CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DO PARANÁ

Programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial

---

DISSERTAÇÃO

apresentada ao CEFET-PR

para obtenção do título de

MESTRE EM CIÊNCIAS

por

MALGARETE RODRIGUES DA COSTA

UMA CONTRIBUIÇÃO AO ESTUDO DO PLANEJAMENTO TEMPORAL EM INTELIGÊNCIA ARTIFICIAL

Banca Examinadora:

Presidente e Orientador:

Prof. LUIS ALLAN KÜNZLE	CEFET-PR

Examinadores:

Prof. MARCOS CASTILHO	UFPR
Prof. ALEXANDRE IBRAHIM DIRENE	UFPR
Prof. FABIANO SILVA	UFPR

Curitiba, 07 março de 2005

MALGARETE RODRIGUES DA COSTA

UMA CONTRIBUIÇÃO AO ESTUDO DO PLANEJAMENTO TEMPORAL EM INTELIGÊNCIA ARTIFICIAL

Dissertação apresentada ao programa de Pós-Graduação em Engenharia Elétrica e Informática Industrial do Centro Federal de Educação Tecnológica do Paraná, como requisito parcial para obtenção do título de "Mestre em Ciências".

Área de Concentração: Informática Industrial.

Orientador: Prof. Dr. Luis Allan Künzle

Curitiba

2005

AGRADECIMENTOS

Para algumas pessoas, a finalização deste trabalho representa apenas mais uma dissertação de Mestrado concluída, porém para aqueles que acompanharam este processo, há a certeza, de uma etapa de grandes mudanças e dificuldades, transformando-o numa grande conquista.

Dentre as dificuldades encontradas veio a certeza de que é difícil dizer às pessoas o quanto são importantes e quanto vale a presença delas. Alcançar com exatidão o que se deveria expressar a cada pessoa aqui mencionada será impossível, porém fica o registro um pequeno agradecimento.

Sem dúvidas, as primeiras pessoas mencionadas, são aquelas a quem devo minha vida. São pessoas que nunca puderam me oferecer dinheiro e conforto, porém me ensinaram a imensa fortuna que a honestidade e a perseverança representam ao ser humano e o imenso conforto espiritual que a integridade nos proporciona. Uma homenagem especial a ti, mãezinha Julieta, que, embora o período que estiveste conosco tenha sido curto, deixaste a marca do que é ser uma verdadeira guerreira. Pai e Mãe minha gratidão a vocês será eterna.

Nesse período, tive a sorte de conviver com uma segunda família, a família LSIP. Sim, pois considero nosso relacionamento não apenas profissional, mas uma dádiva de Deus. Marcos, Jean, Andrey e Erik, meu muito obrigado. Valter e Evan um obrigado especial pelo ombro amigo nas horas em que a dureza da rapadura era maior que a doçura, obrigada pelo café, pelas gargalhadas e, claro, pelo apoio nas horas em que a vida virou um grande mutex.

Aqui deixo meu agradecimento especial ao Fabiano. Obrigada pela paciência, pelo incentivo e porque não pelas palavras duras, pois elas foram responsáveis pelo meu despertar e fizeram com que eu "continuasse nadando".

Agradeço aqui a uma pessoa que ainda não compreende o quanto ela foi importante nesse período, Maria Eduarda, devo a ti a dose extra de ânimo que tantas vezes precisei para a conclusão deste trabalho.

E especialmente ao meu Orientador, Luis Allan, pela compreensão, diante dos meus problemas e pelo apoio tanto profissional quanto pessoal.

Obrigada à CAPES, pelo apoio financeiro durante o desenvolvimento desse trabalho.

Agradeço também a todos aqueles que de alguma maneira contribuíram na conclusão dessa dissertação.

Malgarete R. da Costa

Conteúdo

Lista de Figuras

Resumo

Neste trabalho se estuda os principais métodos de planejamento temporal. Propõe soluções baseadas no grafo de planos, bem como soluções baseadas na tradução deste em Redes de Petri Temporais.

Objetivando a contextualização do cenário em que o presente trabalho se insere, é apresentada uma revisão dos algoritmos que fazem o tratamento de problemas de planejamento clássico e o planejamento temporal. Propõe-se um novo método de tratamento temporal sobre o grafo de planos e sua tradução para uma Rede de Petri Temporal.

Palavras chave: Inteligência Artificial, Planejamento Temporal, Redes de Petri Temporais.

Abstract

In this work it is studied the main methods of temporal planning. It is proposed solutions based on graph of plans as well as solutions based on the translation of graph of plans into time Petri nets.

A review is presented about the algorithms for the treatment of classical planning and temporal planning. This review aims at to present the context of this work. Subsequently, it is proposed a new method of temporal treatment for the graph of plans and its translation into a time Petri net.

Keywords: Artificial Intelligence, Temporal Planning, Temporal Petri Nets.

Capítulo 1

Introdução

A revolução tecnológica do último século tem entre seus três principais temas o computador. Avalia-se que a capacidade dos computadores tem duplicado a cada dezoito meses. Porém existem habilidades humanas que os computadores, aliados aos mais eficientes códigos, ainda não são capazes de reproduzir de forma tão perfeita e natural que a do ser humano. Entre estas habilidades está a capacidade de *planejar*.

Buscando encontrar um conjunto de ações que seja solução para algum problema pré-definido, o *Planejamento* passou a ser um dos objetos de estudo na área de *Inteligência Artificial*. Tomando o comportamento humano como base de contextualização, pode-se dizer que um *planejador* é um sistema que, a partir de uma informação inicial, busca uma combinação correta de ações, levando o agente do estado inicialmente definido ao objetivo final.

Ao traçar uma linha do tempo, observa-se que a área de pesquisa sobre planejamento tem suas raízes na década de 60, quando o planejamento automático era feito como prova de teoremas em lógica de Primeira Ordem [38]. Já nos anos 70, a área de pesquisa sobre planejamento apresentou notáveis progressos, tornando-se seu principal marco à passagem da demonstração de teoremas para as técnicas de busca clássica em Inteligência Artificial.

Nos anos 90, o avanço se deu em termos de eficiência e aperfeiçoamento de algoritmos, com seu potencial de aplicação focado nos problemas de nosso cotidiano, já que até então os planejadores eram baseados em técnicas de busca, cuja ineficiência não permitia tratar problemas simples que apresentavam amplos espaços de busca.

Note-se que desde a retomada das pesquisas em Planejamento na década de 90, inúmeras são os trabalhos que direcionam o tratamento de problemas para planejamento, transformando-o numa área multidisciplinar e dinâmica. Com essa multidisciplinaridade criaram-se competições, diversificaram-se os domínios e conseqüentemente ampliou-se a complexidade dos sistemas e surgiram novos métodos, entre eles os baseados em Funções Heurísticas [10], Redes de Petri [50, 15, 51], Satisfação de Restrições [31, 32, 33], Algoritmos Genéticos [35], entre outros.

Atualmente a área de planejamento está subdividida em dois segmentos: o *planejamento clássico* e o *planejamento não clássico*. No primeiro, a preocupação está principalmente na geração de planos que atinjam um conjunto de objetivos previamente definidos em uma situação. Resumidamente, pode-se afirmar que o planejamento clássico distingue-se por:

- Apresentar resultante determinística;
- Fazer uso de estados estáticos para representação do mundo;
- Mudanças do mundo ocorrerem apenas como resultado do efeito de uma ação.

No *planejamento não clássico*, o agente depara-se com situações imprevisíveis, cujo grau de certeza da ação nem sempre é absoluto. Isso pode decorrer de variáveis externas, ou seja, fatores externos que podem interferir no resultado. Entre estes fatores estão fatores climáticos, variações de energia, alterações de ambientes, entre outros.

No contexto do planejamento clássico, em 1971, Fikes e Nilson [16] apresentaram a linguagem STRIPS, uma linguagem simplificada que propunha a integração de um sistema formal a um algoritmo correspondente, ou seja, fazia uso de diferentes estratégias

e explorava diferentes espaços de busca. A partir de STRIPS ocorreu o desenvolvimento de uma ampla variedade de algoritmos para Planejamento, alguns deles serão apresentados ao longo deste trabalho.

Em STRIPS, os autores propunham a solução de um determinado problema utilizando-se de busca tradicional, ou seja, uma busca em uma árvore de estados do mundo, que pode alcançar um tamanho exponencial. Apesar de tratar uma classe restrita de problemas, o trabalho de Fikes e Nilson foi de extrema importância por apresentar um sistema formal que permitiu representar ações e estados de fácil compreensão, dando ao sistema uma independência entre o algoritmo e a linguagem que vai solucionar o problema.

Durante a década de 80, a pesquisa em planejamento se restringiu basicamente a duas abordagens: *a busca em STRIPS* e a por *prova de teorema*. Nestas duas abordagens recaía-se em busca exaustiva com a explosão combinatória de estados, o que tornava ambas ineficientes do ponto de vista da praticidade. Em 1992, Kautz e Selman propuseram a tradução da representação STRIPS para o cálculo proporcional, surgindo assim o *Satplan* [31, 21].

Logo em seguida, Blum e Furst apresentaram o *Graphplan* [9] que, assim como os demais planejadores, apresentava um modelo de planejamento baseado em grafos, porém inovando ao mostrar que, a partir de uma descrição STRIPS, era possível a construção de um grafo cujo espaço de busca do problema era sensivelmente reduzido.

Das inovações propostas no *Satplan* e no *Graphplan* ocorre uma retomada dessa área de pesquisa, motivando o resgate das antigas técnicas de busca. Surgiram então planejadores modernos e expressivos, que buscavam agregar condições de análise mais precisa através de funções heurísticas para guiar a busca.

Surgiu também os planejadores híbridos, conbinando elementos do planejamento com heurística, com as abordagens baseadas em grafo de planos e satisfabilidade. Dentre os inúmeros planejadores que surgiram nos últimos anos podemos citar o IPP [7, 28], Blackbox [34], UCPOP [45], ZENO [29] [6, 56], STAN [36], TGP [52], HSP [10], FF

[25], TP4 [23], SAPA [14], LPG [20], entre outros.

Na figura 1.1, baseada em [53], os planejadores foram agrupados de acordo com a principal técnica de busca.

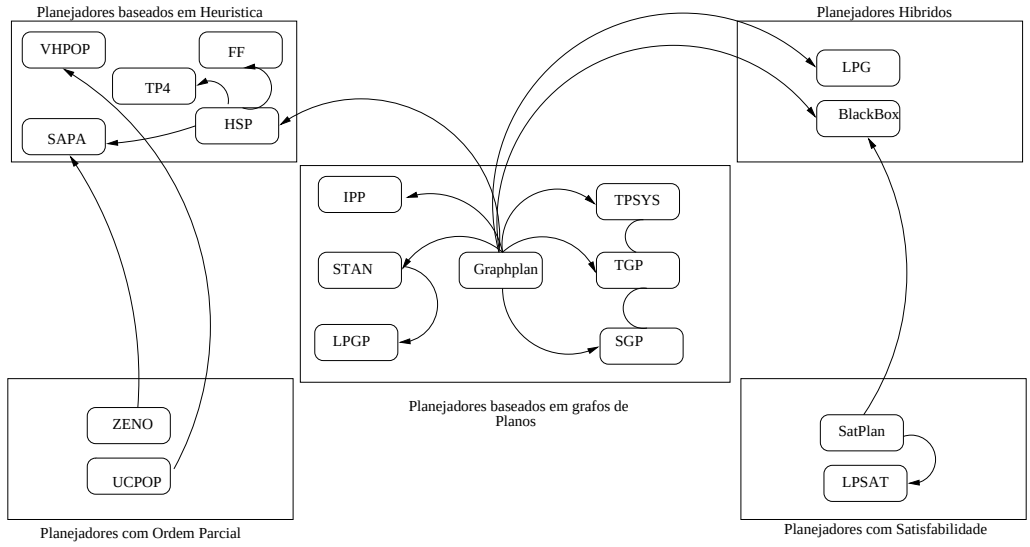

Figura 1.1: Grafo de evolução e derivações de alguns planejadores.

Desde o trabalho de Kautz e Selman, muitos outros algoritmos baseados em grafo de planos foram relacionados com outras áreas de pesquisa, entre elas a Programação Inteira (PI), [54] e Redes de Petri [50]. Observa-se que passados dez anos desde que ocorreu o relacionamento entre as áreas e aperfeiçoamento dos algoritmos, surgiram planejadores com capacidade de administrar situações muito mais práticas do que os iniciais problemas de quebra-cabeça.

Em 1998, o fator impulsionador à retomada ao desenvolvimento dos planejadores foi a competição bienal de planejadores automáticos, que teve sua primeira edição nesse ano. A primeira tarefa dos organizadores da competição foi apresentar uma linguagem de representação que seria comum à todos os competidores, derivada da junção das linguagesn STRIPS e ADL. Esta linguagem foi chamada de PDDL [40], e passou a ser a linguagem de descrição de problemas mais utilizada pela comunidade de planejamento.

Na competição de 1998 foi apresentado o planejador HSP [10] baseado em busca heurística, capaz de produzir planos com extrema eficiência e rapidez. Na competição de 2000, o grande destaque foi para o sistema FF (Fast-Forward Planning System)[24]. O sistema FF utiliza-se da já conhecida heurística do HSP [10], e obteve um grande destaque, mostrando ser um sistema robusto e rápido. Nas duas competições que se seguiram, 2000 e 2002, foi apresentada uma gama muito grande de novos planejadores que enfatizavam a resolução de problemas práticos. Particularmente, em 2002, deu-se maior destaque a problemas que incluíam o raciocínio numérico e temporal.

O planejamento temporal é o foco principal desta dissertação de mestrado. O objetivo principal do trabalho consistiu em estudar as especificações temporais presentes nas linguagens de descrição de problemas e propor soluções para sua inserção nos métodos desenvolvidos pela equipe de pesquisa, de forma a obter planos que atendam às restrições temporais. Desta forma, o capítulo dois desta dissertação se revê os principais conceitos de Planejamento em Inteligência Artificial, das linguagens de descrição de problemas e das estratégias de busca mais importantes. Como são inúmeros os planejadores que se utilizam do Grafo de Planos, inclusive o Petriplan, desenvolvido pela equipe na qual este trabalho está inserido, o capítulo três apresenta de forma detalhada estes métodos. No capítulo quatro a linguagem PDDL+, que inclui a duração das ações como parte da descrição temporal, é estudada, assim como são descritos os principais planejadores que se propõem a tratar quantitativamente as restrições temporais.

Finalmente, no capítulo 5 são apresentadas duas abordagens de geração de planos temporais. A primeira baseada sobre o Grafo de Planos e a segunda baseada sobre o Petriplan. Um exemplo federador permite ilustrar, para diferentes métodos, as possibilidades de análise de planos cuja dimensão temporal é tratada de forma explícita. Uma comparação dos resultados e das diferentes características dos métodos propostos conclui o capítulo.

Capítulo 2

Linguagens de Descrição e Planejamento em IA

Neste capítulo serão apresentados conceitos fundamentais para a área de planejamento, linguagens de descrição, bem como alguns planejadores que representam a área de planejamento clássico.

O objetivo desse capítulo é apresentar ao leitor características gerais das linguagens de descrição, características e limitações de alguns planejadores tidos como eixo central no planejamento clássico e formalizar alguns conceitos que representam a descrição de um planejador automático.

2.1 Conceitos Fundamentais

Sempre que projeta-se o desenvolvimento de um resolvedor automático de problemas, ou seja, um *planejador*, alguns conceitos estão presentes. Tais conceitos objetivam encontrar uma classe de modelos matemáticos que farão a definição do problema e apresentarão as possíveis formas de solução. Entre estes estão os conceitos de *planejador, plano e domínio.*

2.1.1 Planejador

De maneira informal um *planejador* é um algoritmo que necessita de uma entrada inicial, um objetivo e uma descrição do comportamento das ações em uma linguagem formal, ou seja, um problema de planejamento é uma tupla $\mathcal{P} = < \mathcal{O}, \mathcal{I}, \mathcal{G} >$, sendo $\mathcal{O}$ a descrição do conjunto de ações, $\mathcal{I}$ a descrição do estado inicial do problema e $\mathcal{G}$ o estado meta. Um plano solução para $\mathcal{P}$ é uma seqüência de ações de $\mathcal{O}$ que transformam $\mathcal{I}$ em $\mathcal{G}$.

Para melhor compreender o conceito acima, imagine uma empresa de logística que precisa transportar um pacote (pk_2) que está na localidade 1 (loc_1) para a localidade 2 (loc_2) e outro pacote (pk_1) que está na localidade 2 (loc_2) para a localidade 1 (loc_1). Para fazer esse transporte dispõe-se de um único caminhão (t_1). Logo pode-se afirmar que um problema particular para o domínio envolvido pode ser definido pela dupla $\mathcal{I}$ e $\mathcal{G}$. Sendo o estado inicial $\mathcal{I}$ representado pela situação descrita acima, ou seja, a localidade onde o pacote encontra-se inicialmente. Tem-se também que $\mathcal{G}$ é onde a empresa deve entregar os pacotes, estado objetivo e $\mathbb{O}$ são todas as possíveis ações a serem executadas para que os pacotes estejam nas localidades estipuladas.

2.1.2 Plano

Para o planejamento, o problema é obter uma seqüência de ações chamada *Plano*; e esta seqüência, se aplicada a algum mundo, satisfaz as descrições iniciais e alcance o objetivo. De modo informal, um planejador é um algoritmo que apresenta a capacidade de resolução de um problema de planejamento e possui as três entradas que serão codificadas em alguma linguagem formal, ou seja, a codificação gera a partir da entrada inicial um *plano* como saída.

2.1.3 Domínio

Conforme descrito nas subseções 2.1.1 e 2.1.2, um problema de planejamento é um algoritmo que faz a formalização deste problema, e atráves de um conjunto de premissas, busca encontrar de forma automática uma solução para o problema, quando esta existir, a esse conjunto de premissas dá-se o nome *domínio*.

A definição de um *domínio* é descrita básicamente pelas ações, ou seja, o conjunto $\mathbb{O}$, que apresenta o problema de planejamento $\mathcal{P} = < \mathcal{O}, \mathcal{I}, \mathcal{G}$. Sendo que $\mathcal{I}$ e $\mathcal{G}$ são responsáveis pela definição de um problema em particular para o domínio em estudo, ou seja, podem haver inúmeros problemas para um mesmo domínio, variando conforme as combinações feitas sobre os estados iniciais e finais.

Logo, o *Domínio* de um problema pode ser interpretado como o cenário utilizado por um protótipo para representar os elementos que compõem o problema.

A representação do *Domínio* de um problema pode ser feita de várias maneiras, ou seja, pode-se representar as informações de um problema de planejamento utilizando-se de alguma das linguagens de descrição que serão discutidas e apresentadas na seção a seguir.

2.2 Representação STRIPS

STRIPS (Stanford Research Institute Problem Solver) [16] é o precursor dos atuais sistemas de planejamento. Este foi o primeiro sistema a introduzir a noção de estado de mundo como um conjunto de predicados. Sua primeira versão foi apresentada em 1971 e foi destinada à solução de algumas aplicações, entre elas possibilitar que o robo Shakey contruísse planos para manusear blocos.

O ponto marcante de STRIPS é a incorporação de uma linguagem de descrição cujo modelo de representação é compacto e ter uma representação simples para domínios de planejamento. Baseada em cálculo proposicional, ela permite descrever mecanismos

e ações, e especificar estados do mundo. Estes são representados por conjunções de literais instanciados. O estado inicial ou final é obtido pela aplicação de predicados sobre constantes, também chamados proposições.

Em STRIPS, a definição de domínio e do conjunto de operadores relacionados ao problema de planejamento é dado por $\mathcal{P}_p = < L, U, S_i, S_f >$.

Onde:

- $\mathcal{L}$ é a representação da linguagem de primeira ordem.
- $\mathcal{U}$ é o conjunto de operadores.
- $\mathcal{S}_i, \mathcal{S}_f$, são respectivamente a definição inicial do problema e o estado objetivo.

Tomando STRIPS para formalizar um problema de planejamento $\mathcal{P}_p$, podemos descrevê-lo como um conjunto de ações, cada qual constituída de três partes:

- Primeiramente devemos ter a descrição da ação, com sua identificação e a definição de seus parâmetros;
- Posteriormente devemos descrever as pré-condições, sendo que estas devem ser necessariamente verdadeiras para que a ação proposta possa ser aplicada;
- Por último, tem-se os efeitos que a ação executada irá gerar descritos por uma conjunção que pode incluir literais positivos ou negativos. Tais literais descrevem as alterações que devem ser realizadas sobre o mundo objetivando obter um outro após a execução da ação.

Em termos de limitação, os sistemas de planejamento baseados em STRIPS possuem restrições quanto à capacidade de representação da linguagem, pois supõem que todos os sub-objetivos são independentes uns dos outros, ou seja, considera a linearidade. Entretanto, existem interações de sub-objetivos que podem forçar que eles tenham que

ser obtidos em uma certa ordem; ou ainda sub-objetivos que precisam ser satisfeitos juntos como, por exemplo a **Anomalia de Sussman** [26].

