Planejamento automático aplicado a problemas dependentes de recursos

Este exemplar corresponde à redação final da Dissertação a ser defendida por Felipe Vieira Silva e aprovada pela Banca Examinadora.

Fortaleza, 11 de agosto de 2003.

Ana Teresa de Castro Martins
Universidade Federal do Ceará

Francisco Tavares
Instituto Nacional de Pesquisas Espaciais

Tarcísio Pequeno
Universidade Federal do Ceará

Dissertação apresentada ao Mestrado em Ciência da Computação, UFC, como requisito parcial para a obtenção do título de Mestre em Ciência da Computação.

Universidade Federal do Ceará
Mestrado em Ciência da Computação

Dissertação de Mestrado

Planejamento automático aplicado a problemas dependentes de recursos

---

por

Felipe Vieira Silva

orientação
Dra. Ana Teresa de Castro Martins
Departamento de Computação, UFC/CE

Laboratório de Inteligênica Artificial
UFC / CE - Brasil

Agradeço em especial à minha família, a amada Alena e ao meu filho Rodrigo, pessoas que sempre estiveram sorrindo ao meu lado e que me deram toda a felicidade que eu precisava para concluir este trabalho. Aos meus pais e irmãs, que de longe acompanharam a minha saga nordestina, um lastro de segurança sem fronteiras, incondicional.

À minha banca examinadora, serena e motivadora nos momentos finais do meu trabalho:

Ana Tereza, educada e profissional em todos os momentos - incluindo as incertezas que rondam a cabeça de um mestrando na reta final - uma verdadeira amiga. Ao admirado professor Tarcísio, com seu jeito vivido de olhar as coisas, de conhecimento cobiçado - um exemplo a seguir. Professor Tavares, com sua tecnologia admirável, conceitos modernos e educação em adequar o discurso a todos os níveis de platéia, desde o estudante mais ingênuo ao cientista mais pernóstico.

Todos estes mestres me ensinaram o prazer da ciência, me permitiram perspectivas profissionais que eu não imaginava no começo do meu mestrado. A todos eles, o meu eterno reconhecimento.

Aos demais professores do departamento de computação da UFC, em especial à Marcelino Pequeno, Cláudia Linhares, Ricardo Corrêa, Fernando Carvalho, Júlio Wilson e ainda outros que me cederam o conhecimento e me apoiaram incondicionalmente sempre que precisei.

Agradecimento especial ao Eng. Assis Filho, profissional nobre que me recebeu na FUNCEME sempre que o trabalho exigiu, um incentivo à integração entre universidades e empresas. Ao professor Porfírio, com seus bumerangues e dicas sobre inteligência artificial e vivência acadêmica. As empresas que me deram espaço para mostrar meu trabalho: a Softexport, o Instituto Atlântico, todos os colegas de profissão que encontrei pelo caminho.

Ao Ceará, seu povo povo nobre de tradições e cultura apoiados no Sol e na alegria de viver. Cada sorriso, cada gosto e cheiro, cada beleza revelada nas palavras amigas de desconhecidos e acolhedores irmãos. Pra sempre o orgulho de ter conhecido, a felicidade de ter salgado a alma em suas praias e cidades.

Por fim os novos amigos, aqueles que me abriram as portas de Fortaleza e me ensinaram os caminhos da felicidade por aqui. Os colegas do mestrado, a pousada em Iracema, o futebolzinho no Diário e todas as outras pessoas que me receberam

como manda a tradição local: seriedade e companherismo. A peleja, com suas eternas discussões e, em especial, aos meus amigos Fernando Rodrigues, Rodrigo Lopes e Allan Reffson.

Resumo

Esta dissertação relata a investigação preliminar sobre uma área de pesquisa pouco difundida no ambiente acadêmico brasileiro: o planejamento automático [11, 51, 129]. Apesar dos primeiros planejadores datarem dos anos setenta, recentes avanços nessa área promoveram uma euforia na comunidade internacional que foi pouco refletida no cenário nacional. A partir dessa constatação, este trabalho situa a evolução do planejamento automático nos últimos trinta anos a partir da enumeração de suas principais técnicas.

Um modelo de planejador é discutido, incluindo a implementação de um protótipo e a reprodução de experimentos publicados por pesquisadores internacionais. O esforço de repoduzir em laboratório os conceitos previstos teoricamente permitiram uma visão crítica sobre o estado da arte na geração de planos. Dentre as críticas, a baixa aplicabilidade dos planos gerados ao mundo real aparece como destaque e uma solução é sugerida: algoritmos planejadores com suporte a recursos.

Um problema relacionado à administração de recursos hídricos no Estado do Ceará é apresentado e adotado como métrica de qualidade das idéias propostas. O desempenho do protótipo e de outros planejadores frente a esse problema encerra a obra, sugerindo perspectivas em relação ao uso efetivo de planejadores no mundo real.

(209 palavras)

Abstract

This dissertation reports a preliminary investigation about a research area which is little spread in brazilian academic environment: planning [11, 51, 129]. Despite first planners were made in the 70's, recent improvements in this area promoted an excitement in the international community which was little reflected in the national scenery. From this observation, this work situates the evolution of planning in the last thirty years by the enumeration of its main techniques.

A planner model is discussed, including a prototype development and the reproduction of experiments published by international researchers. The effort of reproducing in laboratory the theoretically predicted concepts allowed a critical view over the state of art of plans generating. Among all critics, low applicability of generated plans in real world stands out and a solution is suggested: planner algorithms with support to resources.

A problem related to the managing of water resources in the state of Ceará is presented and adopted as a parameter of quality of the proposed ideas. The performance of the prototype and of other planners face to this problem closes this work, suggesting perspectives in relation to the effective use of planners in real world.

(209 words)

Sumário

Lista de Tabelas

Lista de Figuras

Lista de exemplos

1

Introdução

O Ceará, pela sua localização geográfica, encontra-se em uma região com dois períodos climáticos distintos e bem definidos: o período das chuvas, entre dezembro e março, e o período seco, nos demais meses do ano. Esta característica semi-árida imprime ao Estado o dever de administrar seus recursos hídricos da forma mais racional possível, e exige de seus centros de pesquisa a busca de tecnologias que assim o permitam. Departamentos públicos como FUNCEME - Fundação Cearense de Metereologia, e UFC - Universidade Federal do Ceará, ocupam-se em monitorar e desenvolver soluções para os problemas relativos à água no Estado [24]. Dentre os estudos atualmente desenvolvidos encontra-se o Projeto WAVES, cooperação Brasil-Alemanha de fomento à pesquisa sobre a disponibilidade e qualidade da água na região do semi-árido nordestino (veja Apêndice A). Integrando os projetos financiados pelo WAVES, nossa pesquisa foi originalmente motivada por um problema relativo à administração do sistema de reservatórios de água do Ceará:

> Como distribuir a água do sistema de reservatórios do Ceará, maximizando a satisfação social e econômica do Estado [1], quando o volume de água disponível é geralmente inferior à demanda requerida no sistema ?

À primeira vista, esse dilema sugere soluções puramente quantitativas [76], relacionadas à distribuição do volume disponível de água ao número de pontos de distribuição. Essa prematura sugestão, entretanto, demonstra fragilidade quando consideramos que o recurso envolvido na solução do problema é vital ao ser humano: a água. Não parece sensato elocubrar soluções puramente quantitativas para um problema que tem impacto direto na capacidade de sobrevivência e desenvolvimento do ser

[1] Os aspectos sociais e econômicos inerentes à gestão de recursos hídricos não serão abordados nesta dissertação, mas textos introdutórios sobre esse assunto podem ser encontrados no livro sobre gestão de águas da Associação Brasileira de Recursos Hídricos [15]

humano, enquanto sociedade e enquanto indivíduo. De fato, as decisões observadas no mundo real em relação a problemas de distribuição de água sempre consideram a qualidade [23] das demandas como critério prioritário na tomada dessas decisões. Ou seja, a função da água distribuída - abastecimento de cidades, indústria, irrigação agrícola ou turismo, entre outras - acaba gerando um complexo sistema de pesos e responsabilidades que define a melhor ou mais correta distribuição a ser adotada.

Historicamente, o conhecimento empírico vem sendo adotado como guia na administração de sistemas dos reservatórios de água cearenses. As ferramentas computacionais, embora existentes e tecnicamente avançadas, são predominantemente quantitativas e por isso acabam subjugadas por decisões políticas do governo local. Normalizando e avaliando o resultado dessas decisões notamos que elas são eficientes, embora não haja nenhum modelo formal ou técnica que justifique tais decisões. Creditamos a essa eficiência uma única razão: a inferência humana. Tal inferência tem o seu modelo e controle ainda desconhecidos em sua completude, mas possui resultados incomparáveis em velocidade e qualidade à melhor das máquinas que o homem já produziu.

De fato, o propósito desta dissertação é justamente relatar a pesquisa de técnicas que solucionam problemas de forma semelhante à abordagem demonstrada pelo homem para esses mesmos problemas. A abordagem de problemas através da implementação de mecanismo semelhantes ao comportamento humano, ação ou raciocínio, remete à ontologia da Inteligência Artificial [80, 92, 93, 110], área da ciência cujos conceitos e paradoxos estarão implícitos no conteúdo desta dissertação.

1.1 Abordagens quantitativas e qualitativas a problemas do mundo real

Neste trabalho, o problema da administração de reservatórios de água terá a função de baliza [2] para a discussão entre as abordagens quantitativas e qualitativas de problemas do mundo real - o tema central da dissertação.

[2]Na literatura internacional, os autores usam o termo inglês *benchmark* para referir problemas que permitem a comparação entre técnicas diversas que visam o mesmo propósito. Aqui preferimos manter o termo "baliza", pois as técnicas apresentadas no texto, apesar de suportarem a abordagem do mesmo problema, são distintas em áreas de conhecimento e em sua função original.

Tradicionalmente, os aspectos qualitativos e quantitativos de um problema são separados e tratados em áreas específicas da Inteligência Artificial (IA). De um lado, temos os simbólicos, pesquisadores que tentam modelar um problema através da sugestão de sistemas lógicos que permitam a solução correta e completa desse problema por meios qualitativos. Por outro lado, temos grupos de estudiosos que buscam a representação numérica de um problema para então aplicar algoritmos de cálculo da solução ótima desse problema dentro de uma abordagem quantitativa.

Apesar de compartilharem a mesma raíz matemática, esses dois grupos acabaram historicamente se distanciando e hoje gozam de igual respeito acadêmico, contabilizando suas falhas e sucessos. Algumas vezes porém, fica evidente a necessidade de interseção de seus conceitos na abordagem de problemas do mundo real - o problema de água é um belo exemplo disso.

1.2 A administração de sistemas de reservatórios de água

A administração da distribuição de água em um sistema de reservatórios consiste na decisão sobre o volume de água a ser distribuído em um determinado período de tempo, levando-se em conta a demanda, a capacidade de transmissão e a projeção sobre o volume de água disponível nesse período. Numa descrição informal, o problema possui os seguintes componentes:

- Um conjunto de reservatórios de água, com uma capacidade de armazenamento e vazão máxima pré-estabelecidas;
- Um conjunto de centros de demanda (centros urbanos, fazendas ou zonas industriais);
- Um conjunto de dutos que ligam os reservatórios aos centros de demanda.

A Figura 1.1 apresenta um sistema de distribuição de água e um diagrama identificando seus componentes.

Aparentemente simples, essa questão torna-se complexa principalmente em regiões como o semi-árido nordestino, onde a disponibilidade de água é sempre crítica

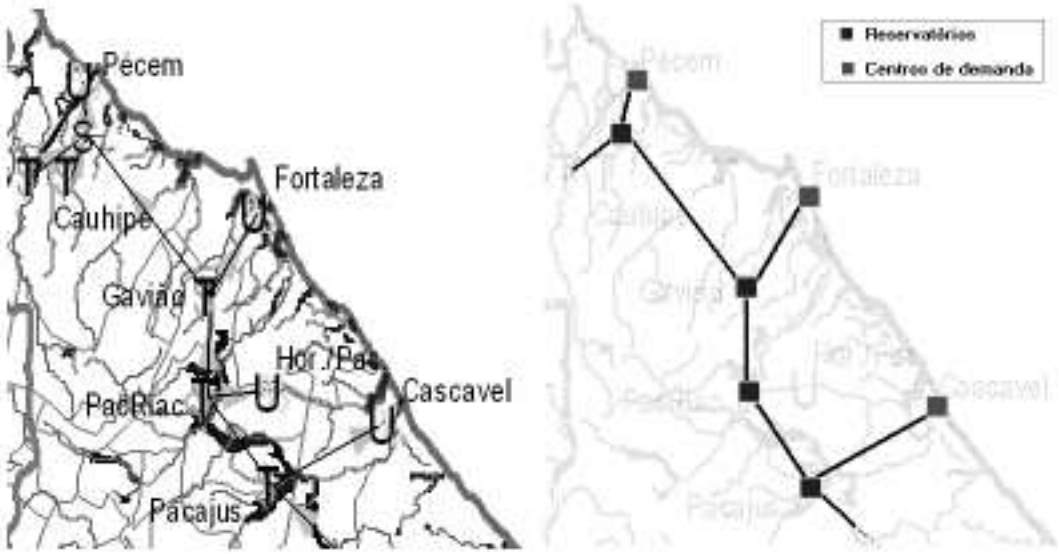

Figura 1.1. *Representação parcial do sistema de distribuição de água do Ceará*

em relação à demanda. Atualmente, a administração do sistema de distribuição de água no Ceará depende da orientação técnica dos engenheiros que gerenciam os reservatórios e de decisões políticas do Governo do Estado (veja Apêndice B). Observa-se nesse ambiente administrativo a carência de ferramentas computacionais de apoio à tomada de decisões, e uma conseqüente dependência humana nos processos de decisão - a experiência dos técnicos envolvidos é que vem garantindo a qualidade do sistema de distribuição. A expansão do sistema de reservatórios e a renovação natural da mão de obra que os controla permite a repetição de falhas e dificulta cada vez mais a tomada de decisões e a previsão a longo prazo do comportamento desse sistema. Dentro desse contexto sugere-se a implementação de uma ferramenta de apoio à tomada de decisões utilizando uma modelagem do conhecimento adquirido ao longo dos anos pelos técnicos que administram o sistema de distribuição de água e técnicas de Inteligência Artificial. Duas áreas de estudo foram consideradas para o projeto dessa ferramenta de apoio: Fluxo de Redes e Planejamento Automático.

1.2.1 A abordagem tradicional do problema

Do ponto de vista quantitativo, o problema da distribuição de água pode ser abordado por técnicas de otimização como, por exemplo, algoritmos de fluxo máximo e mínimo (veja Cormen[18], capítulo 27). Tais algoritmos, baseados em redes de

fluxo (grafos direcionados), procuram otimizar a distribuição do fluxo em redes, considerando apenas a capacidade de fluxo de cada aresta. No caso de sistema de distribuição de água, podemos considerar a capacidade de fluxo entre dois nodos como a demanda requerida em uma das extremidades, ou infinito, no caso da aresta ligar dois reservatórios.

O sistema de reservatórios que aparece na Figura 1.1 pode ser representado, por exemplo, pelo grafo não direcionado que aparece no lado esquerdo da Figura 1.2. Note que os nodos com linhas duplas representam os reservatórios e os nodos com linhas simples representam os centros de demanda, em uma configuração idêntica a que aparece no mapa anterior. Como o fluxo tem mais de uma origem e mais de um destino, necessitamos incluir dois novos nodos na rede: uma super origem, criada com arestas para todas os reservatórios da rede e um super destino, que recebe arestas de todos os centros de demanda da rede. Esse dois novos nodos reduzem o problema de fluxo com múltiplas origens e destinos para o problema original de fluxo, com uma única origem (R) e um único destino (D). Atribuíndo uma capacidade ilimitada às arestas que ligam a super origem e o super destino ao grafo original, garantimos que o fluxo calculado pelos dois grafos será equivalente. Não provaremos essa relação aqui [3], pois os problemas de fluxo são amplamente abordados na literatura e servirão apenas para estabelecer o contexto em que as técnicas de planejamento auomático podem ser adotadas como soluções alternativas.

