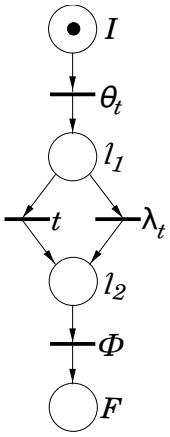


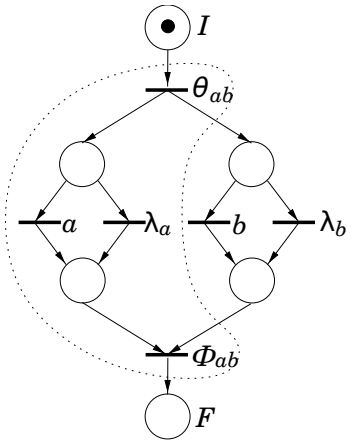

Figura 2.16: Estrutura de controle para o disparo de uma transição

A partir da estrutura de controle para o disparo de uma transição, é possível realizar a composição de estruturas de controle para as relações de ordenação descritas anteriormente para duas ações $a$ e $b$. Estas estruturas são apresentadas na figura 2.17.

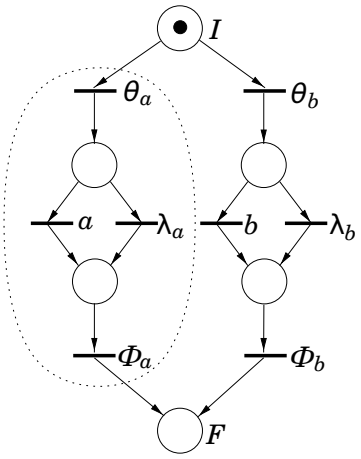
Na figura 2.17(a) a transição $\theta ab$ e $\Phi ab$ possibilitam a execução da relação de ordenação do tipo $a$ concorrente com $b$ ($a \shortparallel b$). As transições $a$ e $b$ podem disparar em paralelo; ou somente dispara a transição $a$; ou somente a transição $b$; ou $a$ e $b$ não disparam.

Do mesmo modo na figura 2.17(b), os arcos que conectam o lugar $I$ com as transições $\theta a$ e $\theta b$, acompanhado dos arcos que conectam $\Phi a$ e $\Phi b$ com o lugar $F$, possibilitam execução a relação de ordenação do tipo $a$ *mutex* $b$ ($a \diamondsuit b$).

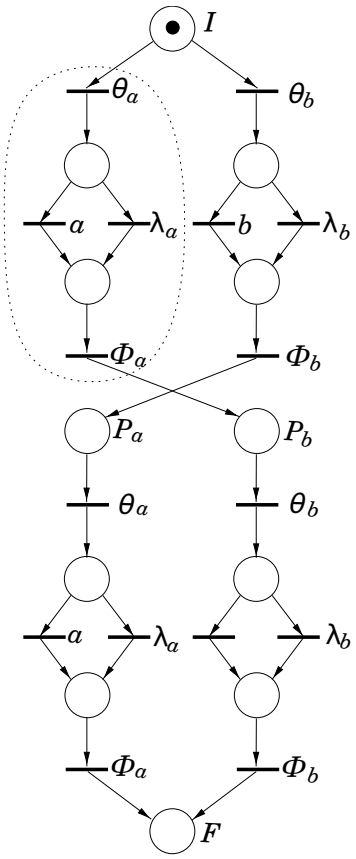
Nas figuras 2.17(c) e 2.17(d) os lugares $P_a$ ou $P_b$ são inseridos. Este lugares realizam o encadeamento e a ordenação das estruturas de controle entre os passos de uma camada, possibilitando a execução das relações de ordenação do tipo $a$ não-concorrente com $b$ ($a \nparallel b$) e do tipo $a$ precede $b$ ($a \triangleleft b$).



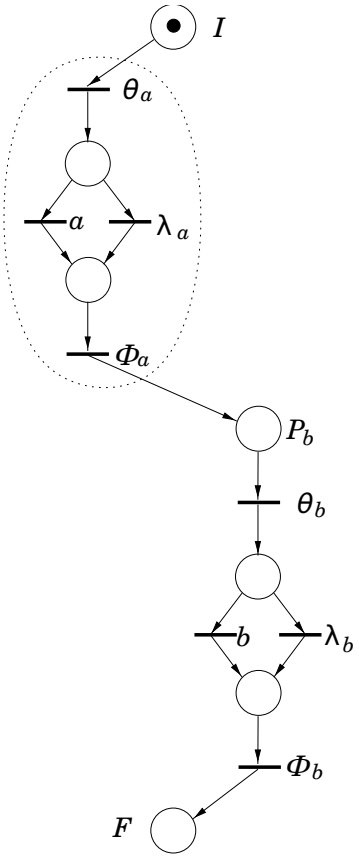

(a) Ação $a$ concorrente com ação $b$ ($a \shuffle b$)




(b) Ações $a$ e $b$ são mutex ($a \lozenge b$)




(c) Ações $a$ e $b$ não-concorrentes ($a \nmid b$)




(d) Ação $a$ precede ação $b$ ($a \lhd b$)

Figura 2.17: Disparo de uma transição em rede de Petri

Na composição de um conjunto de estruturas de controle é necessário inserir um lugar $I'$ entre os passos de uma camada, para garantir que apenas uma subestrutura seja executada na parte inferior da rede.

Para isso todas as subestruturas de controle do primeiro passo da camada devem estar ligadas com $I'$, como apresentado no exemplo de composição das estruturas de controle,

para as relações de ordenação $(a \nrightarrow b)$, $(a \nrightarrow c)$, $(a \diamondsuit d)$, $(b \shortparallel c)$, $(c \triangleleft d)$ e $(d \triangleleft b)$, que pode ser visualizado na figura 2.18(a).

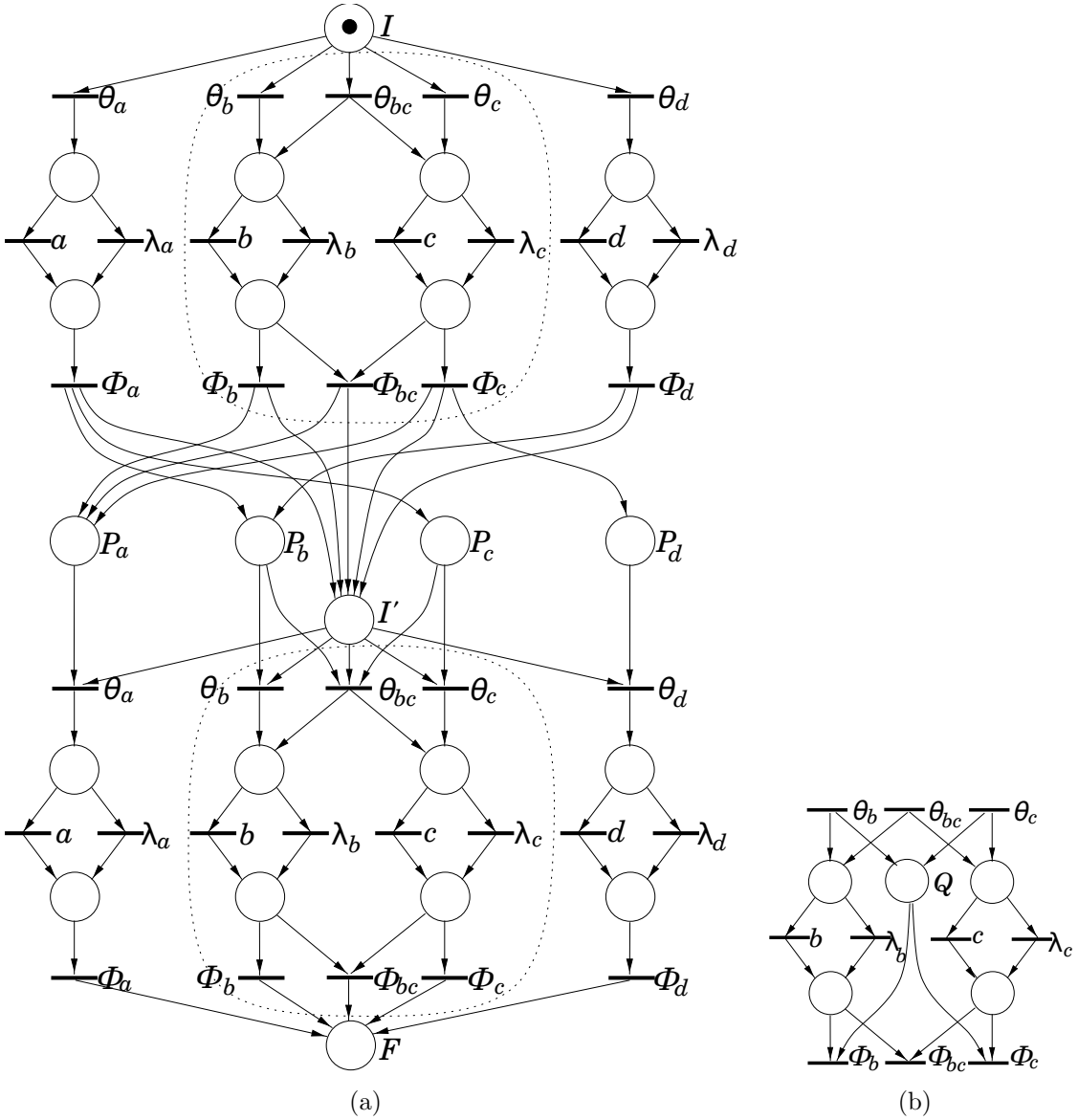



Figura 2.18: Exemplo de composição de estruturas de controle

A composição da estrutura de controle do tipo concorrente com outras estruturas de controle ainda pode resultar no disparo de transições inconsistentes, mesmo com a inclusão do lugar $I'$.

Por exemplo, na figura 2.18(a), o disparo da transição $\theta bc$ pode resultar no disparo da transição $\Phi b$ e $\Phi c$, resultando em duas marcas no lugar $I'$, o que possibilita a execução de mais de uma sub-estrutura no segundo passo da camada.

A solução para este problema é dada na figura 2.18(b), onde a inclusão dos arcos $(\theta_b$,

$Q$), ($\theta_c$, $Q$), ($Q$, $\Phi_b$) e ($Q$, $\Phi_c$) eliminam a possibilidade do disparo de transições de modo inconsistente. Se ocorrer o disparo de $\theta bc$, somente $\Phi bc$ pode disparar, consequentemente existirá apenas uma marca em $I'$.

Enfim, após definir o modelo da rede de planos, é possível definir o algoritmo de construção desta rede a partir do PDDL. O algoritmo de construção da uma rede de planos $R$ com $k$ camadas, que recebe como parâmetros o conjunto de ações do problema de planejamento $A$ e o estado inicial $S_o$, é dado pelo algoritmo 1 [24]:

**Algoritmo 1** Algoritmo para construção da rede de planos

RedeDePlanos($k$,$A$,$So$)

```
1:  Inclui So na rede de planos R;
2:  i ← 1;
3:  S ← So;
4:  while i ≤ k do
5:      A ← AçõesValidas(A,S);
6:      Inclui A na rede de planos;
7:      I ← CalculaInconsistências(A);
8:      C ← EstruturaDeControle(I);
9:      Inclui C na rede de planos R;
10:     S ← S∪ EfeitoDe(A);
11:     i ← i + 1;
12: end while
    Retorne R;
```

É possível verificar no algoritmo 1 (linha 1), que o estado inicial do problema é inserido a rede de planos. Na linha 2, o contador de camadas é inicializado com 1, pois o estado inicial é a primeira camada da rede. Na linha 3, o conjunto de proposições da rede de planos $S$ recebe as proposições que representam o estado inicial do problema. Em seguida o laço do enquanto (while) é repetido até que a rede de planos tenha o número de camadas igual à variável $k$.

A cada iteração do laço é obtido um subconjunto das ações $A$ do problema de planejamento, que contém suas pré-condições presentes na rede de planos. Estas ações são então inseridas na rede (linha 6) de acordo com as restrições apresentadas nesta seção. Por meio deste conjunto de ações pode-se realizar o cálculo das inconsistências, cujo algoritmo deve verificar se há inconsistência, duas a duas, entre todas as ações já inseridas.

O algoritmo de cálculo das inconsistências deve iniciar da ação mais restritiva para a

menos restritiva, retornando um conjunto de relações de ordenação $I$. Estas relações de ordenações possibilitam a construção das estruturas de controle.

As estruturas de controle podem ser inseridas na rede de planos (linha 9), bem como os lugares efeito das ações inseridas (linha 10). Por fim o número de camadas é incrementado (linha 11).

Para a construção de uma rede de planos consistente houve a necessidade de realizar uma análise mais aprimorada. Detalhamentos da construção da estrutura de controle e da inclusão de novas ações na rede serão apresentados na próxima seção.

### 2.2.3 Análise da rede de planos

Para realizar a análise de construção da rede de planos e identificar particularidades não descritas no algoritmo 1 foi utilizado como exemplo o problema de logística descrito no capítulo 1, o qual é um problema de logística de um comerciante que possui três lojas $l_1, l_2, l_3$, sendo que existe um caminho direto entre elas. A loja $l_2$ contém o pacote $p_1$, a loja $l_1$ contém o pacote $p_2$ e o comerciante tem a sua disposição um caminhão $c$ que está na loja $l_3$ (estado inicial). O comerciante precisa que $p_1$ esteja em $l_1$ e $p_2$ esteja em $l_2$ (estado objetivo). Este problema é definido em PDDL na introdução do capítulo 2, nas figuras 2.2, 2.3 e 2.4.

A rede de planos do problema do comerciante, composta pelo estado inicial e uma camada, é apresentada na figura 2.19. Esta camada contém a estrutura de controle e o conjunto de lugares que representam as proposições.

A estrutura de controle, na figura 2.19 foi construída a partir da relação de ordenação $dirigir(c, l_3, l_2) \diamondsuit dirigir(c, l_3, l_1)$, a qual tem apenas um passo na camada.

Dois lugares foram necessários para representar a proposição $em(c, l_3)$, pois $dirigir(c, l_3, l_2)$ e $dirigir(c, l_3, l_1)$ tem esta proposição como pré-condição.