STRIPS também apresenta o problema de utilizar métodos de busca exaustiva, o que acarreta explosão combinatória do número de estados na solução dos problemas, inviabilizando a sua aplicação em planos que envolvam problemas do mundo real. Por outro lado, STRIPS apresenta como grande vantagem a simplicidade de representação, que pode ser combinada com estratégias de busca otimizadas.

2.3 ADL - Action Description Language

Devido às limitações de STRIPS, surgiram inúmeras derivações que se originaram com base em sua estrutura, entre elas a linguagem ADL (Action Description Language) [44], uma das mais citadas na literatura.

ADL tem como base um modelo algébrico para caracterizar os estados do mundo e supõe um replanejamento do cálculo de situações mediante a definição de uma ação *(a)* como uma tupla $\langle Pre, Add, Del, Update(s,t)\rangle$ onde:

- *Pre* são as pré-condições de a, *Add* e *Del* representam conjuntos de formúlas com conjuntos de tuplas a acrescentar e excluir respectivamente, ao executar a ação a;
- *Update(s, t)*, faz a descrição do conjunto dos pares de estados $\langle s,t\rangle$ de forma que ao executar a ação a no estado s obtem-se o estado t.

Em termos de ampliações, quando comparada à linguagem STRIPS, podemos observar que em ADL ocorre:

1. Inclusão de efeitos condicionais;
2. Inclusão de efeitos quantitativos; e

3. Inclusão de novas características nas pré-condições, tais como disjuntores, fórmulas quantitativas, além de outros operadores lógicos.

As amplicações acima descritas permitem a redução do número de ações instanciadas, aumentando assim a eficiência do planejador, conforme pode-se observar detalhadamente em [27].

2.4 PDDL - Planning Domain Definition Language

A linguagem PDDL (The Planning Domain Definition Language)[40] foi especialmente desenvolvida para a Competição de Planejadores realizada em 1998, a mesma foi desenvolvida com o objetivo de ser uma especificação neutra dos problemas de planejamento, ou seja, não favorecer nenhum planejador específico. A PDDL originou-se da combinação de alguns formalismos existentes em planejamento [2, 16, 17, 44, 58], revelou-se uma linguagem com alta capacidade de representação, com capacidade de representar universos dinâmicos, axiomas de domínio, efeitos condicionais, quantificadores universais, gerenciamento de múltiplos problemas em domínios múltiplos, composição ou destruição dos objetivos e ações hierárquicas compostas por sub-ações e submetas.

Com essa nova representação, a comunidade científica objetivava comparar o desempenho dos planejadores da época, para que problemas e domínios pudessem ser descritos de maneira universal e aplicados a qualquer planejador capaz de interpretar a linguagem. Para tanto era necessário padronizar a descrição de domínios e problemas. Com sua capacidade de representação, a PDDL garantiu a integração de forma simples entre diferentes planejadores, bem como, uma representação equivalente ao conjunto destes formalismos.

Conforme já apresentado na subseção 2.1.1, um problema de planejamento é representado por uma tupla $\mathcal{P} = < \mathcal{O}, \mathcal{I}, \mathcal{G} >$. Baseando-se no modelo formal de planejamento, PDDL descreve o domínio, tendo como base a descrição das ações. Se rela-

cionado ao domínio, pode-se afirmar que $\mathcal{O}$ faz referência ao problema de planejamento, enquanto $\mathcal{I}$ e $\mathcal{G}$ fazem a definição para o domínio em análise.

Um domínio em PDDL é representado por uma declaração que contém: o nome, o conjunto de predicados presentes, o conjunto de características necessárias para sua definição e as ações que poderão ser por ele executadas.

Da mesma maneira que existem parâmetros para a descrição do domínio, a representação de problemas em PDDL também exige algumas declarações em sua descrição, tais como: nome do problema, pertinência de domínio, conjunto de objetos que estarão presentes no domínio, estado inicial e estado final.

Ao longo dos últimos anos, a PDDL mostrou-se uma linguagem simples, porém a mesma é formada por um número de itens elevado, possibilitando a representação das funcionalidades por ela proposta. Sua sintaxe é claramente difinida no manual, porém semanticamente informal, sendo necessário um acompanhamento dos manuais ao longo da evolução das linguagens e sistemas que a PDDL substitui.

2.5 Estratégias de busca em planejamento

Várias gerações de pesquisadores já dedicaram seus esforços no complexo desafio que tornou-se o planejamento automático em IA. Atualmente existem inúmeros planejadores apresentados pela comunidade científica, bem como inúmeras técnicas sob as quais os planejadores são implementados com o intuito de obter resultados cada vez mais satisfatórios. A seguir serão abordados o planejamento heurístico e o planejamento por satisfatibilidade, pois ambos são estratégias de busca contempladas em inúmeros planejadores classificados dentro do planejamento clássico.

2.5.1 Planejamento Heurístico

A busca de uma possível solução para um problema de planejamento baseada em processos exaustivos detém-se em situações de explosão combinatória quando aplicada a problemas do mundo real.

Uma alternativa é a utilização de procedimentos de busca não completa, baseados em funções heurísticas. Tal solução não é exaustiva, porém gera um planejador sem garantia de solução genérica.

A função heurística tem como objetivo central o direcionamento de um procedimento de busca, que seja capaz de prever, de forma eficiente, o custo de sair de um determinado estado para chegar ao estado meta.

Em planejamento, uma função com as características acima mencionadas tem que ser capaz de medir o custo entre qualquer estado do mundo e o estado final de forma otimizada.

O sistema HSP (Heuristic Search Planner) [11], bem como toda a família de planejadores baseados em busca heurística, utilizam-se de heurísticas calculadas resolvendo uma simplificação do problema. Isto permite estimar com maior precisão os custos que irão orientar a tomada de decisão. Algumas dessas simplificações podem nos levar a cotas inferiores se comparadas àquelas que permitem fazer um planejamento ótimo. Utilizando-se algoritmos gerais de busca geralmente chega-se a soluções sub-ótimas.

O sistema HSP tem como principal característica a utilização de um método heurístico, em que a função de estimação é calculada automaticamente com base nas informações extraídas de uma estrutura similar ao grafo de planos[53].

Para guiar o procedimento de busca, o HSP faz uso da técnica *subida de encosta*. O procedimento de busca adotado pela técnica *subida de encosta* é genérico para todas as buscas heurísticas. É um procedimento relativamente simples, pois calcula, a cada passo, o custo dos filhos do nó atual da árvore de busca, selecionando o de menor custo e que será utilizado para realizar a expansão no próximo passo. Sucessivamente, o

processo é executado até que o estado meta seja alcançado. Para se obter melhores resultados o procedimento de busca é incrementado com alguns recursos, tais como a memorização de nós já visitados, a fim de evitar o reprocessamento, e uma estratégia para que a busca se reinicialize em um outro estado quando um mínimo local é alcançado [11].

Embora a utilização da técnica de *subida de encosta* não forneça um procedimento completo na resolução de problemas envolvendo planejamento, o HSP permite obter ótimos resultados em alguns domínios, conforme os resultados obtidos na competição de 1998 [39].

Dentre os planejadores que utilizam-se de técnicas de busca heurística, um dos que merece destaque por sua eficiência é o FF (Fast Forward Planning System) [25]. O FF faz uso de funções heurísticas para guiar o processo. Desenvolvido na Universidade de Freiburg e apresentado por Jorg Hoffmann no AIPS/2000, o sistema FF utiliza-se do grafo de planos como base para sua função heurística.

No processo de busca, o FF se utiliza de uma função heurística, fazendo uma estimativa desde o estado em que se encontra até o estado objetivo. De maneira geral, pode-se dizer que o sistema FF desenvolveu uma heurística com base no *grafo de planos relaxado*, gerado pelo *Graphplan*, que é muito similar à heurística utilizada pelo sistema HSP, o que permite estimar a distância em números de ações desde o estado atual até o estado objetivo.

Apesar do FF basear-se na heurística do sistema HSP, ele apresenta uma função de estimativa diferenciada de HSP, pois só faz uso das interações com valor positivo na construção dos objetivos. Além de utilizar um plano relaxado tomando um grafo de planejamento sem a presença de efeitos negativos.

Dessa maneira, o sistema FF gera um plano, entretanto ao invés de determinar as relações de *mutex* entre ações, usa o grafo para obter a heurística que irá conduzir o processo de busca para a solução.

Outro diferencial marcante entre HSP e FF, é o fato que o FF não contempla ações que ocorrem em paralelo. FF também não contempla ações cuja duração e qualidade de planos ficam muito distante do que foi definido como ótimo para o problema.

Como já mencionado, seus princípios básicos são os mesmos do HSP apresentado por Bonet (Bonet, Loerincs & Geffner, 1997), ou seja, o FF pode ser visto como um sucessor avançado do HSP que se difere por apresentar:

- Um método mais sofisticado para a avaliação da heurística, levando em conta interações positivas entre fatos;
- Uma estratégia de busca local, utilizando-se de uma sistemática para não cair em platôs ou mínimos locais;
- Um método de identificação para os sucessores de um nó de busca que parece ser útil adquirindo a meta.

Desde o seu desenvolvimento, o FF tem se destacado por seus resultados na resolução de problemas de planejamento clássico. Seu sucesso deu origem ao Metric-FF [24], cuja extensão preocupa-se com características numéricas.

Embora o sistema FF tenha obtido os melhores resultados na competição de 2000, ele não obteve os mesmos efeitos na competição de 2002, por não agregar uma métrica capaz de manipular problemas com restrições temporais. Conseqüentemente outros sistemas que apresentavam capacidade de manipular tais restrições, solucionaram mais problemas no decorrer da competição, obtendo assim os melhores resultados.

2.5.2 Planejamento com Satisfatibilidade

O planejamento baseado em Satisfatibilidade (SAT) [31], juntamente com o planejamento baseado em grafo de planos, pode ser apontado como uma das aproximações relevantes para o planejamento clássico.

Uma instância SAT pode ser definida como um conjunto de restrições ou literais que assumem um valor verdadeiro ou falso. O planejamento baseado em satisfatibilidade pode ser definido como a transformação de um problema em um conjunto de axiomas, cuja propriedade principal é o fato de que qualquer valoração que faça com que este conjunto seja satisfatível, corresponderá a um plano válido.

Planejadores que utilizavam esta estratégia de busca ficaram abandondos por um longo período devido o baixo desempenho destas técnicas. Somente com o desenvolvimento de métodos de maior eficiência [31, 32] para a prova baseada em satisfabilidade eles passaram a ser utilizados para a resolução de problemas de planejamento.

Diversos experimentos mostraram que a tradução de problemas de planejamento para instâncias SAT pode gerar planejadores com desempenho muito próximo aos de melhor desempenho da competição de 98 [33].

O principal problema da busca de uma solução de um problema de planejamento utilizando satisfabilidade é a ordem das ações, uma vez que em planejamento temos uma componente temporal inerente, ou seja, no plano temos uma seqüência de ações ordenadas cronologicamente, enquanto que em satisfabilidade, o problema apresenta-se de modo estático com tempo fixo e discreto das instâncias.

Uma da vantagens na utilização do SAT é a facilidade que se encontra para especificar condições e restrições em qualquer dos estados intermediários do mundo, gerando modelos com maior precisão sobre os planos.

Na literatura sobre planejamento que utiliza-se de algoritmos para satisfatibilidade, pode-se apontar o SATPLAN [31]. Pode-se resumir SATPLAN como um planejador que, partindo de um problema de planejamento descrito em PDDL, faz a tradução para um problema de satisfatibilidade resolvendo o mesmo por meio de algum método já conhecido.

Outro planejador que faz uso da estratégia de satisfatibilidade é o BLACKBOX [30]. Proposto por Kautz e Selmam, o BLACKBOX obtém a instância SAT a partir do grafo

de planos.

Entre os planejadores que fazem uso dos algoritmos para satisfatibilidade também encontra-se o LPSAT [1]. LPSAT utiliza-se de uma extensão do PDDL, que corresponde ao nível dois PDDL 2.1 combinando instância SAT com desigualdades lineares.

Capítulo 3

Planejadores Graphplan e Petriplan

Por diversas décadas, muitos sistemas de planejamento foram desenvolvidos com o intuito de automatizar soluções para diversos problemas. Porém, na década de 90, é que obtiveram-se sistemas de planejamento com resultados satisfatórios. Os novos planejadores apresentados estabeleceram um novo paradigma na área, como pioneiro podemos citar o sistema *Graphplan* [8, 9].

Dentre os diversos planejadores concretizados a partir do Graphplan está o *Petriplan*, cuja primeira versão foi apresentada em [50], onde se faz a tradução do grafo de planos gerado pelo Graphplan para uma Rede de Petri. Os conceitos sobre Redes de Petri são encontradas no Anexo B, deste trabalho.

O objetivo desse capítulo é apresentar de forma mais detalhada os planejadores Graphplan e o Petriplan, bem como um exemplo de aplicação. A partir dos resultados obtidos pretendemos avaliar suas vantagens e desvantagens quando usados na resolução de problemas com características de situações reais.

3.1 Graphplan

Se analisada a literatura sobre a evolução da área de planejamento nota-se que até meados de 1994, todos os algoritmos para planejadores tinham seus alicerces derivados de três classes:

- Os que derivavam de STRIPS;
- Os que usavam procedimentos heurísticos;
- E os que usavam procedimentos baseados em prova SAT.

Em 1995, surge uma nova abordagem para os planejadores automáticos cuja base estava alicerçada sobre um grafo de planos, o Graphplan [8, 9]. O Graphplan apresentou resultados bem mais eficientes que qualquer outro planejador da época, podendo ser considerado um ponto de inflexão no modelo de planejamento clássico. Para obter tal agilidade, Blum e Fust utilizaram-se de uma idéia simples que consistia em dividir o processo de construção em duas etapas:

- Expansão do grafo;
- Extração da solução.

Para isso, o algoritmo executa repetidamente estas duas fases até que uma solução seja obtida.

3.2 Grafo de Planos

A composição de um grafo de planos é formado por camadas, onde cada camada é definida pelos nós que a compõem. Esses nós são clássificados como nós *proposição* ou *ação*.

Os *nós proposição* estão sempre localizados nas camadas pares do grafo (incluindo a camada zero), representando as proposições possíveis do problema em questão. Já os *nós ações* localizam-se nas camadas ímpares, sendo estas as instâncias cujas pré-condições devem ser satisfeitas pelas proposições da camada a ela antecedente. Na camada de número zero estão armazenados todos os literais positivos do estado inicial do problema em questão. Já nas camadas ímpares encontram-se as instâncias de ações, cujas pré-condições são satisfeitas pelas proposições contidas na camada *n - 1*(sendo n a camada atual), ou seja, a camada par. Graficamente os nós que incluem uma proposição no grafo são representados por um círculo, enquanto os nós que fazem a inserção de uma ação são representados por um retângulo.

As arestas de um grafo têm a finalidade de conectar os nós proposição aos nós ação da camada seguinte. São classificadas de quatro maneiras:

1. *Arestas proposição-ação:* conectam os nós proposição aos nós ação da camada seguinte, ligando as pré-condições das ações às proposições a ela correspondente;

2. *Arestas ação-proposição:* conectam os nós dos efeitos das ações aos nós proposição da camada posterior, ligando os efeitos das ações de uma camada às proposições correspontes a ela na camada seguinte;

3. *Arestas de manutenção:* conectam dois nós proposição, ou seja, para cada proposição presente na camada n é aplicada a referida ação de manutenção, resultando na existência dessas proposições na camada $n + 2$. As arestas de manutenção são incluídas no grafo para que se tenha a garantia de que uma ação que foi executada na camada n seja válida na próxima camada;

4. *Arestas de exclusão mútua:* representam as relações de exclusão existente entre ações ou proposições.

Podemos observar o grafo de planos representado na figura 3.1, onde os círculos que

representam os nós proposição os quadrados representam os nós ação e as arestas que fazem as conexões entre as camadas.

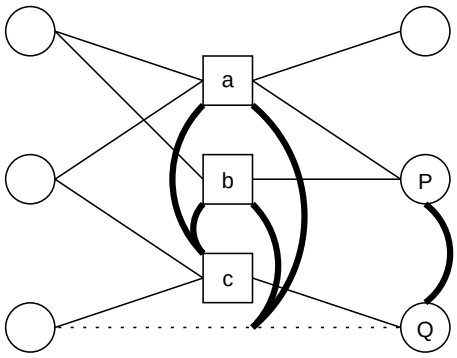

Figura 3.1: Exemplo de expanção do grafo.

A espessura das arestas que ligam os nós ação aos nós proposição, representa o tipo de conexão existente entre os nós, por exemplo, as arestas sólidas representam a conexão entre diferentes tipos de nós (ação-proposição, proposição-ação), as arestas de maior espessura representam as relações de exclusão mútua (*mutex*), entre ações ou proposições que podem ocorrer em virtude de situações contraditórias, conforme será descrito no decorrer deste capítulo, enquanto as arestas pontilhadas representam as arestas de manutenção, cujo objetivo é manter numa camada n as proposições existentes na camada *n - 2*.

As camadas proposição do grafo contêm os estados do mundo. Dessa forma, a construção do grafo tem início ao atribuirmos a primeira camada de proposições que irão representar o estado inicial do problema. Para a primeira expansão ser efetuada, verifica-se quais são as ações que têm suas pré-condições satisfeitas pelo estado inicial do problema e estas são inseridas no grafo. As expansões seguintes ocorreram adicionando-se uma nova camada de proposição contendo os efeitos das ações que foram incluídas na camada anterior. A opção por parar a expansão do grafo é feita quando obtém-se o estado meta, ou quando a camada de proposição alcançada é idêntica a camada de

proposições $n-2$.

3.3 Expansão do Grafo

A expansão do grafo consiste em adicionarmos uma nova camada de nós com instâncias de ações, que têm suas pré-condições satisfeitas na camada anterior $(n-1)$. Todas as proposições da camada atual são mantidas na nova camada através das *ações de manutenção*.

Dessa forma quando iniciamos um grafo temos apenas uma camada de proposições que irá representar o estado inicial do problema. Todas as ações cujas pré-condições estão presentes na camada atual irão compor uma nova camada de ações. A próxima camada a ser adicionada será a camada de proposições composta pelos literais originados dos efeitos das ações da camada anterior.

No decorrer de uma expansão do grafo podemos nos deparar com inconsistências e conflitos. Isso acontece em decorrência de inconsistências existentes entre proposições e ações, gerando restrições nas camadas do grafo. Um exemplo muito comum de inconsistência em um grafo, é termos em uma mesma camada o efeito de uma ação que é a negação do efeito da outra, conforme podemos observar na figura 3.2.

Visando o controle de tais ocorrências define-se a relação como uma exclusão mútua *(mutex)* e sua representação é feita por uma aresta que liga ações contidas na mesma camada. As situações de relações mutuamente exclusivas podem apresentar-se de quatro maneiras diferentes.

Por definição duas ações são mutuamente exclusivas em uma camada n se:

Inconsistência: O efeito de uma ação é a negação do efeito da outra.

Na figura 3.2 podemos observar que a ação b tem como efeito a proposição $\neg T$ sendo que esta é a negação de um efeito da ação a, logo as ações a e b são mutuamente exclusivas na camada.

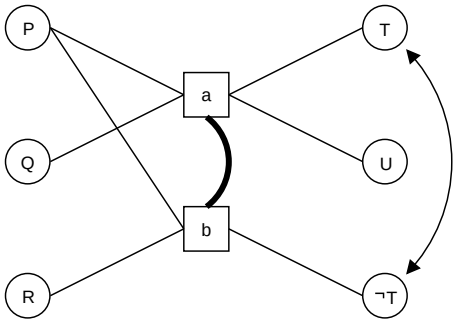

Figura 3.2: Representação Gráfica de Inconsistência

Interferência: O efeito de uma ação é a negação de uma pré-condição de outra.

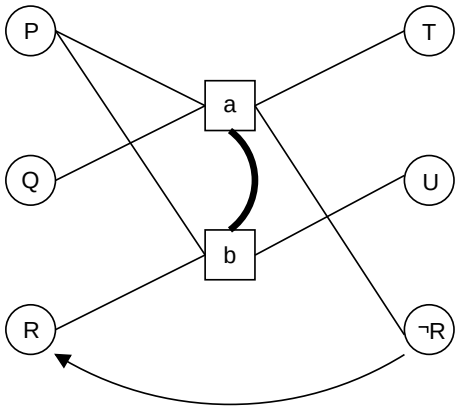

Figura 3.3: Representação Gráfica de Interferência

Na figura 3.3 a ação a gera como efeito à proposição $\neg R$, sendo esta a negação de uma pré-condição de b, fazendo das mesmas ações mutuamente exclusivas.