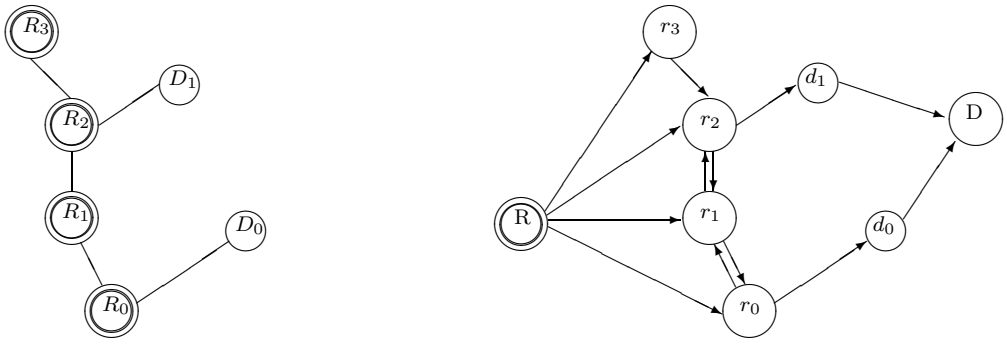

Figura 1.2. *O sistema de reservatórios representado através de uma rede. À esquerda aparece a representação simples do problema, e à direita a representação considerando dois novos nodos: uma* super origem *e um* super destino.

[3] O leitor poderá encontrar essa prova e um material completo sobre fluxo de redes em [18, 34].

1.3 O planejamento automático

Outra forma de abordagem para problemas do mundo real, como a administração de sistemas de reservatórios de água, é o uso de uma técnica batizada de Planejamento Automático [84, 81, 49, 56]. O planejamento automático, que será formalizado no próximo capítulo, parte da idéia de simular o comportamento de um agente em um determinado ambiente, e enumerar as ações adotadas por esse agente para alcançar um conjunto de objetivos préviamente definidos. Se o agente conseguir alcançar os seus objetivos, a enumeração de suas ações é chamado de plano para alcançar esses objetivos. Caso contrário, dizemos que não foi possível gerar um plano para o problema. As características do ambiente são observáveis em intervalos de tempo que chamaremos de visões ou, simplesmente, estados do ambiente. Inicialmente observamos o estado do intervalo de tempo zero, dito Estado Inicial, e o objetivo do agente é interagir com esse ambiente na busca de uma determinada visão, conhecida como Estado Objetivo. Essa interação é feita através da aplicação de regras de inferência, instanciadas a partir de um conjunto de esquemas de ação préviamente conhecido. A cada intervalo de tempo, o agente deve decidir quais as melhores instâncias a serem aplicadas ao estado atual e em que ordem aplicá-las.

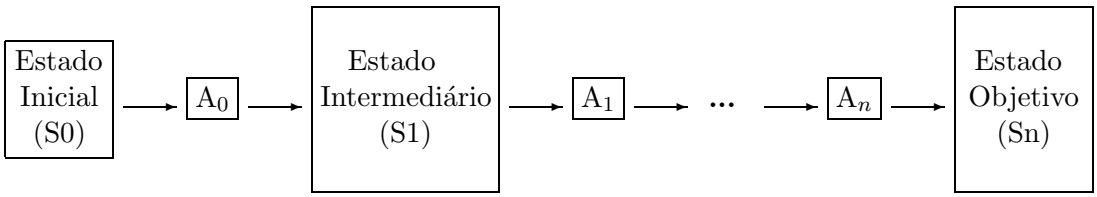

Figura 1.3. *A função de um plano é permitir a um agente que se encontra em um determinado estado S0, dito inicial, alcançar um outro estado S_n, dito objetivo, através da execução das ações $\{A_0, A_1, A_2, \ldots, A_{n-2}, A_{n-1}, A_n\}$*

A Figura 1.3 apresenta a idéia geral de planejamento automático, com as seguintes características:

i Estados, simbolizados pela letra S, são coleções não ordenadas de fatos.

ii Ações, simbolizadas na figura pela letra A, são regras de inferência - para que as conseqüências de uma ação tornem-se fatos, necessita-se verificar a ausência

de exceções a essa regra. No planejamento automático mantem-se o conceito de exceções, porém passa-se a exigir que os pré-requisitos de uma regra estejam presentes antes que este fato seja introduzido no domínio.

iii Existe ainda o aspecto temporal relacionado ao planejamento, ou seja, além da necessidade da enumeração de ações aplicáveis a um domínio, faz-se necessário a identificação da ordem dessas ações. A maioria dos sistemas planejadores usam o conceito de intervalo de tempo para situar as ações e os estados.

Esta descrição original, considerando apenas um agente em um domínio onde as ações são seqüenciais, mas aspectos mais complexos sobre a geração de planos são introduzidos quando consideramos domínios mais complexos, com ações concorrentes e/ou mutuamente exclusivas. Tal complexidade, que no capítulo 2 mostraremos ser NP-Hard, é uma barreira encontrada pelos pesquisadores para a construção de uma ferramenta aplicável ao mundo real. Outra dificuldade é a expressividade das linguagens disponíveis para a representação da causalidade dos fatos no mundo real [79] e, principalmente, as limitações e custos dos recursos compartilhados pelas ações de um plano.

1.4 Problema em aberto - o consumo de recursos

A maior parte dos planejadores atuais depende da estratégia de seleção da ação a ser aplicada a um determinado estado intermediário. Como será apresentado no próximo capítulo, essa seleção raramente leva em conta o custo real dessas ações, ou seja, os recursos consumidos pela sua aplicação no problema do mundo real.

Imaginemos por exemplo o caso do sistema de reservatórios descrito acima. O problema só é tratável pelos planejadores atuais se ignorarmos os aspectos reais de transferência de água entre dois pontos, como evaporação, tempo da transferência, necessidade de bombeamento e o valor financeiro associado à água.

Inúmeros trabalhos apontam a dificiência dos planejadores em considerar os recursos associados ao domínio de um problema [64, 3, 100, 5, 76]. A forte tendência da comunidade que estuda planejamento em adaptar os algoritmos disponíveis ao suporte à recursos nos motivou a identificar alternativas para esse problema. O resultado desta análise, bem como o desenvolvimento de um protótipo planejador

baseado em recursos, aparece descrito no Capítulo 3 como uma das contribuições previstas em nosso trabalho.

1.5 Contribuições

i Revisão sobre Planejamento Automático - uma área de pesquisa pouco difundida no cenário acadêmico brasileiro apesar da recente evidência no cenário internacional. Identificação das principais técnicas e dos principais pesquisadores em Planejamento Automático.

ii Modelagem e análise de um sistema planejador, utilizando-se diagramas UML. A discussão sobre essa modelagem provê um ponto de partida para estudantes e/ou grupos de pesquisa interessados no projeto e implementação de ferramentas de apoio à tomada de decisões baseadas em planejamento automático.

iii Análise da aplicabilidade de sistemas de Planejamento Automático a problemas do mundo real e em relação a outras técnicas de apoio a tomada de decisões. Identificação dos obstáculos atuais e tendências para o desenvolvimento de pesquisas relacionadas a planejamento automático.

1.6 Estrutura da dissertação

Essa dissertação[4] traz um levantamento geral sobre o Planejamento Automático enquanto área de pesquisa pouco difundida no Brasil. Para tal, o seu conteúdo foi dividido nas seguinte seções:

Capítulo 2: apresenta-se um breve histórico do estudo de planejamento automático, conceituando formalmente o assunto e classificando as suas principais áreas de pesquisa. Ferramentas tidas como referência são enumeradas a partir da cronologia em que foram desenvolvidas e descritas em detalhes quanto as suas características técnicas. A análise do desempenho e da forma de representação do

[4] As fontes desse texto foram formatadas no padrão LaTeX2e [67], utilizando-se o editor TextPad 4.4.1, e compiladas com MikTex 1.20e sob o sistema operacional MS Windows 2000.

conhecimento dessas técnicas sugere um modelo utilizado como referência nos demais capítulos da dissertação.

Capítulo 3: a partir das técnicas propostas no Capítulo 2, um modelo de planejador automático é apresentado. Esse modelo é analisado quanto a sua estrutura de controle (complexidade, corretude, completude) e em relação as características esperadas de programas desenvolvidos a partir dele (desempenho, consumo de memória, escalabilidade e portabilidade). O capítulo estabelece um guia para a implementação dos componentes básicos de um planejador automático, considerando o paradigma de Orientação a Objetos [25, 19].

Capítulo 4: apresenta-se a descrição em PDDL do problema de administração de reservatórios de água no Ceará, e os respectivos resultados da aplicação dessa descrição no protótipo do planejador modelado no Capítulo 3.

Capítulo 5: sumarizam-se os principais tópicos da dissertação, apresentando-se sugestões de trabalhos futuros.

2

Conceitos e Técnicas de Planejamento Automático

Conforme dito na introdução desta obra, planejamento automático é um tema central em IA, pois propõe o desenvolvimento de um *solucionador* genérico de problemas. Várias gerações de pesquisadores já dedicaram esforços nesse complexo desafio, deixando como legado uma vasta bibliografia. Esta seção traz uma uma revisão sobre essa bibliografia, apresentando um breve histórico da área e identifcando as técnicas mais recentemente adotadas pela comunidade científica internacional.

2.1 Linha do tempo

A descrição de problemas em termos de enumeração de passos para a satisfação de um objetivo apareceu no final dos anos cinqüenta, com problemas como a *ida ao aeroporto* [78] e os *missionários e canibais* [73]. O problema da *ida ao aeroporto*, por exemplo, foi definido por John McCarthy a partir da seguinte idéia:

> *Suponha-se que eu esteja no escritório da minha casa e deseje ir ao aeroporto. Considere-se que o meu carro esteja na garagem da minha casa. A solução desse problema seria: caminhar até o carro e dirigir o carro até o aeroporto.*

Nessa época, a dificuldade era centrada na correta formalização do problema atráves de premissas que pudessem ser usadas para a geração automática da solução. Quando existiam, os algoritmos que usavam essas premissas na satisfação de objetivos eram complexos e específicos para cada problema. Alguns anos mais tarde, o próprio McCarthy propôs uma linguagem formal na tentativa de encapsular todos os componentes dos problemas temporais: o Cálculo Situacional [79]. Apesar da

nova linguagem apresentar um avanço considerável em relação à representação de conhecimento, ainda faltava um algoritmo capaz de utilizar tal representação para a solução de problemas de planejamento. Foi então que, no começo dos anos setenta, Fikes e Nilsson lançaram o famoso planejador STRIPS [29]. De formulação simples, o STRIPS marca o início do estudo de planejamento automático a partir da idéia dos operadores ADD-DEL-PRE (veja seção 2.4.1 abaixo). Como mostra a Figura 2.1, os operadores STRIPS permaneceram quase duas décadas como paradigma vigente mas, ainda assim, havia problemas difíceis de serem resolvidos por eles. A complexidade alta de seu algoritmo - busca exaustiva no espaço de estados - fez com que muitas melhorias fossem propostas. A criação de novas linguagens de representação, específicas para planejamento, foi relegada a segundo plano - embora ainda hoje exista uma ampla comunidade de teóricos trabalhando nesse sentido.

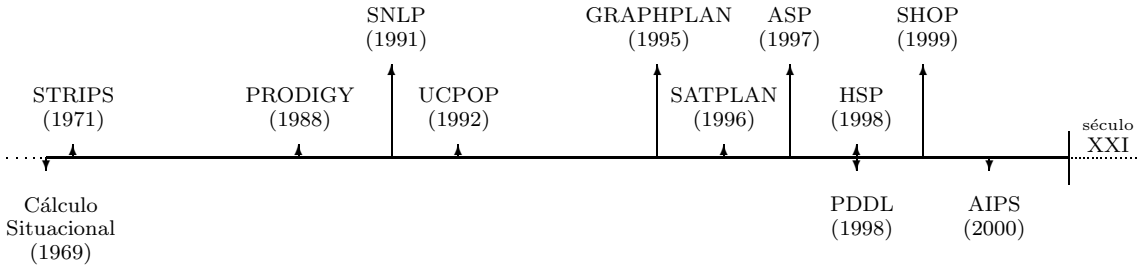

Figura 2.1. *Evolução no estudo de planejamento automático.*

Os anos oitenta foram marcados por um certo desânimo quanto à capacidade da geração automática de planos, e alguns poucos resultados só apareceram no final da década. O planejador mais famoso do início da década de noventa foi o Prodigy [89], que adotava uma engenhosa heurística de planejamento incluindo reordenação de operadores, aprendizagem automática [86] e busca em paralelo de sub-objetivos. Apesar desses esforços, os pesquisadores não conseguiam um mecanismo robusto o bastante para resolver problemas genéricos.

A década de noventa começou com a busca do aumento da velocidade da geração de planos e muitos estudos sobre linguagens e técnicas alternativas ao STRIPS. Nessa época, surgiram alguns conceitos válidos ainda hoje, como planejamento em ordem

parcial, uso de variáveis na definição do esquema de ações, efeitos condicionais e linguagens baseadas em causalidade entre os operadores. SNLP [77] e UCPOP [96] completavam com o PRODIGY o grupo de planejadores baseados em STRIPS de maior sucesso.

A partir de 1995, a história do planejamento automático sofreu novo impulso quando Avrim Blum lançou o GRAPHPLAN [9] - um mecanismo de representação de intervalos de tempo e extração de planos através de grafos. A simplicidade do novo planejador, aliado ao seu desempenho muito superior aos planejadores da época, estimulou o desenvolvimento de muitos outros planejadores a partir de sua idéia central [52, 117, 60, 105]. Tal sucesso permitiu a recuperação de idéias associadas a planejamento por satisfatibilidade. Um ano mais tarde, Henry Kautz e Bart Selman lançaram o SATPLAN [57, 56], um planejador que traduzia o conhecimento do GRAPHPLAN ao problema da satisfatibilidade. Tal transformação na forma de representar os problemas se mostrou mais eficiente que o GRAPHPLAN na maioria dos problemas em que os dois sistemas eram comparados.

Em 1998, Hector Geffner propôs a técnica mais rápida de planejamento conhecida até os dias de hoje: o planejamento por busca heurística - HSP [10]. A grande contribuição de Geffner foi a definição de uma função de custo que permitia verificar a distância dos estados intermediários ao estado objetivo.

Nessa mesma época, começaram a surgir esforços na definição de um modelo comum de testes para que os planejadores pudessem ser comparados entre si - surgiu a PDDL: *Planning Domain Description Language* [82, 4], linguagem de descrição de domínios em planejamento automático. A partir da PDDL, foram estabelecidos critérios e escolhidos problemas exigentes (*benchmarks* [46]) para a realização de uma competição de planejadores automáticos - *AIPS Artificial Intelligence Planning Systems* [84]. A partir de 1998, foi instituida a competição bi-anual de planejamento automático, conhecida por IPC - *International Planning Competition*. Os primeiros passos de nossa pesquisa foram guiados pela identificação dos destaques da competição de 1998 e 2000. Procuramos identificar não as ferramentas em si, mas as abordagens mais eficientes em relação aos critérios comparativos do AIPS:

- O número de problemas resolvidos por cada planejador.
- O tempo requerido para cada solução.

- A qualidade do plano gerado.

Este terceiro critério, dito *qualidade*, tem a ver com o senso comum humano sobre as ações esperadas diante de um determinado problema. O estabelecimento deste *senso comum* é uma tarefa subjetiva diante das interpretações possíveis de um plano - até mesmo planos gerados pelo homem, como uma rota de viagem ou execução de jogos, que são avaliados conforme a visão de quem os analisa. No caso do AIPS, os aspectos focados foram a redundância e o número de ações dos planos gerados.

Outra contribuição das competições foi o estabelecimento de termos comuns para referenciar os elementos constituintes de problemas de planejamento e suas soluções. Devido a sua recente popularidade, o planejamento automático agrega um grande número de pesquisadores oriundos de outras áreas de pesquisa, com diferentes nomenclaturas e formatos estabelecidos para explicar seus experimentos. As competições obrigaram essa heterogênea comunidade a usar termos comuns, alguns deles descritos abaixo.