Quando na figura 2.19 é adicionada a estrutura de controle do primeiro passo da segunda camada gera-se uma nova rede, cuja subrede é apresentada na figura 2.20. As

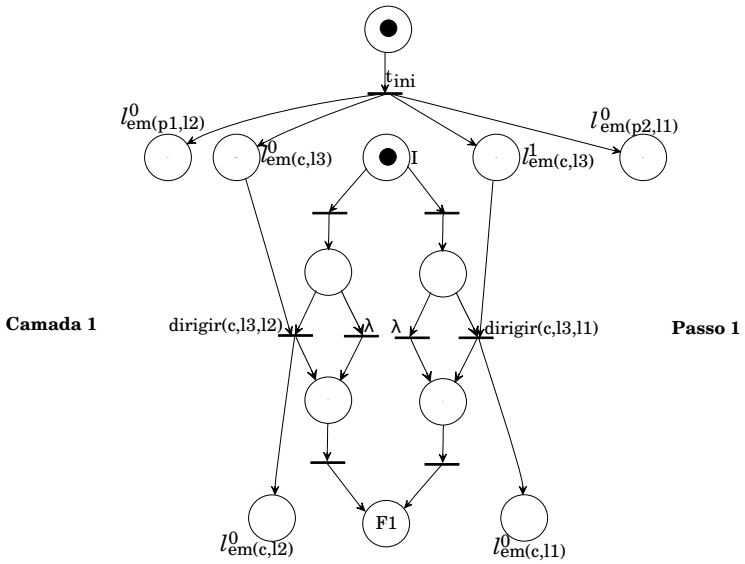



Figura 2.19: Rede de planos com uma camada

relações de ordenação desta subrede são: $dirigir(c, l_3, l_2) \Diamond dirigir(c, l_3, l_1)$, $dirigir(c, l_3, l_2) \shuffle carregar(p_2, c, l_1)$, $dirigir(c, l_3, l_1) \shuffle carregar(p_2, c, l_1)$, $dirigir(c, l_3, l_2) \shuffle carregar(p_1, c, l_2)$, $dirigir(c, l_3, l_1) \shuffle carregar(p_1, c, l_2)$, $carregar(p_1, c, l_2) \shuffle carregar(p_2, c, l_1)$.

As relações de ordenação foram obtidas a partir do cálculo das inconsistências mais restritivas para as menos restritivas.

Na construção da rede de planos da figura 2.20 não é possível conectar o lugar $F_1$ com o lugar $I$, para reaproveitar as estruturas de controle da primeira camada, pois a rede de planos é uma rede de Petri acíclica.

Assim as estruturas de controle das ações da primeira camada estão contidas na segunda camada, pois as pré-condições destas ações estão contidas na rede. Mas a relação de ordenação $dirigir(c, l_3, l_2) \Diamond dirigir(c, l_3, l_1)$ é representada na segunda camada de modo a aproveitar as estruturas de paralelismo das outras relações de ordenação.

Com as relações de ordenação $dirigir(c, l_3, l_2) \Diamond dirigir(c, l_3, l_1)$, $dirigir(c, l_3, l_2) \shuffle carregar(p_2, c, l_1)$, $dirigir(c, l_3, l_1) \shuffle carregar(p_2, c, l_1)$, $dirigir(c, l_3, l_2) \shuffle carregar(p_1, c, l_2)$, $dirigir(c, l_3, l_1) \shuffle carregar(p_1, c, l_2)$, $carregar(p_1, c, l_2) \shuffle carregar(p_2, c, l_1)$, a rede de planos com duas camadas é a rede apresentada na figura 2.21.

Para a inclusão do segundo passo da camada 2 na rede de planos da figura 2.21 foi necessário respeitar as definições a seguir. Estas foram obtidas por meio de estudos

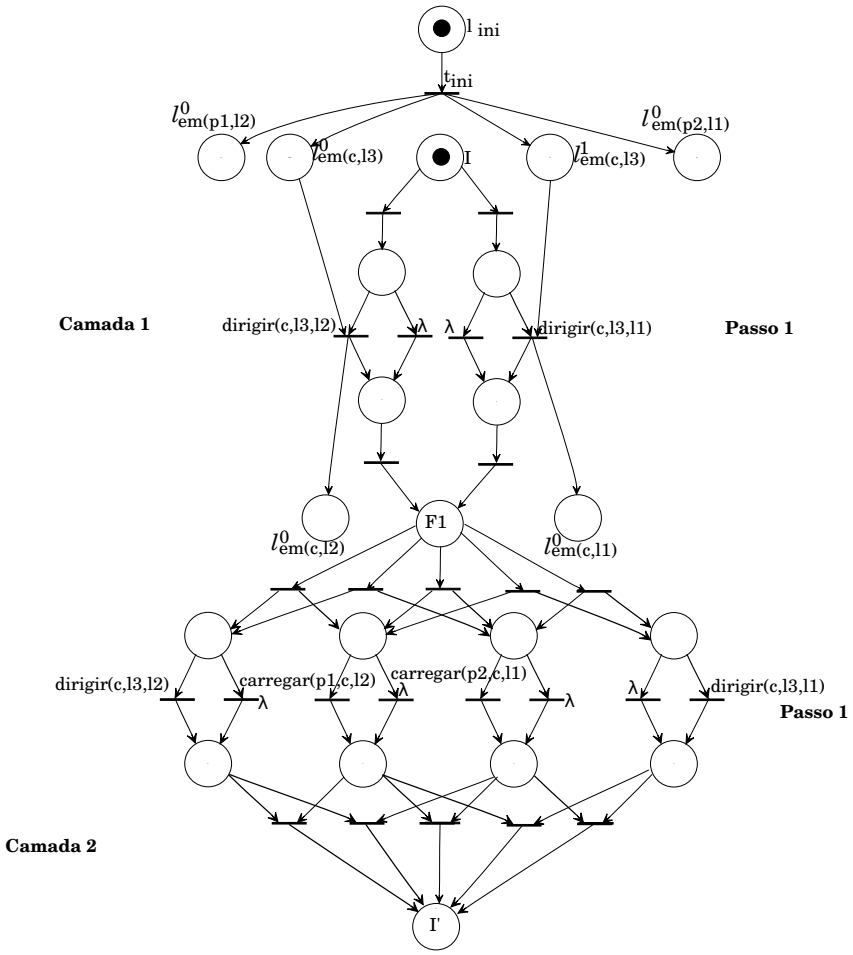



Figura 2.20: Rede de planos com uma camada e com um passo da segunda camada

aprofundados da rede de planos, que não foram descritos de modo explicito no trabalho de Silva [24].

**Definição 1** *A estrutura de controle de uma ação A é inserida no segundo passo de uma camada, se A pertence à pelo menos duas relações de ordenação paralelas, ou se A pertence a pelo menos uma relação de ordenação não concorrente ou pertence ao segundo passo de uma relação de precedência.*

Como as ações $dirigir(c, l_3, l_2)$, $dirigir(c, l_3, l_1)$, $carregar(p_2, c, l_1)$ e $carregar(p_1, c, l_2)$ do exemplo pertencem a duas relações de ordenação paralelas, no segundo passo da camada 2 da figura 2.21, existe uma estrutura de controle para cada uma destas ações.

**Definição 2** *Para cada transição de uma subestrutura de controle do primeiro passo de uma camada verifica-se, de acordo com as relações de ordenação, a possibilidade de*

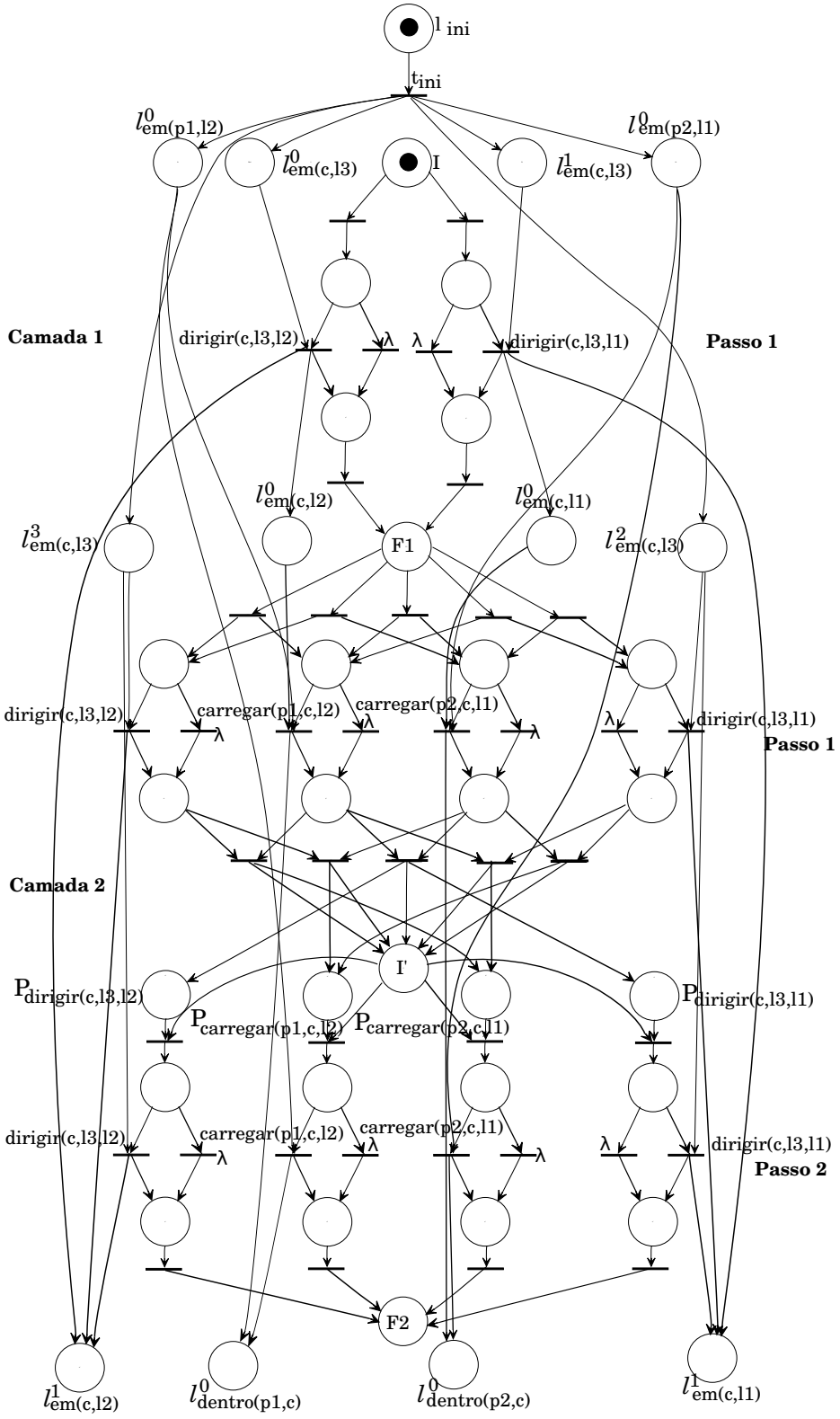



Figura 2.21: Rede de planos com duas camadas

*ligação desta subestrutura com o lugar $P$ de todas as ações do segundo passo da camada.*

Por exemplo, na figura 2.21 a estrutura de controle concorrente $dirigir(c, l_3, l_2)$ e

$carregar(p_1, c, l_2)$ é ligada com $P_{carregar(p2,c,l1)}$, porque $carregar(p_2, c, l_1)$ é paralelo com $dirigir(c, l_3, l_2)$ e $carregar(p_1, c, l_2)$. As demais ligações das estruturas de controle do primeiro passo com o segundo passo da camada 2 seguem a mesma ideia.

**Definição 3** *Para todos os lugares $P's$ que são ligados a pelo menos uma subestrutura em comum do primeiro passo, são criadas estruturas de controle concorrentes entre as ações dos $P's$ no segundo passo, se a relação de ordenação entre estas ações for concorrente.*

No exemplo da figura 2.21 apenas $dirigir(c, l_3, l_2)$ e $dirigir(c, l_3, l_1)$ recebem ligação de uma mesma subestrutura do primeiro passo, mas estas ações são mutuamente exclusivas, logo, no segundo passo da camada 2 não há nenhum paralelismo entre as ações.

Como esperado, as relações de ordenação não evitam as inconsistências que eram resolvidas pelos conflitos de competição de necessidade. Por exemplo, as ações $dirigir(c, l_3, l_2)$ e $carregar(p_1, c, l_2)$ não podem ser aplicadas em paralelo, entretanto a relação de ordenação entre estas ações é paralela.

Na rede de planos esta inconsistência é controlada pelo fluxo de marcas, portanto este fluxo sempre deve gerar estados consistentes. Se um lugar contiver uma marca que foi gerada por meio de ações que negam um conjunto de proposições $p$, o conjunto de lugares que representam $p$ não podem estar marcados.

Seguindo esta ideia foi feita uma análise do conjunto de lugares com relação ao fluxo de marcas na rede de planos da figura 2.21.

Quando a transição $t_{ini}$ dispara, uma marca é criada em: $l^0_{em(p_1,l_2)}$, $l^0_{em(p_2,l_1)}$, $l^0_{em(c,l_3)}$, $l^1_{em(c,l_3)}$, $l^2_{em(c,l_3)}$, $l^3_{em(c,l_3)}$.

Se $dirigir(c, l_3, l_1)$ disparar na primeira camada, a marca que está no lugar $l^1_{em(c,l_3)}$ é consumida, mas a marca dos outros lugares que representam $em(c, l_3)$ não são consumidas, o que pode gerar inconsistência na rede não controladas pelas relações de ordenação.

Como existe uma marca em $l^3_{em(c,l_3)}$, a transição $dirigir(c, l_3, l_2)$ pode disparar na segunda camada, gerando marcas nos lugares que representam a proposição $em(c, l_2)$, o que é inconsistente com o resultado obtido na primeira camada.

A solução deste tipo de inconsistência é uma contribuição deste trabalho de mestrado. Para resolver a inconsistência de manutenção das proposições válidas entre as camadas

na rede de planos, um lugar $l$ pertencente ao segundo passo da camada $n$, só pode ser marcado se:

- existir uma ação no segundo passo de $n$ cuja transição ao ser disparada cria uma marca em $l$;
- existir uma ação no primeiro passo de $n$ cuja transição ao ser disparada cria uma marca em $l$, sendo que outra ação que remove a proposição representada por $l$ não teve sua transição disparada no segundo passo de $n$;
- existir uma ação em $n-1$ cuja transição ao ser disparada cria uma marca em $l$, sendo que o disparo das transições em $n$ não representam ações que removam $l$.

Para atender essas novas regras de manutenção para a rede de planos, criou-se um mecanismo para manutenção das proposições, o qual é descrito a seguir e exemplificado na figura 2.22.