Competição por necessidades: as ações têm pré-condição que são mutuamente exclusivas na camada *n - 1.*

Na figura 3.4 as ações a e b exigem pré-condições que são mutuamente exclusivas na camada anterior (Q e ¬Q), fazendo delas ações mutuamente exclusivas.

Da mesma forma que entre ações, as relações de exclusão também são definidas para as proposições. Uma relação de exclusão entre proposições significa que as mesmas não

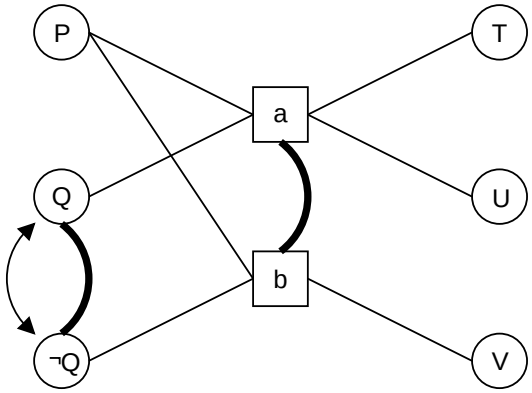

Figura 3.4: Representação Gráfica de Competição por Necessidade

poderão ser obtidas simultaneamente na mesma camada. Por definição temos que duas proposições são mutuamente exclusivas em uma camada n se qualquer das condições a seguir for verificada:

- Uma proposição é a negação da outra, conforme podemos observar na figura 3.4.
- As formas de obtenção das proposições são mutuamente exclusivas entre si, ou seja, as ações da camada n - *1* que apresentam como efeito tais proposições são duas a duas mutuamente exclusivas.

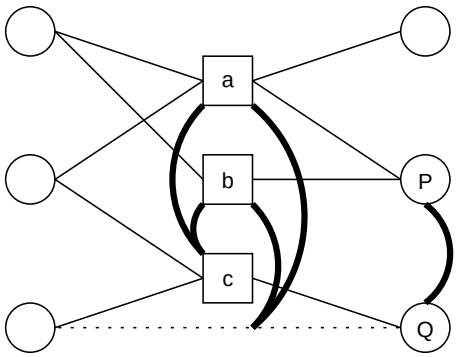

Figura 3.5: Representação Gráfica de Proposições Mutuamente Exclusivas

Nesse caso temos as proposições P e Q, que são mutuamente exclusivas em decorrência da exclusão existente entre as ações que dão origem a P (a e b) e as ações que dão origem a Q (c e a ação de manutenção)

3.4 Extração da Solução Através de Um Grafo de Planos

A extração de uma solução do grafo só é possível quando, após uma expansão, encontramos na última camada todos os literais que havíamos definido inicialmente como meta do problema.

A presença destes literais nos indica a possibilidade de existência de uma solução para o problema em análise. Se o grafo de planos até a expansão atual não contiver uma solução, expandimos mais uma camada do grafo com o objetivo de, com essa expansão e suas novas ações, gerar uma solução. Para tal verificação usa-se uma estratégia de *busca regressiva*, que consiste em analisar cada sub-meta da camada *n* e encontrar uma ação, na camada *n - 1*, que a contenha como efeito.

Considerando *a* uma ação e sendo ela consistente, ou seja, não é relação de exclusão com as demais ações escolhidas anteriormente, passamos então para a análise da próxima sub-meta, porém se detectarmos qualquer caso de exclusão das enumeradas na seção 3.3, retornamos e escolhemos outra ação. A partir do momento que encontramos um conjunto de ações consistentes para a camada *n - 1*, de forma recursiva vamos para a camada *n - 2*, agora considerando como sub-metas as pré-condições das ações escolhidas na camada *n - 1*, usamos esse processo recursivo até alcançarmos a camada zero. Se o procedimento nos permitir alcançar um conjunto de ações consistentes nas camadas anteriores, temos então uma solução para o problema. Caso o *retrocesso* falhe após tentar todas as possibilidades, faz-se necessária uma nova expansão do grafo.

Se após uma nova expansão do grafo observarmos que as duas últimas camadas

ímpares não sofreram nenhuma alteração, ou seja, nenhum mutex foi eliminado, assim podemos concluir que nenhum plano poderá ser encontrado para o problema em questão.

Para melhor compreender os conceitos acima abordados será apresentado na subseção 3.8.1 um exemplo de aplicação.

3.5 Petriplan

Conforme mencionamos no início desse capítulo, a partir do Graphplan originaram-se vários planejadores entre eles o *Petriplan*.

O algoritmo Petriplan utiliza-se de uma estrutura semelhante do Graphplan, porém faz a tradução do grafo de planos para uma Rede de Petri, objetivando a redução do espaço de busca.

3.6 O Algoritmo Petriplan

Em [50] é apresentado o algoritmo *Petriplan*, cuja estrutura muito se assemelha ao *Graphplan*, porém a definição do algoritmo é feita em três etapas.

Na primeira fase o Petriplan faz a construção do grafo de planos utilizando-se das mesmas regras do *Graphplan*. A passagem por esta fase de pré-processamento tem como objetivo a simplificação do problema para as fases seguintes.

Já na segunda fase de extração da solução do problema, o grafo de planos é transformado em uma RdP em que o problema original de Planejamento passa a ser analisado como um problema de busca de uma seqüência de disparos da transição que garanta a alcançabilidade de sub-marcação.

Finalmente na terceira fase do algoritmo o problema de alcançabilidade definido na fase anterior é convertido para um problema de Programação Inteira (P.I.), que então é resolvido por sistemas específicos de otimização.

Após execução da terceira etapa, se nenhuma solução for encontrada para o problema de P.I., o algoritmo irá retornar para a primeira fase e uma nova expansão do grafo será realizada. Este laço irá permanecer de forma a encontrar uma solução ou até que uma condição de falha seja apresentada na primeira etapa.

3.7 A Tradução para uma RdP

Conforme apresentada na seção 3.2 um grafo de planos é obtido alternando-se camadas de proposições e ações, a seguir será apresentada a transformação do grafo de planos em uma RdP. Será visto que obter um plano válido de um grafo de planos torna-se equivalente a resolução de um problema de alcançabilidade.

Vale resaltar que um problema de alcançabilidade em RdP, é definido pela obtenção de uma seqüência de transições que sejam disparáveis para se obter a marcação definida como objetivo partindo da marcação inicial. Já o problema de alcançabilidade de submarcação, é definido pela obtenção de uma seqüência de transições que possam ser disparadas para a obtenção de um subconjunto de lugares marcados que compõem a marcação objetivo.

Observando um grafo de planos nota-se que o mesmo pode ser fragmentado em cinco estruturas básicas, no algoritmo apresentado em [50], o autor traduz essas estruturas, para estruturas equivalentes em RdP. As cinco estruturas que compõem o grafo são:

- nós ação;
- nós proposição;
- arestas de nós proposição;
- arestas de nós ação para nós proposição; e
- relações de exclusão mútua.

A tradução para um RdP segundo [50] se dá da seguinte maneira:

3.7.1 Nós Ação

A tradução de um nó ação do grafo se faz através de uma única transição na RdP, conforme a representação mostrada na figura 3.6.

Figura 3.6: Tradução de um nó ação em uma transição na RdP.

Após a tradução, a transição só estará habilitada se, e somente se, todas as pré-condições forem validadas, ou seja, se existirem marcas em todos os lugares que representam as arestas pré-condições ligadas à ação representada pela transição.

3.7.2 Nós Proposição

Um nó proposição é traduzido em um lugar e uma transição com um arco do lugar para a transição, conforme a representação abaixo.

Um nó proposição, é traduzido por um lugar ligado a uma transição. A transição se faz necessária para que todos os lugares que representam arestas pré-condição recebam marcas após o disparo.

3.7.3 Arestas Pré-Condição

A tradução da aresta pré-condição se faz através de um lugar com dois arcos, um vindo da transição que representa o nó proposição e outro que vai para a transição que

representa o nó ação, conforme representação gráfica em 3.7:

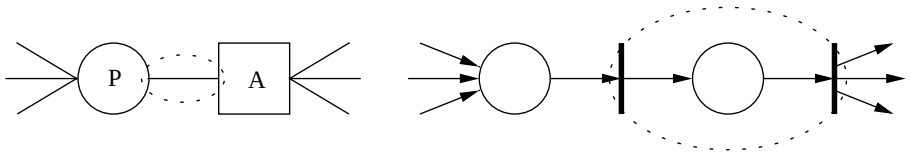

Figura 3.7: Tradução das arestas pré-condição em um lugar com dois arcos na RdP.

As arestas pré-condição são representadas por um lugar devido ao fato de um nó proposição do grafo poder ser pré-condição de mais de uma ação, assim cada transição que representa ações tem um conjunto de arestas pré-condição independente. Caso esta estrutura não fosse utilizada, o disparo de uma transição ação desabilitaria outra transição ação que tivesse uma pré-condição em comum, pois o disparo da primeira removeria as marcas de todos os lugares representando sua pré-condição.

3.7.4 Aresta Efeito

Uma aresta efeito é traduzida em um arco, que vai da transição que está representando o nó ação, para o lugar representando nó proposição conforme representação gráfica em 3.8.

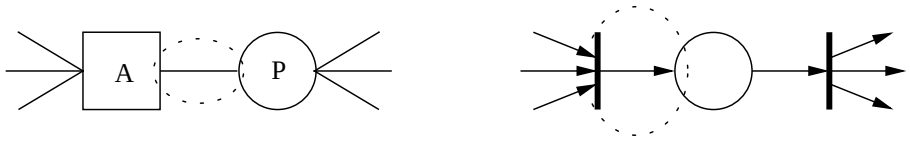

Figura 3.8: Tradução das arestas efeito na RdP.

3.7.5 Exclusão Mútua

A relação binária de exclusão mútua entre duas ações se faz através de um lugar marcado que é pré-condição dessas ações.

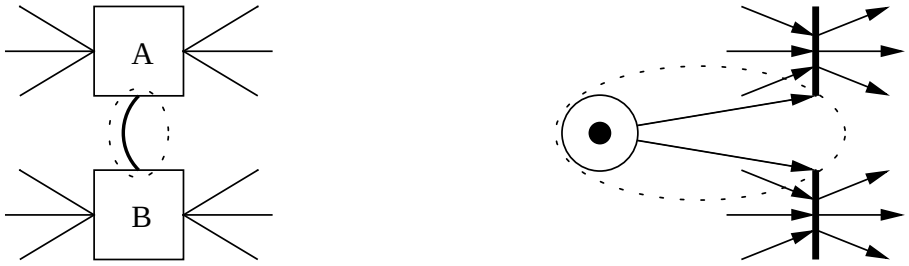

Figura 3.9: Tradução das relações de exclusão mútua na RdP.

Como podemos observar na figura 3.9, o lugar que representa a exclusão mútua possui uma única marca. O disparo de qualquer uma das transições que representa as ações desabilita a outra transição.

Inicialmente temos a rede marcada com peso inicial um. A existência de uma marca em determinado lugar na rede, indica uma proposição válida, ou ainda, um *mutex* entre duas ações.

Se algum lugar da rede receber mais de uma marca, podemos concluir que há mais de um caminho que torna válida a proposição representada por esse lugar.

Finalmente, o estado inicial do problema é traduzido pelo inclusão de uma marca em cada um dos lugares que representam as proposições iniciais além das marcas presentes nos mutex.

3.8 Exemplo de Aplicação

O exemplo a seguir tem como objetivo ilustrar os conceitos acima abordados, bem como avaliar a questão de desempenho dos dois planejadores quando analisados sob os aspectos de planejamento clássico em IA.

3.8.1 Descrição do Problema

O exemplo proposto consiste em um problema de logística. Este exemplo é um dos problemas clássicos da área de planejamento, podendo ser encontrado na íntegra no artigo da PDDL2.1 proposto por *Fox* e *Derek* [19]. Utilizaremos entretanto uma versão simplificada deste problema, restringindo os *recursos* [1] disponíveis evitando uma explosão combinatória de estados na busca da solução.

O problema aqui proposto consiste em transportar pacotes entre localidades. Para efetuarmos tal transporte estaremos considerando apenas os elementos: *truck* (t_n), *locality* (loc_n), *package* (pk_n). Com estes elementos três ações são passíveis de serem executadas:

- *Load*, ação de carregar um pacote em um caminhão;
- *Unload*, ação de descarregar um pacote de um caminhão; e
- *Drive*, ação de dirigir um caminhão.

As ações acima mencionadas serão executadas se, e somente se, suas pré-condições forem atendidas. Dessa forma temos que:

- A ação *Load* será executada se existir uma localidade *(loc)*, um caminhão *(t)* nesta localidade e se nesta localidade existir um pacote *(pk)* disponível para carregar.
- A ação *Unload* será executada se existir um pacote *(pk)* dentro de um caminhão *(t)* e este caminhão estiver em uma localidade *(loc)*.
- A ação *Drive* será executada se existir um caminhão *(t)*, uma localidade de origem e uma localidade destino.

Para o problema proposto temos a seguinte descrição inicial:

[1] Recursos são todos os elementos necessários para a execução das ações exigidas pelo problemas em questão.

- Existe um caminhão na localidade 1.
- O pacote 1 está na localidade 2.
- O pacote 2 está na localidade 1.

O problema proposto objetiva obtermos:

- O pacote 1 na localidade 1.
- O pacote 2 na localidade 2.

A seguir estaremos buscando a solução para o problema a partir dos planejadores propostos inicialmente nesse capítulo.

O problema proposto será descrito em PDDL que consistirá de três partes: descrição do domínio, descrição do problema, estado inicial e estado meta.

3.8.2 Descrição do Problema Inicial em PDDL

Conforme descrito em [8, 9] Graphplan necessita de um estado *Inicial*, cuja descrição é feita atráves da linguagem PDDL. A descrição do problema acima apresentado em PDDL será encontrada, na literatura, na seguinte forma:

```
(define (problem logistic)
  (:domain logistic)
  (:objects pk1 pk2 loc1 loc2 t1)
  (:init (obj pk1)
         (obj pk2)
         (truck t1)
         (location loc1)
         (location loc2)
         (at t1  loc1)
```

```
            (at pk1 loc2)
            (at pk2 loc1))
```

Assim como temos a descrição do problema inicial também temos a descrição do nosso estado *Objetivo*. Cuja descrição em PDDL é a seguinte:

```
 (:goal (and  (at pk1 loc1)
              (at pk2 loc2))))
```

Conforme descrito em [8, 9] para que seja possível sairmos do estado Inicial e alcançarmos nosso estado Objetivo, faz-se necessário o cumprimento de algumas pré-condições, na descrição em PDDL, o exemplo acima proposto fica com a seguinte estrutura:

```
 (define (domain logistic)
 (: predicates (OBJ ?obj)
               (TRUCK ?truck)
               (LOCATION ?loc))
(:action LOAD-TRUCK
  :parameters
   (?obj
    ?truck
    ?loc)
  :precondition
   (and (OBJ ?obj) (TRUCK ?truck) (LOCATION ?loc)
   (at ?truck ?loc) (at ?obj ?loc))
  :effect
   (and (not (at ?obj ?loc)) (in ?obj ?truck)))
(:action UNLOAD-TRUCK
```

```
  :parameters
   (?obj
    ?truck
    ?loc)
  :precondition
   (and (OBJ ?obj) (TRUCK ?truck) (LOCATION ?loc)
        (at ?truck ?loc) (in ?obj ?truck))
  :effect
   (and (not (in ?obj ?truck)) (at ?obj ?loc)))
(:action DRIVE-TRUCK
  :parameters
   (?truck
    ?loc-from
    ?loc-to)
  :precondition
   (and (TRUCK ?truck) (LOCATION ?loc-from) (LOCATION ?loc-to)
   (at ?truck ?loc-from))
  :effect
   (and (not (at ?truck ?loc-from)) (at ?truck ?loc-to)))
)
```

O exemplo citado, após passar pelas fases descritas no capítulo anterior, irá nos gerar um grafo, podendo este ser ou não solução do problema. Do exemplo abordado, as camadas iniciais são representadas na figura 3.10 :

Neste exemplo, o estado objetivo é alcançado na décima primeira camada, ou seja, ao alcançarmos a décima primeira camada, nos utilizando do processo de backtraking,

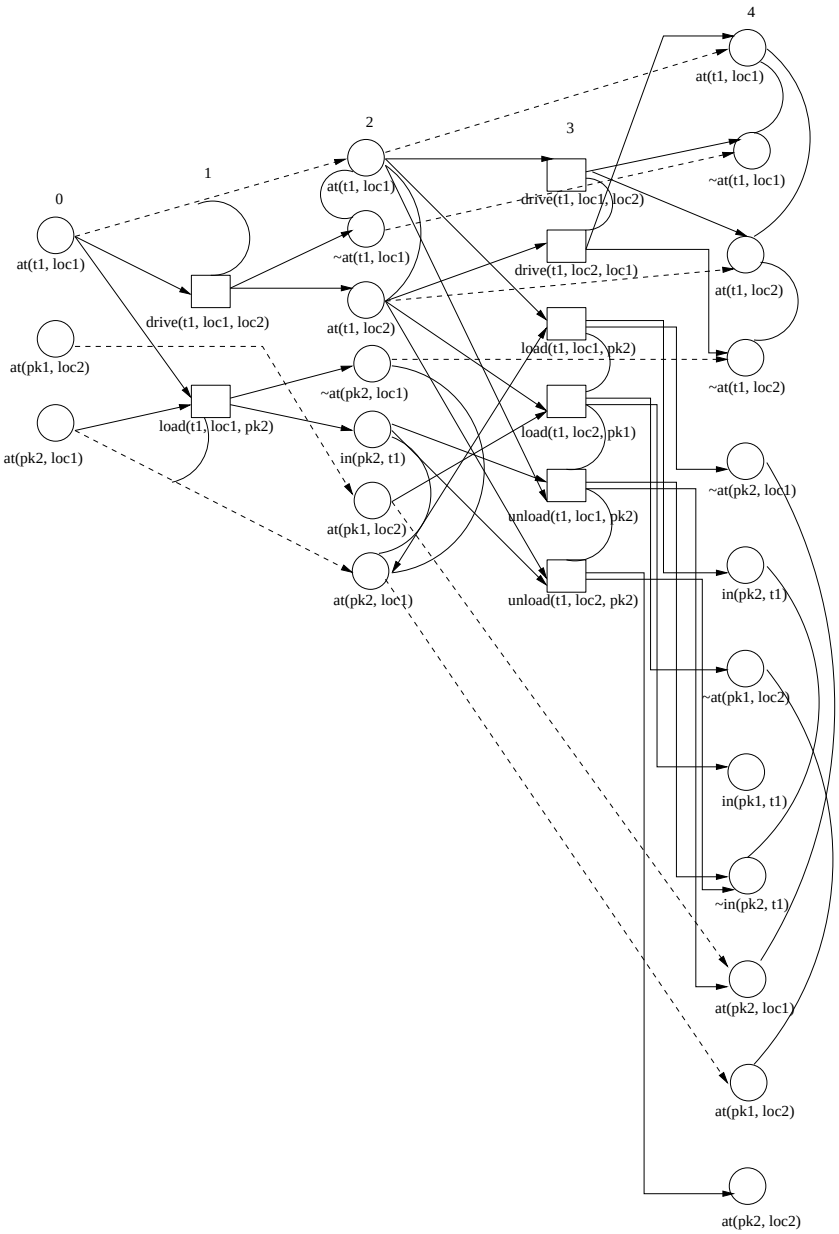

Figura 3.10: Grafo gerado pelo Graphplan até camada 5

podemos retornar à camada inicial sem nos depararmos com situações de inconsistência (*mutex*).

3.9 Solução do Problema de Logística usando Petriplan

Para que possamos encontrar a solução do problema abordado primeiramente será necessária a tradução do mesmo para uma RdP.

Nos utilizando dos conceitos da seção 3.7, podemos traduzir o Graphplan em uma Rede de Petri.

O estado inicial do problema de planejamento é representado ao inserirmos uma marca nos lugares que estão representando a camada zero do grafo, conforme mostra a camada $P0$ na figura 3.11.

A tradução gera uma RdP marcada que está apresentada na 3.11.