2.2 Termos fundamentais em planejamento automático

Apesar de ainda não haver uma nomenclatura padrão em planejamento automático, algumas palavras são compartilhadas pela maioria das publicações nessa área. Tais palavras sugerem um consenso sobre os conceitos em planejamento e serão aqui adotadas como termos fundamentais da área:

Fluents: na lógica clássica, caracterizamos os objetos que compõem um mundo através de *fatos*. No planejamento automático, entretanto, essas informações são associadas a instantes de tempo, ou seja, o que é válido em um momento pode ser inválido no momento seguinte. Essa não-monotonicidade é identificada nos problemas de planejameto automático pelo termo *fluent*, ou seja, um fluent é uma proposição cujo valor verdade é dado em função do tempo. Outro aspecto interessante é a imprevisibilidade associada a um fluent, que pode aparecer em intervalos intermitentes de tempo, dependendo dos operadores instanciados pelo planejador. Alguns autores usam a noção de visibilidade para descrever o comportamento dos fluents, ou seja, *visíveis* quando o fluent é verdadeiro em um determinado instante de tempo e *não visíveis*, caso contrário.

Estados: sendo $\mathcal{L}$ uma linguagem da Lógica Clássica de 1^a ordem, um estado S_t é um conjunto finito de fluents pertencentes a $\mathcal{L}$ e que são verdadeiros (visíveis) no instante de tempo t. Note-se que essa definição assume um mundo fechado[1], ou seja, todo fluent $f \notin S_t$ é considerado falso no instante de tempo t.

Operadores: também chamados de ações, os operadores são esquemas que representam regras de inferência não instanciadas. Ou seja, um regra do tipo $\alpha \supset \beta$ onde α e β são conjunções de variáveis. Durante o processo de planejamento, o sistema deve instanciar essas variáveis com fluents selecionados a partir de algum critério - esse processo de instanciação e seleção dos operadores é o cerne da complexidade associada aos problemas de planejamento.

Domínio: o domínio é composto pela enumeração dos predicados e do conjunto de operadores aplicáveis a um determinado ambiente.

Problema: o problema é composto pela descrição do estado inicial e do estado objetivo associado a um ambiente. Ou seja, descreve-se o estado atual de um mundo (ou visão atual), e o estado esperado após a aplicação de um plano.

Plano: é a lista ordenada de operadores instanciados por um planejador e que permite a transição do estado atual ao estado objetivo de um problema.

A partir dos termos citados acima, podemos conceituar o termo *planejamento automático* através da Definição 2.1.

Definição 2.1. *Planejamento automático: é a enumeração do conjunto de instâncias de operadores aplicáveis ao domínio de um problema e que permitem a transição do estado inicial, dito S_0, para o estado desejado e conhecido como Estado Objetivo (S_n) deste problema.*

A definição acima pressupõe o uso de lógica clássica e de um conjunto limitado de objetos válidos no domínio de um problema de planejamento. Apesar dessa definição ser genérica e válida para a maioria dos sistemas planejadores conhecidos, modelos alternativos são sugeridos, como planejamento temporal [6], baseado na

[1] O conceito de *mundo fechado* é analisado em detalhes por Raymond Reiter em [103]

plausibilidade dos fluents, baseado em hierarquia de tarefas [125] e planejamento por checagem de modelos [17], entre outros. A investigação exaustiva de todas essas vertentes ocuparia muito espaço desta dissertação, principalmente se levarmos em conta o caráter empírico associado a maioria desses estudos - investigações em curso e/ou que não possuem um conceito reconhecido à luz da comunidade internacional.

Dos resultados observados até hoje, duas correntes de pesquisa apresentam modelos estáveis: o planejamento determinístico e o planejamento probabilístico. A próxima seção traz uma breve definição dessas linhas de pesquisa, não excluindo os demais modelos como capazes de estabelecerem futuros conceitos sobre planejamento.

2.3 Modelos de planejamento automático

2.3.1 Modelo determinístico

O modelo clássico de planejamento automático [29, 91, 14] é caracterizado pela descrição finita de um domínio e um conjunto de ações aplicáveis a este domínio. Esse modelo prevê que todas as alterações no domínio são conseqüências diretas das ações e que estas conseqüências são discretas e previsíveis. Um problema descrito em um modelo determinístico consiste em encontrar a seqüência de ações que permite a transição do estado inicial ao estado objetivo. O modelo determinístico é definido como segue:

- Um espaço de estados - S
- Um estado inicial - $s_0 \in S$
- Um conjunto de ações aplicáveis a cada estado do domínio - $A(s), s \in S$
- Uma função de transição entre os estados - $f(s, a), s \in S, a \in A(s)$
- O custo da aplicação de cada ação - $c(a, s) \geq 0$
- Um conjunto não vazio de estados objetivos - $G \subseteq S$

A solução de um problema de planejamento determinístico é uma seqüência de ações $a_0, a_1, ..., a_n$ que gera uma transição $s_0, s_1 = f(s_0, a_0), ..., s_{n+1} = f(s_n, a_n)$ para s_{n+1}

igual ao estado objetivo. A solução ótima é obtida quando o custo total $\sum_{i=0}^{n} c(s_i, a_i)$ é mínimo.

A maioria dos planejadores baseados em ações determinísticas utilizam linguagens de alto nível baseadas nos operadores STRIPS [29]. Além disso, o desempenho de planejadores determinísticos depende de heurísticas para a aceleração da busca no espaço de planos.

2.3.2 Modelo probabilístico

Os modelos probabilísticos permitem a representação de ações com conseqüências associadas à distribuição de probabilidades. O modelo probabilístico prevê que as conseqüências das ações não são previsíveis, mas são totalmente observáveis. O modelo probabilístico é definido como:

- Um espaço de estados - S
- Um estado inicial - $s_0 \in S$
- Um conjunto de ações aplicáveis a cada estado do domínio - $A(s), s \in S$
- Uma distribuição de probabilidades associada aos estados alcançáveis a partir do estado atual - $P_a(s', s), s \in S, a \in A(s)$
- O custo da aplicação de cada ação - $c(a, s) \geq 0$
- Um conjunto não vazio de estados objetivos - $G \subseteq S$

Neste tipo de planejamento não se calcula um plano absoluto, capaz de atingir o estado objetivo, mas sim um plano associado a uma probabilidade de atingir o estado objetivo. De fato, os planejadores que usam probabilidade costumam ter como saída um conjunto de planos com probabilidade maior do que zero para a conquista do objetivo.

Devido à alta complexidade inerente aos problemas de planejamento automático, os planejadores que lidam com conhecimento não monotônico são observados com certo ceticismo. Não sendo nosso objetivo discutir a filosofia motivadora das diferentes tácnicas de pesquisa, e reconhecendo o planejamento como um problema NP-Completo, adotamos o planejamento determinístico como foco de nossa pesquisa.

Essa escolha foi também sugerida pela ausência de cultura prévia em planejamento, permitindo o estabelecimento de um sentimento original sobre ferramentas planejadoras a partir das técnicas mais simples. Embora o planejamento determinístico lide com problemas fechados, ou seja, com um número limitado e conhecido de fluents e operadores, a indecidibilidade [27] de sua solução permite novas idéias e se mostra adequada ao problema motivador da administração do sistema de reservatórios.

Na Seção 2.4 temos a enumeração das principais técnicas de planejamento determinístico, permitindo a crítica e sugestão de um modelo próprio a ser apresentado no Capítulo 3.

2.4 Técnicas de planejamento automático

2.4.1 STRIPS

Em 1971, foi lançado o famoso planejador automático baseado nos operadores STRIPS [29] . O ponto alto desse planejador é o esquema de representação de ações baseado nos operadores Pré-Condição, Adição e Eliminação. A partir desses operadores, uma ação α é definida como:

- Identificação da ação, um predicado n-ário podendo conter variáveis.
- *Pré-condições*(PC): um conjunto de literais presentes no estado atual que justificam a aplicação da ação α.
- *Eliminação*(D): um conjunto de literais que são eliminados da descrição do próximo estado pela aplicação da ação α no estado atual.
- *Adição*(A): um conjunto de literais que deve ser adicionado ao próximo estado após a aplicação da ação α no estado atual.

Além do esquema de representação de ações, a definição do problema é completa com os conjuntos de predicados que descrevem os estados Inicial e Final do problema. Para esboçar o funcionamento do planejador STRIPS, utilizamos o exemplo do mundo dos blocos que aparece na Figura 2.2.

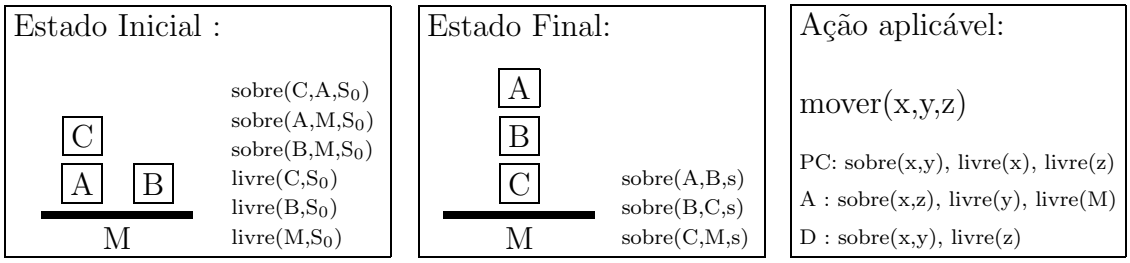

Figura 2.2. *Descrição de um problema do mundo dos blocos utilizado pelo planejador STRIPS.*

O algoritmo do STRIPS utiliza a estratégia de divisão e conquista [18]: escolhe um dos literais do Estado Final e aplica uma seqüência de ações ao estado atual até atingir um estado que contenha esse literal. Depois seleciona o próximo literal do objetivo, e assim sucessivamente até encontrar um estado que contenha todos os literais do Estado Objetivo. Esse processo pode ser descrito informalmente como:

i Seleciona um esquema cujas Pré-Condições sejam satisfeitas no estado atual. A prioridade é para os esquemas em que o conjunto *Adição* contenha literais do Estado Objetivo que ainda não foram satisfeitos.

ii Armazena o nome da ação aplicada na lista de ações aplicadas.

iii Gera um novo estado a partir do atual, adicionando os literais do conjunto *Adição* e eliminando os literais do conjunto *Eliminação*.

iv Se todos os literais do Estado Objetivo estiverem presentes no estado atual, encerra e retorna a lista de ações aplicadas.

v Caso contrário, retorna ao passo 1.

No nosso exemplo do mundo dos blocos, existem três objetivos: *sobre(A,B,s), sobre(B,C,s) e sobre(C,M,s).* O programa tenta unificar as variáveis *x, y e z* de modo que o conjunto *Adição* contenha um ou mais desses objetivos. Digamos que o programa selecione a instância *mover(B,M,C)*, satisfazendo o segundo objetivo com a geração do estado S_1=$\{$*sobre(B,C,*S_1*), sobre(C,A,*S_1*), sobre(A,M,*S_1*), livre(B,* S_1*)*$\}$. Nota-se que apesar de obter um dos objetivos, o planejador realizou uma ação "ingênua", que levará o programa a repetir passos durante a busca do plano desejado. No segundo passo, apenas uma ação é aplicável: *mover(B,C,M)*, gerando

S_2 = {*sobre(B,M,*S_2*), sobre(C,A,*S_2*), sobre(A,M,*S_2*), livre(B,* S_2*, livre(C,* S_2*)*}. Digamos que as próximas ações selecionadas pelo programa sejam *mover(C,A,M)*, *mover(B,M,C)* e *mover(A,M,B)*. O plano encontrado seria:

mover(B,M,C) · *mover*(B,C,M) · *mover*(C,A,M) · *mover*(B,M,C) · *mover*(A,M,B)

Observe-se que esse plano apresenta a repetição de ações, no caso *mover*(B,M,C). O problema de repetir ações durante a busca de um plano é conhecido como Anomalia de Sussman [121, 93], e pode levar o algoritmo a um ciclo sem saída.

A estratégia de busca progressiva utilizada no STRIPS é aparentemente ingênua diante do tamanho do espaço de busca dos problemas de planejamento. Apesar disso, muitos outros planejadores utilizaram essa estratégia, compensando a falta de orientação na busca com heurísticas de encadeamento [30] ou linguagens mais expressivas [90]. O maior sucesso entre os planejadores baseados na idéia original do planejador STRIPS foi o PRODIGY.

2.4.2 PRODIGY

O planejador Prodigy [89, 30] foi desenvolvido no final dos anos oitenta por um projeto homônimo da Universidade de Carneggie Mellon[2].

Esse trabalho preserva a idéia inicial dos operadores STRIPS, representando os estados de um mundo a partir de um conjunto de literais e definindo uma série de operadores (*pré-condição*↔*ação*↔*consequências*) aplicáveis a esses estados. A geração de planos pelo PRODIGY é baseada em busca com *backtracking*, realizada a partir de duas rotinas complementares: a simulação de planos e a regressão encadeada de estados. A simulação de planos é a aplicação de ações a partir do Estado Inicial, procurando atingir o Estado Objetivo (tal qual o STRIPS). A Regressão é uma heurística que gera estados intermediários, a partir do conjunto de pré-condições das ações que supostamente geraram o Estado Final. A Figura 2.3 mostra o esquema de geração do PRODIGY: uma busca em profundidade realizada nos dois sentidos: do estado inicial ao final e vice-versa.

[2]O projeto Prodigy ficou famoso pelo pioneirismo no uso de aprendizagem automática, planejamento em tempo real e por tratar o planejamento automático como um problema de busca. O sucesso do projeto pode ser creditado também ao grande número de pesquisadores envolvidos: Steven Minton, Jaime Carbonell, Jim Blythe, Xuemei Wang, Manuela Veloso, Dan Kahn, Oren Etzioni, Dan Kuokka e Daniel Borrajo, entre outros.

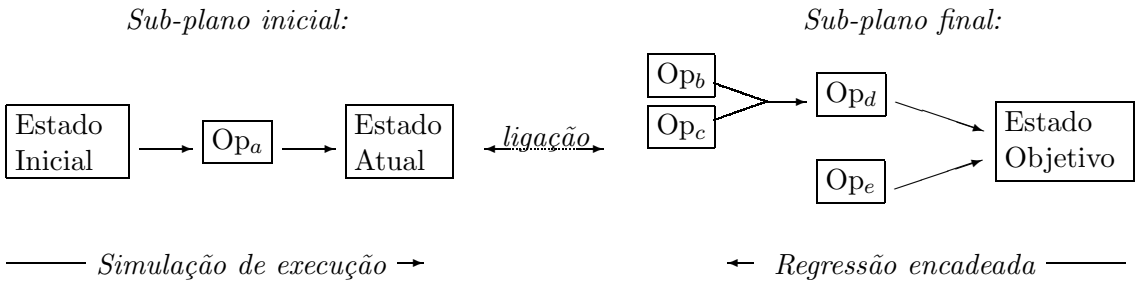

Figura 2.3. *Esquema de geração de planos do PRODIGY*

O Prodigy gera planos através da seguinte rotina:

i Se o estado objetivo é satisfeito pelo estado atual, então retornar o sub-plano inicial.

ii Executar uma das duas opções (*backtracking* - as duas opções devem ser consideradas):

(i) modifica o sub-plano final ou

(ii) liga um operador do sub-plano final ao sub-plano inicial:

- ▹ seleciona um operador α sem nenhum antecessor no sub-plano final e cujas pré-condições sejam satisfeitas no estado atual do sub-plano inicial - (*backtracking* - todos os operadores do sub-plano final devem ser considerados);
- ▹ instancia esse operador α;
- ▹ aplique α ao estado atual do sub-plano inicial, gerando um novo estado atual;

iii chame o algoritmo recursivamente até que todos os operadores do sub-plano final estejam ligados ao sub-plano inicial, ou o sub-plano inicial satisfaça o estado objetivo.

Apesar de extremamente veloz se comparado aos planejadores de sua época, o Prodigy herdou do STRIPS alguns problemas em relação ao espaço de busca. Uma das principais dificuldades do Prodigy é a ordenação dos operadores que forma o

sub-plano final. Uma vez que um desses operadores é movido ao sub-plano inicial, ele passa a ser antecessor dos demais operadores do sub-plano final. Quando dois ou mais operadores puderem ser ligados ao sub-plano inicial, o algoritmo deve escolher a ordem de aplicação entre esses operadores. Para a garantia da completude, todas as combinações devem ser possíveis (*backtracking*), aumentando muito a complexidade do algoritmo para o pior caso [88].