Seja um conjunto de ações válidas $A$, para uma camada $n$, onde a ação $a \in A$ e $t_1$ e $t_2$ são as transições que representam $a$ em $n$, respectivamente, no passo 1 e no passo 2 de $n$. Seja também um conjunto de proposições $P$, onde a proposição $p \in P$ e $p$ é removida por $a$. A manutenção de $p$ é realizada do seguinte modo:

1. para realizar a manutenção de $p$ no primeiro passo de $n$, cria-se um lugar $\epsilon_p^{n.1}$, pré-condição da transição $\epsilon t_p^{n.1}$. Esta transição tem como efeito todos os lugares que representam $p$ e que são pré-condição de transições do segundo passo de $n$;
2. se $p$ também é removida no segundo passo de $n$ cria-se o lugar $\epsilon_p^{n.2}$, pré-condição da transição $\epsilon t_p^{n.2}$ e efeito de $\epsilon t_p^{n.1}$. O lugar $\epsilon_p^{n.2}$ também recebe um arco de todo $t_1$ cujo $a$ adiciona $p$. A transição $\epsilon t_p^{n.2}$ tem como efeito todos o lugares que representam $p$ e que são pré-condição de transições primeiro passo de $n+1$;
3. se $p$ também é removida em $n-1$, a transição $\epsilon t_p^{n-1.x}$ da camada $n-1$ tem como efeito $\epsilon_p^{n.1}$ para o maior $x$ da camada. Se o maior $x$ é igual a 1 ($p$ é removido somente no primeiro passo de $n-1$), então o lugar $\epsilon_p^{n.1}$ recebe um arco de todas as transições

do primeiro e do segundo passo de $n-1$, cujas ações adicionam $p$. Se o maior $x$ é igual a 2, então o lugar $\epsilon_p^{n.1}$ recebe um arco de todas as transições do segundo passo de $n-1$, cujas ações adicionam $p$;

4. se $p$ não é removida em $n-1$, $\epsilon_p^{n.1}$ pode ser substituído pelo lugar $l_p^k$, cujo $k$ é o maior para $p$ nas camadas anteriores a $n$, se $l_p^k$ não é utilizado como pré-condição por ações que não removem $p$.;

5. para todo $t_1$ cujo $a$ remove uma proposição $p$, cria-se um arco ($\epsilon_p^{n.1}$, $t_1$);

6. para todo $t_2$ cujo $a$ remove uma proposição $p$, cria-se um arco ($\epsilon_p^{n.2}$, $t_2$).

Na figura 2.22 é apresentado um exemplo das novas regras de manutenção das proposições válidas. Não foram adicionadas as estruturas de manutenções da proposição $em(p_2, l_1)$ para facilitar a compreensão das alterações realizadas.

Por exemplo, na figura 2.22 foi criado o lugar $\epsilon_{em(c,l3)}^{2.1}$, pré-condição da transição $\epsilon t_{em(c,l3)}^{2.1}$ (item 1). Esta transição, por sua vez, esta ligada a estrutura de manutenção da proposição $em(c, l3)$ do segundo passo da camada 2 (item 2) e o lugar $\epsilon_{em(c,l3)}^{2.1}$ é efeito da estrutura de manutenção de $em(c, l3)$ da camada 1 (item 3). Conforme definição do item 4, o lugar $l_{em(p1,l2)}^0$ realiza a mesma função de um lugar $\epsilon_{em(p1,l2)}^{2.1}$.

De acordo com a definição do item 5, o lugar $\epsilon_{em(c,l3)}^{2.1}$ é pré-condição das transições $dirigir(c, l_3, l_2)$ e $dirigir(c, l_3, l_1)$. O conflito gerado em $\epsilon_{em(c,l3)}^{2.1}$ reflete as informações de que $dirigir(c, l_3, l_2)$ e $dirigir(c, l_3, l_1)$ são mutuamente exclusivas.

Com base na definição de ações mutuamente exclusivas da rede de planos e nas novas regras de manutenção é possível realizar uma simplificação na rede, a qual é descrita na definição a seguir.

**Definição 4** *Para todas as ações A, onde as ação $a_1$ e $a_2$ pertencem a A e $a_1$ e $a_2$ removem uma proposição p, que também é pré-condição de $a_1$ e $a_2$, o arco ($l_p^k$, $t_{a1}$) e ($l_p^k$, $t_{a2}$), onde $t_{a1}$ é a transição que representa $a_1$ e $t_{a2}$ é a transição que representa $a_2$, podem, respectivamente, serem substituídos pelos arcos ($\epsilon_p^{n.1}$, $t_{a1}$) e ($\epsilon_p^{n.1}$, $t_{a2}$), no primeiro passo de n, e pelos arcos ($\epsilon_p^{n.2}$, $t_{a1}$) e ($\epsilon_p^{n.2}$, $t_{a2}$), no segundo passo de n, sendo que os lugares $l_p^k$'s pode então serem removidos.*

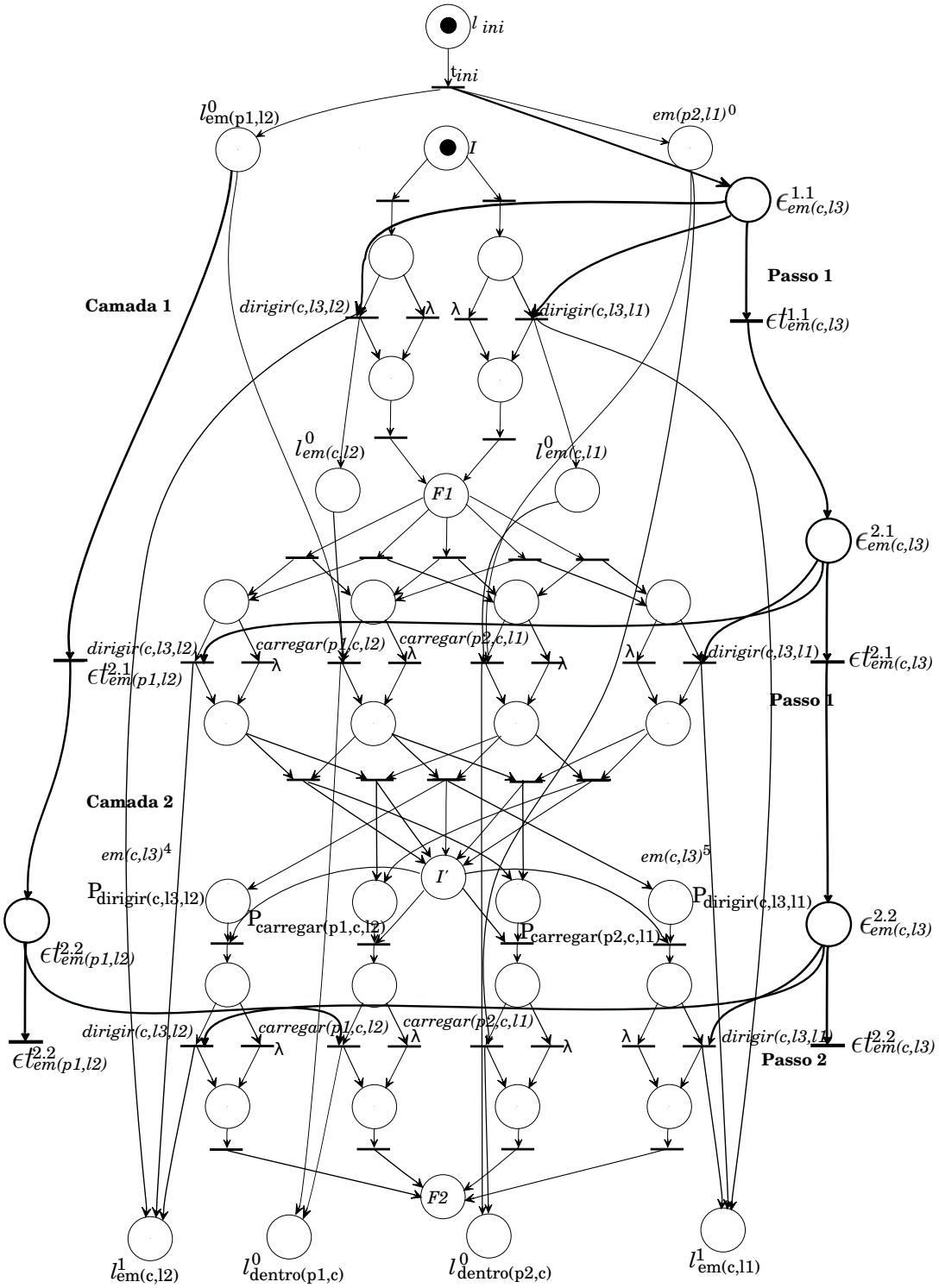



Figura 2.22: Rede de planos com manutenção das proposições

Esta simplificação é possível, pois as ações $a_1$ e $a_2$ são mutuamente exclusivas. O conflito estrutural gerado não impedirá o paralelismo entre ações na camada.

É devido a esta simplificação que, por exemplo, na figura 2.22 não existe um arco entre copias do lugar que representa a proposição $em(c, l_3)$ e as ações $dirigir(c, l_3, l_1)$ e $dirigir(c, l_3, l_2)$. O lugar $\epsilon^{2.1}_{em(c,l3)}$ é pré-condição das transições $dirigir(c, l_3, l_2)$ e da transição $dirigir(c, l_3, l_1)$.

Analisando a manutenção das proposições válidas na figura 2.22, constata-se que quando a transição $t_{ini}$ dispara, uma marca é criada em: $em(p_1, l_2)^0$, $em(p_2, l_1)^0$ e $\epsilon^{1.1}_{em(c,l3)}$.

Por exemplo, se a transição $dirigir(c, l_3, l_1)$ disparar, a marca de $\epsilon^{1.1}_{em(c,l3)}$ é consumida e uma marca é criada no lugar $l^0_{em(c,l_1)}$. Consequentemente a transição $\epsilon t^{2.1}_{em(c,l3)}$ não poderá disparar, impedindo que as transições da camada 2 que tiverem $\epsilon^{2.1}_{em(c,l3)}$ sejam disparadas.

Caso contrário, se $dirigir(c, l_3, l_1)$ e $dirigir(c, l_3, l_2)$ não dispararem, $\epsilon t^{1.1}_{em(c,l3)}$ pode disparar, consumindo uma marca de $\epsilon^{1.1}_{em(c,l3)}$ e gerando uma marca em $\epsilon^{2.1}_{em(c,l3)}$, possibilitando o disparo das transições $dirigir(c, l_3, l_1)$ e $dirigir(c, l_3, l_2)$ na camada 2.

Embora não apresentado na figura 2.22, as regras de manutenção das proposições possibilitam o disparo de transições do segundo passo de uma camada, devido ao disparo de transições do primeiro passo desta camada. O lugar de manutenção (por exemplo, $\epsilon^{n.2}_p$, onde $p$ é uma proposição representada no segundo passo da camada $n$) é efeito de transições do primeiro passo de $n$ e é pré-condição de transições do segundo passo de $n$. Deste modo, criou-se a ligação entre o primeiro e o segundo passo de $n$. Esta ligação captura resultados do primeiro passo que podem ser imediatamente utilizados como pré-condição no segundo passo.

Outra proposta é que o conjunto de lugares que representam uma proposição $p$, que são utilizados como pré-condição no segundo passo de uma camada, sejam efeitos das transições do primeiro passo da camada, desde que $p$ seja adicionada por alguma ação presente na rede. Assim a ligação entre o primeiro e o segundo passo de uma camada não se restringe apenas aos lugares de manutenção. Os efeitos do primeiro passo podem ser as pré-condições das ações do segundo passo de uma camada.

Em resumo, uma das vantagens ao se utilizar a rede de planos é a simplificação da

estrutura de representação do problema de planejamento, por meio da eliminação das ações de manutenção, *mutex* de competição de necessidade. Estas simplificações são possíveis mediante ao aproveitamento da dinâmica das redes de Petri, a qual permite representar fluxo de informações.

Outra vantagem é a representação de relações de ordenação, as quais possibilitam adicionar em uma camada ordenações entre pares de ações, que são mais ricas em informações do que o *mutex* de interferência do grafo de planos.

Uma desvantagem é com relação ao critério de parada na expansão da rede quando não é possível encontrar plano. No grafo de planos, se a última camada é igual à penúltima camada de proposições, inclusive nos *mutex*, não existe plano. Na rede de planos não existe *mutex* de proposições e não existe *mutex* de competição de necessidade.

No critério de parada da rede de planos é verificado se a última camada contém exatamente as mesmas proposições da penúltima camada. Neste instante inicia-se a contagem do número de camadas seguintes, que não pode ser superior a combinação do número de proposições duas a duas. Este critério de parada pode fazer com que a rede de planos pare com mais camadas do que o grafo de planos quando não existe solução para um determinado problema de planejamento.

Para a ideal apresentação deste trabalho é importante descrever trabalhos científicos referentes às estruturas convertidas para instância SAT como método de solução para o problema de planejamento. O SATPLAN foi um marco histórico nesta linha de pesquisa e novos resultados foram obtidos com a conversão de redes de Petri de planejamento para instâncias SAT.

## 2.3 Estruturas convertidas para uma instância SAT

Em 1992, Kautz e Selman impulsionaram as pesquisas em planejamento em IA quando propuseram o SATPLAN, um planejador que transforma o problema de planejamento definido em STRIPS em uma instância SAT em CNF [13].

Outro padrão que também pode ser utilizado nos métodos de solução do problema de planejamento é o formato ISCAS, pois existem resolvedores SAT eficientes que utilizam este padrão como entrada.

### 2.3.1 SATPLAN

No SATPLAN a tradução do problema de planejamento para uma instancia SAT é feita partindo-se do princípio de que uma proposição e uma ação pertencem a um determinado instante de tempo, pois um plano é uma sequência cronológica de ações e o problema de satisfatibilidade é um problema estático de valoração. Os literais gerados nesta tradução pertencem a um instante de tempo particular [13].

Se a teoria (fórmula proposicional em CNF) gerada for satisfatível, existe um plano cuja sequência de ações é dada pela valoração do resolvedor SAT, caso contrário a profundidade do espaço de estados é incrementada e a teoria é gerada novamente.

Essa teoria é gerada a partir das seguintes regras:

1. o estado inicial é especificado pela conjunção dos literais nele contido para o instante de tempo 0;

2. o estado final é especificado pela conjunção dos literais nele contido para o instante de tempo igual a profundidade do espaço de estados;

3. uma ação implica na conjunção dos seus efeitos.

Embora as regras para construção da teoria apresentadas garantam que os literais do estado inicial e final sejam verdadeiros para qualquer valoração satisfatível (item 1 e 2), elas não são suficientes para obtenção de planos válidos.