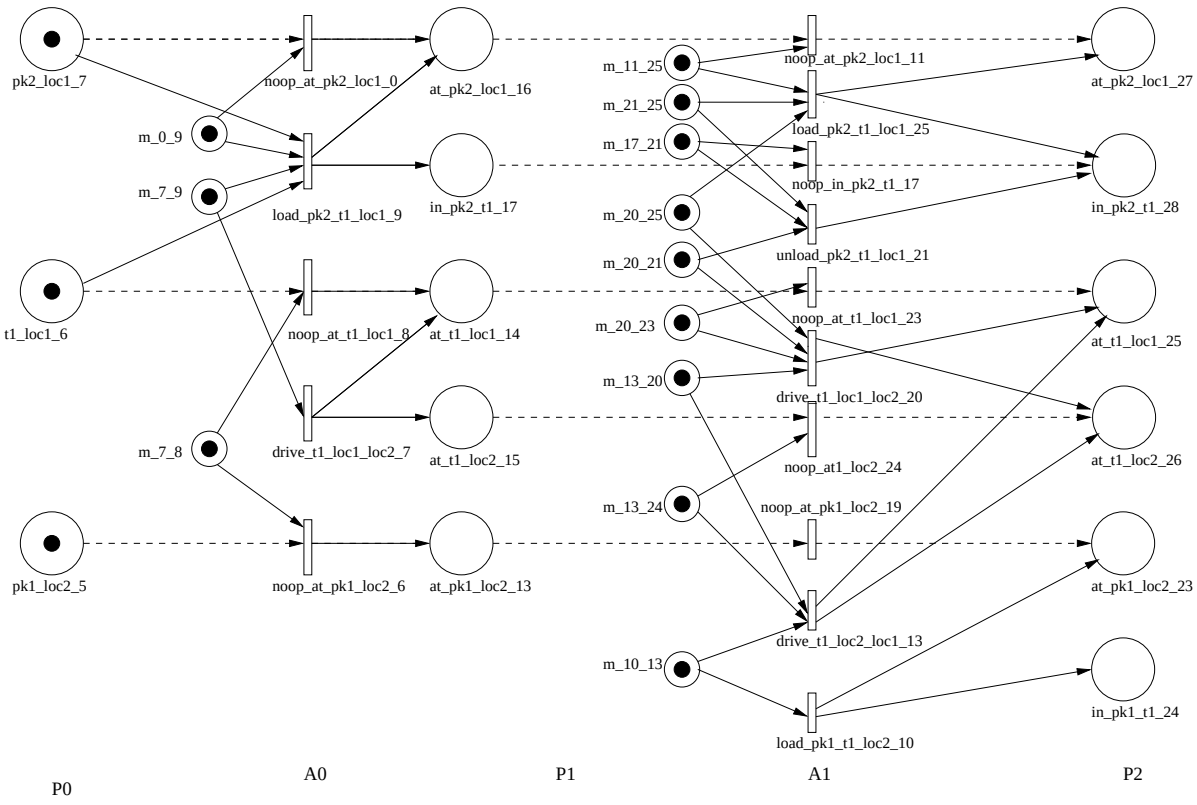

Figura 3.11: Rede de Petri gerada pelo Petriplan até camada 5

Nas camadas iniciais geradas pelo Petriplan conforme podemos observar na figura 3.11 é possível notar que a transição t_1, que representa a ação *load*, está habilitada, uma vez que os lugares que são pré-condições para seu disparo contêm marcas. O mesmo podemos observar com a transição t_2 *(drive)*.

A segunda camada da rede gerada pelo Petriplan é composta pelas transições que podem ser disparadas a partir dos lugares marcados da primeira camada. A terceira camada será composta pelos lugares gerados pelo disparo dessas transições e pelos lugares denominados *manutenção* que consistem basicamente na repetição das pré-condições existentes na primeira camada. Esse processo de expansão é executado até serem encontrados todos os literais definidos inicialmente como estado objetivo. Neste momento busca-se resolver o problema de alcançabilidade da marcação correspondente ao estado objetivo, a partir da marcação inicial. Se o estado objetivo não é alcançável, novas expansões devem ser executadas. Em caso de sucesso, é necessário identificar a seqüência de disparo de transições capaz de levar a rede do estado inicial ao estado objetivo. Essa seqüência corresponderá ao plano que resolve o problema de planejamento.

Para o problema aqui apresentado, o estado objetivo é alcançado na décima primera camada do grafo. No grafo obtido aplicamos todas as simplificações propostas em [15], e obtemos a seqüência t_1, t_2, t_3, t_4 , t_5 e t_6.

3.10 Vantagens do Petriplan

Mediante as análises dos resultados obtidos, podemos observar que a solução obtida para o problema de *Logística*, quando nos utilizamos do algoritmo Graphplan, nos fornece uma solução, cuja representação gráfica parcial nos gera a figura 3.10.

Nos baseando na solução gerada pelo Graphplan e aplicando sobre este mesmo exemplo o sistema Petriplan, podemos concluir que:

- Petriplan permite obter após as simplificações, uma representação mais compacta

do problema;

- A representação em RdP que resolve o problema de alcançabilidade do estado objetivo a partir do estado inicial se constitui em um modelo formal da solução;
- Trabalhando sobre este modelo é possível obter não somente as ações que permitem inferir relações de causalidade e de concorrência entre ações;
- Sobre este modelo podem ser agregadas informações adicionais relativas ao planejamento não-clássico, tais como dados temporais, incerteza, dados probabilísticos entre outras. O modelo com estas informações adicionais, passa a ser uma RdP de alto nível, sobre a qual, a melhor solução pode ser buscada.

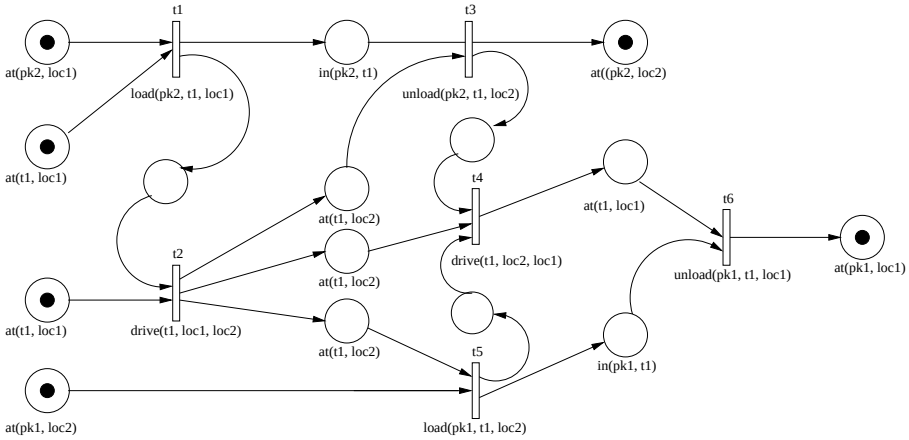

Figura 3.12: Rede de Petri solução para o Problema de Logística

Objetivando agregar as informações acima concluídas, foi proposto uma ordenação das ações na figura 3.11 para que a mesma seja executada de maneira sincronizada obedecendo uma ordenação. Aplicarmos essa relação de ordem significa dizermos que a transição t_2 só estará habilitada após o disparo de t_1, ou seja, a ação $at(t_1, loc_1)$ nunca poderá habilitar t_2 antes de ter habilitado t_1. Outra ordenação que podemos realizar

no exemplo proposto é na ação $at(t_1,\ loc_2)$, na qual as primeiras transições a serem habilitadas devem ser t_3 e t_5, para então, t_4 ser habilitada.

Tomando a figura 3.11 e aplicando a relação de ordem, ou seja, aplicando um sincronismo na execução da ações, teremos como resultado a RdP abaixo representada.

Capítulo 4

Planejamento Temporal

Neste capítulo será abordado o desenvolvimento do *planejamento temporal*. Será mostrado que o planejamento temporal é um modelo extendido, que tem suas raízes no planejamento clássico, agregando nos domínios, além da necessidade da enumeração das ações, a ordenação das mesmas. Serão apresentados resumidamente alguns planejadores temporais citados na literatura e a linguagem de descrição PDDL+.

4.1 PDDL+

Ao ser traçada a linha do tempo em termos de evolução dos planejadores automáticos, nota-se a evolução da expressividade dos mesmos.

Em paralelo à evolução dos planejadores ocorre a evolução das linguagens de representação de problemas de planejamento, como, por exemplo, a linguagem PDDL.

Mediante a necessidade de avaliação dos Planejadores na solução de problemas reais durante o IPC de 2002 foi proposta uma extensão da PDDL [19] denominada PDDL+.

A PDDL+ foi subdividida em 5 níveis, sendo que os níveis compreendidos 1 e 4 são chamados coletivamente de PDDL2.1. A decisão por dividir o PDDL+ para tratar de características específicas, incorporadas aos domínios, ao longo da evolução

das pesquisas nesta área. Resumidamente, pode-se apresentar os 5 níveis da seguinte maneira:

- **Nível 1** ⇒ Trata-se do PDDL em sua versão original;
- **Nível 2** ⇒ Adicionam-se ao PDDL original variáveis numéricas e a capacidade de realizar atualizações sobre os valores;
- **Nível 3** ⇒ Adição do tratamento sobre ações com duração. As ações e efeitos passam a acontecer em um tempo "t", pré-determinado pelo agente. Nesse nível são tratadas ações cujo tempo de execução é dado no modo discreto;
- **Nível 4** ⇒ Adição do tratamento de ações com duração contínua. Por exemplo, ao executarmos a ação de dirigir um caminhão, precisamos ter como recurso o combustível, o efeito de consumi-lo gera um efeito contínuo no decorrer da execução, ou seja, a cada instante o nível do combustível terá um valor associado. A literatura comumente apresenta de forma unificada os níveis 3 e 4;
- **Nível 5** ⇒ Capacidade de representação de eventos exógenos [1], cujos domínios apresentem variáveis temporais discretas e contínuas. Este nível também é definido como *"real-time"*, sendo o mesmo uma proposta alternativa para o modelo temporal, porém não usado nas competições.

Ao longo desse trabalho se opera com variáveis temporais cujas características são melhor comportadas pelo nível 3. Um detalhamento desse nível pode ser observado no anexo A deste trabalho.

4.2 Planejadores Temporais

Com a retomada das pesquisas na área de Planejamento em Inteligência Artificial, a comunidade científica focou-se no desenvolvimento de planejadores capazes de encon-

[1] Um evento exógeno é um acontecimento externo ao sistema, que afeta seu comportamento.

trar bons resultado quando aplicados a problemas reais. Para que isso fosse possível, iniciaram-se pesquisas sobre a inserção de diferentes tipos de informação nos problemas de planejamento. Entre estas informações a análise temporal inerente ao problema.

O principal diferencial entre Planejamento *Clássico* e *Temporal* é o fato de ser necessário um conjunto com maior completude no que se refere às restrições ao operarmos ações que envolvem características temporais. No Planejamento Temporal o aumento do grau de complexidade [13] acontece em virtude de ser preciso não apenas escolher uma seqüência de ações, mas a seqüência que resolve o problema respeitando os critérios temporais a ela inerentes. Dessa forma, a busca não se limita a encontrar uma seqüência que seja a solução para o problema, mas em geral, aquela que possa ser executada no menor intervalo de tempo.

Sob a ótica de planejamento, pode-se dizer que entre os objetivos do Planejamento Temporal não está somente minimizar o número de passos ou ações do mesmo, mas a possibilidade encontrar uma seqüência de ações cujo tempo de duração seja mínimo, podendo esta ser obtida ou não.

Segundo Garrido em [53], estruturalmente a seqüência de procedimentos no Planejamento Temporal é representada na figura 4.1.

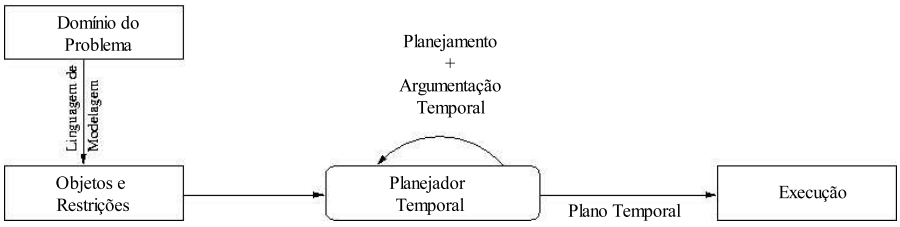

Figura 4.1: Modelo Estrutural de um Planejador Temporal

Analisando a figura 4.1, observa-se que a partir do momento que se tem definido o domínio do problema, aplica-se uma linguagem de modelagem, de acordo com a métrica

adotada pelo planejador e que será usado na seqüência da solução do problema. Feita esta modelagem, efetua-se uma análise das restrições inerentes ao problema. Após definidas as restrições do problema, executa-se o domínio em um planejador que contemple o tratamento de argumentos temporais. Nessa etapa, alguns planejadores fazem uso de funções heurísticas para selecionar as ações que irão gerar o plano temporal.

A seguir serão abordados alguns planejadores e suas principais características.

4.2.1 Planejador TGP

Na literatura sobre planejadores temporais observa-se que é vasto o número de planejadores que foram propostos buscando tratar esta extensão do planejamento clássico. Um dos primeiros planejadores que buscou efetuar o tratamento de características temporais foi o TGP (Temporal Graphplan [52]).

TGP foi desenvolvido na Universidade de Washington, sendo um dos primeiros planejadores a propor com êxito a inserção de tempo nas ações em um planejador derivado do Graphplan. TGP tem como base a construção simplificada de um grafo de planejamento e um cálculo rigoroso das relações de exclusão *(mutex)* existentes entre ações com duração distinta. Das características relevantes de TGP, podemos destacar:

- Generalização do grafo de planos substituindo as camadas por intervalos temporais.
- Incremento dos argumentos de exclusão mútua em um grafo de plano temporal.

Quanto aos passos que TGP executa, primeiramente ele irá generalizar a representação do grafo de planos, para então trabalhar o tempo, ou seja, o critério para encontrar o melhor plano é a duração das ações.

No novo grafo as ações e proposições aparecem uma única vez porém indicando a data em que ocorrem. O mesmo ocorre com os mutex, porém a data registrada é aquela

em que o mutex desaparece. Devido ao fato de não armazenar *"noop"*. [2] O plano gerado por TGP registra um espaço de busca reduzido, resultado da monotonicidade apresentada.

Como conseqüência à inclusão de variáveis temporais têm-se novas relações de exclusão. As novas relações de exclusão são classificadas em *mutex eternos (emutex)* e *mutex condicional (cmutex)*, representando as relações de exclusão que uma vez inclusas no grafo se farão presentes até que a expansão seja finalizada ou ainda relações de exclusão que ao longo da expansão são eliminadas respectivamente.

Na segunda etapa, TGP faz o tratamento de ações com duração distinta, desfazendo assim a simetria existente no grafo de planos do Graphplan e exigindo um cálculo de mutex mais minucioso.

Por fazer um tratamento completo e otimizado das ações envolvidas no problema e por utilizar-se de um modelo conservativo TGP agrega maior precisão quando aplicado a problemas reais.

4.2.2 Planejador LPGP

Dentre os planejadores que contemplam o tratamento de ações temporais há o planejador LPGP (Linear Programing Graphplan [22, 18]) surgido de uma das extensões da linguagem PDDL, mais precisamente da PDDL2.1.

Comparado a outros planejadores temporais, por exemplo ao TGP (Temporal Graphplan), sua representação do tempo faz-se unicamente através das camadas de proposições. Tal representação é obtida através da divisão das ações instantâneas e seu tempo de duração, uma com condições e efeitos do início da ação original e a outra com condições e efeitos do fim da ação.

[2] No Operation - Ação de manutenção que tem como objetivo manter em uma camada x as proposições de uma camada x - *2*. Como exemplo podemos imaginar uma ação cujo efeito é a própria pré-condição.

Buscando garantir as condições invariantes de uma ação, faz-se a inclusão de um novo tipo de ação. Esta nova ação irá agregar condições para que as mesmas se mantenham fixas durante a execução da ação inicial. A esse tipo de ação dá-se o nome de *ação invariante*.

As ações invariantes deverão estar presentes entre os níveis de ação em que se executam, respectivamente:

- Ações instantâneas de início;
- Fim da ação original com duração.

Da mesma forma que TGP, o LPGP usa um grafo de planejamento compacto. No entanto, LPGP faz uso das ações de *noop* para prolongar a presença das proposições, exceto para proposições que atuam como condições invariantes conduzidas explicitamente por ações invariantes.

Em relação ao grafo, a única mudança que ocorre no LPGP quando comparado ao grafo gerado pelo TGP se dá pela propagação das ações invariantes, uma vez que estas ações não requerem tempo, não necessitando de cálculo de mutex. Dessa forma se faz a busca da seguinte maneira:

- Toda vez que uma ação instantânea de fim acontece, a ação que a originou finda;
- Quando uma ação instantânea de início acontece, a ação original também deve acontecer.

Em LPGP faz-se necessário a inclusão de restrições que estabeleçam a distância temporal entre as camadas das ações de início e de fim, representando a duração de ocorrência dessa ação. Para fazer essa inclusão de restrição o LPGP faz uso do método Simplex.

Se analisado sob a ótica de desvantagens, em relação ao TGP, o LPGP não garante um plano de duração ótimo, pois minimiza o número de mudanças de estados do problema, o que pode não ser coincidente com o plano de duração mínima.

Por exemplo, se na solução de um mesmo problema, uma das soluções tiver muitas ações curtas e outra tiver uma única ação, porém, com duração longa, LPGP irá apresentar um plano com uma única ação. Isto é, LPGP irá apresentar como solução um plano que contenha a menor duração no universo do conjunto de planos que apresente o menor número de passos.

Outro fator que acarreta desvantagens ao LPGP é o fato de promover um aumento do número de ações, uma vez que uma ação gera sempre duas ações instantâneas e uma terceira se esta contar com condicionais invariantes.

4.2.3 Planejador Sapa

Outro planejador que faz o tratamento de características temporais é o SAPA (A domain-independent Heuristic Metric Temporal Planner) [14]. SAPA é baseado em heurística e foi proposto para suprir as limitações temporais que não eram tratadas pelo HSP.

SAPA faz uma extensão considerável do modelo de ações STRIPS. Tal extensão está na possibilidade de medir a duração das ações, o consumo de recursos, assim como as pré-condições e efeitos das ações poderam ocorrer em qualquer momento da execução das ações.

Outra característica do SAPA é que, a partir do momento em que os objetivos do problema estiverem definidos, tem-se um prazo para que sejam encontrados *(deadline goals)*.

SAPA também é rigoroso quanto à ordem de execução das ações, não permitindo que duas ações que se utilizem do mesmo recurso sejam executadas simultaneamente.

Assim como TGP, SAPA faz a construção de um *grafo de planos relaxado*. Dessa

forma, não se faz o cálculo dos mutex, reduzindo ao mínimo o custo de construção do grafo. Posteriormente, este grafo é utilizado para construir um plano temporal relaxado que atenda aos objetivos definidos [53].

Sabe-se que a utilização de um plano relaxado atua como embazamento heurístico, pois permite calcular as distâncias dos objetivos e estados de busca.

O SAPA também fornece informações referentes a métricas relacionadas ao tempo de duração de um plano, ao custo e ao tempo disponível para o cumprimento do prazo estipulado no objetivo. Tais informações permitirão definir heurísticas aceitáveis e não aceitáveis.

Em termos de contribuição pode-se observar que o SAPA oferece alguns ganhos, quando comparado ao TGP, pois o mesmo é capaz de resolver problemas cuja complexidade é maior e em menor tempo, bem como permite definir recursos métricos. Porém, TGP garante a *otimalidade* [3] de todos os planos encontrados, enquanto que o SAPA só pode fazer isso quando a heurística for aceitável, ou seja, quando ocorre uma redução considerável em seus resultados.

4.2.4 Planejador TP4

Dentre os planejadores temporais que se utilizam de heurística para encontrar soluções em problemas nos quais se faz necessária a análise temporal está o TP4 (Heuristc Planning With Time and Resources) [23].

TP4 foi desenvolvido na Universidade Simón Bolivar e faz uso do mesmo modelo conservativo de ações do TGP, porém com recursos consumíveis e renováveis. Sua principal característica é a capacidade de administrar recursos, otimizando a duração global do plano.

Outra característica de TP4 é o fato de modelar ações de manutenção, ou seja, um recurso pode ser utilizado por um longo tempo, apenas informando que as ações de

[3] Otimalidade: palavra adaptada para a língua portuguesa que expressa a condição de ótimo.

manutenção (*noop*) consomem recursos.

Em TP4 é difícil encontrar um plano ótimo para problemas complexos, pois sua aplicação se restringe a ações não conservativas[4], o que o torna um planejador limitado.

Ao analizar o TP4 pode-se observar que o mesmo apresenta um rendimento ligeiramente superior ao TGP e sua eficiência é inferior ao SAPA, pois seus planos apresentam qualidade inferior.

4.2.5 Planejador TPSYS

Outro planejador temporal que merece destaque na literatura sobre planejamento temporal é o TPSYS (Temporal Planning System) [53, 3, 4].

TPSYS foi desenvolvido pela Universidade Politécnica de Valência e surgiu da combinação das idéias de Graphplan e TGP, ou seja, seu modelo vem do planejamento clássico, porém, faz a inclusão e administração do tempo.

TPSYS, em sua primeira versão [3, 4], foi proposto prevendo solução de problemas para *Modelos Conservativos*, enquanto em sua segunda versão TPSYSv2 [53], propõe-se a solucionar problemas para *Modelos não Conservativos*.