O desenvolvimento do PRODIGY teve muito impacto na comunidade de IA, servindo como base de comparação de desempenho com seus sucessores GRAPHPLAN e SATPLAN. Atualmente, a estratégia de busca do PRODIGY persiste em planejadores que utilizam redes hierárquicas de tarefas para a geração de planos [90, 118].

2.4.3 Planejamento por análise de grafos (GRAPHPLAN)

Em 1995, Blum e Furst propuseram o planejamento automático baseado na análise de grafos e lançaram um algoritmo batizado de GRAPHPLAN [9]. Tal algoritmo chamou a atenção da comunidade de IA por dois motivos:

- é um algoritmo simples e robusto, que apresenta uma velocidade de planejamento melhor que seus antecessores Prodigy [30] e UCPOP [96].
- permitiu a automação da redução de problemas de planejamento ao problema da satisfatibilidade proposicional [57, 60].

O GRAPHPLAN funciona em duas fases: a geração de um grafo representando as ações e estados e a extração do plano a partir desse grafo. O grafo de planejamento é constituído de dois tipos de nodos: os nodos proposicionais e os nodos que representam ações, distribuídos em níveis diferentes. Os nodos proposicionais aparecem nos níveis pares e representam o conhecimento do mundo em um determinado instante de tempo. Nos níveis ímpares, aparecem os nodos que representam as ações cujos pré-requisitos estão presentes no nível anterior.

2.4.3.1 Expansão do grafo de planejamento

A expansão do grafo de planejamento é feita a partir da idéia dos operadores STRIPS, com a exceção de que não há lista de exclusão. A cada nova etapa do

processo de expansão são gerados dois novos níveis no grafo: um nível com as ações cujas pré-condições são satisfeitas no nível atual e um nível com as conseqüências dessas ações. Todas as ações com pré-requisitos presentes no nível atual são aplicadas, permitindo o processamento paralelo de ações. Muitas dessas ações são mutuamente exclusivas (*mutex*) e o controle desse conflito é a base da geração de planos por análise de grafos. A exclusão mútua adotada pelo GRAPHPLAN aparece na Figura 2.4 e segue as seguintes definições:

- Duas ações são consideradas *mutex*, caso satisfaçam os seguintes critérios:
  - Efeitos inconsistentes: o efeito de uma ação é a negação do efeito de outra ação;
  - Interferência: uma ação elimina a pré-condição de outra ação.
  - Pré-condições inconsistentes: duas ações do nível i são geradas a partir de pré-condições que são mutuamente exclusivas no nível $i-1$.
- Duas proposições são consideradas mutuamente exclusivas se uma for a negação da outra, ou se suas ações geradoras forem mutuamente exclusivas.

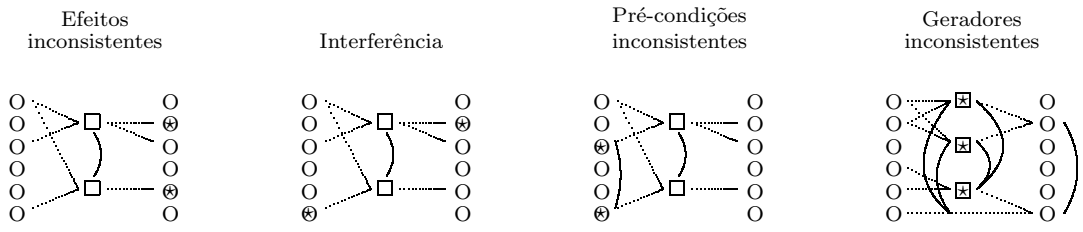

Figura 2.4. *Conflitos entre ações de um mesmo nível em grafos de planejamento. Círculos representam proposições e quadrados representam ações. Arcos representam proposições ou ações mutuamente exclusivas. Estrelas representam ações ou proposições inconsistentes.*

Observe-se que o desenho mais à direita da Figura 2.4 representa as duas proposições mutuamente exclusivas pela herança da inconsistência de suas ações geradoras. Mesmo que essas proposições não sejam diretamente inconsistentes (a negação uma da outra), elas se tornam inconsistentes pelos pais. Nesse mesmo desenho, aparece uma aresta do nível $2i$ diretamente ao nível $2i + 2$ - essa aresta representa uma *Ação de Persistência*. Ações de Persistência utilizam a lei da inércia para supor que uma proposição não sofre alterações entre um estado e o seu sucessor.

2.4.3.2 O problema do jantar surpresa

Para demonstrar a geração do gráfico de planejamento, usamos o exemplo da Tabela 2.1[3]: uma pessoa preparando um jantar-surpresa para a sua esposa, enquanto ela está descansando. Os seus objetivos são: remover o lixo($\neg Lixo$), cozinhar o *Jantar*

Estado Inicial	Lixo ∧ MLimpas ∧ Silêncio
Estado Objetivo	¬Lixo ∧ Jantar ∧ Presente
Cozinhar	*p*: MLimpas *e*: Jantar
Embrulhar	*p*: Silêncio *e*: Presente
Carregar	*p*: *não possui pré-condições* *e*: ¬Lixo ∧ ¬MLimpas
Arrastar	*p*: *não possui pré-condições* *e*: ¬Lixo ∧ ¬Silêncio

Tabela 2.1. *Descrição do problema do jantar.* p = *pré-condições,* e = *efeitos.*

e embrulhar o *Presente*. Existem quatro ações aplicáveis ao domínio: *Cozinhar, Embrulhar, Carregar e Arrastar (o lixo)*. Cozinhar tem a pré-condição das mãos estarem limpas (*MLimpas*) e, como conseqüência, produz o *Jantar*. Embrulhar o *Presente* exige *Silêncio* para evitar acordar a esposa e estragar a surpresa. Carregar o lixo para fora satisfaz $\neg Lixo$ mas suja as mãos ($\neg MLimpas$). Arrastar o lixo para fora evita sujar as mãos, mas o enferrujado carrinho do lixo faz barulho ($\neg Silêncio$). Inicialmente, a pessoa tem as mãos limpas (*MLimpas*), existe lixo acumulado (*Lixo*)

[3]Exemplo criado por Daniel S. Weld em [129]

e a casa está silenciosa (*Silêncio*). Assumimos um mundo fechado [103], negando todas as outras proposições.

A figura 2.5 apresenta o grafo de expansão do problema do jantar-surpresa. O primeiro passo do algoritmo gera o nível 1, com todas as ações cujas pré-condições são satisfeitas no nível 0 (inicial), e o nível 2, com as conseqüências dessas ações. Observe-se que os arcos representam os conflitos entre as ações do nível 1 e o conflito entre as proposições no nível 2. *Carregar* é mutuamente exclusiva com a *Manutenção* de Lixo e de MLimpas, porque gera as suas respectivas negações. *Carregar* gera ¬MLimpas, conflitando por interferência com *Cozinhar*. *Arrastar* é inconsistente com as *Manutenções* de Lixo e de Silêncio por gerar as suas respectivas negações. *Arrastar* também causa inteferência em *Embrulhar* ao gerar a negação de sua pré-condição Silêncio. No nível 2, as exclusões mútuas são entre as proposições de sinal contrário e entre as proposições causadas por ações mutuamente exclusivas.

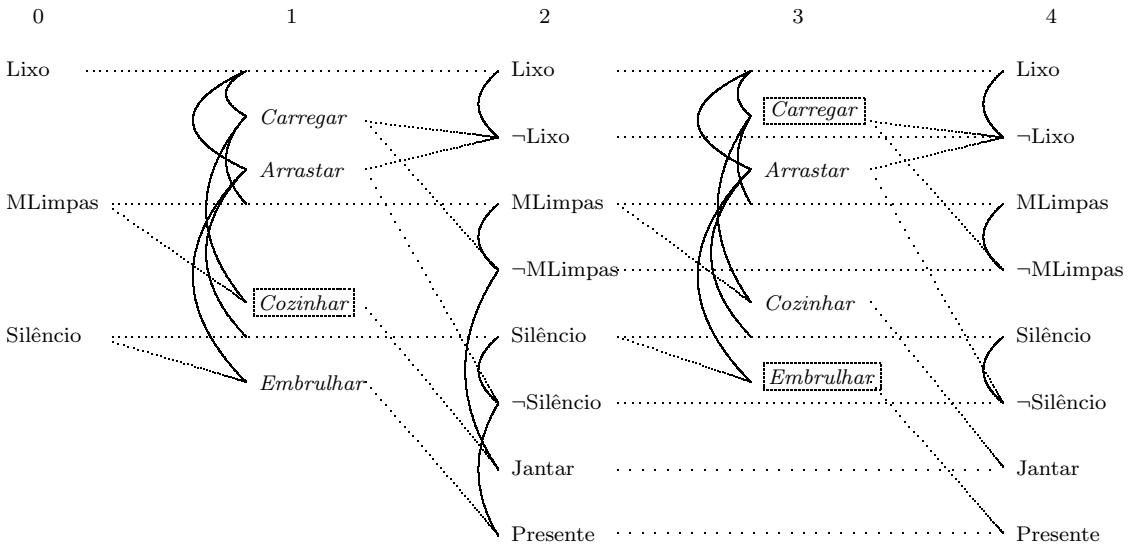

Figura 2.5. *Grafo de planejamento para o problema do jantar.*

O fato de todas as proposições do Estado Objetivo (¬*Lixo* ∧ *Jantar* ∧ *Presente*) estarem presentes no nível 2 e o fato de não haver exclusão mútua entre elas possibilitam a existência de um plano. Nesse caso, o algoritmo passa para a fase II: a extração do plano no grafo atual (com os níveis 0, 1 e 2).

2.4.3.3 Extração da solução

Quando o GRAPHPLAN gera um nível com todos os literais de um objetivo e esses literais não são mutex, então é possível que o grafo contenha um plano válido para o problema em questão. Essa condição é necessária, mas insuficiente para afirmar que o plano existe. Para tal, é necessário que não haja conflito entre as ações que geraram as proposições do objetivo. A verificação da ausência de conflitos é a chamada *Extração da Solução*: um processo de busca que parte do nível mais alto do grafo ao nível mais baixo, verificando a presença de conflitos. Caso algum conflito seja detectado, o algoritmo cancela a extração da solução e retorna ao processo de expansão do grafo até que um outro nível contenha as proposições do objetivo e assim por diante. Para cada proposição do nível i, o GRAPHPLAN seleciona uma ação geradora do nível $i-1$ e verifica se essa ação não é mutuamente exclusiva com as outras ações geradoras selecionadas no mesmo nível $i-1$. Caso não haja conflito, o algoritmo repete o processo com as pré-condições dessas ações selecionadas e assim por diante, até chegar ao nível 0. Caso chegue no nível 0, o plano existe. Note-se que uma proposição do objetivo pode ser gerada por mais de uma ação, obrigando o algoritmo a considerar todas as combinações de ações geradoras.

No caso do exemplo do jantar, existem três proposições do objetivo no nível 2: ¬Lixo foi gerado por *Carregar* e por *Arrastar*, Jantar foi gerado por *Cozinhar* e Presente foi gerado por *Embrulhar*. Neste momento, existem dois grupos de ações geradoras a serem considerados: {*Carregar, Cozinhar e Embrulhar*} e {*Arrastar, Cozinhar e Embrulhar*}. Note-se que nenhum dos grupos é consistente, pois *Carregar* é mutuamente exclusivo com *Cozinhar* e *Arrastar* é mutuamente exclusivo com *Embrulhar*. Como não há um conjunto de ações consistentes, a extração da solução falha e o algoritmo expande o grafo de planejamento gerando os níveis 3 e 4.

Embora nenhum novo literal tenha sido gerado, a diferença entre os níveis 2 e 4 está no número de exclusões mútuas entre os literais. Note-se, por exemplo, que não há mais exclusão mútua entre Jantar e ¬MLimpas. Isso ocorre porque Jantar foi gerado por uma Ação de Persistência que não possui pré-condição conflitante com as ações geradoras de ¬MLimpas: *Carregar* e uma Ação de Persistência. A presença das Ações de Persistência reduz a ocorrência de conflitos entre as ações geradoras, uma vez que essas ações só são mutuamente exclusivas quando possuem pré-condições mutuamente exclusivas.

No nível 4, o algoritmo novamente encontra as condições que possibilitam a extração de um plano, porém com um número um pouco maior de conjuntos de ações geradoras. As ações geradoras são: ¬Lixo é gerado por {*Carregar, Arrastar e Ação de Persistência*}, Jantar é gerado por {*Cozinhar e Ação de Persistência*} e Presente é gerado por {*Embrulhar e Ação de Persistência*}. Todos esses conjuntos devem ser combinados na tentativa da extração da solução, gerando 3x3x2 = 12 combinações. Esse aumento no número de possibilidades de ações geradoras aumenta as chances de que algum plano seja encontrado, mas aumenta também o trabalho de extração desse plano.

Usaremos, como exemplo, o conjunto {*Carregar, Ação de Persistência e Embrulhar*}. Nenhuma dessas ações são mutuamente exclusivas no nível 4, então verificamos a relação entre as suas pré-condições: *Carregar* não possui pré-condições, Jantar é a pré-condição da *Ação de Persistência* e Silêncio é a pré-condição de *Embrulhar*. Temos agora um novo conjunto de literais a verificar no nível 2: Jantar e Silêncio. Todas as combinações de ações geradoras desse conjunto devem ser consideradas, mas o exemplo apresenta apenas 1: *Cozinhar* e *Ação de Persistência*. As ações desse novo conjunto não são mutuamente exclusivas e suas pré-condições são MLimpas e Silêncio, respectivamente. Como essas pré-condições não são mutuamente exclusivas e estão no nível 0, um plano foi encontrado. As ações que representam o plano encontrado estão realçados por retângulos pontilhados na Figura 2.5. Note-se que o plano encontrado está parcialmente ordenado, pois aparecem duas ações no nível 3. No momento de relatar o resultado, o programa pode optar por {*Cozinhar, Carregar e Embrulhar*} ou {*Cozinhar, Embrulhar e Carregar*}.

Além da estratégia original apresentada acima, o GRAPHPLAN foi otimizado e amplamente discutido pela comunidade de IA nos últimos anos. Alguns tópicos continuam estimulando os pesquisadores: a otimização dos algoritmos de regressão [83], o tratamento de conflitos durante a geração do grafo de planejamento [129] e a adaptação do GRAPHPLAN ao suporte a conhecimento incompleto [117], entre outros. Esses avanços não serão descritos aqui, pois representam otimizações e adaptações que não alteram a idéia central do GRAHPLAN.

2.4.4 Planejamento por satisfatibilidade (SATPLAN)

Como todo o problema NP-Completo, o planejamento automático pode ser reduzido a um problema de satisfatibilidade [18]. Partindo deste princípio, Henry Kautz e Bart Selman propuseram em [57] a formalização de problemas de planejamento automático como conjuntos de cláusulas proposicionais. A satisfação desse conjunto de cláusulas por um algoritmo chamado GSAT surpreendeu os autores pela velocidade superior aos demais planejadores da época. Desde então, muitos planejadores vem explorando esta técnica, incluindo os famosos MEDIC [26] e Blackbox [59].

O esquema básico de um planejador baseado em algoritmos de satisfatibilidade é mostrado na Figura 2.6. O *Compilador* recebe a descrição dos estados Inicial e Final, das Ações Aplicáveis e gera uma fórmula normal conjuntiva(FNC) proposicional equivalente a um plano de tamanho estimado n. Uma tabela de símbolos é gerada com a correspondência entre as variáveis proposicionais e as instâncias do plano. O *Simplificador* reduz o tamanho da fórmula através de técnicas de otimização em tempo polinomial: eliminação de literais e propagação de cláusula única [35]. O *Gerador* tenta encontrar uma atribuição que satisfaça a fórmula - um plano. Caso nenhuma atribuição gerada satisfaça o plano atual, o processamento retorna ao *Compilador*, que aumenta o tamanho estimado do plano e gera uma nova fórmula. Caso a fórmula tenha sido satisfeita, o *Decodificador* traduz a fórmula em um plano aplicável ao Estado Inicial do problema.