Pelo item 3, se uma determinada ação $a$ receber o valor verdade, obrigatoriamente os efeitos desta ação, por exemplo $e_1$ e $e_2$, também deverão ser verdadeiros, conforme apresentado na equação 2.7.

$$a \rightarrow (e_1 \wedge e_2)$$

$$\neg a \vee (e_1 \wedge e_2)$$

$$(\neg a \vee e_1) \wedge (\neg a \vee e_2) \tag{2.7}$$

Entretanto, apenas com estas regras, uma determinada ação pode ser verdadeira mesmo que suas pré-condições sejam falsas; e duas ou mais ações podem ser verdadeiras em um instante de tempo de modo inconsistente. As valorações satisfatíveis para a fórmula podem resultar em planos inconsistentes. Modelos que não correspondem a planos válidos são denominados modelos anômalos [13].

Para eliminação dos modelos anômalos foi proposta a inclusão do seguinte conjunto de infomações à fórmula:

- para que uma determinada ação seja válida apenas se suas pré-condições forem válidas, está ação deve implicar na conjunção de suas pré-condições;
- para que apenas uma ação seja válida em um determinado instante de tempo, cada ação deste instante de tempo implica na conjunção da negação das demais ações deste instante de tempo.

Com a adição destas informações à fórmula do SATPLAN foi possível resolver os modelos anômalos. O SATPLAN passou a obter planos válidos em domínios e problemas que o planejador STRIPS não conseguia resolver.

A partir da codificação da estrutura do espaço de estados para fórmulas em CNF, descrita acima e denominada codificação linear (*Linear Encodings*) [10], outras codificações de estruturas de dados foram criadas. Uma delas é a codificação em CNF baseada no grafo de planos (*Graphplan-based Encodings*), que é definida da seguinte forma:

1. os literais do estado inicial representam as proposições da camada 0 e os literais do estado final representam as proposições do estado final contidas na última camada. Os literais do estado inicial e final são cláusulas unitárias na fórmula em CNF;

2. toda proposição de uma camada $i$ implica na disjunção das ações de $i-1$ que a adicionam;

3. as ações de uma camada $i$ implicam na conjunção de suas pré-condições da camada $i-1$;

4. ações mutuamente exclusivas em uma camada $i$ são representadas pela disjunção entre a negação destas ações na camada $i$.

Na codificação baseada no grafo de planos é possível realizar simplificações que resultam em uma cláusula que determina a mudança de estado de uma determinada proposição $p$, pertencente a uma camada $i-1$ e $i+1$. Com parte da fórmula gerada pelos itens 2 e 3, a mudança de estado de $p$ ocorre se ao menos uma ações da camada $i$, que adiciona $p$, for válida.

Por exemplo, seja um grafo de planos, cuja proposição $em(c, l_1, i+1)$ pode ser obtida pelas ações, $dirigir(c, l_3, l_1, i)$, $dirigir(c, l_2, l_1, i)$ e $manutencao(c, l_1, i)$.

A sub-fórmula 2.8 expressa a mudança de estado da proposição $em(c, l_1)$, em função das ações da camada $i$, com exceção da ação de manutenção.

$$(\neg em(c, l_1, i-1) \wedge em(c, l_1, i+1)) \rightarrow (dirigir(c, l_3, l_1, i) \vee dirigir(c, l_2, l_1, i)) \quad (2.8)$$

$$em(c, l_1, i-1) \vee \neg em(c, l_1, i+1) \vee dirigir(c, l_3, l_1, i) \vee dirigir(c, l_2, l_1, i) \quad (2.9)$$

A ação de manutenção não pertence a sub-fórmula 2.8, pois esta ação mantém o estado da proposição $em(c, l_1)$ constante. Ao converter sub-fórmula 2.8 para CNF tem-se a cláusula 2.9, na qual se $em(c, l_1, i+1)$ for verdadeiro, obrigatoriamente uma ação da camada $i$ é verdadeira, ou $em(c, l_1, i-1)$ é verdadeiro.

A fórmula proposicional gerada a partir do grafo de planos também pode ser manipulada a ponto de conter apenas variáveis de ações, ou apenas de proposições. A eliminação destas variáveis pode ser feita pela execução de todas as resoluções possíveis sobre uma variável com a eliminação de todas as cláusulas que contenham esta variável.

Fórmulas proposicionais baseadas no grafo de planos, cujas variáveis representavam apenas ações foram utilizadas no planejador BLACKBOX e no SATPLAN, o ganhador da competição de planejadores para domínios determinísticos de 2004 [12].

Na versão 2004 do SATPLAN, não foi realizada a propagação *mutex* durante a geração do grafo de planos, porque em alguns problemas as fórmulas eram tão grandes, devido às cláusulas *mutex*, que excediam o limite de memória. Apenas os *mutex* de interferência foram codificados [12].

No SATPLAN, ganhador da competição de planejadores para domínios determinísticos de 2006, a codificação do grafo de planos em fórmulas CNF manteve as variáveis geradas a partir das proposições e ações. Também foi realizada a propagação de *mutex*, mas somente as proposições mutuamente exclusivas e os *mutex* de interferência foram codificados. Esta estratégia de codificação possibilitou a resolução de problemas difíceis, com uso eficiente de memória.

No SATPLAN, tanto na versão de 2004, quanto na versão de 2006, é realizado um pós-processamento para remover (alguma das) ações desnecessárias no plano. O pós-processamento é útil porque o plano obtido pode conter ações que não são realmente necessárias para obter o estado objetivo do problema de planejamento [12, 14].

Além da codificação linear e a codificação baseada no grafo de planos, as redes de Petri para solução do problema de planejamento também foram convertidas para instancias SAT. Estas instâncias foram geradas no formato não clausal.

### 2.3.2 Codificação da rede de Petri no formato ISCAS

A codificação da rede de Petri para fórmulas proposicionais utilizava como base a estrutura da primeira versão do método de planejamento com redes de Petri, o PETRI-PLAN [18].

A primeira proposta de Montaño para resolver o problema de planejamento com fórmulas proposicionais geradas a partir das redes de Petri foi representada em NNF

(Forma Normal Negativa). A valoração para as variáveis da fórmula em NNF, foram obtidas por meio de uma fórmula em FNNF (*Factored Negation Normal Form*) equivalente, pois é possível encontrar a valoração SAT para uma fórmula em FNNF em tempo polinomial. A conversão da fórmula em NNF para FNNF é de alta complexidade computacional.

A rede de Petri resultante do problema de planejamento não foi codificada em CNF, pois o formato NNF pode ser obtido de modo natural. Uma transformação para CNF acrescentaria mais um passo computacional que poderia destruir a estrutura original da fórmula [28]. Se a fórmula original permanece intacta, um resolvedor SAT pode explorar a estrutura do problema original [27]

Uma alternativa não clausal explorada por Montaño [19] foi o formato ISCAS, que é um padrão utilizado para codificar circuitos aritméticos.

No formato ISCAS é possível representar variáveis e operadores. As variáveis são os dados de entrada do circuito, sendo definidas pela palavra reservada INPUT. Os operadores são atributos que recebem o resultado de operações lógicas entre uma ou mais variáveis e/ou um ou mais operadores. Dentre outras, as operações lógicas permitidas no formato ISCAS são: $AND$, $NAND$, $OR$, $NOR$, $XOR$ e $NOT$.

O resultado da teoria gerada em ISCAS é definido pela palavra reservada OUTPUT, a qual é a raiz do grafo direcionado acíclico gerado partir desta teoria. Este grafo, gerado a partir de uma fórmula em ISCAS, permite que os nós filhos tenham múltiplos pais.

Por exemplo, a fórmula $((\neg a \vee b) \wedge \neg a)$, é escrita no formato ISCAS de acordo com a figura 2.23.

$$
\begin{array}{|l|}
\hline
INPUT(a) \\
INPUT(b) \\
INPUT(c) \\
OUTPUT(out) \\
NOT_A = NOT(A) \\
D = OR(NOT_A, b) \\
out = AND(D, NOT_A) \\
\hline
\end{array}
$$

Figura 2.23: Exemplo do formato ISCAS

As variáveis de entrada da fórmula 2.23 são as mesmas variáveis da fórmula $((\neg a \vee$

$b) \wedge \neg a)$. Para obter o resultado é necessário representar os operadores $NOT$, $OR$ e $AND$, sendo que o operador $NOT_A$ é argumento dos dois outros operadores na fórmula. Esse reaproveitamento de operadores pode ser utilizado para representar apenas uma vez lugares ou transições da rede de Petri.

Na codificação do problema de planejamento em redes de Petri proposto por Montaño [19], os conflitos da rede de Petri são as variáveis de entrada da fórmula. Estes conflitos são sempre binários, o que permite atribuir a uma das opções a afirmação da variável e a outra a negação da variável, conforme apresentado na figura 2.24.

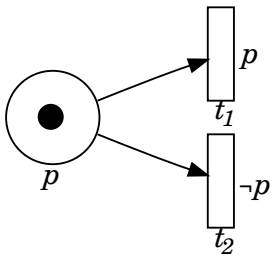
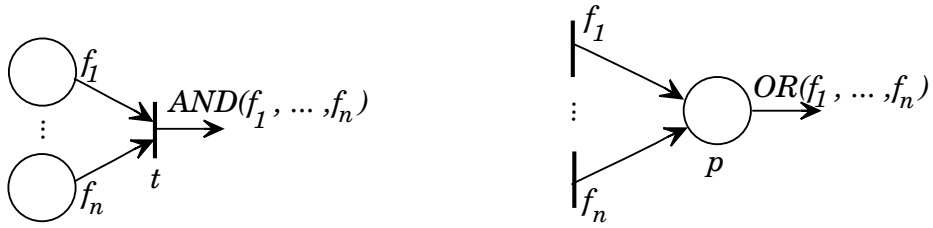



Figura 2.24: Representação do *mutex* da rede de Petri em ISCAS

A tradução da rede de Petri do problema de planejamento para o formato não clausal ISCAS ocorre por meio do caminhamento na rede, a partir dos lugares do estado objetivo até o estado inicial. Caso cada um dos lugares que chegam a uma transição $t$ tenham uma fórmula associada, então o operador $t$, em ISCAS, é igual a conjunção destas fórmulas, como por exemplo representado na figura 2.25(a). Caso cada uma das transições que chegam a um lugar $p$ tenham fórmulas associadas, então o operador $p$ é igual a disjunção destas fórmulas, conforme apresentado na figura 2.25(b).




(a) Lugares que chegam numa transição (b) Transições que chegam a um lugar

Figura 2.25: Tradução da rede de Petri para ISCAS

Nessa codificação para o formato ISCAS, o estado objetivo é uma conjunção dos operadores de cada uma de suas proposições. Estes operadores, por sua vez, são uma

disjunção de possibilidades para se obter a proposição do estado objetivo. Cada uma destas disjunções é uma conjunção das variáveis que estão contidas nestas possibilidades. Como as variáveis são os conflitos da rede, as estruturas que não tem conflitos não são codificadas.

A fórmula gerada por meio desta codificação é a entrada para um resolvedor SAT para fórmulas em ISCAS. O resolvedor SAT utilizado por Montaño foi o NFLSAT.

A valoração satisfatível, gerada pelo NFLSAT, permite realizar podas na rede, de tal modo a resolver todos os conflitos pertinentes, sendo possível obter o plano com o caminhamento na rede, a partir do estado objetivo até o estado inicial.

Em experimentos comparativos com o SATPLAN e a codificação original em FNNF, para os domínios *Blocks*, *Gripper* e *Logistics*, a codificação em ISCAS teve uma melhora significativa com relação a codificação original em FNNF. Os resultados foram mais próximos ao SATPLAN [19].

Para melhorar a codificação do problema, deve-se usar outras técnicas que diminuam o número de operadores da fórmula, ou produzam fórmulas com uma estrutura mais alinhada com as técnicas do resolvedor SAT.

# CAPÍTULO 3

# REPRESENTAÇÃO DA REDE DE PLANOS EM FÓRMULAS PROPOSICIONAIS

Um problema de planejamento em PDDL pode ser resolvido por meio da rede de planos. O mapeamento desta rede para fórmulas proposicionais, juntamente com um resolvedor SAT eficiente, permite encontrar variáveis que tornam a fórmula satisfatíveis e consequentemente encontrar as ações que compõem um plano.

Montaño realizou a codificação da rede de Petri do problema de planejamento para o formato NNF. Ele obteve resultados promissores ao utilizar o formato ISCAS e o resolvedor NFLSAT [19].

Entretanto, fórmulas proposicionais clausais foram utilizadas no SATPLAN. A codificação destas fórmulas foi baseada tanto na árvore de estados, quanto no grafo de planos, inclusive com algumas manipulações e simplificações, conforme detalhado na seção 2.3.1. Atualmente o SATPLAN é uma das mais eficientes técnicas de planejamento.

Em virtude desses resultados, neste trabalho foi definida uma codificação da rede de planos no formato ISCAS e uma codificação em CNF com base na rede de planos. Estas codificações serão detalhadas nas seções a seguir.

## 3.1 Codificação da rede de planos em fórmulas ISCAS

Na codificação da rede de planos para o formato ISCAS, as transições que representam as ações são variáveis da fórmula, sendo identificadas pelo símbolo $t$. Como estas transições são utilizadas para compor as estruturas de controle, $t$ é seguido do seu identificador único $i$ e pelo número da camada $c$ e passo $p$ correspondentes na estrutura de controle. Assim uma ação em ISCAS é representada por $t\_i\_c\_p$.

Por exemplo, a ação $t$ com $i = 0$, pertencente a camada 1 e ao passo 2 da estrutura de controle, traduzida para fórmula ISCAS é $t\_0\_1\_2$.

As proposições do estado inicial também são variáveis da fórmula, sendo identificadas pelo símbolo $l\_i\_k$, onde $i$ é o identificador da proposição e $k$ o índice da cópia do lugar.

Para garantir que as proposições do estado final serão sempre verdadeiras quando um plano for encontrado, ou seja, quando a fórmula for satisfatível, estas proposições devem compor uma conjunção atribuída ao operador que representa o estado final.

Do mesmo modo, para garantir que todas as proposições do estado inicial sejam verdadeiras, mesmo quando o plano encontrado não utilize todas elas como pré-condição, elas devem compor uma conjunção atribuída a um operador que representa o estado inicial. Deste modo, não é necessário representar o lugar e a transição inicial da rede de planos.

O estado inicial e o estado final, por sua vez, compõem uma conjunção com operadores de controle (os operadores de controle serão descritos a seguir), a qual é atribuída ao operador de saída da fórmula.

### 3.1.1 Codificação da estrutura básica da rede de planos

A codificação da estrutura básica do modelo da rede de planos pode ser segmentada para uma rede de planos:

- simples;
- com duas ou mais camadas e duas ou mais ações;
- com transições de manutenção.