De uma forma simplicada, pode-se dizer que planejadores de *Modelos Conservativos* são aqueles que ao constatar um conflito entre duas ações sucessivas, a segunda ação deverá esperar a primeira finalizar para que então seja inicializada a sua execução.

Já em planejadores de *Modelos não Conservativos*, quando um conflito entre ações sucessivas é detectado, faz-se necessário o cálculo do ponto onde ocorre o início da execução da segunda ação. Se o conflito verificado for do tipo pré-condição e efeito de uma ação, pode-se efetuar o cálculo do momento exato em que são necessários e que são produzidos. Com isso pode-se efetuar o cálculo mais eficiente em termos de conflito, isso faz com que se possa adiantar o momento inicial da execução da segunda ação,

[4]Diz-se que um modelo é não conservativo quando dada uma ação x com duração $dur(x) \in R^+$ as condições *SCond(x), Inv(x) e ECond(x)* são condições de início, meio e fim de x respectivamente.

quando sua execução não afetar o resultado da primeira.

TPSYS em suas duas versões utiliza-se para especificar um problema de planejamento de uma tupla $\langle S_i, A, S_f, D_{max} \rangle$, sendo S_i e S_f o estado inicial e final respectivamente. $\boldsymbol{A}$ o conjunto de ações, com duração fixa e positiva e D_{mx} a máxima duração do plano sendo este opcional e não estritamente necessário. A segunda versão do TPSYS também agrega em sua estrutura o conceito de *ação durativa*.

Conforme já foi mencionado TPSYS faz a utilização do modelo conservativo de ações, ou seja, dada uma ação *"x"* a duração *dur(x)* $\in R^+$, as pre-condições *Pre(x)* devem ser cumpridas, ou seja, devem estar presentes no início da ação e manter-se até o final da mesma, exceto em *Del(x)* (sendo Del(x) efeitos negativos). Já em TPSYSv2, utiliza-se um modelo não conservativo de ações, ou seja, dada uma ação x com duração *dur(x)* $\in R^+$, as condições *SCond(x), Inv(x) e ECond(x)* são condições de início, meio e fim de x respectivamente.

Nos planejadores cuja ações apresentam duração faz-se necessária a análise do cumprimento das pré-condições, tanto para início como para o término das ações (conforme pode ser observado no anexo A). Isto é, para que uma ação aconteça, se fazem necessárias não apenas as condições para que ela se inicialize, mas sim que os recursos necessários para sua execução estejam disponíveis durante todo intervalo de tempo de sua execução, bem como para que esta se finalize.

Diante dessas características observa-se que para administrar tempo e recursos na execução das ações depara-se com situações críticas quanto ao gerenciamento das garantias de início e término de uma ação. O detalhamento dessas garantias podem ser encontradas em [53].

Capítulo 5

Planejamento Temporal Baseado em Grafo de Planos e Redes de Petri Temporais

Neste capítulo são propostos dois métodos de planejamento temporal. O primeiro método consiste em uma extensão do grafo de planos em que as camadas, ao invés de corresponderem à execução do conjunto de ações que podem ser aplicadas ao conjunto de proposições válidas, são geradas a cada evento de início ou término de execução de todas as ações aplicáveis e serão rotuladas com a data do evento correspondente. O segundo método consiste também de uma extensão, mas da rede de planos, anexando um rótulo temporal a cada transição que corresponde à execução de uma ação do plano. Um mesmo problema exemplo de planejamento em inteligência artificial será utilizado para ilustrar os dois métodos propostos.

5.1 Uma Nova Proposta para a Inserção de Tempo no Grafo de Planos

O novo modelo de planejamento temporal aqui proposto tem como eixo central o planejador Graphplan original [6], mantendo as características estruturais já descritas na seção 3.1, além de incluir novas estruturas exigidas para a avaliação temporal. A linguagem de descrição utilizada para descrever o problema e o domínio será a PDDL.

De forma semelhante ao Grafo de Planos, a execução do método consiste nas etapas de inclusão de uma nova camada de ações (referente ao início ou à conclusão de ações em uma mesma data, relativa ao início do plano) e extração de uma solução, repetidamente até que um plano seja encontrado ou que não exista solução para o problema.

5.1.1 Grafo de Planos Temporal

Da mesma maneira que o Graphplan descrito na seção 3.1, o novo modelo é composto por dois tipos de nós: *Nó Proposição* e *Nó Ação*. Devido ao tratamento dos eventos temporais no modelo, as ações serão subdivididas em *Início de Ação*, *Manutenção de Ação* e *Finalização de Ação*. As ações de início e finalização estarão presentes nas camadas que correspondem, respectivamente, às datas de início e finalização de uma determinada ocorrência da ação. As ações de manutenção da ação estarão presentes caso existam camadas entre a data de início e a data de finalização da ação (correspondentes a eventos de início e finalização de outras ações do plano).

Considere como exemplo o problema proposto por Weld em [57]. O problema consiste em preparar um jantar surpresa. Os objetivos são: embrulhar um presente, preparar um jantar e remover o lixo. Para isso são consideradas quatro ações: cozinhar, embrulhar, carregar e triturar o lixo. A ação *embrulhar*, tem como pré-condição *silêncio* e sua execução tem como efeito *presente*. A ação *triturar o lixo*, tem como como efeito *não ter lixo* e *não ter silêncio*. A ação *carregar o lixo* também tem como

efeito *não ter lixo* e *não ter mãos limpas*. A ação *cozinhar* requer *mãos limpas* e irá gerar o efeito *jantar*. O estado inicial é definido por três proposições: mãos limpas, lixo e silêncio.

Para este problema, uma das possíveis soluções para o grafo de planos *atemporal* é representada pela figura 5.1. Conforme pode ser observado na figura, em um grafo de planos de um domínio atemporal as ações são consideradas instantâneas, ou seja, se suas pré-condições forem atendidas, a ação é executada, apresententando seus efeitos imediatamente na camada seguinte.

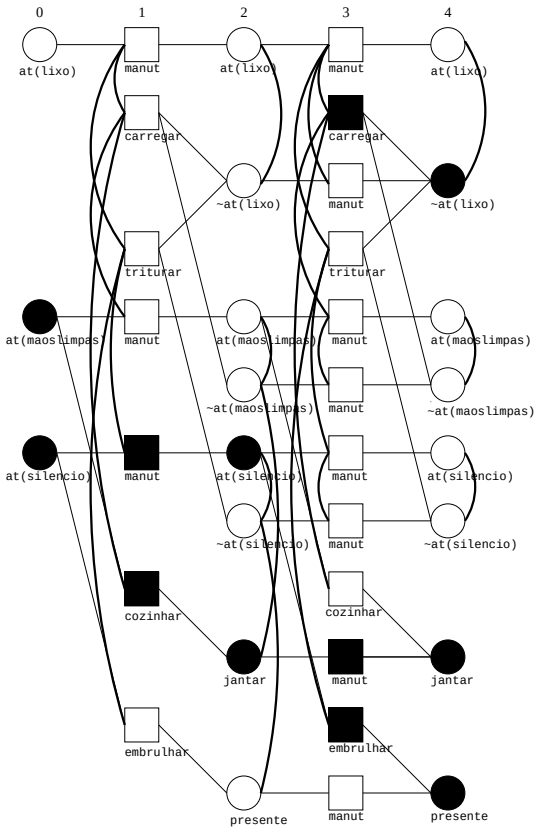

Figura 5.1: Grafo de Planos atemporal para o domínio do jantar, com uma das possíveis soluções.

No modelo de Grafo de Planos a Eventos, que está sendo proposto, a construção

do grafo, na sua estrutura geral, é efetivada de maneira semelhante à descrita na seção 3.2. Uma nova camada é gerada para retratar a ocorrência de um evento de início ou de término de ação e para caracterizar o intervalo (ou o conjunto deles) correspondente à duração da ação. A cada camada é associado um rótulo que corresponde a data de ocorrência dos eventos nela contidos. Diferentes ações que se iniciam em uma mesma data terão seus eventos de início fazendo parte de uma mesma camada. O mesmo ocorre com o término de várias ações em uma mesma data.

A separação das ações em eventos de início e término implica uma expansão adicional do Grafo de Planos a Eventos. Uma vez que no Grafo de Planos têm-se camadas intercaladas de ações e proposições, o término de uma ação em uma determinada data pode gerar como efeito uma proposição que é pré-condição de uma ação que pode ser iniciada nessa mesma data. Ou seja, três camadas, uma de finalização de ação, outra de proposição e uma terceira de início de ação, todas com a mesma data. Para distinguir temporalmente estas três camadas utilizaremos a seguinte convenção: as camadas de início de ações terão como rótulo *(data início evento*$^+$, as camadas de finalização como *data finalização evento*$^-$*)* e as camadas de proposições situadas entre as duas anteriores como *data do evento*. As camadas correspondentes à execução das ações não precisam ser rotuladas, pois correspondem ao intervalo temporal, aberto, entre as datas dos eventos anterior e posterior.

É importante também notar que as pré-condições ficam vinculadas ao evento de início da ação, enquanto que os efeitos somente são gerados após o evento de finalização da ação. Esta característica da abordagem proposta permite que sejam consideradas as especificações mais recentes da linguagem PDDL, onde também podem ser especificados efeitos de início de ação, pré-condições de término de ação, assim como proposições que devem se manter válidas durante a execução da ação.

Tomemos o mesmo problema proposto por [57], porém com os intervalos de execução para cada ação definidos temporalmente com:

- 6 unidades de tempo para a ação cozinhar;
- 3 unidades de tempo para a ação carregar;
- 3 unidades de tempo para a ação triturar e
- 2 unidades de tempo para a ação embrulhar.

Para esses intervalos de tempo o Grafo de Planos a Eventos gerado é o apresentado na figura 5.2.

5.1.2 Expansão do Grafo Temporal

Comparando os dois grafos, atemporal (figura 5.1) e temporal (5.2), podemos observar que a primeira ocorrência da ação *embrulhar* ocupa apenas uma camada do grafo atemporal e três camadas de ações no grafo a eventos. A primeira camada é rotulada com a data 0 e contém as proposições do estado inicial do problema. As ações que podem ser aplicadas no estado inicial têm seus eventos de início presentes na segunda camada, rotulada com 0^+. Na terceira camada estão representadas as proposições mantidas válidas e uma categoria nova de "proposições"que descrevem a execução das ações. Quatro ações foram iniciadas na segunda camada (0^+). Destas, a primeira a ser finalizada é *embrulhar*, com duração de 2 unidades de tempo. Isto faz com que a quarta camada seja rotulada com a data 2^-) e contenha o evento correspondente à finalização da ação *embrulhar*. Como esta é a única ação finalizada nesta data, as demais ações que estão em execução são mantidas com eventos de "manutenção de ação". A quinta camada, datada de [2] contém os efeitos da finalização da ação *embrulhar*, ou seja, a proposição *presente* aparece pela primeira vez no Grafo de Planos a Eventos duas unidades de tempo após o início do plano. Esta já é uma informação temporal quantitativa não disponível no grafo de planos, já que a primeira camada deste contém um conjunto de ações com diferentes durações, o que induz a considerar válidas as proposições da

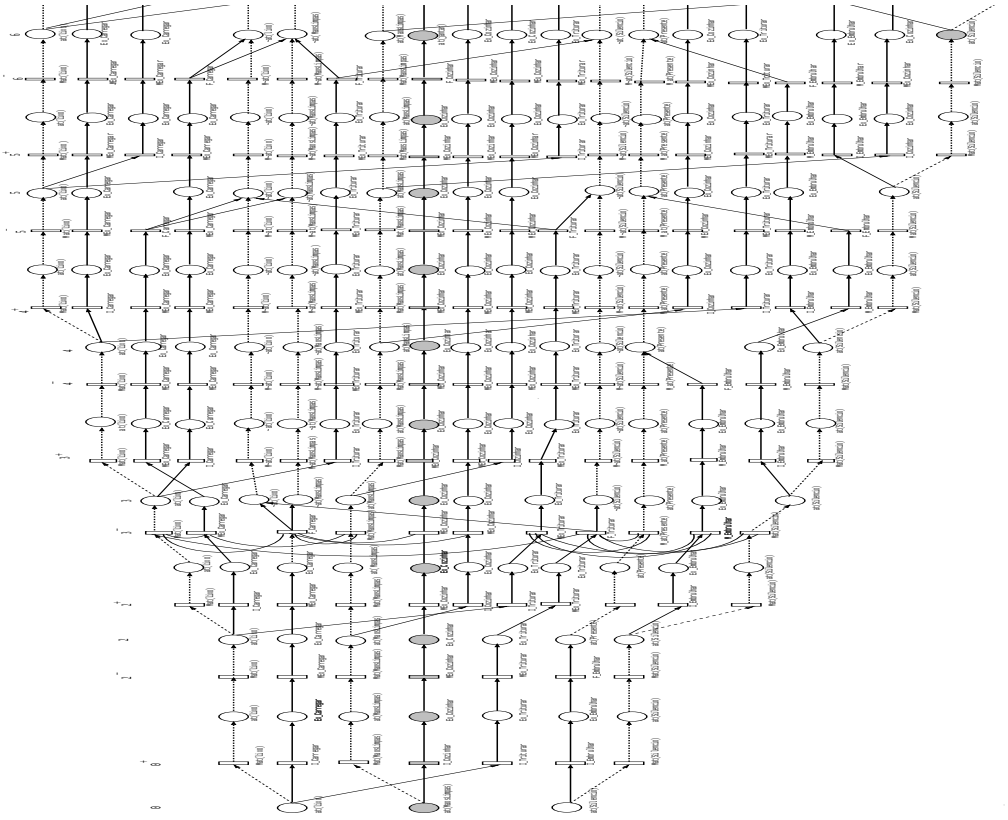

Figura 5.2: Grafo de Planos temporal para o domínio do jantar com uma das possíveis soluções.

camada dois somente ao término da ação com duração mais longa. Na camada seguinte (a sexta, com rótulo temporal (2^+), podemos reaplicar a ação *embrulhar*, uma vez que suas pré-condições estão presentes. Ou seja, a segunda ocorrência, no grafo de planos a eventos, da ação *embrulhar*, tem seu início sobreposto à execução de outras ações do grafo. Como conseqüência, podemos obter planos que são menores que aqueles obtidos com o grafo de planos atemporal, já que as ações podem ter seu início antecipado.

A análise de exclusão mútua entre ações e proposições mantém-se da mesma forma que no Grafo de Planos original. Ela é realizada, entretanto, apenas nos eventos de finalização das ações, quando os efeitos de cada ação são gerados. Devem ser analisadas as inconsistências existentes entre cada evento de finalização de ação e os eventos de início e de manutenção das demais ações. Mantém-se, como no Grafo de Planos, a análise de inconsistência entre proposições e a propagação de inconsistências. No Grafo de Planos a Eventos da figura 5.2 foram apresentados apenas uma parte do conjunto de exclusões mútuas do problema, somente para ilustrar sua aplicação e tornar legível o grafo.

5.1.3 Extração da solução do Grafo de Planos Temporal

Da mesma maneira que no grafo de planos original a expansão é feita até que todos os literais definidos como estado meta estejam presentes na última camada. O processo de busca de solução segue o mesmo processo definido para o Grafo de Planos original. Caso seja encontrada uma solução, basta percorrer o grafo, da primeira à última camada, relacionando as ações nele presentes e suas datas de início e término, que correspondem ao rótulo da camada da qual fazem parte os eventos correspondentes.

O critério de parada na geração de novas camadas, quando o problema não possui solução, é mais complexo que o existente para o grafo de planos original. Isto se deve à existência de ações sobrepostas, o que dificulta a definição de camadas de ações independentes. A solução proposta consiste em identificar, no Grafo de Planos a Eventos, as

camadas de proposições existentes no Grafo de Planos original, que quando comparadas permitem decidir sobre a não existência de solução para o problema.

A primeira camada é comum a ambos os grafos, temporal e atemporal. Todas as ações iniciadas na segunda camada do grafo de planos a eventos correspondem às ações da segunda camada do grafo de planos original. A camada com data correspondente a maior duração entre as ações iniciadas contém as proposições equivalentes à terceira camada do grafo atemporal. O conjunto de ações iniciadas na camada seguinte, a partir das proposições equivalentes, será o mesmo do grafo de planos atemporal e a camada em que todas estarão concluídas será a nova camada de proposições a ser utilizada na decisão sobre a não existência de solução.

Para exemplificar, tomemos o grafo da figura 5.2. Aplicando-se o mesmo processo de extração do Grafo de Planos original obtém-se o conjunto de ações da solução em destaque nesta mesma figura. Este conjunto é apresentado de forma simplificada na figura 5.3, que consiste basicamente na seleção das ações do plano com suas datas de início e finalização.

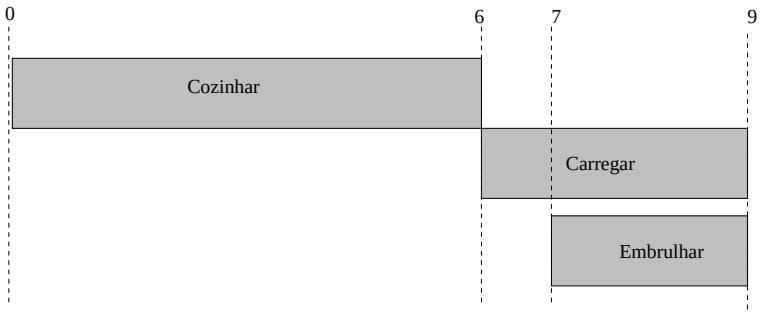

Figura 5.3: Possível Solução do Grafo de Planos a Eventos para o Domínio do Jantar.

5.2 Petriplan Temporal

O planejador Petriplan, apresentado na seção 3.5, consiste na tradução do Grafo de Planos em uma rede de Petri acíclica. A busca da solução é efetuada utilizando-se de métodos de alcançabilidade existentes para as rede de Petri ou a partir da equação fundamental das redes de Petri resolvendo-se um problema de programação inteira.

Aplicando-se o algoritmo Petriplan, o problema do jantar descrito na seção 5.1 retorna a seguinte RdP:

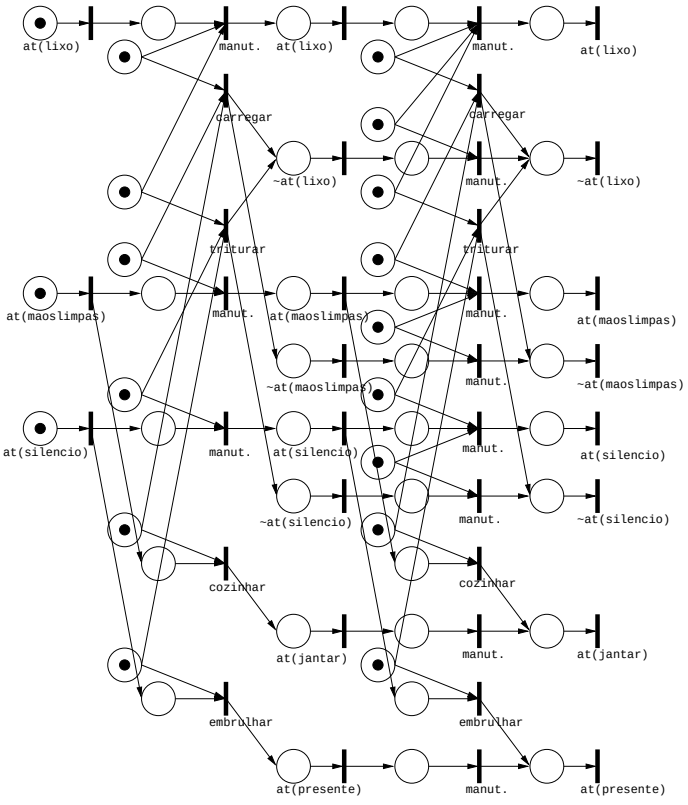

Figura 5.4: Rede de Petri gerada pelo Petriplan em sua primeira versão para o problema do jantar.

Uma primeira abordagem para extender o método Petriplan para planejamento temporal consiste em adotar uma extensão de redes de Petri que seja capaz de associar

às transições um intervalo de tempo correspondente à duração da ação. O modelo de Redes de Petri T-temporizadas [47] é bastante suficiente para as necessidades do planejamento temporal. Desta forma, basta associar a duração do disparo das transições à sub-rede correspondente à solução obtida e efetuar a análise temporal.