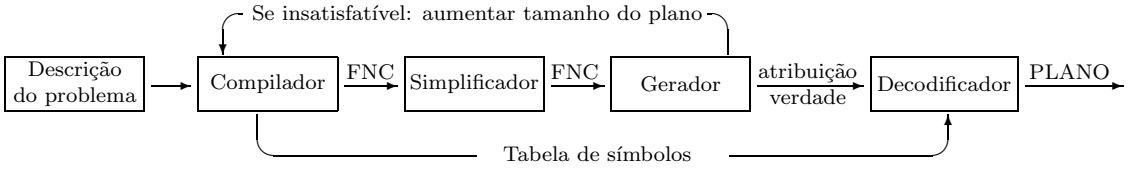

Figura 2.6. *Esquema de geração de planos por satisfatibilidade.*

2.4.4.1 Codificação de planos como FNC

Um compilador SAT traduz uma descrição em alto nível de um problema em uma fórmula normal conjuntiva proposicional. Tal qual os compiladores de linguagens

de programação, o compilador SAT pode ser projetado para gerar rapidamente um código (FNC) ou para dedicar mais tempo à otimização desse código. O tamanho da FNC gerada está associado ao tempo em que o gerador de atribuições-verdade levará para satisfazê-la. Quando falamos em tamanho de uma FNC devemos escolher uma métrica: o número de variáveis, o número de cláusulas ou o número de literais presentes na fórmula. Dois fatores determinam o tamanho da fórmula gerada por um compilador SAT: a forma de representação utilizada e a otimização da fórmula gerada.

Primeiramente, mostramos as opções para a codificação do problema:

- Ações podem ser representadas a partir das codificações conhecidas como: *regular, simply split, overloaded split e bitwise*. Tais representações são detalhadas na Seção 2.4.4.2.
- Axiomas de persistência podem ser representados de forma clássica ou explicativa.

A codificação adotada será aplicada sobre um modelo baseado em *fluents*, com intervalos discretos e não negativos de tempo. Os estados serão vinculados aos instantes pares de tempo, enquanto as ações estarão vinculadas aos instantes ímpares. No exemplo anterior do problema do jantar, a variável proposicional Lixo_0 representa que existe Lixo no estado inicial, $\neg\text{Lixo}_2$ representa que não há lixo após a execução das ações do instante 1 e Carregar_1 representa que a ação Carregar é executada no instante de tempo 1. Os modelos de planejamento por satisfatibilidade são compostos pelos seguintes conjuntos de axiomas:

INICIAL O Estado Inicial de um problema é descrito de forma completa e é vinculado ao instante 0 de tempo. Além disso, assumimos um mundo fechado negando todas as proposições que não aparecem descritas no estado inicial. Exemplo: $(Lixo_0 \wedge MLimpas_0 \wedge Silêncio_0 \wedge \neg Jantar_0 \wedge \neg Presente_0)$.

FINAL As proposições do Estado Objetivo são associadas ao instante de tempo $2n$ para um plano de tamanho estimado n. Supondo que o exemplo do jantar-surpresa possua um plano de tamanho $n = 1$ (equivalente ao primeiro nível de ações no grafo de planejamento utilizado pelo GRAPHPLAN) teremos: $(\neg \text{Lixo}_2 \wedge \text{Jantar}_2 \wedge \text{Presente}_2)$.

AÇÕES A descrição das ações aplicáveis ao domínio do problema é baseado em STRIPS, ou seja, uma ação sucede suas pré-condições e precede suas conseqüências. Para um instante de tempo t entre 1 e $2n - 1$, uma determinada ação possui um axioma que implica em suas pré-condições no instante $t - 1$ e suas conseqüências no instante $t + 1$. Exemplo:

$$(\neg \text{Carregar}_1 \vee Jantar_2) \wedge (\neg Carregar_1 \vee MLimpas_0)$$

Observe-se que o número de cláusulas geradas por tal esquema de representação de ações é vinculado ao número de objetos e ao comprimento estimado do plano. Cada variável proposicional que faz parte das pré-condições ou conseqüências de uma ação gera uma cláusula, e o número de cláusulas deve ser multiplicado pelo número de níveis em que a ação pode ocorrer.

2.4.4.2 Representação de ações

As ações podem ser representadas através das codificações *regular, simply split, overloaded split e bitwise*.

Regular: cada instãncia da ação é representada por uma variável lógica diferente, gerando $n|\text{Ações}||Objetos|^P$ variáveis onde: n é o número de intervalos ímpares de tempo, |Ações| representa o número de ações aplicáveis ao domínio, $|Objetos|$ o número de objetos presentes no domínio e P representa o número máximo de parâmetros para cada esquema de ação.

No exemplo do mundo dos blocos da Figura 2.2, a ação de mover o bloco C do topo de A para o topo de B - mover(C,A,B) - pode ser codificada como $MoverCAM_1$, e junto aos axiomas de persistência temos:

$$(\neg moverCAB_1 \vee (sobreCA_0 \wedge livreB_0 \wedge livreC_0))$$

$$\wedge\ (\neg moverCAB_1\ \vee\ (sobreCB_2\ \wedge\ livreA_2))$$

Embora esta seja uma representação simples, o grande número de variáveis lógicas geradas prejudica o desempenho dos geradores de atribuições verdade. Isso obrigou a criação de codificações com um número menor de variáveis.

Simple Split: esta codificação propõe dividir cada *fluent* n-ário por n *fluents* unários. Por exemplo, a ação mover(C,A,B), no instante de tempo 1 do exemplo anterior, pode ser escrita como: $moverArg1C_1 \wedge moverArg2A_1 \wedge moverArg3B_1$. Considere-se agora uma outra instância da ação *mover(x,y,z)* do mesmo problema, digamos mover(C,A,M): $moverArg1C_1 \wedge moverArg2A_1 \wedge moverArg3M_1$. Note-se que as variáveis lógicas moverArg1C_1 e moverArg2A_1 poderão ser simplificadas, pois aparecem duplicadas na fórmula final. À primeira vista, a substituição parece pouco eficiente, mas, de fato, o número de variáveis lógicas resultantes nesta codificação cai para $n|\text{Ações}||Objetos|P$. Note-se que apenas instâncias do mesmo esquema de ações permitem a simplificação entre as variáveis geradas. Uma alternativa para estender essa simplificação é a *sobrecarga de operadores, conforme demonstramos a seguir.*

Overloaded Split: é uma notação baseada na divisão simples dos *fluents* exceto pelo uso de um meta operador que identifica as ações aplicadas em um instante de tempo. Por exemplo, a ação mover(C,A,B) passa a ser representada por: $AgirMover_1 \wedge Arg1C_1 \wedge Arg2A_1 \wedge Arg3B_1$. Supomos que exista outro esquema de ação instanciável com os mesmos argumentos, digamos desempilhar(C,B): $AgirDesempilhar_1 \wedge Arg1C_1 \wedge Arg2A_1$. Esta notação reduz o número de variáveis para $n(|\text{Ações}| + |Objetos|P)$.

Bitwise: é um padrão de codificação que associa uma conjunção de bits a cada instância de ação aplicável a um determinado nível do plano. Por exemplo, se tivermos quatro ações aplicáveis no instante de tempo 1, $\neg bit1_1 \wedge \neg bit2_1$ representa a primeira ação, $bit1_1 \wedge \neg bit2_1$ representa a segunda ação e assim sucessivamente. O primeiro nível de ações no exemplo do mundo dos blocos da Figura 2.2 pode ser representado como:

bit1	bit2	Ação
0	0	mover(C,A,B)
0	1	mover(C,A,M)
1	0	mover(B,M,C)

A codificação *bitwise* permite a representação da instância de um ação com $\lceil log_2|\text{Ações}||Objetos|^P\rceil$ símbolos proposicionais.

2.4.4.3 Axiomas de persistência

Os axiomas que controlam os *fluents* não afetados pela aplicação de uma ação a um determinado domínio podem ser escritos de forma clássica [80] ou explicativa [44, 106].

Axiomas de persistência clássicos: descrevem os *fluents* inalterados a partir da aplicação de uma ação em um domínio. No nosso exemplo do mundo dos blocos, a ação $mover(C, A, M)$ não altera o status do bloco B:

$$livreB_0 \wedge moverCAM_1 \supset livreB_2$$

Observe-se que é necessário o acréscimo de um axioma para cada instância de ação aplicável ao domínio em cada intervalo ímpar de tempo, e que a persistência de uma informação depende da ocorrência de uma ação que não a afete. Caso não ocorra nenhuma ação em um determinado instante t de tempo, nenhum *fluent* persistirá no instante seguinte a t. Isso força a presença de axiomas que garantam que pelo menos uma ação ocorra(PM1) em cada instante ímpar de tempo.

PM1: os axiomas PM1 são a disjunção de todas as instâncias de ações aplicáveis a cada instante de tempo. Por exemplo:

$$moverCAM_1 \vee moverCAB_1 \vee moverBMC_1$$

O plano resultante da utilização dos axiomas de persistência clássicos é totalmente ordenado e muito semelhante aos planos gerados pela codificação linear proposta por Kautz e Selman em [61].

Axiomas de persistência explicativos: ao invés de enumerar os *fluents* não afetados pela aplicação de uma ação, os axiomas explicativos enumeram as ações responsáveis pelas diferenças entre dois estados consecutivos. Por exemplo:

$$livreB_0 \wedge \neg livreB_2 \supset (moverCAB_1 \vee moverDEB_1 \vee ... \vee moverFGB_1)$$

Cada *fluent* presente em um nível $t \geq 2$ deve ter um axioma de persistência explicativo equivalente. A diferença entre os valores-verdade da variável associada a esse *fluent* nos instantes de tempo $2t$ e $2t-2$ implica na ocorrência de uma ação no instante $t-1$. Caso nenhuma ação tenha ocorrido, os axiomas explicativos serão tratados como operadores nulos (contrapositiva). Dessa forma, não há necessidade de axiomas que garantam a ocorrência de uma ação para cada instante ímpar de tempo.

Os axiomas explicativos não exigem que os *fluents* permaneçam inalterados após uma ação que não os altera, permitindo o paralelismo. Quaisquer ações cujas pré-condições estão presentes em um instante de tempo t e que não possuem efeitos inconsistentes podem ser aplicadas em paralelo no instante $t+1$. Um dos problemas desse paralelismo é a geração de planos não linearizáveis: suponha a ação $A1$ com uma pré-condição α e uma conseqüência β, e outra ação $A2$ com uma pré-condição $\neg\beta$ e uma conseqüência $\neg\alpha$. Essas duas ações podem ser realizadas em paralelo, mas equivalente ao problema de interferência do GRAPHPLAN, não existe uma ordem válida entre elas $\{A1; A2\}$ ou $\{A2; A1\}$. A solução para esse problema é a adoção de axiomas de exclusão entre as ações que não podem ocorrer simultaneamente.

Exclusão: a exclusão pode ser feita adicionando-se disjunções entre instâncias de ações aplicáveis a cada instante ímpar de tempo, por exemplo: $\neg A1 \vee \neg A2$. Os axiomas de exclusão podem ser adicionados a todos os pares de ações, gerando uma ordem total no plano gerado, ou mantendo a capacidade de paralelismo, adicionando-se apenas a pares conflitantes.

Normalmente é usada a exclusão apenas das instâncias conflitantes, mas essa escolha deve levar em conta a codificação utilizada na instanciação dos esquemas de ações. A representação *Simple Split* exige a exclusão entre todos os pares de ações, pois não há mais uma única variável lógica para cada ação. O mesmo problema ocorre com a notação *Overloaded Split*. A notação *Regular* permite o uso de exclusão

apenas de ações conflitantes, enquanto a notação *Bitwise* dispensa os axiomas de exclusão ao garantir naturalmente uma ordem total ao plano gerado.

Escolha da notação: Embora o menor número de variáveis pareça determinar a escolha da notação a ser adotada, alguns estudos indicam que as notações *Regular* e *Simply Split* podem ser melhores que as demais [26]. Isso decorre da aplicação de métodos de simplificação da FNC em tempo linear [35]. Métodos como a fatoração apresentada na Seção 2.4.4.4 não têm impacto nas notações *Bitwise* e *Overloaded Split*, mas conseguem reduzir muito o tamanho das fórmulas geradas com as notações *Regular* e *Simple Split*. A Figura 2.2 apresenta o tamanho das fórmulas geradas para todas as combinações entre codificação de ações e axiomas de persistência.

	n° Variáveis	**Clássico**	**Explicativo**
Regular	$n\mathcal{F} + n\mathcal{A}$	PM1 $O(n\mathcal{F}\mathcal{A})$	Exclusão $O(n\mathcal{F}\mathcal{A} + n\mathcal{A}^2)$
Simple Split	$n\mathcal{F} + n\lvert Ops\rvert A_o\lvert Dom\rvert$	PM1 $O(n\mathcal{F}\mathcal{A}A_o + nA_o^{\mathcal{A}}\mathcal{A})$	Exclusão $O(n\mathcal{F}A_o^{\mathcal{A}} + n(A_o\mathcal{A})^2)$
S. Split fatorado	$n\mathcal{F} + n\lvert Ops\rvert A_o\lvert Dom\rvert$	PM1, não parcial $O(n\mathcal{F}\mathcal{A}A_o + n\lvert Ops\rvert\lvert Dom\rvert^2 A_o)$	exclusão, não parcial $O(n\mathcal{F}A_o^{\mathcal{A}} + n\lvert Ops\rvert^2\lvert Dom\rvert^2 A_o)$
Overloaded Split	$n\mathcal{F} + n(\lvert Ops\rvert + A_o\lvert Dom\rvert)$	PM1 $O(n\mathcal{F}\mathcal{A}A_o + nA_o^{\mathcal{A}}\mathcal{A})$	exclusão $O(n\mathcal{F}(\mathcal{A}A_o)^2 + n\mathcal{F}A_o^{\mathcal{A}}\mathcal{A})$
Over. Split fatorado	$n\mathcal{F} + n(\lvert Ops\rvert + A_o\lvert Dom\rvert + 1)$	PM1, não parcial $O(n\mathcal{F}\mathcal{A}A_o + n\lvert Dom\rvert^2 A_o)$	exclusão, não parcial $O(n\mathcal{F}A_o^{\mathcal{A}}\mathcal{A} + n\lvert Dom\rvert^2(A_o + \lvert Ops\rvert^2))$
Bitwise	$n\mathcal{F} + nlog_2\mathcal{A}$	$O(n\mathcal{F}\mathcal{A}log_2\mathcal{A})$	$O(n\mathcal{F}(log_2\mathcal{A}^{\mathcal{A}}))$

$|Ops|$ – número de operadores
$|Pred|$ – número de predicados
$|Dom|$ – número de objetos no domínio
n – número de instantes de tempo ímpar
A_o – aridade máxima dos operadores
A_p – aridade máxima dos predicados
$\mathcal{A}$ – n° esquemas de ações ($|Ops||Dom|^{A_o}$)
$\mathcal{F}$ – n° de fluents ($|Pred||Dom|^{A_p}$)

Tabela 2.2. *Tamanho para as codificações de ações e axiomas de persistência.*

2.4.4.4 Otimização da FNC equivalente a um plano

A representação das ações e dos axiomas de persistência nos permitem a escolha entre oito modelos distintos de notação: {*Regular, Simple Split, Overloaded Split, Bitwise*} x {*Clássico, Explicativo*}. Embora o menor número de variáveis seja um parâmetro importante na escolha da notação, outros fatores devem ser considerados.

As codificações *Bitwise* e *Overloaded Split* geram o menor número de variáveis lógicas, entretanto aumentam o número de cláusulas ou o tamanho de cada cláusula. Considerem-se os axiomas que garantem a execução de pelo menos uma ação a cada

instante ímpar de tempo: uma disjunção de todas as instâncias de ações aplicáveis ao domínio. Imagine-se um axioma PM1 na notação *Bitwise*, e considere-se um instante de tempo em que todas as instâncias podem ser aplicadas. A substituição de cada variável lógica pelo axioma que representa a sua respectiva ação produz uma disjunção que pode crescer exponencialmente ao ser convertida em FNC. Uma forma de reduzir esta explosão exponencial é o uso de apenas parte da conjunção que representa a instância de uma ação. Esta divisão da instância de uma ação é chamada de fatoração e apresenta excelentes resultados em todas as notações, exceto a *Bitwise*[4].