#### Rede de planos simples

Uma rede de planos simples é uma rede construída a partir do problema de planejamento com apenas uma ação, como é apresentado na figura 3.1. A codificação em ISCAS desta rede de planos é realizada conforme representado na figura 3.2.

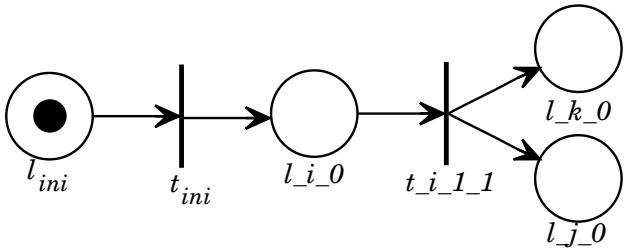



Figura 3.1: Rede de planos simples

1 - $not\_t\_i\_1\_1 = NOT(t\_i\_1\_1)$
2 - $l\_j\_0 = AND(t\_i\_1\_1, l\_i\_0)$
3 - $l\_k\_0 = AND(t\_i\_1\_1, l\_i\_0)$
4 - $Ctr\_a\_i\_1\_1 = OR(not\_t\_i\_1\_1, l\_i\_0)$
5 - $out = AND(initState, goalState, Ctr\_a\_i\_1\_1)$

Figura 3.2: Codificação rede de planos da figura 3.1 em ISCAS

A codificação da rede da figura 3.1 é baseada na criação de um operador para cada um dos efeitos desta transição. Cada um destes operadores recebe a conjunção entre variável que representa a transição com as suas pré-condições, conforme apresentado na figura 3.2.

Para que uma transição não receba o valor verdade sem que suas pré-condições sejam verdadeiras é necessário criar um operador de controle $Ctr\_a\_i\_1\_1$, o qual recebe a disjunção entre a negação desta transição com a sua pré-condição. Este operador é adicionado na conjunção de saída da fórmula.

No SATPLAN, descrito na seção 2.3.1, uma ação implica na conjunção dos seus efeitos. Esta ação também implica na conjunção de suas pré-condições (eliminação de modelo anômalo). Estas duas cláusulas garantem que os efeitos e pré-condições sejam verdadeiros.

Em ISCAS a eliminação desse modelo anômalo é realizada da mesma forma que o SATPLAN. Para isso tem-se a necessidade de criação de um operador de controle.

## Rede de planos com duas ou mais camadas e duas ou mais ações

A idéia da rede de planos simples pode ser incrementada com a inclusão de novas camadas e novas ações. Por exemplo, o processo de expansão da rede de planos da figura

3.1 pode resultar na rede da figura 3.3, cuja codificação em ISCAS é apresentada nas fórmula das figuras 3.2 e 3.4.

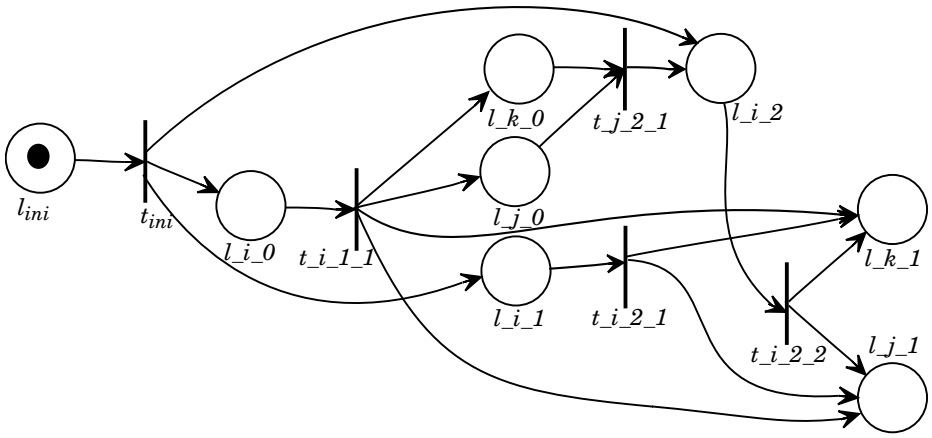



Figura 3.3: Rede de planos com duas ou mais camadas

5 - $not\_t\_i\_2\_1 = NOT(t\_i\_2\_1)$
6 - $a\_i\_2\_1 = AND(t\_i\_2\_1, l\_i\_1)$
7 - $Ctr\_a\_i\_2\_1 = OR(not\_t\_i\_2\_1, l\_i\_1)$

8 - $not\_t\_j\_2\_1 = NOT(t\_j\_2\_1)$
9 - $pre\_t\_j\_2\_1 = AND(l\_j\_0, l\_k\_0)$
10 - $a\_j\_2\_1 = AND(t\_j\_2\_1, pre\_t\_j\_2\_1)$
11 - $Ctr\_a\_j\_2\_1 = OR(not\_t\_j\_2\_1, pre\_t\_j\_2\_1)$
12 - $l\_i\_2 = OR(a\_j\_2\_1, l\_i\_1)$

13 - $not\_t\_i\_2\_2 = NOT(t\_i\_2\_2)$
14 - $a\_i\_2\_2 = AND(t\_i\_2\_2, l\_i\_2)$
15 - $Ctr\_a\_i\_2\_2 = OR(not\_t\_i\_2\_2, l\_i\_2)$

16 - $l\_j\_1 = OR(l\_j\_0, a\_i\_2\_1, a\_i\_2\_2)$
17 - $l\_k\_1 = OR(l\_k\_0, a\_i\_2\_1, a\_i\_2\_2)$
18 - $Ctr\_2 = AND(Ctr\_a\_i\_2\_1, Ctr\_a\_i\_2\_2, Ctr\_a\_j\_2\_1)$
19 - $Ctr = AND(Ctr\_a\_i\_1\_1, Ctr\_2)$
20 - $out = AND(initState, goalState, Ctr)$

Figura 3.4: Codificação de uma rede de planos com duas ou mais camadas

Na expansão da rede o operador de saída da fórmula é alterado, fazendo com que a linha 5 da figura 3.2 seja removida, sendo incluídos os operadores da figura 3.4 em seu lugar. Um destes operadores é o operador de controle das camadas da rede.

Quando uma camada é composta por mais de uma transição cria-se um operador de controle para esta camada (linha 18 da figura 3.4). Numa rede de planos com duas ou mais camadas, os operadores de controle de cada camada compõem uma conjunção (linha 19 da figura 3.4), a qual é um argumento do operador de saída da fórmula. Esta codificação dos operadores de controle evita modificações entre a sequência de fórmulas geradas a partir da rede.

O objetivo dos operadores de controle é evitar que as variáveis que representam as transições recebam o valor verdade, se suas pré-condições não forem verdadeiras. Quando uma transição possui mais de uma pré-condição, uma conjunção destas pré-condições precisa ser criada (linha 9 da figura 3.4). Esta conjunção é um argumento do operador de controle da transição (linha 11 da figura 3.4), assim como da conjunção que determina a validade desta transição (linha 10 da figura 3.4).

Todos os efeitos de uma única transição são resultado da conjunção de validade desta transição, mas quando um lugar é efeito de mais de uma transição, uma disjunção precisa ser criada (linhas 12, 16 e 17 da figura 3.4). Nesta disjunção são incluídos os operadores de validade das transições contidas no passo da camada que o lugar pertence. Se este lugar pertence ao segundo passo de uma determinada camada e no primeiro passo desta camada não existe nenhum lugar que represente esta proposição, então os operadores de validade das transições do primeiro passo também são incluídas na disjunção (linhas 16 e 17 da figura 3.4), caso contrário, para capturar os efeitos das transições do primeiro passo da camada, o operador que representa a proposição cujo $k$ é o maior incluído na rede deve ser incluído na disjunção.

A mesma idéia é utilizada para capturar a validade das transições das camadas anteriores. A disjunção que determina a validade de um efeito é composta por operadores de validade das transições contidas no mesmo passo/camada deste efeito, e pelo operador de validade da última cópia do lugar que representa a proposição no passo/camada anterior.

Para facilitar o mecanismo utilizado para codificar os efeitos de mais de uma transição, na fórmula em ISCAS uma proposição passa a ser representada por no máximo dois operadores numa camada, ou seja, um operador para cada passo. Isso porque as cópias

dos lugares criadas para evitar conflito são necessárias apenas nas redes de Petri.

Criar apenas um operador para cada proposição contida num determinado passo auxilia inclusive na representação das transições de manutenção.

## Rede de planos com transições de manutenção

Com a inclusão das transições de manutenção, a rede de planos da figura 3.3 é representada pela rede de planos da figura 3.5. Por exemplo, estas transições de manutenção foram criadas pois a ação $i$ remove a proposição $i$ e a ação $j$ remove a proposição $k$. A proposição $j$ não é removida por nenhuma ação inserida na rede.

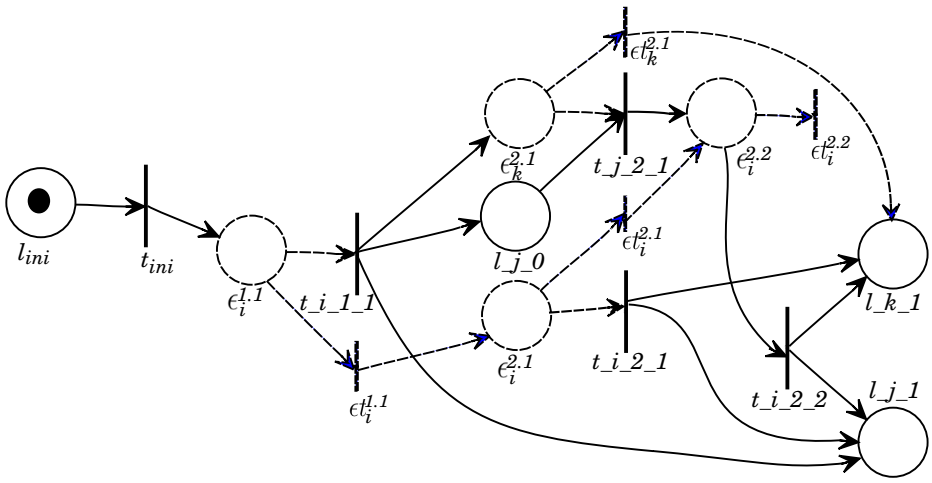



Figura 3.5: Rede de planos com transições de manutenção

As transições de manutenção na rede da figura 3.5 são codificadas para o formato ISCAS por meio de operadores de manutenção representados em negrito na figura 3.6. A estrutura original da fórmula definida na figura 3.4 é mantida com inclusão dos operadores de manutenção.

A transição de manutenção é representada em ISCAS por apenas uma conjunção, quando um lugar é somente efeito desta transição. Os argumentos desta conjunção são as negações das transições que removem a proposição mantida pela transição de manutenção e o operador de manutenção anterior (linha 5 da figura 3.6). Deste modo, o operador que representa o lugar efeito só será verdadeiro se o operador anterior for verdadeiro e nenhuma

1 - $not\_t\_i\_1\_1 = NOT(t\_i\_1\_1)$
2 - $l\_j\_0 = AND(t\_i\_1\_1, l\_i\_0)$
3 - $l\_k\_0 = AND(t\_i\_1\_1, l\_i\_0)$
4 - $Ctr\_a\_i\_1\_1 = OR(not\_t\_i\_1\_1, l\_i\_0)$

**5 - l_i_1 = AND(l_i_0, not_t_i_1_1)**
6 - $not\_t\_i\_2\_1 = NOT(t\_i\_2\_1)$
7 - $a\_i\_2\_1 = AND(t\_i\_2\_1, l\_i\_1)$
8 - $Ctr\_a\_i\_2\_1 = OR(not\_t\_i\_2\_1, l\_i\_1)$

9 - $not\_t\_j\_2\_1 = NOT(t\_j\_2\_1)$
10 - $pre\_t\_j\_2\_1 = AND(l\_j\_0, l\_k\_0)$
11 - $a\_j\_2\_1 = AND(t\_j\_2\_1, pre\_t\_j\_2\_1)$
12 - $Ctr\_a\_j\_2\_1 = OR(not\_t\_j\_2\_1, pre\_t\_j\_2\_1)$

**13 - m_l_i_2 = AND(l_i_1, not_t_i_2_1)**
**14 - l_i_2 = OR(a_j_2_1, m_l_i_1)**
15 - $not\_t\_i\_2\_2 = NOT(t\_i\_2\_2)$
16 - $a\_i\_2\_2 = AND(t\_i\_2\_2, l\_i\_2)$
17 - $Ctr\_a\_i\_2\_2 = OR(not\_t\_i\_2\_2, l\_i\_2)$

18 - $l\_j\_1 = OR(l\_j\_0, a\_i\_2\_1, a\_i\_2\_2)$
**19 - m_l_k_1 = AND(l_k_0, not_t_j_2_1)**
**20 - l_k_1 = OR(m_l_k_1, a_i_2_1, a_i_2_2)**

Figura 3.6: Codificação de uma rede de planos com transições de manutenção

transição da camada e passo correspondente, cuja ação nega a proposição mantida, for verdadeira.

Quando um lugar é efeito da transição de manutenção e de outras transições numa mesma camada, além da conjunção uma disjunção deve ser criada. A disjunção garante que o efeito é resultado do operador de manutenção ou da validade de ações na camada atual (linhas 13, 14, 19 e 20 da figura 3.6).

### 3.1.2 Codificação da estrutura de controle em ISCAS

A estrutura de controle é codificada em ISCAS a partir da relação de ordenação entre duas ações diferentes. O problema de planejamento com apenas uma ação não possui estrutura de controle, pois não possui relação de ordenação.

Na tabela 3.1 são apresentados os subplanos válidos e inválidos, entre duas ações $a$ e $b$, para cada relação de ordenação.

| Rel. de ord. | subplanos válidos | subplanos inválidos |
|---|---|---|
| $a$ e $b$ são concorrentes ($a \shortparallel b$) | $(a\|b), (a), (b), (a;b), (b;a), \lambda$ | |
| $a$ e $b$ são não-concorrentes ($a \nparallel b$) | $(a), (b), (a;b), (b;a), \lambda$ | $(a\|b)$ |
| $a$ precede $b$ ($a \lhd b$) | $(a), (b), (a;b)\lambda$ | $(a\|b), (b;a)$ |
| $a$ e $b$ são *mutex* ($a \diamondsuit b$) | $(a), (b), \lambda$ | $(a\|b), (a;b), (b;a)$ |

Tabela 3.1: Subplanos válidos e inválidos para cada relação de ordenação

Na tabela 3.1, $\lambda$ representa um subplano vazio, subplano em que nenhuma ação é executada. O símbolo $(x)$ indica que o subplano é composto da ação $x$, $(x;y)$ representa que $x$ precede $y$ no subplano e $(x|y)$ indica que $x$ e $y$ são concorrentes no subplano [24].