Para a rede da figura 5.4, temos o seguinte plano solução:

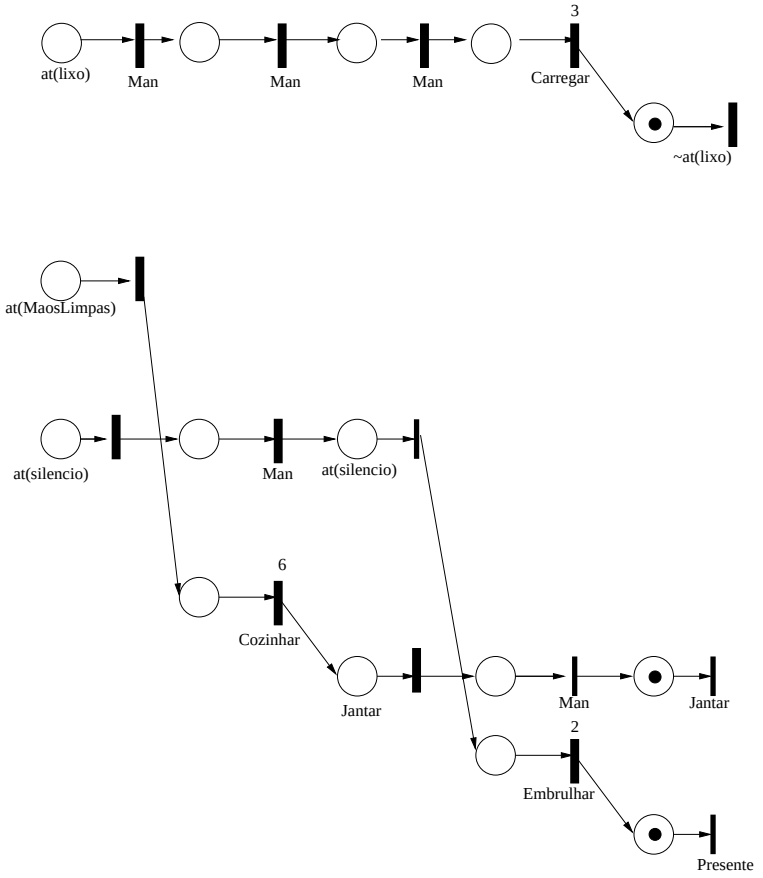

Figura 5.5: Marcação Final da Rede de Petri gerada pelo Petriplan em sua primeira versão para o problema do jantar.

Note que a Rede de Petri gerada para o problema do jantar pelo Petriplan original, obtém a marcação final para o problema do jantar conforme descreve a figura 5.5. Associando a essa marcação, as mesmas durações das ações utilizadas no exemplo anterior,

obtém-se uma rede com custo temporal equivalente ao Grafo de Planos a Eventos, ou seja, 9 unidades de tempo.

5.3 Rede de Planos Temporal

A Rede de Planos [51] é uma nova abordagem para o algoritmo Petriplan, onde a Rede de Petri associada ao problema de planejamento não é mais obtida a partir do grafo de planos, mas diretamente da descrição do problema em PDDL. A Rede de Planos consiste de transições que representam ações, de lugares que representam proposições e de uma estrutura de controle que trata as inconsistências entre as ações. É também uma Rede de Petri acíclica, tal como a abordagem Petriplan, o que implica que as diferentes instâncias de ações e proposições são copiadas, evitanto ciclos na rede.

O mesmo problema do jantar proposto na seção 5.1 é representado em uma rede de planos conforme mostrado nas figuras a seguir. O estado inicial da rede de planos é definido por *at(mãoslimpas)*, *at(lixo)* e *at(silêncio)*, mostrado na figura 5.6.

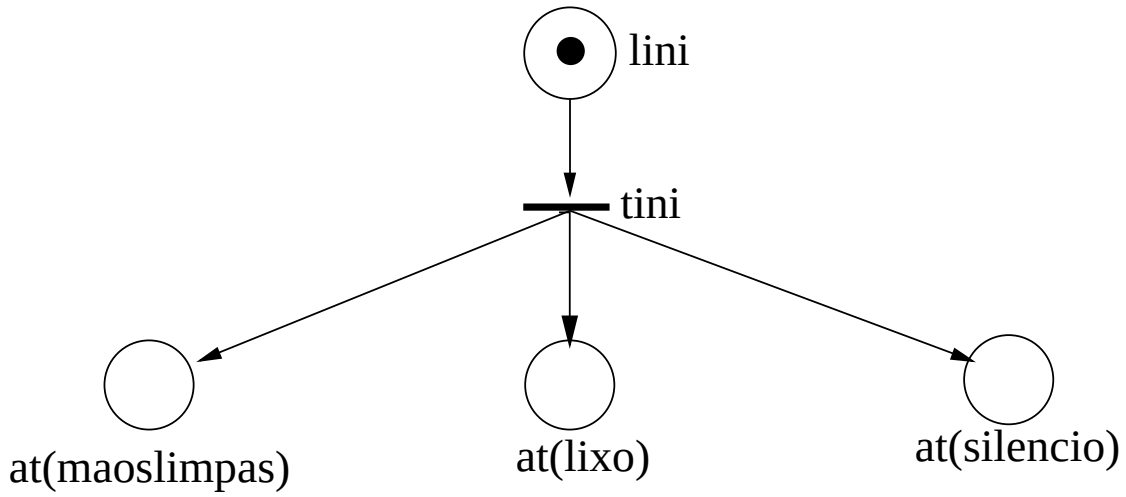

Figura 5.6: Estado inicial para o problema do jantar, gerado pela Rede de Planos.

Na fase seguinte, com a inclusão das ações, seus efeitos e da estrutura de controle,

obtém-se a Rede de Planos da figura 5.7.

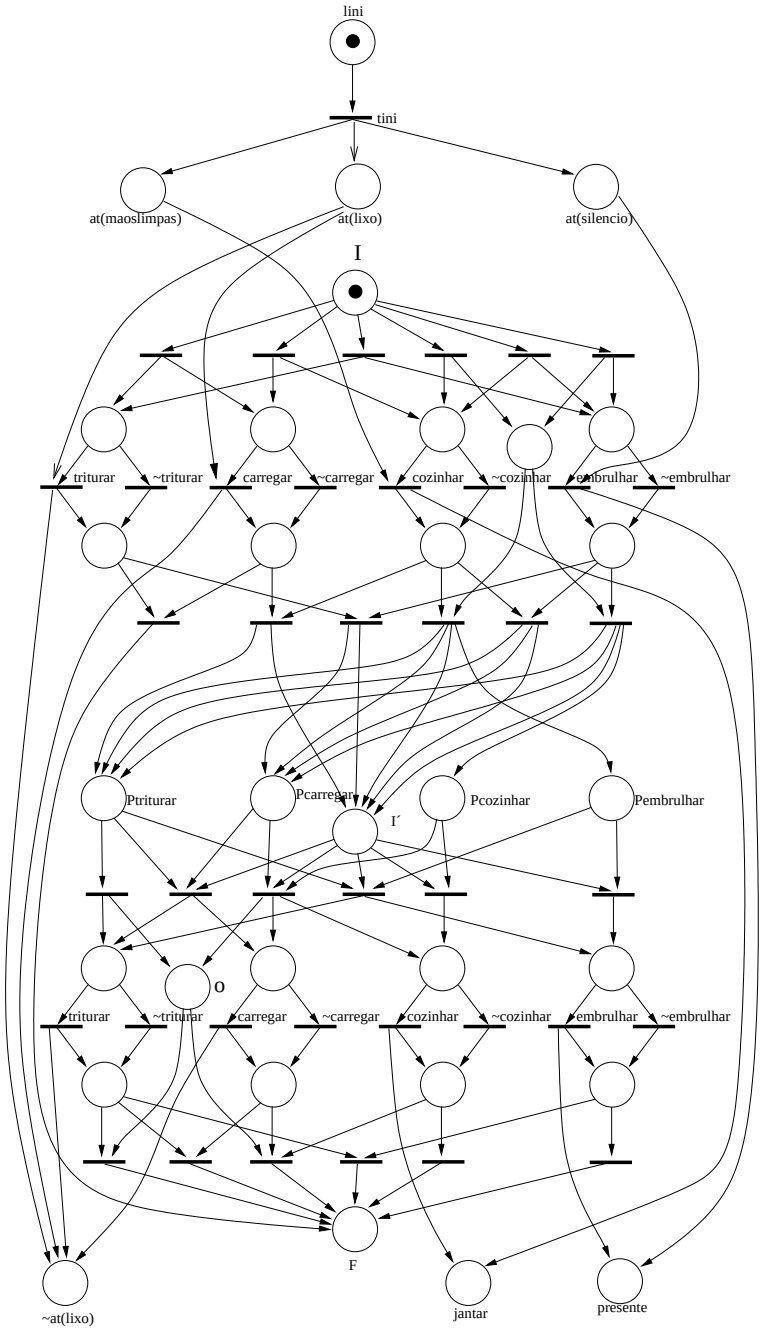

Figura 5.7: Rede de planos após inclusão da segunda camada para o problema do jantar.

É importante notar que com a inclusão desta nova camada estão presentes na rede as proposições referentes ao objetivo proposto inicialmente para o problema e que qualquer algoritmo de alcançabilidade é capaz de identificar uma seqüência de disparos de transições que leva do estado inicial ao estado meta.

Para a Rede de Planos representada na figura 5.7, uma das possíveis sub-rede que nos leva ao estado objetivo é representada pela figura 5.8.

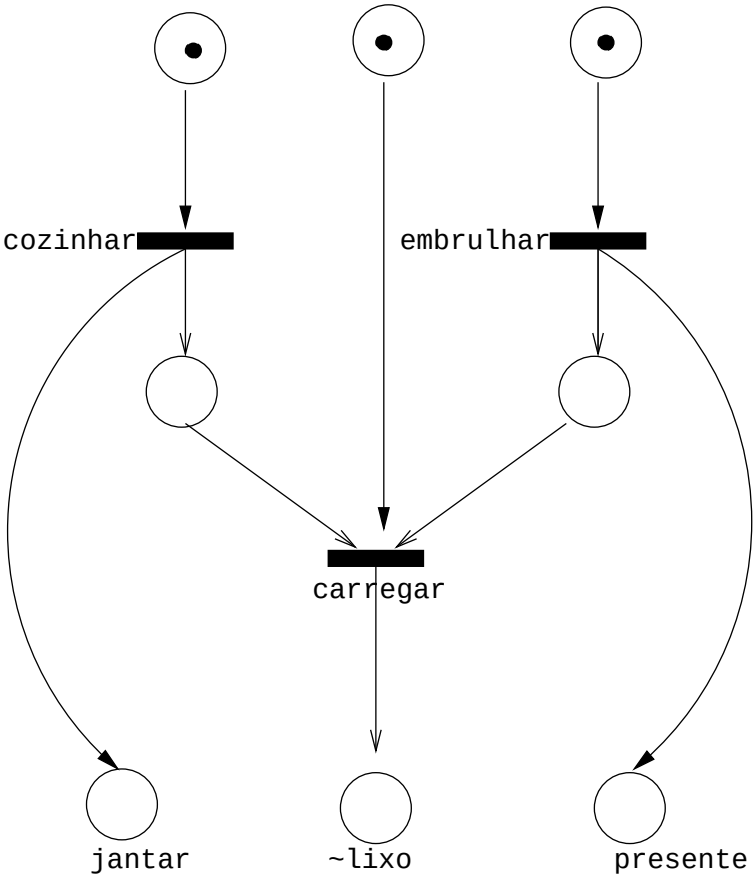

Figura 5.8: Sub-Rede obtida pela Rede de Planos para o problema do jantar.

Note que para alcançar o estado meta para esta sub-rede representada na figura 5.8, efetua-se:

- primeiramente, é executado o disparo de *cozinhar* e *embrulhar*;

- o disparo de *cozinhar* e *embrulhar* nos permite obter imediatamente as sub-metas *jantar* e *presente*, bem como sensibiliza a transição *carregar*;
- pelo disparo de *carregar* obtêm-se a sub-meta *lixo*;

Logo uma das possíveis sub-rede extraída para a Rede de Planos deste problema nos retorna a solução para o problema do jantar a partir da seguinte seqüência de disparo:

- S_1 = *cozinhar//embrulhar*
- S_2 = *carregar*

Observe que para este mesmo problema, utilizando o grafo de planos original, se fazem necessários no mínimo duas expansões para se alcançar o estado meta e apenas uma no caso da Rede de Planos.

5.3.1 Representação do Tempo na Rede Obtida

A mesma estratégia utilizada para incluir uma especificação temporal no Petriplan pode ser utilizada para a Rede de Planos. É suficiente associar a cada transição que representa ações do plano um valor que representa sua duração.

Graficamente a representação temporal feita sobre a rede obtida através do grafo de planos é feita da seguinte maneira:

Na figura 5.9, fez-se a especificação temporal, utilizando-se a mesma estratégia aplicada no Petriplan, e obtém-se a sub-rede T-temporizada representada na figura 5.10:

Aplicando os mesmos tempos para as ações utilizados na seção 5.1, e utilizando os métodos de análise de RdP, propostos por Ramchandani em [47], sobre a seqüência de disparo descrita na sub-rede apresentada na seção 5.3, obtemos uma solução para o problema do jantar em 9 unidades de tempo.

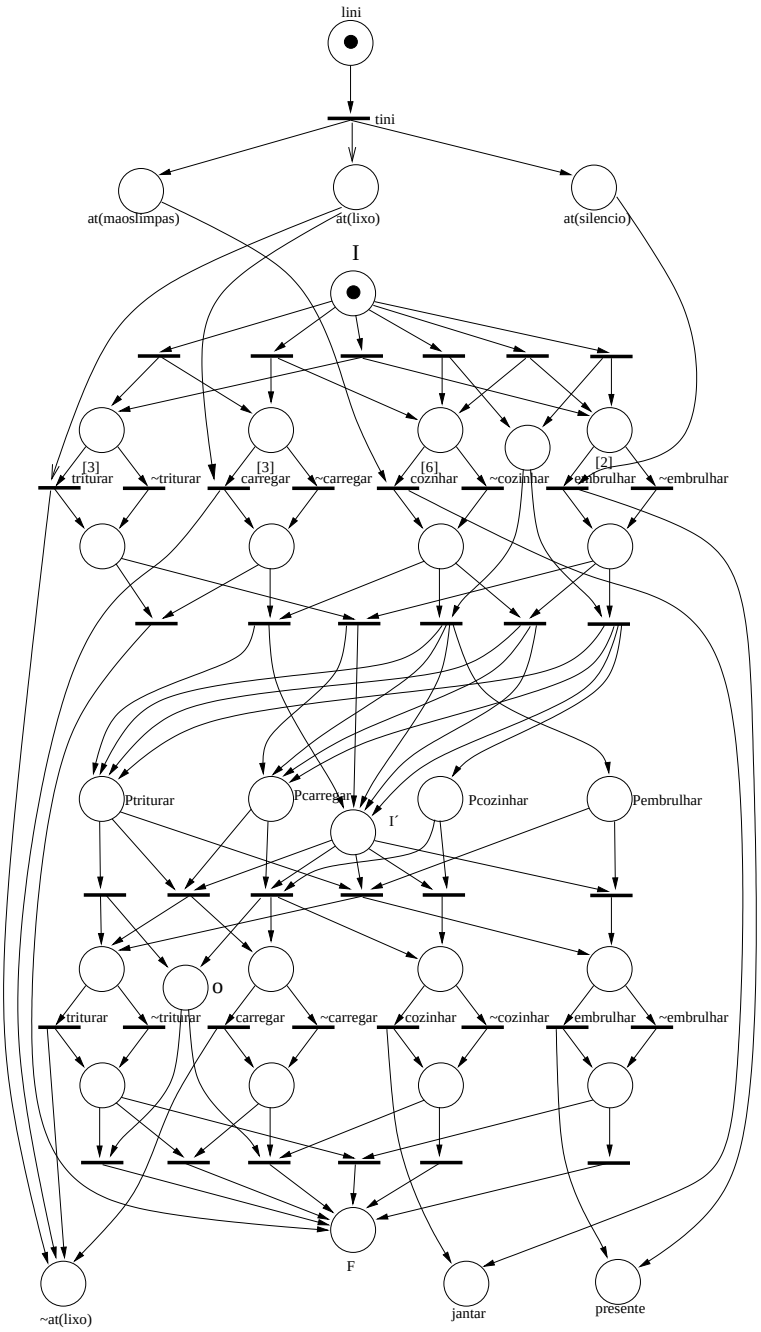

Figura 5.9: Rede de Petri sobre o resultado da Rede de Planos.

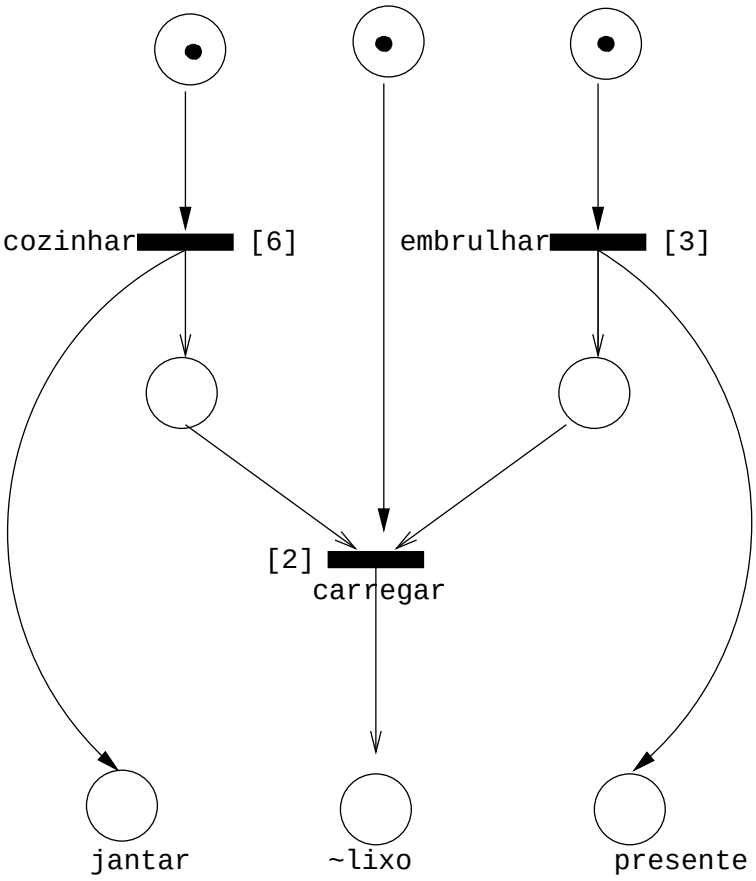

Figura 5.10: Sub-Rede temporizada obtida pela Rede de Planos para o problema do jantar.

5.4 Comparação entre os Métodos Propostos

Para o exemplo apresentado, todos os métodos apresentaram o mesmo resultado temporal. Isto não deve ser entendido como uma equivalência entre os métodos. Basta alterarmos os tempos associados às ações que passaremos a obter resultados distintos daqueles aqui apresentados, com um custo computacional diferente para cada método. Como exemplo, assumamos que as ações *cozinhar* e *embrulhar* tenham uma duração de 6 unidades de tempo e a ação *carregar* dure 1 unidade de tempo. O Grafo de Planos atemporal obtém um plano que executa a ação *cozinhar* seguida da execução

simultânea das ações *carregar* e *embrulhar*. Este plano tem, portanto, uma duração de 12 unidades de tempo. Para o cálculo de duração bastou aplicarmos as novas durações ao plano já existente, que é construído independentemente dos dados temporais. Até mesmo porque o objetivo do Grafo de Planos é obter um plano qualquer que minimize o tamanho do plano em termos do número de camadas, o que não obrigatoriamente implica o melhor plano em termos temporais quantitativos.

No caso do Grafo de Planos a Eventos, seria necessário reconstruir o grafo para os novos valores de duração, uma vez que as datas dos eventos serão alteradas. Entretanto, o novo plano obtido apresenta a execução simultânea das ações *cozinhar* e *embrulhar*, seguidas da execução da ação *carregar*. Este plano tem uma duração de 7 unidades de tempo, conforme pode ser observado na figura 5.11, e corresponde a um plano que minimiza sua duração. Este resultado é obtido diretamente das características do Grafo de Planos a Eventos, que associa a cada camada uma data relativa ao início do plano. Como o Grafo de Planos original minimiza o número de camadas, o Grafo de Planos a Eventos apresentará inevitavelmente um plano que tem a menor duração entre os planos possíveis.

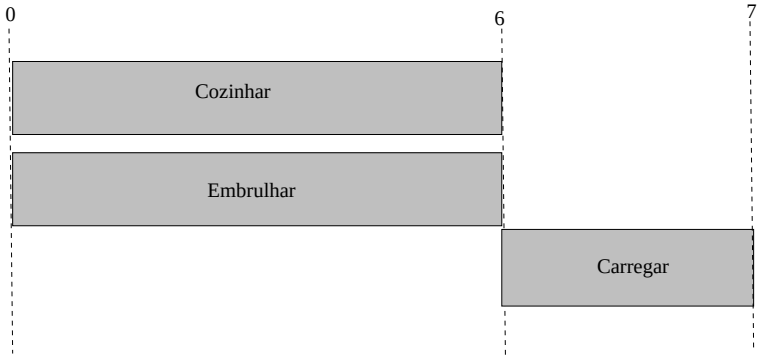

Figura 5.11: Possível Solução do Grafo de Planos a Eventos para o Domínio do Jantar com os Novos Tempos.