Fatoração de axiomas de ação e persistência: a idéia é utilizar apenas as partes da instância de uma ação que tenham influência direta em um determinado *fluent*. Considere-se, por exemplo, o esquema da ação $mover_1(x, y, z)$ do exemplo da Figura2.2, instanciado como $mover_1(C, A, M)$ na notação *Simple Split*:

$$moverArg1C_1 \ \wedge \ moverArg2A_1 \ \wedge \ moverArg3M_1 \ \supset \ livreA_2$$

Observe-se que os argumentos 1 e 3 são irrelevantes na conseqüência de todos os axiomas unificáveis com $mover_1(x, A, z)$, sendo eliminados pela fatoração do axioma original. Esta eliminação gera um novo axioma muito mais compacto:

$$moverArg2A_1 \ \supset \ livreA_2$$

A fatoração dos axiomas de persistência segue o mesmo raciocínio. Considere-se o axioma de persistência clássico apresentado na seção 2.4.4.3, escrito na notação Simple Split:

$$livreB_0 \ \wedge \ moverArg1C_1 \ \wedge \ moverArg2A_1 \ \wedge \ moverArg3M_1 \ \supset \ livreB_2$$

Neste axioma, a informação sobre qual o objeto a ser movido ($moverArg1C_1$) e a origem desse objeto ($moverArg2A_1$) não interfere no status do bloco B, podendo ser excluída do axioma sem prejuízo à sua função:

$$livreB_0 \ \wedge \ moverArg3M_1 \ \supset \ livreB_2$$

[4]A notação *Bitwise* não permite a identificação dos argumentos de uma cláusula, que é a base da otimização por fatoração.

Fatoração de axiomas de exclusão: a fatoração dos axiomas de exclusão é feito a partir da substituição desses axiomas por outros que relacionam apenas os argumentos dessas ações. Ao invés de garantir a exclusão entre duas ações, os novos axiomas garantem a exclusão de partes dessas ações. Considere-se o mesmo exemplo do esquema da ação *mover(x,y,z)* no instante 1 da Figura 2.2: a notação Simple Split nos permite escrever o axioma

$$\neg moverArg1C_1 \ \vee \ \neg moverArg1B_1$$

A presença de um dos argumentos de uma ação é suficiente para assumirmos a ocorrência dessa ação, garantindo a exclusão entre instâncias de um mesmo esquema de ações(operadores). Se quisermos garantir que somente um operador seja aplicado a cada instante ímpar de tempo t, devemos, então, acrescentar um axioma de exclusão para cada par de operadores distintos α e β:

$$\neg \alpha Arg1x_t \ \vee \ \neg \beta Arg1y_t$$

Fatoração dos axiomas PM1: os axiomas que garantem que pelo menos uma ação ocorra a cada instante de tempo ímpar causam uma explosão exponencial ao serem convertidos para FNC. Porém, o uso de apenas um dos argumentos de cada operador permite uma boa redução no tamanho da conjunção que representa a PM1. Ao invés do axioma original $moverCAM_1 \vee moverCAB_1 \vee moverBMC_1$ do exemplo da Figura 2.2, usaremos $moverC_1 \ \vee \ moverB_1$. Note-se que o uso apenas do primeiro argumento dos operadores reduz o número de cláusulas, porque diminui a combinação entre os objetos do domínio dentro de cada predicado.

Execução não parcial de ações: quando fatoramos a instância de uma ação assumimos que a presença de um dos argumentos dessa ação acarreta a presença dos outros argumentos. Essa premissa deve estar expressa também através de um axioma. Para o operador *mover* no instante de tempo 1 do exemplo do mundo dos blocos, temos:

$$(moverArg1C_1 \ \vee \ moveArg1B_1) \Leftrightarrow$$
$$(moverArg2A_1 \ \vee \ moveArg2M_1) \Leftrightarrow$$
$$(moverArg3M_1 \ \vee \ moveArg3B_1 \ \vee \ moveArg3C_1)$$

Com esse tipo de axioma, a presença de qualquer *fluent* que faça parte de uma notação *Simple Split* ou *Overloaded Split* garante a verdade em relação a uma instância do operador *mover*.

2.4.4.5 Algoritmos de satisfatibilidade

Um vez que um problema de planejamento automático seja convertido em uma fórmula normal conjuntiva, aplica-se algum algoritmo de satisfatibilidade para a extração do plano. O estudo de tais algoritmos agrega uma vasta e fiel comunidade científica que a cada ano produz algoritmos mais rápidos e precisos. A discussão acerca das técnicas de satisfatibilidade existentes tomaria demasiado espaço e desviaria o foco sobre planejamento automático, obrigando-nos a uma enunciação breve da idéia central desses algoritmos. Textos mais completos podem ser encontrados em [107, 58].

As técnicas conhecidas para satisfação de FNC podem ser divididas em dois grupos: algoritmos sistemáticos e estocásticos.

Sistemáticos: O algoritmo mais conhecido para satisfação de fórmulas proposicionais é o DPLL [22], que simplifica as fórmulas originais a partir dos conceitos de cláusula unitária e literal puro descritos abaixo.

Sendo ϕ uma fórmula normal conjuntiva, temos: se uma das cláusulas de ϕ for um literal P, podemos assumir que esse literal é verdadeiro - nesse caso dizemos que P é uma cláusula unitária. Se existe um outro literal Q tal que toda a cláusula em ϕ se refere a Q ou $\neg Q$ na mesma polaridade - todas as referências são verdadeiras ou todas falsas - então Q (ou $\neg Q$) é dito um literal puro. No exemplo abaixo, podemos ver esses dois casos:

$$\phi = (A \vee B \vee \neg E) \wedge (B \vee \neg C \vee D) \wedge (\neg A) \wedge (B \vee C \vee E) \wedge (\neg D \vee E)$$

$\neg A$ é uma cláusula unitária e B é um literal puro. Usamos a notação $\phi(u)$ para representar uma fórmula ϕ onde u é verdadeiro. Nesse caso, podemos simplificar todas as disjunções onde o literal u aparece:

$$\phi(\neg A) = (B \vee \neg C \vee D) \wedge (B \vee C \vee E) \wedge (\neg D \vee E)$$

$$\phi(B) = (\neg A) \wedge (\neg D \vee E)$$

O algoritmo DPLL, esboçado na figura 2.7, é um algoritmo de busca em profundidade no espaço parcial de atribuições-verdade, utilizando *backtracking* e as heurísticas de cláusula unitária e literais puros. Métodos como o Tableau [20] e Satz [69] são exemplos de implementações baseadas em DPLL - com as necessárias adequações de estrutura de dados e indexação. Várias heurísticas foram propostas para otimizar a busca em algoritmos baseados em DPLL, algumas delas especialmente talhadas para lidar com fórmulas utilizadas em planejadores automáticos [41, 35] Um dos problemas do DPLL é que a busca em profundidade com *backtracking* pode acarretar um tempo de execução demasiado longo e é diretamente dependente da escolha das variáveis a serem analisadas durante esse processo de busca. Uma alternativa para esses problemas são os algoritmos de satisfatibilidade baseados em probabilidade.

```
1.  função booleana dpll(FNC φ)
2.  se (φ está vazio) então retorna VERDADE
3.  senão se(existe uma cláusula vazia em φ) então retorna FALSO
4.  senão se(existe um literal puro p em φ) então retorna dpll(φ(p))
5.  senão se(existe uma cláusula unitária {p} em φ) então retorna DPLL(φ(p))
6.  senão
7.      v = uma variável de φ
8.      se dpll(φ(v)) então retorna VERDADE
9.      retorna dpll(φ(¬v))
```

Figura 2.7. *Algoritmo DPLL - satisfação de fórmulas normais conjuntivas através dos conceitos de literais puros e cláusulas unitárias.*

Estocásticos: Os métodos estocásticos realizam buscas locais no espaço de atribuições-verdade, incluindo movimentos randômicos como heurística para fugir dos mínimos locais. Como resultado, os métodos estocásticos são incompletos - em caso de problemas difíceis, esses algoritmos podem esgotar o tempo de busca sem encontrar nenhuma atribuição válida para uma determinada fórmula. Nesse caso, não é possível distinguir uma fórmula insatisfatível de outra cuja satisfação existe, mas é difícil de ser identificada. Apesar da incompletude, os algoritmos de satisfação estocásticos são comumente muito mais rápidos do que seus pares sistemáticos. Este aumento de desempenho justifica a escolha desses algoritmos em problemas cujas

fórmulas conjuntivas não sejam muito extensas e/ou quando o tempo de satisfação for um fator prioritário.

A Figura 2.8 apresenta o algoritmo GSAT, um dos mais populares algoritmos estocásticos conhecidos [61, 107]. A idéia central desse algoritmo é a geração randômica de uma atribuição verdade e de um número limitado de *sub-atribuições* que variam apenas no valor de um literal. O literal cujo valor-verdade será trocado é escolhido a partir do impacto que essa mudança terá sobre a fórmula (Hill-Climbing). Observe-se que a alteração do valor-verdade de apenas um literal pode causar uma atribuição com o mesmo grau de satisfação da anterior (movimento lateral) ou até mesmo gerar uma atribuição com uma grau de satisfação pior que o anterior (regressão). Após uma certa quantidade de alterações frustradas no valor verdade dos literais, o algoritmo GSAT reinicia o processo de busca através da geração randômica de uma nova atribuição. Se vários desses reinícios não forem suficientes para a satisfação de uma fórmula, o algoritmo é encerrado sem nenhuma atribuição válida.

Muitas variações do GSAT foram deenvolvidas [62, 36], incluindo o WALKSAT [54, 55], que usa têmpera simulada como heurística.

```
1.  função booleana gsat(FNC φ, int ciclos, int alteracoes)
2.  para i = 1 até ciclos faça
3.      A = atribuição-verdade gerada randômicamente
4.      para j = 1 até alteracoes faça
5.          se A satisfaz φ então retorna VERDADE
6.          α = selecionar um literal cuja troca de valor-verdade
7.              satisfaça o maior número de cláusulas em φ
8.          modificar A invertendo o valor-verdade atribuido para α
9.  retorna FALSO
```

Figura 2.8. *Algoritmo GSAT - busca Hill-Climbing com reinício randômico no espaço de atribuições-verdade para uma determinada fórmula normal conjuntiva.*

2.4.5 Planejamento por heurística de busca (HSP)

Os primeiros trabalhos com heurísticas de busca na geração de planos foram propostos a partir de 1997 por Blai Bonet e Héctor Geffner [13, 10]. A idéia é realizar uma busca progressiva do estado inicial ao final, utilizando uma função heurística

que estima a distância (custo) em passos do Estado Atual ao Estado Objetivo (veja [92, 94]).

Função heurística: Um forma de calcular uma função heurística para um problema P é considerar um problema P^* mais simples e com a solução ótima computável. Considere-se, por exemplo, um problema P de planejamento baseado em STRIPS cuja distância de cada estado intermediário até o Estado Objetivo é dada por $h(s)$. Sabemos que calcular a distância de um estado intermediário até o objetivo em um planejamento STRIPS é um problema NP-difícil. Agora considere-se um outro problema P^*, idêntico a P, mas onde os operadores *Eliminação* são desconsiderados: cada ação aplicável a P^* pode apenas acrescentar *fluents* aos estados intermediários e a solução de P^* é obtida quando um estado intermediário contiver todos os fluents do Estado Objetivo.

Uma vez que a função h^* é o limite inferior para a função original $h(s)$, podemos admitir h^* como função de custo para o problema original P. Usamos o custo $h^*(s)$ da solução ótima de P^* como heurística para a solução de P. No caso de um planejamento P baseado em STRIPS, um problema mais simples P^* é o planejamento onde os operadores *Eliminação* são desconsiderados. Nesse caso, as ações apenas adicionam *fluents* ao próximo estado, facilitando a busca do Estado Objetivo.

Cálculo da função heurística $h^*(s)$: considerando cada estado de um problema de planejamento uma conjunção de *fluents*, podemos determinar quando um *fluent* p é gerado pela aplicação de uma ação a um determinado estado E. Essa ação, cujas pré-condições estão presentes no estado E e que possui p como uma de suas conseqüências, será representada por $E \supset p$. Indutivamente, podemos quantificar o número de regras $E_n \supset p$ necessárias para obtermos p a partir de cada estado intermediário n de um espaço de busca.

Essa "distância" é calculada pela função $g(p, s)$, que representa o número de passos necessários para obtermos p a partir de um estado s: 0 se $p \in s$, infinito se p não é alcançável a partir de s e $i + 1$ quando existir uma regra $E \supset p$.

$$g(p,s) \stackrel{\text{def}}{=} \begin{cases} 0 & \text{se } p \in s \\ i+1 & \text{se para alguma regra } E \supset p,\ \sum_{r \in E}\ g(r,s)\ =\ i \\ \infty & \text{se } p \text{ não é alcançável a partir de } s \end{cases} \tag{2.8}$$

Para uma conjunção de *fluents* P, consideramos a soma dos passos necessários para alcançarmos todos os *fluents* $p \in P$:

$$g(P, s) \stackrel{\text{def}}{=} \sum_{\text{p} \in P} g(p, s) \tag{2.8}$$

Utilizando a Equação 2.8 podemos definir a função de custo para a satisfação do estado objetivo G a partir de um estado s de um determinado problema de planejamento:

$$h^*(s) \stackrel{\text{def}}{=} g(G, s) \tag{2.8}$$

Note-se que a função $h^*(s)$ estima o custo de satisfação do objetivo considerando que os *fluents* que formam o estado objetivo são totalmente independentes. Ou seja, a função $h^*(s)$ é a soma da dificuldade de cada um dos sub-objetivos, calculados individualmente a partir do estado intermediário s.

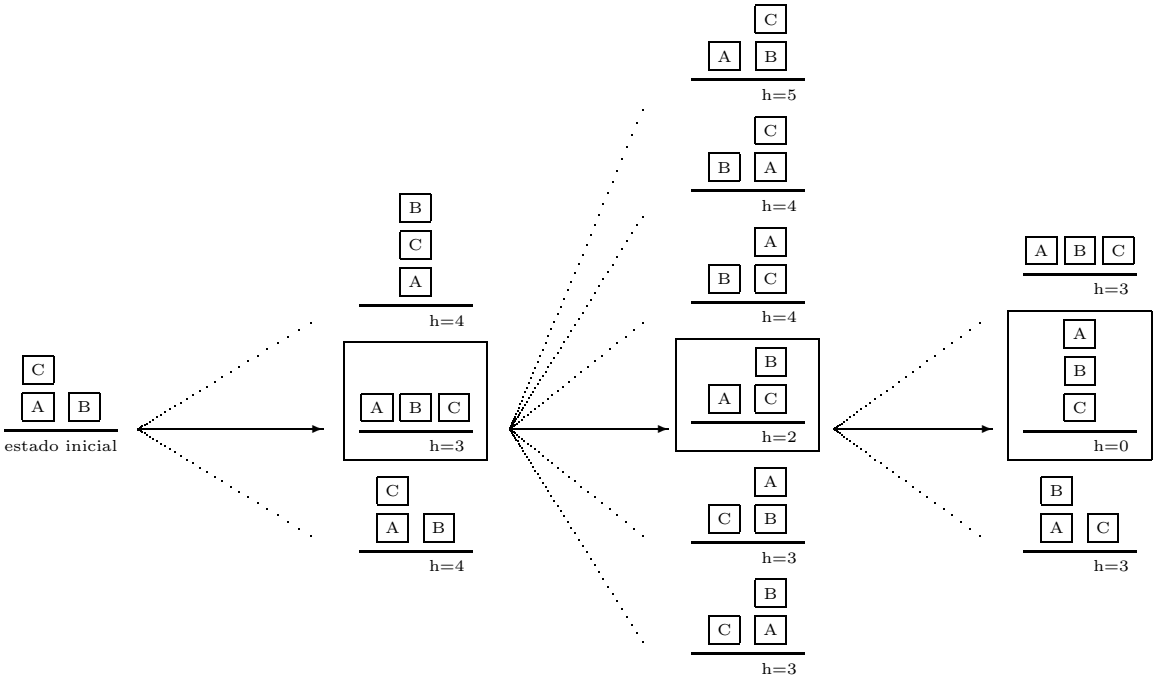

Figura 2.9. *Valores heurísticos para o exemplo do mundo dos blocos.*

A Figura 2.9 mostra um exemplo de planejamento no mundo dos blocos onde Estado Inicial = $\{sobre(C, A), sobre(A, Mesa), sobre(B, Mesa)\}$ e Estado Final =

$\{sobre(A,B), sobre(B,C), sobre(C,Mesa)\}$. Nota-se que os estados emoldurados representam os estados com menor valor da função heurística h*(s), ou seja, que estão mais próximos do objetivo.