Na rede planos, as estruturas de controle permitem o disparo das transições que representam os subplanos válidos de cada relação de ordenação presente na rede. Alguns destes subplanos já são representados pela estrutura básica da rede de planos. A justificativa para tal representação destes subplanos é inserir na rede, mais informação referente ao problema.

Na definição da rede de planos, a competição de necessidade é excluída dos cálculos de inconsistências da rede, assim um conjunto de ações não paralelas é identificado como paralelas. A inclusão destas informações na rede não interfere no resultado final, mas acrescenta estruturas desnecessárias e não coerentes com a solução do problema.

Por outro lado, com a representação dos subplanos inválidos apenas informações coerentes com a solução do problema serão introduzidas na rede.

Outra vantagem em representar subplanos inválidos na estrutura de controle é a possibilidade de duas ou mais transições serem disparadas num mesmo passo da camada. A representação original da rede de planos possibilita que apenas duas transições sejam disparadas em cada passo da camada.

A representação dos subplanos inválidos entre duas ações $a$ e $b$ foi codificada em ISCAS conforme descrito na tabela 3.2. Nesta tabela o prefixo *not_* nos operandos da subfórmula representa a negação da ação na camada e passo correspondentes.

| Rel. de ord. | Fórmula não clausal |
|---|---|
| $a$ e $b$ são *mutex* ($a \diamondsuit b$) | $C_1 = OR(not\_t\_a\_c\_1, not\_t\_b\_c\_1)$<br>$C_2 = OR(not\_t\_a\_c\_1, not\_t\_a\_c\_2)$<br>$C_3 = OR(not\_t\_a\_c\_1, not\_t\_b\_c\_2)$<br>$C_4 = OR(not\_t\_b\_c\_1, not\_t\_a\_c\_2)$<br>$C_5 = OR(not\_t\_b\_c\_1, not\_t\_b\_c\_2)$<br>$C_6 = OR(not\_t\_a\_c\_2, not\_t\_b\_c\_2)$<br>$C_\diamondsuit = AND(C_1, C_2, C_3, C_4, C_5, C_6)$ |
| $a$ precede $b$ ($a \triangleleft b$) | $C_1 = OR(not\_t\_a\_c\_1, not\_t\_b\_c\_1)$<br>$C_2 = OR(not\_t\_a\_c\_1, not\_t\_a\_c\_2)$<br>$C_4 = OR(not\_t\_b\_c\_1, not\_t\_a\_c\_2)$<br>$C_5 = OR(not\_t\_b\_c\_1, not\_t\_b\_c\_2)$<br>$C_6 = OR(not\_t\_a\_c\_2, not\_t\_b\_c\_2)$<br>$C_\triangleleft = AND(C_1, C_2, C_4, C_5, C_6)$ |
| $a$ e $b$ são não-concorrentes ($a \nparallel b$) | $C_1 = OR(not\_t\_a\_c\_1, not\_t\_b\_c\_1)$<br>$C_2 = OR(not\_t\_a\_c\_1, not\_t\_a\_c\_2)$<br>$C_5 = OR(not\_t\_b\_c\_1, not\_t\_b\_c\_2)$<br>$C_6 = OR(not\_t\_a\_c\_2, not\_t\_b\_c\_2)$<br>$C_\nparallel = AND(C_1, C_2, C_5, C_6)$ |
| $a$ e $b$ são concorrentes ($a \shortparallel b$) | |

Tabela 3.2: Codificação das relações de ordenação no formato ISCAS

A representação das relações de ordenação da tabela 3.2 tem por base as cláusulas de conflito da codificação baseada no grafo de planos do SATPLAN, descritas na seção 2.3.1. Como na rede de planos são representadas relações de ordenação, as cláusulas de conflito foram expandidas para ações pertencentes a dois passos. Assim na relação de exclusão mútua é representada por seis disjunções e uma conjunção. Duas disjunções para o conflito em que as ações pertencem ao mesmo passo (operadores $C_1$ e $C_2$), duas para os conflitos de precedência (operadores $C_3$ e $C_4$) e duas para evitar o disparo seguido da mesma ação (operadores $C_5$ e $C_6$).

Para descrever a relação de precedência basta representar os mesmos operadores do *mutex*, com exceção da disjunção que evita a precedência de $a$ com $b$ (operador $C_3$). Do mesmo modo, para representar a não concorrência entre duas ações, os mesmos operadores do *mutex* são utilizados, exceto os de precedência (operadores $C_3$ e $C_4$).

A composição de um conjunto de relações de ordenação é representada na rede de planos pela composição das estruturas de controle descritas na seção 2.2.3 deste docu-

mento. A composição das subfórmulas de controle em ISCAS é realizada simplesmente pela conjunção dos operadores de controle de cada relação de ordenação.

Por exemplo, a partir das relações de ordenação $(a \diamondsuit b)$, $(a \nvdash c)$ e $(b \shortparallel c)$ é gerada a subfórmula de controle apresentada na figura 3.7.

$$
\begin{array}{l}
C_{a\diamondsuit b\_1} = OR(not\_t\_a\_c\_1, not\_t\_b\_c\_1) \\
C_{a\diamondsuit b\_2} = OR(not\_t\_a\_c\_1, not\_t\_a\_c\_2) \\
C_{a\diamondsuit b\_3} = OR(not\_t\_a\_c\_1, not\_t\_b\_c\_2) \\
C_{a\diamondsuit b\_4} = OR(not\_t\_b\_c\_1, not\_t\_a\_c\_2) \\
C_{a\diamondsuit b\_5} = OR(not\_t\_b\_c\_1, not\_t\_b\_c\_2) \\
C_{a\diamondsuit b\_6} = OR(not\_t\_a\_c\_2, not\_t\_b\_c\_2) \\
C_{a\diamondsuit b} = AND(C_{a\diamondsuit b\_1}, C_{a\diamondsuit b\_2}, C_{a\diamondsuit b\_5}, C_{a\diamondsuit b\_6}) \\
C_{a\nvdash c\_1} = OR(not\_t\_a\_c\_1, not\_t\_b\_c\_1) \\
C_{a\nvdash c\_2} = OR(not\_t\_a\_c\_1, not\_t\_a\_c\_2) \\
C_{a\nvdash c\_5} = OR(not\_t\_b\_c\_1, not\_t\_b\_c\_2) \\
C_{a\nvdash c\_6} = OR(not\_t\_a\_c\_2, not\_t\_b\_c\_2) \\
C_{a\nvdash c} = AND(C_{a\nvdash c\_1}, C_{a\nvdash c\_2}, C_{a\nvdash c\_5}, C_{a\nvdash c\_6}) \\
R_1 = AND(C_{a\nvdash b}, C_{a\nvdash c}) \\
out = AND(initState, goalState, Ctr, R_1)
\end{array}
$$

Figura 3.7: Codificação da composição de relações de ordenação

Para o conjunto de relações de ordenação $(a \diamondsuit b)$, $(a \nvdash c)$ e $(b \shortparallel c)$ foi necessário codificar apenas o *mutex* e a relação não concorrente. Cada uma destas relações resulta num conjunto de operadores de controle que devem ser satisfeitos para ser possível encontrar planos válidos. Por isso o resultado da subfórmula de controle é um argumento do operador de saída da fórmula ISCAS.

Supondo que não haja relações de ordenação concorrente na fórmula, o número de operadores de controle para um conjunto de ações $A$, seria igual $c\binom{A}{2}$, onde $c$ é uma constante. O número de operadores de controle é definido por uma combinação, porque o cálculo das relações de ordenação é realizado com todas as ações, combinadas duas a duas. Assintoticamente, no pior caso, o número de operadores de controle é igual $|A|^2$.

A representação da rede de planos em fórmulas proposicionais clausais é crucial para análise de eficiência codificação da rede de planos, em comparação com o SATPLAN. Tendo em vista que as relações de ordenação são representadas em ISCAS por conjunções

de disjunções, apenas a estrutura básica da rede de planos ainda precisa ser traduzida para CNF.

## 3.2 Codificação da rede de planos em fórmulas clausais

O processo de tradução da rede de planos para fórmulas clausais é semelhante ao SATPLAN da competição de planejadores de 2006 (ICAPS 2006) [1]. A principal diferença é o conceito de camada da rede de planos. A definição da codificação da rede de planos em CNF é descrita como segue:

1. os literais do estado inicial representam as proposições da camada 0 e os literais do estado final representam as proposições do estado final contidas na última camada. Os literais do estado inicial e final são clausulas unitárias na fórmula em CNF;

2. todo lugar $l_i^k$ pertencente a um passo de uma determinada camada implica na disjunção das transições deste passo da camada, que tem $l_i^k$ como efeito. Para manter o efeito de transições anteriores, se $k > 0$ a disjunção das transições também contém $l_i^{k-1}$;

3. as transições implicam na conjunção dos lugares que são suas pré-condições;

4. para todas as proposições removidas por ações presentes na rede e para todas estas ações, deve ser criada a cláusula $(\neg l_i^k \vee \neg t)$, onde $l_i^k$ é um lugar da rede que representa uma proposição $i$, removida por uma ação, cuja transição $t$ pertence ao mesmo passo ou camada de $l_i^k$;

5. ações mutuamente exclusivas são representadas por um conjunto de disjunções.

Para facilitar a manutenção dos efeitos de transições anteriores e a inclusão das cláusulas do item 2, uma proposição é representada por no máximo dois operadores numa camada, sendo deste modo um operador para cada passo.

A manutenção da validade de uma proposição por mais de uma camada é realizada por meio do conjunto de variáveis que representam esta proposição na fórmula. Por

isso a variável que representa a proposição na camada ou passo anterior faz parte da disjunção das transições no item 2. Representação semelhante foi descrita na seção 2.3.1 e exemplificada na equação 2.9.

Contudo a manutenção da validade de uma proposição só é garantida pelo *mutex* de manutenção definido no item 4. Na codificação baseada no grafo de planos (seção 2.3.1) a manutenção da validade de uma proposição entre as camadas é realizada por meio das ações de manutenção e dos *mutex* de competição de necessidade e de proposições. Como na rede de planos a manutenção é feita de modo diferente, teve-se a necessidade de criar a cláusula de manutenção.

Na tabela 3.3 é feito um comparativo entre as diferenças da codificação em CNF da rede de planos e o SATPLAN. Muitas destas diferenças são resultados da estrutura utilizada como base para a codificação.

| **Codificação baseada na rede de planos** | **SATPLAN** |
|---|---|
| Estrutura base: rede de planos | Estrutura base: grafo de planos |
| Calcula e codifica relações de ordenação | Calcula *mutex* de interferência, competição de necessidade e proposição, mas só codifica *mutex* de interferência e de proposição |
| Camada relaxada | Camada não relaxada |
| Codificação de cláusulas de manutenção | Codificação de ações de manutenção e *mutex* de proposição |

Tabela 3.3: Comparação entre o SATPLAN e codificação baseada na rede de planos

As relações de ordenação são representadas por cláusulas binárias. As mesmas disjunções utilizadas na codificação da rede de planos no formato ISCAS são utilizadas no formato CNF.

Para duas ações mutuamente exclusivas $a$ e $b$, de uma camada $i$ composta por dois passos são criadas as seguintes clausulas: $(\neg a_i^1 \vee \neg b_i^1)$, $(\neg a_i^1 \vee \neg a_i^2)$, $(\neg a_i^1 \vee \neg b_i^2)$, $(\neg b_i^1 \vee \neg a_i^2)$, $(\neg b_i^1 \vee \neg b_i^2)$, $(\neg a_i^2 \vee \neg b_i^2)$. Se a ação $a$ precede a ação $b$, são criadas as mesmas cláusulas do *mutex*, com exceção de $(\neg a_i^1 \vee \neg b_i^2)$. Se $a$ é não-concorrente com $b$, são criadas as mesmas cláusulas do *mutex*, exceto $(\neg a_i^1 \vee \neg b_i^2)$ e $(\neg b_i^1 \vee \neg a_i^2)$. Para a relação de ordenação de concorrência não são criadas cláusulas adicionais.

As diferenças entre a codificação da rede de planos e o SATPLAN resultam em fórmulas com um diferente número de variáveis. Na equação 3.1 é descrita a comparação entre o número de variáveis $R_i$ geradas até uma camada $i$ com o número de variáveis do SATPLAN $S_i$, que contém $A_m$ variáveis criadas a partir das ações de manutenção.

$$R_i \leq 2(S_i - A_m) \tag{3.1}$$

Como na codificação da rede de planos as ações de manutenção não geram variáveis, na equação 3.1 é necessário subtrair as variáveis geradas a partir destas ações do total de variáveis do SATPLAN. O resultado desta subtração é multiplicado por dois, pois na rede de planos uma camada é representada em dois passos de ações. O número de variáveis $R_i$ é menor ou igual que o resultado da equação, porque pode existir um conjunto de ações que não estão contidas no segundo passo de algumas camadas.

# CAPÍTULO 4

# RESULTADOS EXPERIMENTAIS

Neste capítulo são apresentados alguns resultados obtidos com o método de planejamento baseado em satisfatibilidade tanto clausal, quanto não-clausal da rede de planos. A comparação com planejadores que seguem a mesma linha metodológica é fundamental.

Montaño apresentou resultados promissores com os estudos de redes de Petri para planejamento convertidas para fórmulas não-clausais (ISCAS)[19].

Neste trabalho foi elaborado um método de conversão da rede de planos para o formato ISCAS, com o qual foram produzidos resultados experimentais que puderam ser comparados com os resultados de Montaño.

Também foram realizados experimentos com a rede de planos mapeada para CNF, cujos resultados foram comparados com o planejador SATPLAN.

Todos os experimentos deste trabalho foram realizados em um computador com um processador AMD de 2.8 GHz, com 8 núcleos, 10 GB de memória RAM e sistema operacional Debian GNU/Linux.

Nos experimentos foi observado o tempo de execução dos planejadores, sendo que o *timeout* é de 1800 segundos.

Os planos foram validados com *Plan Validator*[3], o qual tem como entrada o domínio, o problema e o plano. A saída apresenta informações sobre o plano válido, como o número de ações do plano. Se o plano for inconsistente o *Plan Validator* indica erro e permite verificar quais ações são conflitantes.

Todos os experimentos realizados neste trabalho, entre outros testes empíricos, apresentaram planos válidos, garantindo a validade do planejador aqui desenvolvido.