Com relação aos métodos baseados em Redes de Petri temos restrições semelhantes.

Como o algoritmo Petriplan constrói uma Rede de Petri equivalente ao Grafo de Planos, sem considerações temporais explícitas, a rede obtida pode não expressar de forma correta a concorrência entre as ações, o que implica a obtenção de resultados temporais não satisfatórios.

A Rede de Planos, por outro lado, é construída diretamente a partir da descrição do problema e é capaz de identificar de forma abrangente o conjunto de inconsistências entre a ações. A relação de ordem entre as ações permite que apenas a análise temporal seja refeita para novas durações, sem a necessidade de reconstruir a Rede de Planos. No exemplo em questão, a Rede de Planos do problema do jantar com as novas durações apresenta um plano cuja duração é de 7 unidades de tempo.

Capítulo 6

Conclusão

Este trabalho investigou o planejamento temporal em inteligência artificial. Para tanto, foram apresentados os principais conceitos relacionados com o tema de planejamento e as principais abordagens existentes para resolver problemas a ele relacionados. A abordagem baseada no Grafo de Planos foi estudada com maior profundidade. Também foram apresentadas as diferentes linguagens de descrição de problemas e sua evolução recente, sobretudo na especificação de restrições temporais relativas à duração de ações na construção de planos.

Entre os métodos derivados do Grafo de Planos foi estudado com maior atenção o método Petriplan, proposto pela equipe de pesquisa que envolve o LSIP (Laboratório de Sistemas Inteligentes de Produção) do CPGEI/CEFET-PR e o LIC (Laboratório de Inteligência Computacional) do Departamento de Informática da UFPR. Para tanto, foram estudadas as Redes de Petri, assim como suas extensões temporais.

O desenvolvimento do trabalho resultou na proposta de duas abordagens para a questão temporal em planejamento em inteligência artificial. A primeira baseada apenas no Grafo de Planos e a segunda na Rede de Planos, uma evolução do Petriplan, também desenvolvida na equipe de pesquisa. Ambas as propostas geram planos temporalmente robustos, isto é, que minimizam sua duração total. A primeira com elevado custo

computacional, pois implica gerar em camadas no Grafo de Planos a Eventos para cada data em que ocorre início ou término de ações. Qualquer modificação nos dados temporais implica gerar um novo grafo. Uma vantagem inerente a este método está em utilizar grande parte das soluções já existentes para o Grafo de Planos, como os algoritmos de elevado desempenho para o processo de busca. Outra vantagem está na possibilidade de se definir como objetivo uma duração máxima para o plano. Caso seja alcançada uma camada com tempo superior ao máximo definido, pode-se afirmar a impossibilidade de solução dentro do objetivo fixado.

A grande vantagem da segunda abordagem está na capacidade da Rede de Planos de representar, de forma mais completa e correta, que as demais abordagens já desenvolvidas, as diferentes relações de ordem existentes entre as ações que compõem os planos que resolvem o problema de planejamento. Desta forma, a inclusão do tempo ou a modificação dos dados temporais associados às ações pode ser efetuada diretamente sobre a rede, sem custo computacional adicional.

Ambas as abordagens propostas podem facilmente ser extendidas para atender às novas especificações temporais da linguagem PDDL, tais como efeitos de início, pré-condições para término e efeitos que permanecem válidos somente durante a execução das ações.

Como trabalhos futuros, indicamos:

- Implementação dos métodos propostos e integração na plantaforma IPE [37];

Bibliografia

[1] Wolfman S. A. and Weld D. S. The lpsat engine & its application to resource planning. *IJCAI*, 1999.

[2] Y. Sun A. Barret, J. Penberthy and D. Weld. *UCPOP v4.0 user's manual*. Informe Técnico TR 93-09-06d, Dep. of Computer Science and Engineering, University of Washington, Seattle, WA., 1996.

[3] E. Onaindía A. Garrido and F. Barber. A temporal planning system for time-optimal planning. In *10th Portuguese Conference on AI (EPIA-2001)*, pages 379–392, Springer-Verlang, 2001.

[4] M. Fox A. Garrido and D. Long. A temporal planning system to manege level 3 durative actions of pddl2.1. In *En Proc. 20th UK Planning and Scheduling SIG Workshop*, pages 127–138, 2001.

[5] M. Fox A. Garrido and D. Long. A temporal planning system for durative actions of pddl2.1. In *Proc. European Conference on AI (ECAI-2002)*, pages 586–590, Amsterdam, 2002.

[6] C. Anderson, D. E. Smith, and D. Weld. Conditional effects in graphplan. In *Proc. of AIPS98*, 1998.

[7] Y.Dimopoulos B. Nebel and J.Köehler. Ignoring irrelevant facts and operators in plan generation. *Proc. European Conference on Planning*, pages 338–350, 1997.

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

[9] A. Blum and M. Furst. Fast planning through planning graph analysis. *Journal of Artificial Intelligence*, 90(1-2):281–300, 1997.

[10] B. Bonet and H. Geffner. Hsp: Planning as heuristic search. *Entry at the AIPS-98 Planning Competition*, 1998.

[11] B. Bonet and H. Geffner. Planning as heuristic search. *Artificial Intelligence*, pages 5–33, 2001.

[12] J. Cardoso and R. Valette. *Redes de Petri*. Editora da UFSC, 1997.

[13] J. Franck D. E. Smith and A.K. Jonsson. Bridging the gap between planning and scheduling. *Knowledge Engineering Reviw, 15(1)*, 2000.

[14] M. B. Do and S. Kambhampati. Sapa: A domain-independent heuristic metric temporal planner. *Proc. European Conference on Planning ECP-2001*, 2001.

[15] M. A. Castilho F. Silva and L. A. Kunzle. Petriplan: a new algorithm for plan generation (preliminary report). *Lecture Notes in Artificial Intelligence*, 1952:86–95, 2000. International Joint Conference IBERAMIA'2000 - SBIA'2000.

[16] R. Fikes and N. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Journal of Artificial Intelligence*, pages 3–4, 1971.

[17] E. Fink and M. Veloso. *Prodigy Planning Algorithm*. Informe Técnico CMU-94-123, Carnegie Mellon University, 1994.

[18] M. Fox and D. Long. Fast temporal planning in a graphplan framework. *Proc. Workshop on Planning for Temporal Domains (AIPS-2002)*, 2002.

[19] M. Fox and D. Long. Pddl2.1: An extension pddl for expressing temporal planning domains. 2003.

[20] A. Gerevini and I. Serina. Lpg: A planner bad on local search for planning graphs with action costs. *En Proc. 6th Int. Conference on AI Planning and Scheduling (AIPS-2002)*, pages 281–290, 2002. AAAI Press.

[21] Y.Dimopoulos H. Kautz, B. N. Selman and J.Köehler. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the AAAI-96*, Portland, OR, 1996.

[22] K. Halsey. Literature survey. a review of temporal planning. *Informe técnico*, 2002.

[23] P. Haslum and H. Geffner. Heuristc planning with time and resources. *Proc. Euporean Conference on Planning (ECP-2001)*, pages 121–132, 2001.

[24] J. Hoffmann. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intelligence Research*, 20, 2003.

[25] J. Hoffmann and B. Nebel. The ff planning system: Fast plan generation through heurisc search. *Jornal of Artificial Inteliggence Research*, pages 253–302, 2001.

[26] J. Hendler J. Allen and A. Tate. Readings in planning. *Morgan Kaufmann*, 1990.

[27] J. Hoffmann J. Koehler, B. Nebel and Y. Dimopoulos. Extending planning graphs to an adl subset. Technical report, ICS - University of Freiburg, 1997. TR 88.

[28] B.Nebel J. Köehler and Y.Dimopoulos. Extending planning graphs to an adl subset. *Lexture Notes in Artificial Intelligence (1348*, pages 273–285, 1997.

[29] J.Penberthy and D.Weld. Temporal planning with continuous change. *Proc 12th Nat. Conference on Artificial Intelligence (AAAI-94*, pages 1010–1015.

[30] H. Kautz and B.Selman. Blockbox: A new approach to the aplicacation of theorem proving to problem solving. *En Proc. Workshop on Planning as Combinatorial Search*, 1998.

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

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

[33] H. Kautz and B. Selman. The role of domain-specific knowledge in the planning as satisfiability framework. *Proceedings of the 4th Int. Conf. on Artificial Intelligence Planning Systems (AIPS-98)*, 1998.

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

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

[36] D. Long and M. Fox. Efficient implementation of the plan graph in STAN. *Journal of Artificial Intelligence Research - JAIR*, 10:87–115, 1999.

[37] J. E. Marynoswki. Ipe: Ambiente para estudo de planejadores. Master's thesis, Dinf-UFPR, 2004.

[38] J. McCarthy and J. Hayes. Some philosophical problems from the standpoint of artificial intelligent. *Machine Intelligence*, pages 463–502, 1969.

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

[40] D. McDermott. Pddl - the planning domain definition language. *AIPS-98 - Planning Competition Comittee*, 1998.

[41] P. Merlin. *A Study of Recoverability of Computer Systems*. PhD thesis, University of California IRVINE, Computer Science, 1974.

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

[43] M. Nidd. Time extensions of petri nets. 1994.

[44] E. P. D. Pednault. Adl: Exploring the middle ground between strips and the situation calculus. In *Proceedings of Knowledge Representation Conf.*, 1989.

[45] J. Penberthy and D. Weld. UCPOP: A sound, complete, partial-order planner for adl. In *Proceedings of the Third International Conference on Principles of Knowledge Representation and Reasoning*, 1992.

[46] C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut ffir Instrumentelle Mathematik, 1962. English translation available as Communication with Automata, Tech. Rep. RADC-TR-65-377, Vol. 1, Suppl. 1, Applied Data Research, Princeton, NJ, 1966.

[47] C. Ramchandani. *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets*. PhD thesis, Cambridge, Mass: MIT, Dep. Electrical Enginnering, 1974.

[48] V. Schastai. Métodos de análise para redes de petri temporais. Master's thesis, Centro Federal de Educação Técnológica do Paraná, 2002.

[49] V. Schastai and L. A. Künzle. Análise de redes de petri temporais combinando grafo de classes e processos. *Congresso Brasileiro de Automática*, 2-5 setembro 2002.

[50] F. Silva. Algoritmos para planejamento baseada em strips. Master's thesis, Universidade Federal do Paraná, 2000.

[51] F. Silva. *Rede de Planos: Uma Proposta para a Solução de Problemas de Planejamento em Inteligência Artificial Usando Redes de Petri.* PhD thesis, Centro Federal de Educação Técnológica do Paraná - CEFET/PR, 2005.

[52] D. E. Smith and D. S. Weld. Temporal planning with mutual exclusion reasoning. *En Proc 16th Int.Joint Conference on AI (IJCAI-99*, pages 326–337, 1999.

[53] A.Garrido T. *Planificacion Temporal Independent del Dominio. Una Aproximación Basada em Grafos de Planificacion.* PhD thesis, Universidade Politécnica de Valencia, 2003.

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

[55] E. A. Lima V. Schastai and L. A. Kunzle. Sequence analyses for time petri nets. *Workshop on Discrete Event Systems - WODES/2004*, 2004.

[56] A. Weld and D. E. Smith. Extending graphplan to handle uncertainty and sensing actions. *En Proc. AAAI-98*, pages 897–904, 1998.

[57] D. Weld. Recent advances in ai planning. *AI Magazine*, 1999.

[58] D. Wilkins. Practical planning: Extending the classical ai planning paradigm. *Morgan Kaufmann*, 1988.

Apêndice A

PDDL2.1 E A DESCRIÇÃO TEMPORAL

Conforme mencionado na seção 4.1 a PDDL+ foi dividida em 5 níveis, sendo o nível 3 objeto desse trabalho, este apêndice destina-se ao detalhamento desse nível.

PDDL 2.1 Nível 3 - Ações com Duração Discreta

Decorrente da inserção de novas informações nos domínios com capacidade de tratamento sobre as variáveis temporais uma nova sintaxe se faz necessária no nível 3, a seguinte sintaxe é definida:

```
(:durative-action        <NameOfAction>
 :parameters             <ListOfArgumentsWithTypes>
 :duration               <LogicalExprOnDurationVariable>
 :condition              <LogicalExpr>
 :effect                 <LogicalExpr>
)
```

Onde cada nova variável tem a seguinte finalidade:

- <NameOfAction> - Listar as ações com duração;
- <ListOfArgumentsWithTypes> - Listar as declarações de variáveis juntamente com os tipos;
- <LogicalExprOnDurationVariable> - Listar as combinações, proposições e restrições temporais na duração de uma ação, onde a restrição numérica pode ser uma desigualdade;
- <LogicalExpr> - Lista de combinações de proposições e fluentes das restrições.

A construção de modelos de relacionamento temporal é feita detalhando condições e efeitos onde os comentários temporais podem ser *at start, over all* ou *at end*. Uma restrição temporal pode ser registrada em *at star* ou em *at end*.

Semanticamente uma condicional e um efeito, incluídos em *at star*, correspondem a uma ação instantânea, que ocorre no momento de inicialização da ação com duração. Uma condicional, incluída em *over all*, corresponde a uma invariante que é exigida para garantir a execução da ação no tempo de sua execução, porém não em situações de inicialização ou finalização. Condicionais e efeitos incluídos em *at end* corresponde a uma ação instantânea que no momento da finalização de uma ação com duração.

A seguir descreve-se um pequeno fragmento do domínio de logística apresentadao na seção 3.8.2, porém, incluindo variáveis temporais. O objetivo é apresentar o momento em que ocorre a inserção do tempo no PDDL. O fragmento descreve a ação *Drive-Truck*.

```
(:durative-action drive
 :parameters (pk1 ?pk2-obj ?t-truck ?loc ?loc-loc)
 :duration (= ?duration (driving-time ?loc1 ?loc2})                    (0)
 :condition (and (at start (at ?obj ?loc1))
                 (at start (at ?truck ?loc1))                          (1)
                 (over all (parked ?truck ?loc))                       (2)
```

```
 (over all (life drive-time ?truck))
 (at end (t ?loc2))                                                    (03)
                 )


  :effect (and
          (at start (not (at ?t ?loc1)))                                (4)
          (at start (not (parked ?t ?loc1)))
  (at start (not (at ?t ?loc1)))
          (at start (not (at ?pk1 ?loc1)))
          (at start (life drive-time ?t))
          (at end (at ?t ?loc2))                                        (5)
          (at end (at ?pk1 ?loc2))
  (at end (not (live drive-time ?t)))
  (at end (not (parked ?t ?loc1)))
          (* (live drive-time ?loc1 ?loc2))                             (6)
          )
)
```

No fragmento do domínio é adicionada uma identifição numérica às linhas, objetivando descrever a notação utilizada no domínio.

- (0), É uma restrição temporal: garante que o tempo de deslocamento entre duas localidades nunca seja superior ao estipulado pelo agente, é uma pré-condição para que (6) seja atualizada. Permite que esta ação seja concorrente com todas as ações que afetam o deslocamento do caminhão durante a execução da ação;
- (1) e (4), são as condições e os efeitos;
- (2), são os invariantes de intervalo;

- (3) (4) e (6), são os efeitos gerados pela ação executada, bem como o tempo decorrido na execução da ação.

Conforme já mencionado anteriormente, a extensão de PDDL2.1 nível 3 tem como objetivo principal a avaliação de características temporais dos domínios, bem como avaliação de planos concorrentes. A representação da concorrência entre ações gera um modelo mais preciso das transições de estados no decorrer da execução das ações. Em conseqüência, novas relações de exclusão são geradas, uma nova nomeclatura é inserida, novos efeitos são obtidos e novas precondições são exigidas, tornando assim, PDDL2.1 nível 3, mais complexo em termos de análise, porém, mais preciso em termos de resultados obtidos [19, 53].

Por tratar-se de ações com duração, primeiramente deve-se apresentar a definição de *Ação Durativa* conforme [19, 5, 53].

Definição A.1 ***Ação com Duração:*** *Em PDDL2.1, uma ação* $\boldsymbol{a}$*, com duração, se define como uma tupla* $\langle\ SCond, Inv, Econd, SEff, EEff, dur\ \rangle$*, sendo SCond, Inv e Econd os conjuntos de condições de início, intervalo de execução e finalização da ação* $\boldsymbol{a}$*, respectivamente. O conjunto* $SEff = \{Sadd \bigcup Sdel\}$*, representa os efeitos que serão gerados no início de* $\boldsymbol{a}$ *,sendo Sadd e Sdel são efeitos positivos e efeitos negativos, respectivamente. Enquanto, EEff* $=\ \{\ Eadd \bigcup Edel\ \}$ *representa os efeitos que serão adicionados à ação* $\boldsymbol{a}$*, ao final de sua execução, sendo Eadd e Edel efeitos positivos e negativos respectivamente. Para finalizar* $dur \in \Re^+$*, que representa a duração da execução de* $\boldsymbol{a}$*,* $dur \in \Re^+$ *reais positivos.*

A partir da definição de duração de uma ação, pode-se observar que no nível 3 de PDDL2.1 passa a não ter que analisar apenas condições de início para que uma ação ocorra, mas sim, a existência de recursos no seu intervalo de execução, bem como condições de finalização da mesma. Como conseqüência dessas ações, tem-se novos efeitos, sendo que estes efeitos podem se apresentar, tanto no início de uma ação, quanto durante sua execução e sua finalização.

Outra particularidade que pode-se notar é que, as condições invariantes *Inv*, não precisam necessariamente estarem presentes para que a ação **a** se inicie, ou seja, as condições podem ser geradas concorrentemente à execução de **a**, uma vez que estas somente serão necessárias durante a execução de **a**. Diante disso faz-se necessário distinguir condições invariantes de não invariantes na análise de um plano.

Outro aspecto presente no nível 3 do PDDL2.1 é o fato de uma ação poder apresentar efeitos iniciais, ou seja, efeitos gerados imediatamente ao início da execução da ação, acrescentando com isso a possibilidade de concorrência em um plano. Por exemplo, ao analisar o problema proposto na subseção 3.8.1, pode-se ter *Drive*(t_1, loc_1, loc_2), que necessita da proposição *at*(t_1, loc_1) antes que a ação *Drive* seja executada. Esta ação, após executada, irá gerar as proposições { at(t_1, loc_1) e $\sim$ at(t_1, loc_1) }, ao final da execução da ação. Diante dessa ação, tem-se que a proposição t_1 torna-se inacessível até que a ação seja finalizada. Isto faz com outras ações, que dependam deste recurso, não sejam executadas concorrentemente. A proposição $\sim$at(t_1, loc_1) pode estar excluindo diversos planos válidos. Dessa forma, o que PDDL2.1 propõe é a inserção como efeito inicial da proposição $\sim$at(t_1, loc_1), evitando, com isso, que diversos planos inválidos continuem sendo gerados enquanto a ação *Drive* esteja sendo executada.

Uma representação gráfica destes novos tipos de pré-condição pode ser observada na figura A.1 [53].

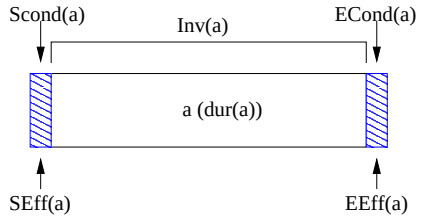

Figura A.1: Elementos de uma ação durativa do nível 3 do PDDL2.1

Novas Relações de Exclusão

Em termos de relações de exclusão PDDL2.1 apresenta novos mutex, gerados em função dos novos conceitos introduzidos nos efeitos e nas ações condicionais.

Os novos mutex que passam a ocorrer em PDDL2.1 são definidos como mutex estáticos e dinâmicos.

Definição A.2 ***Mutex Estático*** *Diz-se que um mutex é estático quando o mesmo se mantém, independentemente do instante temporal da ação que ocorre. São denominados estáticos, pois dependem exclusivamente de ações do domínio independente do tempo.*

Como exemplo, observar-se na figura 3.10: quando a ação $drive(t_1, loc_1, loc_2)$ é executada, são gerados os efeitos at(t_1, loc_1) e $\sim$ at(t_1, loc_1). Este mutex é permanente e não se altera devido ao fato dele ser gerado por uma ação de manutenção, a qual se fará presente até a última camada do grafo. Como resultado tem-se conflitos gerados em virtude da simultaneidade dos efeitos gerados pelas mesmas.

Definição A.3 ***Mutex Dinâmico*** *Diz-se que um mutex é dinâmico se o mesmo for temporal e desaparecer no decorrer da ação.*

Um exemplo de *Mutex Dinâmico* também pode ser observado na figura 3.10, onde tem-se na camada 1 a ação $load(t_1, loc_1, pk_2)$, que irá gerar um mutex entre as ações at(pk_2, loc_1) e in(pk_2, t_1), porém, este mutex irá desaparecer a partir do instante em que surgir $\sim$in(pk_2, t_1).