2.4.6 Planejamento baseado em heurística de busca invertida (GRT)

Após a bem sucedida introdução de heurísticas de busca no planejamento automático, variações na heurística utilizada e nos algoritmos de busca foram naturalmente propostos pela comunidade científica internacional. Dentre essas pesquisas, destaca-se o planejador GRT - Greedy Regression Table [97]. Desenvolvido na Grécia [5], pelos pesquisadores Ioannis Refanidis e Ioannis Vlahas, o GRT propõe a aplicação inversa da heurística de busca do HSP, calculando a distância do estado objetivo a cada um dos fluents presentes no espaço de busca do problem. Para tal, os pesquisadores sugeriram a reescrita dos operadores válidos no domínio de um problema, de forma inversa aos originais, onde, dado um operador original α, calcula-se um operador inverso α' a partir das seguintes fórmulas:

- P(α') = A(α) + P(α) - D(α)
- D(α') = A(α)
- A(α') = D(α)

Podemos observar o exemplo do operador *empilhar* no domínio do mundo dos blocos, mostrado na tabela 2.3. Observe-se que o efeito dos operadores que aparecem na tabela é exatamante oposto, ou seja, a aplicação de um deles anula as conseqüências do outro e vice-versa. Uma vez calculado, o conjunto de operadores

Operador **original**: *empilhar*	Operador **invertido**: *desempilhar*
PRE: livre(C) $\wedge$ suspenso(A)	PRE: sobre(A, C) $\wedge$ garraLivre()
DEL: livre(C) $\wedge$ suspenso(A)	DEL: sobre(A, C) $\wedge$ garraLivre()
ADD: sobre(A, C) $\wedge$ garraLivre()	ADD: livre(C) $\wedge$ suspenso(A)

Tabela 2.3. *Exemplo de inversão de operadores no planejador GRT*

invertidos pode ser usado para gerar o espaço de busca visível do estado objetivo, ou

[5] Aristotle University of Thessaloniki (www.csd.auth.gr)

seja, o conjunto de todos os fluents gerados pela aplicação sucessiva dos operadores invertidos a partir do estado objetivo de um problema. Esse processo obedece ao seguinte algoritmo:

i Calcular os operadores inversos para o domínio do problema.

ii Criar uma tabela contendo duas colunas: a primeira contendo uma lista de fatos e a segunda contendo a distância desses fatos ao estado objetivo.

iii Adicionar à tabela os fatos do estado objetivo, com distância zero.

iv A partir dos fluents da tabela, instanciar um operador inverso Γ e calcular o seu valor heurístico como a soma do custo de seus pré-requisitos mais um $(\Sigma_{PRE} + 1)$.

v Para cada fato ϕ conseqüente desse operador:

   (i) se ϕ não está contido na tabela, incluir ϕ na tabela com o valor heurístico de Γ como distância ao objetivo.

   (ii) caso contrário, se ϕ está contido na tabela, mas o valor heurístico de Γ for menor que o valor de ϕ registrado na tabela, atualizar o valor de ϕ para o valor heurístico de Γ (algoritmo guloso).

vi Repetir os passos iv e v até que nenhum nova mudança seja registrada na tabela de regressão gulosa.

Note-se que o nome de tabela gulosa se deve ao fato de que a tabela é preenchida sempre com o menor valor calculado, ou seja, valores heurísticos gravados na tabela podem ser substituídos em caso de um cálculo menor. Tal qual no HSP, a velocidade do cálculo da tabela de regressão é proporcional ao número de fatos do estado objetivo e pelo número de operadores válidos no domínio do problema. Outro aspecto relevante é a escolha dos operadores no passo *iv*. Conforme o operador escolhido, mais rapidamente a tabela será preenchida com seus valores ótimos.

2.5 O desempenho dos planejadores no AIPS

A competição de planejamento permitiu um sentimento sobre a eficiência das diversas técnicas existentes, incluindo as acima detalhadas. Os resultados completos

destas competições opdem ser encontrados em [4, 82]. Do ponto de vista de nossa pesquisa, o mais relevante é frisar que os planejadores baseados em busca heurística apresentaram um desempenho de destaque nas competições. Considerando que a heurística de busca proposta por Geffner no HSP e aprimorada por Refanidis no GRT é bem mais simples que as demais técnicas e que o desempenho de seus planejadores ficaram acima da média nas competições de 1998 e 2000, procuramos desenvolver um protótipo inspirado nestas técnicas. As decisões sobre o protótipo, bem como a discussão sobre um projeto ideal para uma ferramenta planejadora baseada em busca heurística será apresentado na próxima seção desta dissertação.

3

Modelagem de um sistema planejador baseado em busca heurística

Durante as pesquisas desta dissertação, os planejadores apresentados no Capítulo 2 foram estudados quanto a sua funcionalidade, aspectos computacionais e conceituais. O objetivo não era confrontá-los como no AIPS, mas sim estudar as estruturas de dados e seus algoritmos a fim de identificar um padrão de projeto aceitável para o desenvolvimento de um novo planejador. O código aberto da maioria desses planejadores é distribuído otimizado para competição e, portanto, pouco legível. Os artigos que descrevem os seus algoritmos são abstratos ou econômicos quanto aos detalhes de projeto e desenvolvimento e, raramente, é publicada alguma documentação de suporte a esse programas. Essas dificuldades mostraram-se evidentes quando protótipos construídos a partir desses relatórios técnicos apresentaram desempenho quatro a cinco vezes inferior aos planejadores originais. Mesmo admitindo diferenças na linguagem utilizada e na maturidade técnica da equipe de desenvolvimento, os softwares não refletiam o comportamento previsto nos algoritmos publicados. Alguns autores, ao serem consultados sobre o desempenho de suas heurísticas, admitiram truques de otimização que não constavam nos relatórios publicados em congressos. A dificuldade de conciliar desempenho e clareza no código motiva uma das primeiras decisões do projeto a ser descrito neste capítulo: o uso da Orientação a Objetos. Com isso, foi descartada a geração de um código competitivo segundo os critérios atuais do AIPS, mas permitiu a geração de um código mais acessível quanto aos detalhes inerentes à implementação de um planejador automático.

Uma vez relaxado o critério velocidade, nossa pesquisa concentrou a atenção nos modelos adotados pelos demais planejadores, na representação interna do conhecimento e na qualidade dos planos gerados. Escolhemos os planejadores baseados em busca heurística pela sua simplicidade e baixa dependência de algoritmos exter-

nos ao planejamento, como o caso de planejamento por satisfatibilidade. Dentre os planejadores heurísticos, o GRT foi escolhido por ter apresentado um melhor desempenho em relação ao HSP no AIPS98, e pelo fato do material disponível sobre ao GRT estar melhor organizado e documentado em relação ao conteúdo sobre o HSP

Antes de descrevermos os experimentos com o GRT, geração de protótipos e o modelo proposto para o desenvolvimento de um programa planejador, detalharemos uma característica negativa dos planejadores baseados em heurísticas: a baixa aplicabilidade dos planos gerados ao mundo real.

3.1 Aplicabilidade dos planejadores heurísticos ao mundo real

O planejador GRT baseia a escolha da ação a ser aplicada em um determinado estado de um problema a partir da distância heurística de suas conseqüências ao estado objetivo. Vamos considerar o seguinte problema:

> Suponha um turista que deseja viajar de avião entre as cidades de Fortaleza e Rio de Janeiro, e que a rota do vôo deva ser calculada por um planejador do tipo GRT.

O Exemplo 3.1 traz a definição deste domínio e uma instância do problema com cinco rotas, traduzindo para linguagem PDDL. Note-se que o domínio desconsidera o consumo de combustível e tempo durante o cálculo do trajeto e que a descrição do problema prevê apenas duas rotas possíveis de serem calculadas pelo planejador:

- Fortaleza - Salvador - Vitória - Rio de Janeiro
- Fortaleza - Miami - Rio de Janeiro

Observe-se que o trecho *Fortaleza - Salvador* deixa o passageiro a duas escalas do objetivo, enquanto o trecho *Fortaleza - Miami* o deixa apenas a uma escala do objetivo. Aplicando-se as definições acima ao planejador GRT, obtém-se o seguinte resultado:

```
rotaFortalezaRio,100,2,
(viajar airbus fortaleza miami),(viajar airbus miami rio)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;; DESCRIÇÃO DO DOMÍNIO
(define (domain trafegoAereo)
(:predicates (aviao ?aviao) (localizacao ?aviao ?cidade) (rota ?origem ?destino))
(:action viajar
  :parameters (?aviao ?a1 ?a2)
  :precondition (and (aviao ?aviao) (localizacao ?aviao ?a1) (rota ?a1 ?a2))
  :effect (and (localizacao ?aviao ?a2) (not (localizacao ?aviao ?a1)))))

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;; DESCRIÇÃO DO PROBLEMA
(define (problem  rotaFortalezaRio)
(:domain  trafegoAereo)
(:objects airbus fortaleza salvador vitoria rio miami)
(:init    (aviao airbus) (rota fortaleza salvador) (rota salvador vitoria)
          (rota vitoria rio) (rota miami rio) (rota fortaleza miami)
          (localizacao airbus fortaleza))
(:goal    (and (localizacao airbus rio))))
```

Exemplo 3.1. *O problema do tráfego aéreo*

Seguindo a heurística de distâncias ao objetivo, o planejador escolhe a rota mais curta como o plano gerado. Isto está correto, segundo a heurística adotada, mas dificilmente seria aceito pelo passageiro, uma vez que o preço da passagem e o tempo de vôo aumentariam muito com a escala internacional. Ou seja, o plano gerado não é aplicável ao mundo real.

Essa baixa aplicabilidade dos planos gerados por busca heurística é conseqüência da análise restrita à visibilidade dos fluents em um determinado instante de tempo, ou seja, a estrutura e o significado de um fluent não é modificado ao longo do tempo. A posição do avião no exemplo anterior é tratado como um fato isolado, e o número de escalas necessárias para atingir o objetivo ignora a diferença de custo entre as escalas disponíveis. Não há como o planejador saber que, após alguma escala, a quantidade de tempo e combustível disponível foi modificada - atribuindo um aspecto ingênuo aos planos gerados por busca heurística.

Atualmente, a modelagem do domínio dos problemas é feita considerando a relação entre os esquemas de ações, ou seja, as conseqüências desses esquemas em relação a um objetivo. Isso permite que planejadores como o GRT e o HSP calculem

rapidamente os problemas de testes[1], mas impede o seu uso no mundo real.

Observamos que o vínculo dos operadores ao custo de sua aplicação permitiria a geração de planos verossímeis, promovendo os planejadores baseados em heurísticas de busca a ferramentas de apoio à tomada de decisões no mundo real. Para isso, os algoritmos planejadores necessitam modificações de projeto, passando não apenas a considerar a visibilidade dos fluents nos estados intermediários da busca mas também verificar o custo dessa visibilidade em relação a restrições previamente definidas no domínio do problema. Na idéia original, admite-se a presença de um fluent em um determinado estado a partir de suas pré-condições no estado anterior e de sua distância heurística em relação ao objetivo. Assim como o suporte a custos das ações, o sistema precisa também verificar as restrições em relação ao custo da inclusão ou remoção de um fluent em um determinado estado. Isso, naturalmente, aumenta a complexidade do algoritmo planejador, além de exigir novas estruturas de dados capazes de manipular informações com estrutura matemática - números.

3.2 Representando o custo das ações

Por custo de uma ação, assume-se a quantidade de recursos que a aplicação desta ação agrega ou reduz a um ou mais fluents de um domínio. O primeiro desafio em gerar um planejador sensível ao consumo de recursos é a representação dos recursos associados a cada um dos fluents do domínio. Inicialmente consideramos duas representações: discretização de recursos e aritmética de recursos.

3.2.1 Discretização de recursos

Discretiza-se a quantidade original de um determinado recurso em unidades, ou seja, converte-se um fluent associado a um determinado recurso em um conjunto de fluents associados a quantidades unitárias desse recurso. Para exemplificar essa abordagem, retomaremos o exemplo do plano de vôo apresentado no começo deste capítulo, considerando o recurso combustível. O domínio poderia ser reescrito como:

- Existe apenas um avião disponível para o planejamento da rota, ou seja, a

[1] Do jargão internacional *toy example*

quantidade de combustível disponível para o trajeto é única e não há reabastecimento.

- O tanque de combustível do avião contém cinco unidades de combustível, identificadas pelos símbolos predicados: *c1, c2, c3, c4* e *c5*.
- Cada escala tem uma quantidade específica de consumo de combustível, conforme enumerado abaixo:

Rota:	**Consumo de combustível:**
Fortaleza - Salvador	1 unidade
Salvador - Vitória	1 unidade
Vitória - Rio de Janeiro	1 unidade
Fortaleza - Miami	3 unidades
Miami - Rio de Janeiro	3 unidades

Traduzindo a descrição para linguagem PDDL, e diferenciando os operadores em viagens nacionais e internacionais, teremos o seguinte domínio e problema:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;; DESCRIÇÃO DO DOMÍNIO
(define (domain trafegoAereo)
(:requirements :strips)
(:predicates (combustivel ?u)
             (aviao ?aviao)
             (localizacao ?aviao ?cidade)
             (aeroportoNacional ?aeroporto)
             (aeroportoInternacional ?aeroporto)
             (rota ?origem ?destino))

(:action voarRotaDomestica
  :parameters   (?aviao ?a1 ?a2 ?u)

  :precondition (and (aeroportoNacional ?a1) (aeroportoNacional ?a2)
    (rota ?a1 ?a2)
                (aviao ?aviao) (localizacao ?aviao ?a1) (combustivel ?u))

  :effect (and (localizacao ?aviao ?a2) (not (localizacao ?aviao ?a1))
```

```
          (not (combustivel ?u))))

(:action viajarParaExterior
  :parameters (?aviao ?a1 ?a2 ?u ?v ?x)

  :precondition (and (aeroportoNacional ?a1) (aeroportoInternacional ?a2)
                (aviao ?aviao) (localizacao ?aviao ?a1) (combustivel ?u)
                (combustivel ?v) (combustivel ?x) (rota ?a1 ?a2))

  :effect (and (localizacao ?aviao ?a2) (not (localizacao ?aviao ?a1))
          (not (combustivel ?u)) (not (combustivel ?v)) (not (combustivel ?x))))

(:action retornarDoExterior
  :parameters (?aviao ?a1 ?a2 ?u ?v ?x)

  :precondition (and (aeroportoInternacional ?a1) (aeroportoNacional ?a2)
                (aviao ?aviao) (localizacao ?aviao ?a1) (combustivel ?u)
                (combustivel ?v) (combustivel ?x) (rota ?a1 ?a2))

  :effect (and (localizacao ?aviao ?a2) (not (localizacao ?aviao ?a1))
          (not (combustivel ?u)) (not (combustivel ?v)) (not (combustivel ?x))))
)


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;; DESCRIÇÃO DO PROBLEMA
(define (problem  rotaFortalezaRio)
(:domain  trafegoAereo)
(:objects airbus fortaleza salvador vitoria rio miami c1 c2 c3 c4 c5)

(:init    (aviao airbus) (aeroportoInternacional miami)
          (aeroportoNacional fortaleza) (aeroportoNacional salvador)
          (aeroportoNacional vitoria) (aeroportoNacional rio)
          (combustivel c1) (combustivel c2) (combustivel c3)
          (combustivel c4) (combustivel c5)
          (rota fortaleza salvador) (rota salvador vitoria)
          (rota vitoria rio) (rota fortaleza miami) (rota miami fortaleza)
          (localizacao airbus fortaleza))