## 4.1 Experimentos realizados com a codificação da rede planos em ISCAS

Nos experimentos com a codificação da rede de planos em ISCAS buscou-se obter resultados comparativos com a codificação do PETRIPLAN em ISCAS, por isso nestes experimentos foram utilizados os problemas de planejamento da pesquisa de Montaño. O objetivo em se utilizar este conjunto de problemas é identificar o ganho de desempenho com a utilização da rede de planos em relação ao PETRIPLAN.

Montaño realizou experimentos comparativos entre o PETRIPLAN codificado em ISCAS, a codificação original do PETRIPLAN em FNNF e o SATPLAN. A codificação em ISCAS teve melhora significativa com relação ao FNNF, mas não superou o SATPLAN [19].

Com o objetivo de comparar a rede de planos e o PETRIPLAN, nos experimentos realizados neste trabalho o resolvedor NFLSAT foi utilizado tanto para o PETRIPLAN, quanto para a Rede de Planos. A ideia é identificar a qualidade da estrutura de codificação das fórmulas ISCAS geradas. Esta qualidade é representada pelo tempo de execução gasto pelos planejadores para resolver os problemas de planejamento.

Os resultados experimentais dos domínios *blocks*, *Gripper* e *Logistics* são apresentados na tabela 4.1, onde é comprovada a eficiência da rede de planos. Em todos os experimentos o ganho no tempo de execução da rede de planos em comparação com o PETRIPLAN é maior que 50%.

Para os domínios dos experimentos, as fórmulas geradas com base na rede de planos são mais fáceis de serem resolvidas do que as fórmulas geradas a partir do PETRIPLAN. A simplificação destas fórmulas é resultado da estrutura de dados utilizada e da estrutura de codificação das fórmulas geradas a partir do problema de planejamento.

Por exemplo, com a nova codificação de fómulas ISCAS a partir da rede de planos foi reduzido em mais de 75% o número de variáveis da fórmula. No PETRIPLAN os *mutex* são codificados como variáveis. O número de *mutex* é maior do que o número de ações.

| | PETRIPLAN | | | Rede de Planos | | | SATPLAN |
|---|---|---|---|---|---|---|---|
| Problemas | Var. | Plano | Tempo (s) | Var. | Plano | Tempo (s) | tempo |
| *Blocks 1* | 479 | 3 | 0,0 | 78 | 3 | 0,01 | 0,02 |
| *Blocks 2* | 1094 | 7 | 0,2 | 251 | 7 | 0,07 | 0,04 |
| *Blocks 3* | 2452 | 9 | 0,9 | 440 | 9 | 0,1 | 0,09 |
| *Blocks 4* | 32698 | 27 | 73,7 | 2996 | 30 | 15,4 | 0,1 |
| *Gripper 1* | 111 | 5 | 0,03 | 40 | 5 | 0,01 | 0,001 |
| *Gripper 2* | 1511 | 1 | 0,3 | 296 | 11 | 0,1 | 0,1 |
| *Gripper 3* | 5423 | 2 | 6,3 | 740 | 17 | 1,5 | 0,8 |
| *Gripper 4* | 12967 | 22 | 188,8 | 1376 | 23 | 84 | 4,918 |
| *Logistics 1* | 12959 | 41 | 33,2 | 2239 | 48 | 14,9 | 0,3 |
| *Logistics 2* | 14894 | 85 | 47,6 | 3378 | 68 | 15 | 2,8 |

Tabela 4.1: Comparação entre a rede de planos e o PETRIPLAN codificados em ISCAS

Na rede de planos apenas as transições que representam ações são codificadas como variáveis. A ideia é representar o problema de planejamento como um circuito, cujas variáveis de entrada são ações e as proposições do estado inicial. As operações lógicas deste circuito determinam as restrições para obtenção da solução do problema de planejamento.

As restrições (operadores) provenientes das relações de ordenação são operadores informativos. O objetivo destes operadores é facilitar a obtenção de uma valoração satisfatível. A inclusão destes operadores e o aproveitamento da dinâmica da rede de Petri para não representação dos *mutex* de competição de necessidade foram as principais contribuições da rede de planos e proporcionaram os resultados obtidos neste trabalho.

Na tabela 4.1 também é apresentado o comprimento dos planos gerados pelos planejadores. A rede de planos gera planos mais longos, porque o conceito de camada na rede de planos é relaxado para representar as relações de ordenação. Com no máximo duas variáveis para cada ação em cada camada, aumenta-se o número de planos válidos, dos quais o resolvedor SAT pode encontrar o mais longo.

Embora a utilização da rede de planos tenha aperfeiçoado o planejador baseado em redes de Petri, a codificação da rede de planos em ISCAS não conseguiu superar o SATPLAN. Em todos os experimentos desta seção a rede de planos teve um tempo de execução

superior ao SATPLAN.

Nos experimentos com a codificação da rede de planos em CNF os resultados foram mais promissores. Em alguns domínios a utilização da estrutura da rede de planos e a sua codificação em CNF possibilitaram uma redução no tempo de execução do planejador, de tal modo que se obtivesse tempos de execução melhores que o SATPLAN.

## 4.2 Experimentos realizados com a codificação da rede de planos em CNF

Foram realizados experimentos do mapeamento da rede de planos para fórmulas em CNF com um conjunto de 11 domínios da competição de planejadores obtidos no site da *ICAPS Competitions*[1].

O resolvedor SAT utilizado tanto para as fórmulas da rede de planos, quanto para as fórmulas do SATPLAN foi o resolvedor *lingeling* o qual foi um dos três primeiros ganhadores da competição internacional de resolvedores SAT de 2011 [2].

Os resultados obtidos com a codificação de fórmulas defendida neste trabalho foram comparados com a codificação utilizada pelo SATPLAN na competição de 2006 (*thin-bp-based*)[1]. Os domínios onde a rede de planos foi mais eficiente do que o SATPLAN são apresentados na tabela 4.2. Os domínios onde o SATPLAN foi mais eficiente são apresentados na tabela 4.3.

A rede de planos é uma estrutura que proporciona uma promissora codificação em CNF, pois em quatro de onze domínios obteve-se um tempo de execução inferior ao tempo de execução do SATPLAN. Nestes domínios, as cláusulas geradas a partir das relações de ordenação auxiliam mais o resolvedor SAT a encontrar uma valoração satisfatível, do que os *mutex* de proposições obtidos a partir do grafo de planos.

A complexidade de resolução de uma fórmula em CNF não depende exclusivamente da densidade, ou do número de cláusulas ou do número de variáveis. Nos experimentos

| | SATPLAN | | | | Rede de planos | | | |
|---|---|---|---|---|---|---|---|---|
| Problemas | Var. | Cls. | Plano | Tempo | Var. | Cls. | Plano | Tempo |
| *Logistics 1* | 13312 | 106658 | 115 | 6,4 | 7653 | 86507 | 116 | 4,4 |
| *Logistics 2* | 24539 | 221131 | 151 | 110.4 | 13303 | 172509 | 153 | 29.5 |
| *Logistics 3* | – | – | – | timeout | 24834 | 461445 | 224 | 950.6 |
| *TPP 1* | 25606 | 674472 | 97 | 30,3 | 33765 | 2846513 | 114 | 17,9 |
| *TPP 2* | 35189 | 998982 | 137 | 39,5 | 55038 | 5234978 | 155 | 34,3 |
| *TPP 3* | 43884 | 2665474 | 145 | 1608,1 | 68306 | 10445701 | 159 | 101,3 |
| *Storage 1* | 4550 | 252355 | 22 | 27,2 | 3984 | 581353 | 18 | 26,5 |
| *Storage 2* | 7369 | 362345 | 18 | 271,1 | 5453 | 786927 | 22 | 156,7 |
| *Storage 3* | – | – | – | timeout | 9508 | 2663990 | 29 | 321,2 |
| *Pipeworld 1* | 21647 | 4487456 | 30 | 260,2 | 16066 | 8438456 | 30 | 142,3 |
| *Pipeworld 2* | 26158 | 6635534 | 30 | 335,1 | 19414 | 12701246 | 30 | 93,6 |
| *Pipeworld 3* | 35025 | 7581085 | 44 | 465,3 | 26300 | 15137493 | 44 | 325,6 |

Tabela 4.2: Resultados em que a rede de planos vence o SATPLAN

apresentados nas tabelas 4.2 e 4.3, a codificação da rede de planos teve fórmulas com densidade superior às fórmulas geradas pelo SATPLAN, com exceção do problema *Storage 1*. Com exceção dos domínios *Gripper* e *Logistics* e do problemas *Storage 1* e *Driverlog 1*, o número de cláusulas geradas nas fórmulas do SATPLAN é menor que o número de cláusulas geradas nas fórmulas da rede de planos. As fórmulas da rede de planos têm menos variáveis do que as fórmulas do SATPLAN, exceto no domínio *TPP* e nos problemas *Depots 2*, *Freecell 1* e *Freecell 2*.

Em todos os domínios das tabelas 4.2 e 4.3 é apresentado apenas o número de cláusulas e variáveis da fórmula em que se encontrou uma valoração satisfatível, demonstrando que o número de cláusulas e variáveis não refletem a complexidade de obtenção da solução do problema.

A complexidade de solução das fórmulas do problema de planejamento depende das características do domínio combinada com a estrutura de condificação das fórmulas, visto que nos domínios da tabela 4.2 a estrutura de codificação da rede de planos gerou fórmulas mais fáceis de serem resolvidas pelo *lingeling* do que a codificação do SATPLAN. Nos domínios da tabela 4.3 a codificação do SATPLAN foi mais adequada.

Outra característica crucial para solução de fórmulas do problema de planejamento são as cláusulas binárias de conflito. Neste trabalho é apresentada uma ordenação parcial

| | SATPLAN | | | | Rede de planos | | | |
|---|---|---|---|---|---|---|---|---|
| Problemas | Var. | Cls. | Plano | Tempo | Var. | Cls. | Plano | Tempo |
| *Elevator 1* | 2473 | 33787 | 19 | 2,5 | 2094 | 65723 | 19 | 33,8 |
| *Elevator 2* | 4257 | 69858 | 23 | 7,2 | 3675 | 141083 | 23 | 186,5 |
| *Elevator 3* | 6689 | 128525 | 27 | 119,9 | 5864 | 265995 | 27 | 801,5 |
| *Depots 1* | 5698 | 216423 | 29 | 5,5 | 3696 | 428006 | 33 | 6,3 |
| *Depots 2* | 5698 | 216423 | 73 | 55,7 | 20564 | 7075691 | 81 | 393,6 |
| *Driverlog 1* | 18163 | 560442 | 52 | 28 | 7905 | 300311 | 70 | 750,6 |
| *Driverlog 2* | 14049 | 248447 | 40 | 26,9 | 12186 | 479272 | 48 | 1319,5 |
| *Driverlog 3* | 76564 | 3677227 | 100 | 1681,3 | – | – | – | timeout |
| *Freecell 1* | 5306 | 977707 | 18 | 19 | 5566 | 2120183 | 22 | 26,2 |
| *Freecell 2* | 4490 | 769271 | 21 | 7,9 | 8810 | 4361075 | 27 | 125,8 |
| *Freecell 3* | 17582 | 6114100 | 30 | 253 | 15919 | 11159318 | 36 | 667,0 |
| *Satelite 1* | 8605 | 158171 | 35 | 5 | 6160 | 239080 | 40 | 35,9 |
| *Satelite 2* | 14897 | 313380 | 43 | 48,7 | 10521 | 498887 | 53 | 109,0 |
| *Satelite 3* | 18881 | 479761 | 43 | 38,9 | 14399 | 824851 | 48 | 263,4 |
| *Gripper 1* | 1956 | 18609 | 23 | 4,9 | 1376 | 16695 | 23 | 17,2 |
| *Gripper 2* | 3088 | 34589 | 29 | 50,4 | 2204 | 31581 | 29 | 96,6 |
| *Gripper 3* | 4476 | 57593 | 35 | 291,5 | 3224 | 53051 | 35 | 1321,3 |
| *Blocks 1* | 4044 | 53577 | 25 | 0,2 | 2996 | 54254 | 39 | 4,2 |
| *Blocks 2* | 16014 | 328034 | 35 | 20,4 | 12775 | 396452 | 57 | 134,4 |
| *Blocks 3* | 40844 | 1126334 | 43 | 101,2 | – | – | – | timeout |

Tabela 4.3: Resultados em que a rede de planos perde do SATPLAN

para o *mutex* de inconsistência e a não utilização do *mutex* de competição de necessidade. A ordenação parcial resultou num tempo de execução melhor do que SATPLAN nos domínios da tabela 4.2.

A estrutura de codificação e o tipo de informação adicionada à fórmula são as principais causas dos resultados experimentais deste trabalho. Nos domínios de *Logistics*, *TPP*, *Storage* e *Pipeworld*, a codificação da rede de planos é mais eficiente do que a codificação baseada no grafo de planos. Nos domínios *Elevador*, *Depots*, *Driverlog*, *Freecell*, *Satelite*, *Gripper* e *Blocks*, a codificação a partir do grafo de planos é mais eficiente que a codificação proposta neste trabalho.

Outro dado relevante na tabela 4.2 é o comprimento dos planos gerados pelos planejadores. Os planos gerados pelo SATPLAN são sempre menores ou iguais ao comprimento dos planos gerados pela rede de planos.

Por meio de testes empíricos realizados com o *BLACKBOX*, Kautz e Selman consta-

taram que encontrar planos ótimos é mais difícil que encontrar planos sub-ótimos [11].

O SATPLAN tem pós-processamento para remover (alguma das) ações desnecessárias no plano. O pós-processamento é útil porque o plano obtido pode conter ações que não são realmente necessárias para obter o estado objetivo do problema de planejamento [14].

Neste trabalho não foi incluída a etapa de pós-processamento, mas mesmo assim foram obtidos planos com o mesmo número de ações que o SATPLAN nos domínios *Gripper*, *Elevador* e *Pipeworld*. Nos outros oito domínios os planos gerados pelo SATPLAN foram menores.

Também foi realizada uma análise no conjunto de informações da estruturas de controle que foram inseridas na fórmula. Foram realizados experimentos com a combinação da estrutura de controle do *mutex* com os outros tipos de estruturas de controle, conforme apresentado na tabela 4.4. A codificação que inclui apenas as informações da estrutura de controle do *mutex* é a que mais se destaca nos experimentos.

As informações geradas a partir da estruturas de controle do *mutex* da rede de planos é mais rica do que as geradas a partir do *mutex* de inconsistência no grafo de planos. No *mutex* da rede de planos, além de incluir os conflitos entre ações de uma camada, também são representados os conflitos de ordenação entre as ações na camada.