Ao trabalhar com ações que apresentem duração, torna-se necessário efetuar o cálculo de mutex entre as mesmas, em função de suas diversas formas de execução. A formalização desses calculos poderão ser observados em [53].

Apêndice B

REDES DE PETRI

Redes de Petri Convencionais

A teoria sobre Redes de Petri surgiu em 1962 como resultado do trabalho da tese *Kommunication Mit Automaten* (Comunicação com Autômatos) defendida por Carl Adam Petri [46]. Entre 1968 e 1976 foram lançadas as RdP por um grupo de pesquisadores orientados por Anatol W. Holt do Instituto de Tecnologia de Massachussetts (MIT), nos Estados Unidos, sendo estas definidas como *Redes de Petri convencionais*.

Desde então as Redes de Petri têm sido amplamente utilizadas para modelar, analisar e controlar sistemas dinâmicos e eventos discretos. Segundo Cardoso & Valette [12] uma RdP é adaptável a aplicações em que os eventos envolvem de forma concorrente e tem como vantagem, quando utilizadas como modelo formal para a busca de soluções em termos de aplicações próximas de problemas reais:

- Os estados e eventos são representados de modo explícito;
- Existência de maior flexibilidade na descrição de ordem parcial entre vários eventos;
- Possibilidade de descrição precisa e formal das sincronizações tornando assim o

funcionamento mais seguro;

- Unificação das ferramentas utilizadas para especificação, análise, modelagem e avaliação de desempenho.

Murata em [42] também apresenta algumas características que considera relevantes no que tange a utilização de uma RdP. Algumas dessas características são:

- Utilização de vários níveis de abstração para modelagem;
- Eventos sincronizados;
- A forma sistemática de verificação das propriedades do sistema;
- Possibilidade de modelagem de sistemas concorrentes e paralelos;
- A visualização através da representação gráfica.

Definindo uma Rede de Petri

Pode-se definir uma RdP como uma ferramenta formal, particularmente bem adequada à representação de paralelismo, concorrência, conflito e relações causais em sistemas dinâmicos e eventos discretos.

Nesta seção apresentaremos uma definição formal das RdP como uma coleção de matrizes de vetores cujos componentes são números inteiros e seu comportamento caracterizado por programação linear.

De modo formal podemos definir uma RdP, como sendo uma quintupla N = (P, T, Pre, Post, M), sendo:

- P = (p_1 , p_2, ...p_n), um conjunto finito de lugares definido como o conjunto de lugares de N;

- T=(t_1,t_2,...t_n), um conjunto finito de transições definido como o conjunto de transições de N, sendo $P \cap T = \emptyset$;
- Pre: PxT $\rightarrow$ N é a aplicação de entrada sendo N o conjunto dos números naturais;
- Post: PxT $\rightarrow$ N é a aplicação de saída, sendo N o conjunto dos números naturais;
- M: P $\rightarrow$ N é a marcação inicial de N.

Graficamente podemos representar uma RdP convencional N =(P, T, Pre, Post, M) através do seguinte gráfico:

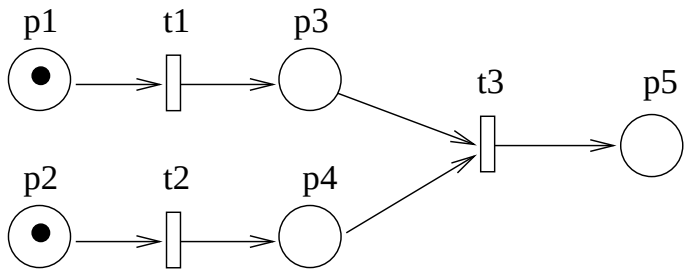

Figura B.1: Rede de Petri convencional

Na figura B.1 os círculos representam os lugares e as barras verticais representam as transições. Os *Lugares* em uma RdP são os nós que descrevem os estados (um lugar é um estado parcial) e as *Transições* descrevem as mudanças de estado. *Pre* é a função de incidência e descreve os arcos orientados que conectam lugares a transições. Para cada transição t, ela representa o fragmento do estado no qual o sistema tem que estar antes da mudança de estado correspondente à ocorrência de t. $Pre(p,t)$ é o de cada arco (p,t). $Pre(p,t) = 0$ denota ausência de um arco entre o lugar p e a transição t.

A função de incidência *Post* descreve os arcos orientados que fazem a conexão das transições aos lugares. Ela representa cada transição t, o fragmento do estado no qual o sistema estará após a mudança de estado correspondente à ocorrência de t. $Post(p,t)$ é o peso do arco (t,p). $Post(p,t) = 0$ denota ausência de um arco entre t ep.

Vetorialmente, ha representação de uma RdP, *Pre* e *Post* são matrizes de n linhas, sendo as linhas compostas pelos lugares e m colunas, sendo as colunas compostas pelas transições e seus elementos pertencentes a N. O vetor *Pre(.,t)* denota os arcos de entrada de uma transição t com seus pesos e o vetor *Post (.,t)* denota os arcos de saída da transição t com seus pesos.

Em uma RdP a dinâmica é dada pelo *disparo* das transições habilitadas, cuja ocorrência corresponde a mudanças de estado do sistema modelado pela rede. A transição t de uma RdP N e está habilitada para uma marcação M, se e somente, se $M \geq Pre(.,t)$. esta condição de habilitação expressada sob a forma de uma desigualdade entre vetores, é equivalente a $\forall p \in P, M(p) \geq Pre(p,t)$.

Para que ocorra o disparo de uma transição em Rdp é necessário que a mesma esteja habilitada. Se M é uma marcação de N que habilita uma transição t, e M' a marcação derivada do disparo de t a partir de M, então $M' = M + Post(.,t) - Pre(.,t)$. Notemos que o disparo da transição t a partir de uma marcação M deriva a marcação M': $M \rightarrow^t M'$.

Generalizando a fórmula para calcular uma nova marcação após o disparo de uma seqüência s de transições, considere a matriz $C = Post - Pre$, chamada de *matriz de incidência* da RdP, e um vetor $\overline{s}$, chamado *vetor característico* de uma seqüência de disparos s($\overline{s}$: T $\rightarrow$ N, tal que $\overline{s}(t)$ é o número de vezes que a transição t aparece na sequência s). O número de transições T define a dimensão do vetor $\overline{s}$. Então, a nova marcação M_g de uma marcação M, após o disparo de uma seqüência s de transições, é calculada por:

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

Esta equação é denotada ***Equação Fundamental*** de N. Podemos usar a equação fundamental para determinar uma vetor $\overline{s}$ para uma dada rede N e duas marcações de M e M_g. A solução que satisfaz a equação deve ser um vetor inteiro não-negativo, e a existência deste vetor é apenas uma condição necessária para que a marcação M_g seja

alcançável a partir da marcação M. Esta condição torna-se necessária e suficiente para as *RdP Acíclicas* [42], sendo estas uma subclasse de Redes de Petri que não possuem circuitos dirigidos.

A relação de alcançabilidade entre marcações de uma transição disparável pode ser estendida, por transitividade, para a alcançabilidade do disparo de uma seqüência de transições. Dessa forma, em uma RdP N, é dito que a marcação M_g é alcançável a partir da marcação M se, e somente se, existe uma seqüência de transições S tal que: $M \longrightarrow^{8} M_g$. O conjunto alcançabilidade de uma RdP marcada (N, M_0) é o conjunto $R(N, M_0)$ tal que $(M \in R(N, M_0)) \Leftrightarrow (\exists_s M_0 \longrightarrow^{8} M)$.

Chama-se de *problema de alcançabilidade* o problema de encontrar uma seqüência de disparos s que alcança uma da marcação M_g a partir de M_0 se $\mathrm{M}_g \in R(N, M_0)$. O *problema de alcançabilidade de submarcação* consiste em encontrar uma seqüência de disparos para alcançar um subconjunto de lugares $M_s \subset M_g$ onde $M_g \in R(N, M_0)$. É provado que o problema de alcançabilidade é decidível [42].

Redes de Petri e a Representação do Tempo

Da mesma maneira que os Planejadores Clássicos, as Redes de Petri não trazem em sua definição original o conceito de tempo de forma explícita, ou seja, as RdP clássicas trazem apenas as propriedades *qualitativas* do problema (não relacionadas ao tempo) de um sistema. No entanto, para que se tenha uma avaliação de desempenho de maior completude, bem como a apresentação formal de problemas que envolvem escalocanamento ou programação em sistemas dinâmicos, ou seja, propriedades *quantitativas*, é aconselhável que nas transições e/ou lugares sejam associados tempos.

Na literatura, modelos que trabalham efetuando um retardamento do tempo na rede é têm o nome de *Modelos Determinísticos*. Os modelos determinísticos encontram-se divididos em duas classes: *Redes de Petri Temporais (Time Petri Net - TPN)* e *Redes*

de Petri Temporizadas (Timed Petri Net - RdPN) . De modo simplificado, pode-se dizer que as *Redes de Petri Temporizadas* trazem agregadas uma duração ou um tempo de disparo que é associado a cada transição da rede [47]. Nas RdPN as transições são sensibilizadas de forma semelhante às RdP clássicas, após serem sensibilizadas as transições disparam instantaneamente, porém as fichas só serão depositadas nos lugares de saída decorrido t unidades de tempo após o disparo, sendo t o tempo de disparo associado à transição.

Já as *Redes de Petri Temporais (Time Petri Net - TPN)* trabalham associando as transições da rede intervalos de tempo, ou seja, as TPN operam com o tempo no modo *contínuo*, enquanto as RdPN operam com o tempo no modo *discreto*. Segundo [12, 43] o modelo TPN engloba o modelo RdPN, de forma que é possível representar um instante de disparo t como um intervalo *[t, t]*. Diante dessa possibilidade de transformação, neste trabalho serão utilizadas as RPN devido ao fato de as mesmas serem um modelo mais geral, oferecendo um universo de análise mais completo para as informações que deseja-se obter. Em decorrência disso, na seção a seguir serão apresentados os conceitos básicos de TPN.

Redes de Petri Temporais

As Redes de Petri Temporais *(Time Petri Net - TPN)* foram propostas por Merlin em 1974 [41]. Segundo Merlin, uma rede de Petri Temporal é obtida quando são associados a cada transição dois valores de tempo (a , b), sendo estes números reais que correspondem a uma duração miníma de sensibilização de a. Caso permaneça continuamente sensibilizada até b, a mesma deverá obrigatóriamente disparar.

O intervalo descrito por (a , b) representa os limites de disparo $(\emptyset_{min}, \emptyset_{max})$, onde $\emptyset_{mim}$ expressa o tempo mínimo necessário que uma transição deve permanecer habilitada antes do disparo e $\emptyset_{max}$ representa o tempo máximo que a transição permanecerá

habilitada.

Uma rede de Petri Temporal representada formalmente, é uma sêxtupla $NT = (P, T, Pre, Post, M,\ I)$, onde $(P, T, Pre, Post, M)$ é a representação de uma Rede de Petri Convencional e I é matematicamente descrito como: $I : T \rightarrow (\mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \{\infty\}))$ | $I(t) = [a, b]$, sendo $\mathbb{Q}^+$ o conjunto dos números racionais não negativos. I faz a associação de um intervalo de disparo $I(t) = [a, b]$ a cada transição t da rede, em que a e $b \in \mathbb{Q}^+$, com $0 \leq a \leq b \leq \infty$ e com $a \neq \infty$ [41]. a é dito limite estático inferior (LEI) e b, limite estático superior (LES) de cada transição da rede.

A introdução do conceito estático no texto acima tem como objetivo esclarecer que em uma rede de Petri Temporal, por razões dos limites temporais associados a cada transição, somente a marcação não é suficiente para definir, sem ambigüidade, um estado da rede. Além da marcação, faz-se necessário descrever os limites temporais das transições habilitadas por essa marcação, pois os mesmos devem ser respeitados. Isso significa que os limites de uma transição da rede, quando analisada de forma isolada, requerem que a transição, isoladamente, necessite de um tempo mínimo (LEI) e um tempo máximo (LES) para disparar, conforme pode ser observado na figura B.2.

Conforme pode ser observado com maior detalhamento em [48], ao ser representado o comportamento dinâmico da Rede, deve-se considerar tanto a evolução temporal (passagem do tempo) como a possibilidade de outras transições que estejam habilitadas e possam vir a impor restrições aos intervalos de tempo. Decorrente disso, os limites das transições resultantes do comportamento dinâmico da Rede serão chamados limite inferior de disparo (LID) e limite superior de disparo (LSD).

A representação de uma Rede de Petri Temporal, de maneira geral, pode ser descrita como $E = (M, IT)$ onde M é a marcação da Rede e IT representa o mapeamento das transições habilitadas pela marcação M juntamente com seus intervalos de disparo, dessa forma pode-se dizer que, $IT : HAB(M) \rightarrow (\mathbb{Q}^+ \times (\mathbb{Q}^+ \cup \{\infty\}))$ | $IT(t) = [c, d]$, em que $0 \leq c \leq d \leq \infty$, e $c \neq \infty$. Sendo c o limite inferior de disparo (LID) e d limite

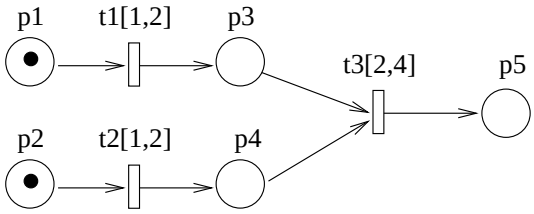

Figura B.2: Rede de Petri Temporal

superior de disparo (LSD) de t.

Análise de Uma Rede de Petri Temporal

No decorrer dos anos de pesquisa sobre RdP alguns métodos de análise foram propostos, entre eles encontra-se a metodologia proposta por Valter Schastai em [48, 49, 55]. Esta metodologia avalia a rede sem fazer com que ocorra aumento de imprecisão temporal.

A metodologia proposta em [49] faz a análise do grafo de classes e constrói para cada sequência a ser analisada no grafo um sistema de inequações. O conjunto de inequações gerado tem por objetivo fazer a descrição de todas as possíveis datas de disparo das transições da seqüência.

As restrições temporais das transições contidas nas seqüências serão representadas pelas inequações que compõem este sistema. A obtenção das restrições se dá através das informações que se extraem do grafo de classes.

Utilizando-se do sistema de inequações que é construído para uma sequência pode-se obter informações que descrevem as datas máximas e mínimas para que um determinado evento aconteça ou a maior e menor distância temporal entre dois eventos [49].

Obtendo um Sistema de Inequações

A obtenção de um sistema de inequações segundo [49] se dá quando percorrido o grafo de classes. O caminho percorrido no grafo descreve a seqüência que será analisada.

O início da seqüência será marcado a partir da primeira transição que está habilitada (classe inicial) finalizando na última transição da seqüência (classe alcançada).

A composição do sistema será efetivado a partir das primeiras inequações presentes no sistema de restrições. Tais restrições são obtidas da classe inicial do caminho, sendo uma cada transição t_i presente na classe.

As inequações obtidas são apresentadas como $a_i \leq t_i \leq b_i$, sendo a_i o limite inferior (LID), da transição t_i na classe e sendo b_i passível de assumir um dos dois valores: o limite superior da classe (LSE(C)), se a transição t_i for utilizada para se obter a próxima classe do caminho, ou caso contrário o limite superior de disparo (LSD) da transição t_i na classe.

As inequações têm por objetivo transportar para o sistema de restrições o estado em que a primeira transição da seqüência irá disparar e a partir das demais classes existentes no caminho novas inequações serão adicionadas ao sistema de inequações.

Para cada classe, toda transição t_i que esteja presente na classe, irá gerar uma nova inequação do tipo $a_i \leq e_{ti}$ - $e_{tk} \leq b_i$, sendo que a_i e b_i obedecem os mesmos critérios definidos anteriormente enquanto que t_k representa a transição que alcançou a classe.

A obtenção dessas inequações tem como objetivo garantir a dependência entre a data de disparo de uma determinada transição da seqüência e a data de disparo das transições anteriores. Ter essa dependência significa evitar o aumento da imprecisão da data de disparo das transições concorrentes, bem como garantir a ordem temporal da ocorrência das transições.

Exemplo de Análise de uma RdP Temporal

A figura B.3 será analisada temporalmente usando a metodologia proposta em [49].

No exemplo apresentado na figura B.3 a sequência sdt = (t_1, θ_1) (t_3, θ_2) (t_4, θ_3) (t_5, θ_4) (t_2, θ_5), (t_6 θ_6), sendo que *sdt* a sequência de disparo temporal, será analisada usando somente o grafo de classes.

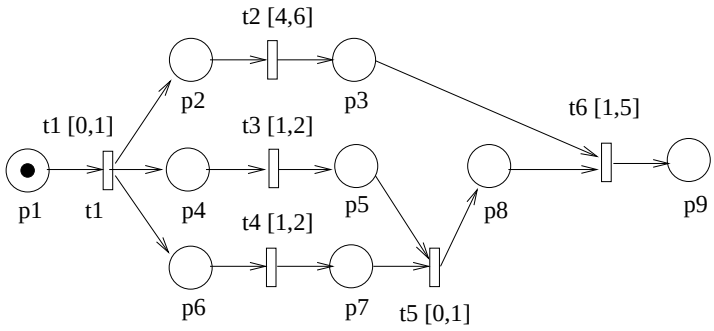

Figura B.3: Rede de Petri Temporal

O grafo de classes que se obtém a partir da rede ilustrada na figura B.3 é o grafo B.4.

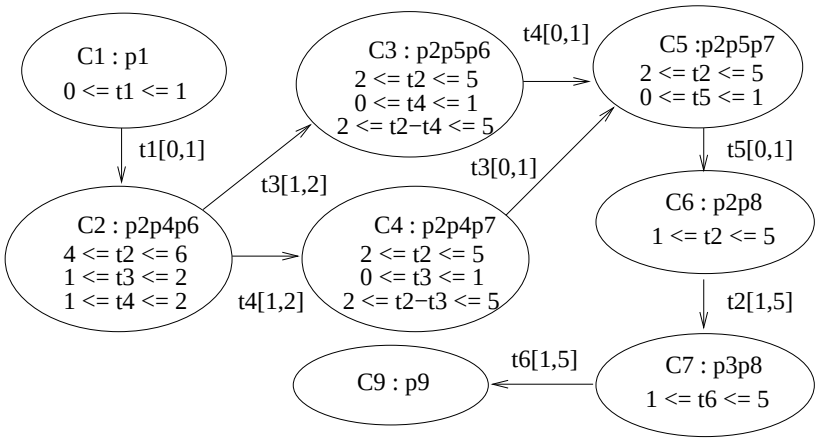

Figura B.4: Grafo de Classes

Do grafo B.4, o sistema de inequações que será gerado pela metodologia proposta em [49] será obtido tendo a primeira inequação resultante da classe inicial do caminho, pois representa a seqüência de transições do grafo. A inequação obtida faz a descrição das possíveis datas de disparo para a transição t_1.

As demais inequações serão obtidas quando percorrido o caminho das demais classes, sendo que as mesmas estarão representando as depêndencias nas datas de disparo das

transições que se fazem presentes na seqüência.

Do exemplo apresentado na figura da rede apresentada na figura B.3, o conjunto de identificadores das restrições para a seqüência é o seguinte:

Intervalo	Restrição Temporal
I_1	$0 \leq e_{1t_1} \leq 1$
I_2	$4 \leq e_{1t_2} - e_{1t_1} \leq 6$
I_3	$1 \leq e_{1t_3} - e_{1t_1} \leq 2$
I_4	$1 \leq e_{1t_4} - e_{1t_1} \leq 2$
I_5	$2 \leq e_{1t_2} - e_{1t_3} \leq 5$
I_6	$0 \leq e_{1t_4} - e_{1t_3} \leq 1$
I_7	$2 \leq e_{1t_2} - e_{1t_4} \leq 5$
I_8	$0 \leq e_{1t_5} - e_{1t_4} \leq 1$
I_9	$1 \leq e_{1t_2} - e_{1t_5} \leq 5$
I_{10}	$1 \leq e_{1t_6} - e_{1t_2} \leq 5$

Tabela B.1: Restrições Temporias para a Seqüência: ((t_1, θ_1) (t_3, θ_2) (t_4, θ_3) (t_5, θ_4) (t_2, θ_5), (t_6 θ_6))

A rede descrita na figura B.4, se analisada por outras metodologias teria como data máxima de disparo 15 unidades de tempo, enquanto que se a mesma for analisada pelo grafo de classes em decorrência do paralelismo existente, pode-se disparar t_6 em até 12 unidades de tempo.

Além dessa possível redução de unidades de tempo, também poderá ser obtido atráves do sistema de inequações outras análises, tais como, a busca pela maior seqüência temporal entre as transições t_1 e t_6 conforme mostrado em [49].