Entretanto as informações produzidas com base nas outras estruturas de controle também são importantes. Na tabela 4.4 são apresentadas várias situações onde as informações da estrutura de precedência e não-concorrente são fundamentais. Situações onde a combinação de duas estruturas de controle supera a representação da estrutura de controle completa e a representação do *mutex*. Nos problemas como, *Logistics 3*, *Storage 3*, *Deposts 2* e *Gripper 3*, pode-se dizer que inserir na fórmula todo o conjunto de informações provenientes da estrutura de controle prejudica a busca pelo plano, do mesmo modo que a ausência das informações das estruturas de controle de precedência ou não-concorrência.

Por exemplo, no problema *Gripper 3* a estrutura de controle completa e a estrutura do *mutex* apresentaram resultados semelhantes. A combinação de precedência com *mutex* foi mais eficiente. A combinação não-concorrente com *mutex* apresentou o pior resultado.

No domínio *Freecell* as informações da estrutura de controle do *mutex* apresentaram

| Problemas | Completa | *Mutex* e Não-concorrentes | *Mutex* e Precedentes | *Mutex* |
|---|---|---|---|---|
| *Logistics 1* | **4,4** | 7,36 | 6,81 | 6,3 |
| *Logistics 2* | 29,5 | 37,8 | 34 | **27,9** |
| *Logistics 3* | 950,6 | 894,1 | **675,9** | 805,3 |
| *TPP 1* | 17,9 | 25,1 | **16,9** | 20,2 |
| *TPP 2* | 34,3 | 40,8 | 31,7 | **30,5** |
| *TPP 3* | **101,4** | 141,2 | 110,2 | 108,5 |
| *Storage 1* | 26,5 | **12,53** | 18,8 | 17,7 |
| *Storage 2* | 156,7 | 144,5 | **118,2** | 136,7 |
| *Storage 3* | 321,2 | **295,2** | 302,1 | 352,9 |
| *Pipeworld 1* | 142,3 | **70,8** | 125,1 | 75,7 |
| *Pipeworld 2* | 93,6 | 161,5 | 204,7 | **84,0** |
| *Pipeworld 3* | 325,7 | 311,7 | 307,4 | **186,9** |
| *Elevador 1* | 33,8 | 31,7 | **23,6** | 33,2 |
| *Elevador 2* | 186,5 | 168,3 | **153,4** | 155,3 |
| *Elevador 3* | 801,5 | 887,7 | 664,8 | **661,5** |
| *Depots 1* | 6,3 | 5,90 | 4,5 | **4,3** |
| *Depots 2* | 393,6 | 343,6 | **284,9** | 470,2 |
| *Driverlog 1* | 750,6 | 524,3 | 480,2 | **466,2** |
| *Driverlog 2* | 1319,5 | timeout | 1663,6 | **1112,8** |
| *Driverlog 3* | timeout | timeout | timeout | timeout |
| *Freecell 1* | 26,2 | 25,5 | 27,6 | **20,7** |
| *Freecell 2* | 125,8 | 106,7 | 110,3 | **96,1** |
| *Freecell 3* | 667,0 | **180,7** | 824,7 | 198,1 |
| *Satelite 1* | 35,9 | 32,7 | 32,0 | **27,4** |
| *Satelite 2* | **109,0** | 193,3 | 180,4 | 154,8 |
| *Satelite 3* | 263,4 | 281,4 | 222,2 | **190,1** |
| *Gripper 1* | 17,1 | 12 | **10,9** | 24,5 |
| *Gripper 2* | 96,5 | **67,1** | 79,4 | 171,8 |
| *Gripper 3* | 1321,3 | timeout | **609,2** | 1075 |
| *Blocks 1* | 4,2 | 4,6 | **1,8** | 2,176 |
| *Blocks 2* | 134,4 | 113,3 | 106,1 | **91,816** |
| *Blocks 3* | timeout | 1725,4 | timeout | **1455,166** |

Tabela 4.4: Análise das informações da estrutura de controle

os melhores resultados. No problema *Freecell 3*, a estrutura do *mutex* e a combinação da estrutura não-concorrente com *mutex* tiveram tempo de execução inferior ao SATPLAN.

Para alguns domínios, inserir algumas informações provenientes da estrutura de controle auxilia na busca pelo plano. Estas informações ajudam na busca por uma valoração satisfatível. Contudo a busca por uma valoração satisfatível pode ser dificultada com a inclusão de muitas informações.

O conjunto de informações proveniente das estruturas de controle que auxiliam a busca pelo plano depende do domínio e do problema de planejamento. Em alguns casos, muitas informações podem prejudicar a busca por uma valoração satisfatível, em outros casos a ausência de informações prejudica esta busca.

# CAPÍTULO 5

# CONCLUSÃO

Neste trabalho foi apresentado um planejador eficiente baseado em redes de Petri. A formalização do problema de planejamento em redes de Petri com base nas definições rede de planos permitiu realizar simplificações significativas e implementar ordenações parciais entre ações. Estas contribuições da rede de planos codificadas em fórmulas proposicionais clausais e não clausais auxiliam o resolvedor SAT na busca por uma valoração satisfatível, a qual representa a solução de um problema de planejamento. A eficiência do planejamento por satisfatibilidade clausal e não clausal baseada na rede de planos foi comprovada, em alguns domínios, por meio de experimentos comparativos com os planejadores PETRIPLAN e SATPLAN.

Nos estudos realizados em torno dos métodos de solução do problema de planejamento, descritos neste documento, foi dado maior enfoque a rede de planos, porque o objetivo deste trabalho foi o de obter um planejador eficiente em redes de Petri. A rede de planos é um modelo que utiliza o poder de representação das redes de Petri para simplificar as estruturas de solução do problema de planejamento. Na definição deste modelo foi proposto um novo método de construção da rede e um método para o tratamento de conflitos que aperfeiçoa o *mutex* de interferência do grafo de planos. Neste documento foram apresentadas melhorias no modelo original da rede, de modo a solucionar problemas construção da rede de planos.

Neste trabalho também foram definidos dois métodos de codificação da rede de planos em fórmulas lógicas. Um deles no formato não-clausal em ISCAS, outro em CNF.

A codificação da rede de planos em ISCAS foi 50% mais rápida que o PETRIPLAN, nos domínios *Blocks*, *Gripper* e *Logistics*. A ideia de representar o problema de planejamento baseado na rede de planos como um circuito cujas variáveis de entrada são ações

e as proposições do estado inicial aperfeiçoou a codificação das fórmulas ISCAS. As restrições (operadores) provenientes das relações de ordenação facilitaram a obtenção de uma valoração satisfatível. A inclusão destes operadores e o aproveitamento da dinâmica da rede de Petri para a não representação dos *mutex* de competição de necessidade foram as principais contribuições da rede de planos e proporcionaram os resultados obtidos neste trabalho.

Embora a utilização da rede de planos tenha aperfeiçoado o planejador baseado em redes de Petri, a codificação da rede de planos em ISCAS não conseguiu superar o SATPLAN. Entretanto, a codificação da rede de planos em CNF obteve tempos de execução melhores do que o SATPLAN nos domínios *Logistics*, *TPP*, *Storage* e *Pipeworld*. Nos domínios *Elevador*, *Depots*, *Driverlog*, *Freecell*, *Satelite*, *Gripper* e *Blocks*, o SATPLAN apresentou melhores tempos de execução.

A rede de planos é uma estrutura que proporcionou a definição de uma promissora codificação em CNF, pois em quatro de onze domínios obteve-se um tempo de execução inferior ao tempo de execução do SATPLAN. Nestes quatro domínios, as cláusulas geradas a partir das relações de ordenação auxiliam mais o resolvedor SAT a encontrar uma valoração satisfatível, do que os *mutex* de proposições obtidos a partir do grafo de planos.

A complexidade de solução das fórmulas do problema de planejamento depende das características do domínio combinada com a estrutura de codificação das fórmulas. A estrutura de codificação e o tipo de informação adicionada à fórmula são as principais causas dos resultados obtidos neste trabalho.

Um trabalho futuro a ser realizado diz respeito a pesquisa dos domínios e problemas de planejamento da competição de planejadores e nos resolvedores SAT, buscado identificar quais informações são relevantes para a busca do plano. Pelos experimentos realizados neste trabalho sabe-se que para alguns domínios e problemas, algumas informações da estrutura de controle auxiliam a busca por uma valoração satisfatível. Entretanto, quando um conjunto maior de informações é adicionado à fórmula a busca é prejudicada.

Outro trabalho futuro diz respeito a criação de relações de ordenação de proposições. Na versão de 2006, o SATPLAN passou a codificar, na fórmula em CNF, apenas o *mutex*

de proposições e de interferência. Esta estratégia de codificação possibilitou a resolução de problemas difíceis, com uso eficiente de memória. O *mutex* de proposições é mais informativo, pois o conflito entre proposições é resultado de conflitos entre duas ou mais ações. Do mesmo modo, as relações de ordenação de proposições poderão ser resultado das relações de ordenação de duas ou mais ações. Assim, será feito um uso eficiente de memória na inclusão de informações adicionais a fórmula.

Outro estudo que pode ser aprofundado é o aproveitamento de cláusulas apreendidas do resolvedor SAT. Em testes realizados ao longo deste trabalho obteve-se uma melhora no tempo de execução quando estas cláusulas foram adicionadas à fórmula seguinte na expansão da rede. Uma desvantagem deste tipo de estudo é o acoplamento do módulo de construção da rede de planos e geração de fórmula com o resolvedor SAT.

Também é possível realizar pesquisas em torno da inclusão de heurísticas de planejamento no resolvedor SAT. Planejadores com o FF obtiveram bons resultados, quando utilizaram o grafo de planos relaxado como base para heurística de busca do plano no espaço de estados. É possível desenvolver heurísticas baseadas na rede de planos e incluí-las num resolvedor SAT específico para solução de problemas de planejamento representados na rede de planos.

Enfim, este trabalho de dissertação de mestrado apresentou resultados promissores e com ideias para pesquisas futuras na área de planejamento em IA. Esta é uma área de alta complexidade computacional, mas é de interesse para empresas, indústrias, entre outras organizações da sociedade em geral. Por isso a busca por soluções eficientes deve continuar a ser empreendida.

# BIBLIOGRAFIA

[1] International conference on automated planning and scheduling (ICAPS). http://ipc.icaps-conference.org.

[2] The international SAT competitions web page. http://www.satcompetition.org.

[3] VAL, the automatic validation tool for pddl, including pddl3 and pddl+ strathclyde planning group. http://planning.cis.strath.ac.uk/VAL.

[4] Avrim Blum e Merrick Furst. Fast planning through planning graph analysis. *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI 95)*, páginas 1636–1642, 1995.

[5] Blai Bonet e Héctor Geffner. HSP: Heuristic search planner. *Entry at the AIPS 98 Planning Competition*, june de 1998.

[6] Tom Bylander. The computational complexity of propositional STRIPS planning. *Artificial Intelligence*, 69(1-2):165–204, 1994.

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

[8] R. O. Fikes e N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3–4), 1971.

[9] Jörg Hoffmann e Bernhard Nebel. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research*, 14:253–302, 2001.

[10] Henry Kautz e Bart Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, páginas 1194–1201. AAAI Press, 1996.

[11] Henry Kautz e Bart Selman. Unifying SAT-based and graph-based planning. Jack Minker, editor, *Workshop on Logic-Based Artificial Intelligence, Washington, DC, June 14–16, 1999*, College Park, Maryland, 1999. Computer Science Department, University of Maryland.

[12] Henry A. Kautz. SATPLAN04: Planning as satisfiability. *Abstracts of the 4th International Planning Competition*, 2004.

[13] Henry A. Kautz e Bart Selman. Planning as satisfiability. *Proceedings of the Tenth European Conference on Artificial Intelligence (ECAI'92)*, páginas 359–363, 1992.

[14] Henry A. Kautz, Bart Selman, e Jöerg Hoffmann. SatPlan: Planning as satisfiability. *Abstracts of the 5th International Planning Competition*, 2006.

[15] R. J. Lipton. The reachability problem requires exponential space. Relatório técnico, Yale University, Department of Computer Science, 1976.

[16] John McCarthy e Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. B. Meltzer e D. Michie, editors, *Machine Intelligence 4*, páginas 463–502. Edinburgh University Press, 1969. reprinted in McC90.

[17] Drew Mcdermott, Malik Ghallab, Adele Howe, Craig Knoblock, Ashwin Ram, Manuela Veloso, Daniel Weld, e David Wilkins. PDDL - the planning domain definition language. Relatório técnico, Yale Center for Computational Vision and Control, 1998.

[18] Razer Anthon Nizer Rojas Montaño. Aplicações de fórmulas não-clausais em planejamento com redes de petri. Dissertação de Mestrado, UFPR, 2006.

[19] Razer Anthon Nizer Rojas Montaño, Marcos Castilho, Fabiano Silva, e Luis Kunzle. Usando Redes de Petri e Resolvedores ISCAS para Tratar Planejamento como Satisfatibilidade. *ENIA - Encontro Nacional de Inteligência Artificial*, 2011.

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

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

[22] Murray Shanahan. *Solving the frame problem: a mathematical investigation of the common sense law of inertia*. MIT Press, Cambridge, MA, USA, 1997.

[23] Fabiano Silva. Algoritmo para planificação baseada em STRIPS. Dissertação de Mestrado, UFPR, 2000.

[24] Fabiano Silva. *Rede de planos: uma proposta para a solução de problemas de planejamento em inteligência artificial usando redes de Petri*. Tese de Doutorado, CEFET-PR, 2005.

[25] Fabiano Silva, Marcos Alexandre Castilho, e Luis Allan Künzle. Petriplan: A new algorithm for plan generation (preliminary report). Maria Monard e Jaime Sichman, editors, *Advances in Artificial Intelligence*, volume 1952 of *Lecture Notes in Computer Science*, páginas 86–95. Springer Berlin / Heidelberg, 2000.

[26] I. A. Stewart. On the reachability problem for some classes of Petri nets. 1992.

[27] Christian Thiffault, Fahiem Bacchus, e Toby Walsh. Solving non-clausal formulas with dpll search. Mark Wallace, editor, *CP*, volume 3258 of *Lecture Notes in Computer Science*, páginas 663–678. Springer, 2004.

[28] G. S. Tseitin. On the complexity of derivation in propositional calculus. *Studies in constructive mathematics and mathematical logic*, 2(115-125):10–13, 1968.

[29] Daniel S. Weld. Recent advances in ai planning. *AI MAGAZINE*, 20:93–123, 1999.