(:goal    (and (localizacao airbus rio))))
```

Exemplo 3.2. *Problema do tráfego aéreo com recursos discretizados*

Observe-se que a descrição do problema inclui agora cinco objetos do tipo combustível: c1, c2, c3, c4 e c5. Além disso, todos os operadores válidos no domínio tem o efeito de consumir esses objetos - rota internacional consome três unidades de combustível, enquanto as rotas nacionais consomem uma unidade de combustível.

A partir das descrições acima, o planejador GRT gerou o seguinte resultado:

```
rotaFortalezaRio,120,3,
(voarRotaDomestica airbus fortaleza salvador c1)
(voarRotaDomestica airbus salvador vitoria c2)
(voarRotaDomestica airbus vitoria rio c3)
```

Interpretação: em cento e vinte milisegundos, o planejador calculou três instâncias de operadores capazes de satisfazer o objetivo. O resultado equivale à rota esperada, uma vez que o vínculo das escalas aos respectivos consumos de combustível impede o planejador de escolher rotas longas como a que inclui Miami.

Note-se que os recursos, no caso o combustível, estão sendo representados junto aos demais fluents do domínio. Isso permite o vínculo entre a execução das ações e o seu custo, uma vez que as ações podem remover fluents dos estados intermediários, simulando o seu consumo. Isso simplifica muito o trabalho de representação de recursos e também dispensa a mudança nos algoritmos planejadores.

Apesar da simplicidade da representação de recursos como um conjunto de fluents válidos do domínio, essa representação é pouco intuitiva e não permite uma visão agrupada desses recursos. No exemplo acima, os fluents c1, c2, c3, c4 e c5 não têm uma relação entre si, apesar de juntos modelarem um único objeto no mundo real. Durante o planejamento, não é possível a tomada de decisões sobre o volume total do combustível disponível, nem sobre o volume total consumido em uma determinada rota que inclua mais de uma escala. Nota-se que a informação sobre o volume total de combustível foi perdida, restando apenas a análise discreta das *partes do combustível*. Além disso, o exemplo utilizado foi propositalmente pequeno para facilitar a discussão sobre o uso de discretização de recursos. O mesmo problema pode ser apresentado, considerando-se um tanque de combustível com milhares de unidades e um número bem maior de rotas disponíveis - isso causaria um número excessivo de fluents a serem manipulados pelo planejador.

Como mostrado no capítulo anterior, a complexidade dos algoritmos de busca heurística está diretamente relacionada com a quantidade de objetos do problema.

Portanto, quando a discretização de recursos gera um número elevado de fluents, o tempo gasto no planejamento tende a se tornar inaceitável. Uma alternativa para esse problema é adotar uma aritmética de recursos, ou seja, manipular as quantidades de recursos associados a cada operador, a partir de uma abordagem matemática.

3.2.2 Aritmética de recursos

O problema descrito acima seria muito mais fiel ao comportamento real de um avião, se a quantidade de combustível consumida em cada rota fosse calculada a partir do consumo previsto para esse avião - contínuo e calculado a partir de alguma equação matemática. Quando consideramos a representação de *quantidades* no mundo real, algum mecanismo de suporte a equações matemáticas se faz necessário. O grau dessas equações depende da natureza do problema tratado, desde simples aritmética de primeiro grau até equações de ordem superior.

Inicialmente, restingiremos o nosso estudo à manipulação de equações de primeiro grau, modificando o mecanismo de busca heurística para o suporte à valores associados aos fluents adicionados ou removidos dos estados intermediários. Chamaremos os valores associados aos fluents de *recursos* e o procedimento responsável pela sua manipulação de *aritmética de recursos*.

O primeiro aspecto a ser considerado é a representação dos recursos dentro da descrição do domínio e do problem em PDDL. Adotamos o símbolo # como modificador que representa recursos, ou seja, fluents que tenham o sustenido como prefixo devem ser tratados como recursos pelo planejador. Além deste modificador, o planejador foi modificado para reconhecer o conjunto de símbolos apresentados na Tabela 3.1. Observe-se que os símbolos são formados por um caractere apenas - isso foi adotado para facilitar o reconhecimento desses símbolos durante a leitura e interpretação dos domínios e problemas em PDDL.

Utilizando essa simbologia, o Exemplo 3.3 apresenta a versão do problema de tráfego aéreo com aritmética de recursos.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;; DESCRIÇÃO DO DOMÍNIO
(define (domain trafegoAereo)
```

símbolo	significado	símbolo	significado
$+$	soma	$-$	subtração
$*$	multiplicação	$/$	divisão
$>$	maior que	$<$	menor que
$:$	atribuição	$=$	igualdade
$]$	maior ou igual que	$[$	menor ou igual que
$\wedge$	potência	$(\ )$	delimitadores
$\%$	mínimo entre dois valores	$\&$	máximo entre dois valores

Tabela 3.1. *Símbolos usados para a manipulação de recursos no planejador proposto.*

```
(:requirements :strips)
(:predicates (#combustivel)
             (aviao ?aviao)
             (localizacao ?aviao ?cidade)
             (aeroportoNacional ?aeroporto)
             (aeroportoInternacional ?aeroporto)
             (rota ?origem ?destino))

(:action voarEscalaNacional
  :parameters   (?aviao ?a1 ?a2)

  :precondition (and (aeroportoNacional ?a1) (aeroportoNacional ?a2)
    (rota ?a1 ?a2) (#combustivel ] 300)
                (aviao ?aviao) (localizacao ?aviao ?a1))

  :effect (and (localizacao ?aviao ?a2) (not (localizacao ?aviao ?a1))
          (#combustivel : #combustivel - 300)))

(:action voarEscalaInternacional
  :parameters (?aviao ?a1 ?a2)

  :precondition (and (aeroportoNacional ?a1) (aeroportoInternacional ?a2)
                (aviao ?aviao) (localizacao ?aviao ?a1) (#combustivel ] 1000)
                (rota ?a1 ?a2))

  :effect (and (localizacao ?aviao ?a2) (not (localizacao ?aviao ?a1))
          (#combustivel : #combustivel - 1000)))
)
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;;;;;;;;;; DESCRIÇÃO DO PROBLEMA
(define (problem  rotaFortalezaRio)
(:domain  trafegoAereo)
(:objects airbus fortaleza salvador vitoria rio miami #tanque_airbus)

(:init    (aviao airbus) (aeroportoInternacional miami)
          (aeroportoNacional fortaleza) (aeroportoNacional salvador)
          (aeroportoNacional vitoria) (aeroportoNacional rio)
          (#tanque_airbus : 2000)
          (rota fortaleza salvador) (rota salvador vitoria)
          (rota vitoria rio) (rota fortaleza miami) (rota miami fortaleza)
          (localizacao airbus fortaleza))

(:goal    (and (localizacao airbus rio) (#tanque_airbus > 0))))
```

Exemplo 3.3. *Problema do tráfego aéreo com aritmética de recursos*

Observe-se que o domínio inclui o fluent denominado #combustivel capaz de representar uma determinada quantidade de combustível, e que os operadores dependem do valor associado a esse fluent para serem aplicados. Além disso, os operadores podem modificar o valor desse recurso, ou seja, consumir ou gerar o recurso.

Quando pensamos nessa abordagem de forma progressiva, ou seja, aplicando instâncias de operadores a partir de um estado inicial, fica simples visualizar a manipulação do recurso combustível de recursos. Podemos considerar, por exemplo, uma quantidade inicial de combustível em mil unidades e uma nova quantidade de setecentas unidades após a aplicação de uma instância do operador voarEscalaNacional.

Pensando, agora, na forma como planejadores baseados em GRT funcionam, devemos encontrar uma forma de representar os fluents associados a recursos também nos operadors invertidos, ou seja, devemos criar um procedimento de inversão de operadores que permitam a influência reversa da variação dos valores associados a recursos. Tal procedimento pode se tornar extremamente complexo, se considerarmos símbolos funcionais que usem como parâmetro outros fluents do domínio. Para os primeiros experimentos com essa abordagem, limitamos o uso de aritmética básica e a ausência de funções, exceto as de maior, menor e potência definidas na Tabela 3.1. Com isso, chegamos às relações de inversão apresentadas na Tabela 3.2. Além da igualdade utilizada nessa tabela, os operadores *maior (>)* e *menor (<)* também

	Operador **progressivo**	Operador **regressivo**
pré-condição:	<precondições>	$\#a = k$
conseqüência:	$\#a = k$	<precondições>
pré-condição:	$\#a = \#b$	$\#b = \#a - \Delta$
conseqüência:	$\#a = \#a + \Delta$	$\#a = \#b$
pré-condição:	$\#a = \#b$	$\#b = \#a + \Delta$
conseqüência:	$\#a = \#a - \Delta$	$\#a = \#b$
pré-condição:	$\#a = \#b$	$\#b = \#a/\Delta$
conseqüência:	$\#a = \#a * \Delta$	$\#a = \#b$
pré-condição:	$\#a = \#b$	$\#b = \#a * \Delta$
conseqüência:	$\#a = \#a/\Delta$	$\#a = \#b$
Δ - constante ou fórmula não contendo $\#a$ ou $\#b$		k - constante numérica

Tabela 3.2. *Inversão de operadores com fluents associados a recursos.*

suportam as mesmas relações. De fato, a forma mais natural de abduzirmos um operador a partir de um conjunto de fluents associados a recursos parece ser o raciocínio sobre a ordem de grandeza desses fluents entre dois estados intermediários adjacentes. Supondo um fluent associado a um recurso, podemos imaginar que após a aplicação de um determinado operador, a quantidade deste recurso sofra uma variação, tornando-se menor ou maior do que a quantidade anterior.

Uma vez que um operador seja inversamente aplicado ao conjunto de fluents na geração da tabela de regressão, deveríamos ter uma modificação na quantidade do recurso associado a esse fluent. Porém, isso contraria a idéia original do GRT, onde a geração da tabela despreza a lista de remoção dos operadores invertidos. Surge outra decisão de projeto, isto é, a volatilidade dos recursos durante a geração da tabela de regressão. Ou seja, um recurso não terá uma quantidade única manipulada durante a geração da tabela de regressão, mas sim uma lista de valores possíveis - cada valor com o seu respectivo custo heurístico.

Para que uma relação de ordem de grandeza entre dois fluents seja verificada durante a busca regressiva, é necessária a verificação da combinação das duas listas de valores associadas a esses fluents. Isso, por si só, causa um aumento de complexidade da ordem de $O(n^2)$, onde n é o maior número de valores possíveis entre o dois fluents. Além disso, temos o acréscimo de um algoritmo de controle de listas encadeadas na rotina principal do planejador.

Uma alternativa para esse problema é uma abordagem semelhante a que é usada em compiladores [1], com a criação de novas variáveis, cópias das referências dos fluents durante o cálculo. Nesse caso, essas referências deveriam constar em uma lista associada à referência original, permitindo a múltipla verificação e a criação de novas variáveis a cada novo valor possível. Após experimentos preliminares, detectou-se que a diferença de complexidade entre as duas opções é desprezível, adotando-se a primeira opção como padrão no projeto.

Outro problema que advém da manipulação de recursos é a necessidade de limitarmos o intervalo em que as quantidades desses recursos podem variar.

3.2.2.1 Intervalo de variação na quantidade de um recurso

Da mesma forma como os operadores invertidos do GRT eram aplicados sistematicamente até a estabilização dos fluents calculados, os operadores que manipulam recursos devem ter alguma condição de parada. Imagine-se um operador que acrescenta uma unidade de um determinado recurso à quantidade desse recurso associada a um fluent, e que esse operador possa ser aplicado indefinidamente de acordo com as suas demais pré-condições - o recurso associado ao fluent tenderá a uma quantidade infinita. Não existe um padrão definido de como limitar a quantidade de um determinado recurso. Esse limite pode ser incluído na descrição do problema, atribuíndo uma quantidade fixa de recursos nos estados inicial e final de um problema. Outra forma de evitar uma varição ilimitada de um recurso é definir os operadores com condicionantes a determinados limites desse recurso.

3.3 Projeto

A maioria dos códigos disponíveis hoje é produzida em ANSI C, com uso intensivo de registradores, ponteiros de memória e variáveis globais, visando sempre a velocidade como objetivo primário dos programas. Nenhum dos planejadores testados apresentava interface gráfica ou relatórios *passo-a-passo* que permitissem o acompanhamento da geração de planos. Os sistemas eram todos como caixas-preta, rodando a partir de linhas de comando e gerando arquivos texto com o plano resultante de suas buscas ou alguma mensagem de erro.

A dificuldade inicial em identificar o funcionamento de tais planejadores, e a

baixa popularidade de tais ferramentas entre os pesquisadores brasileiros no motivou a gerar a interface gráfica da Figura 3.1, simplificando a visualização e interpretação do conhecimento envolvido na geração de planos. Esse cuidado com o a *interface humano-computador* [21] é uma tendência mundial observada na maioria das ferramentas planejadoras criadas recentemente, como GIPO [113] e ASPEN [16].

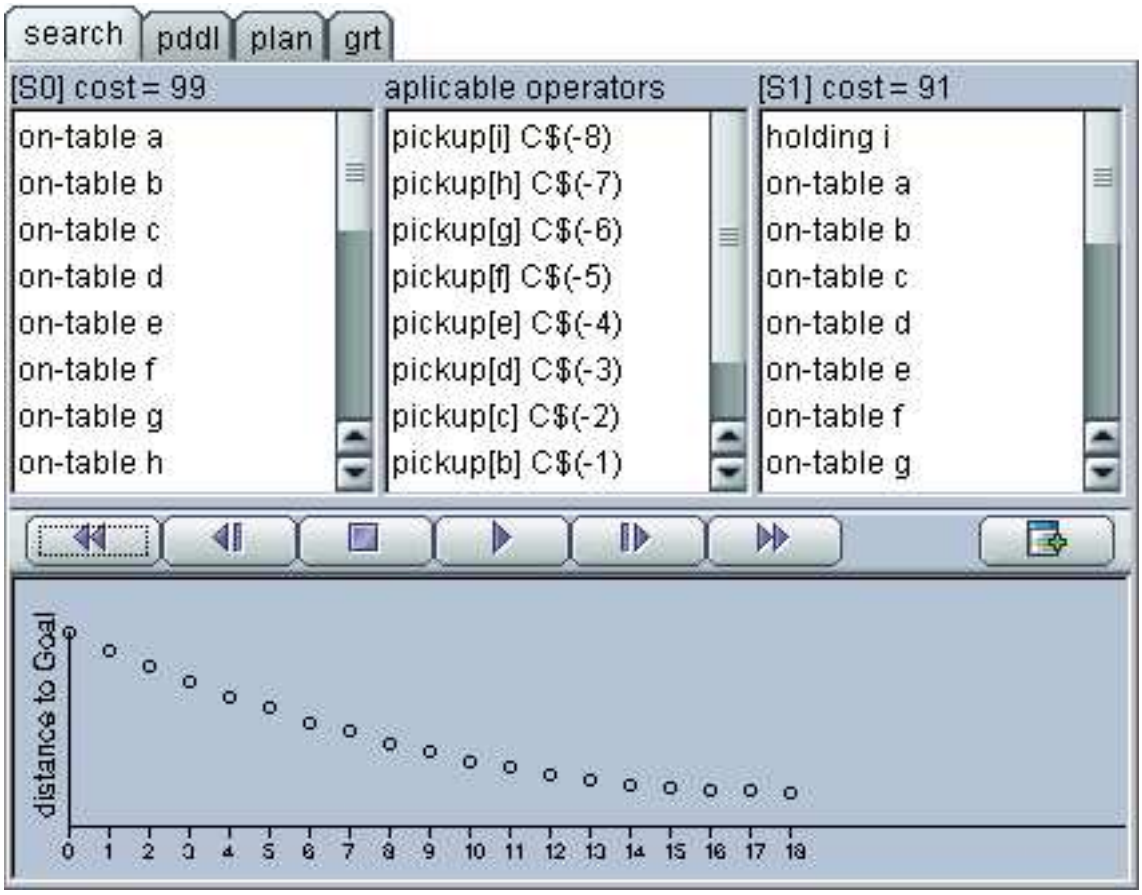

Figura 3.1. *Interface gráfica do protótipo*

Note que na tela principal da interface aparecem os estados intermediários da busca e uma barra de ferramentas que permite ao usuário navegar desde o estado inicial até o estado objetivo do problema. Logo abaixo temos um gráfico onde as ordenadas representam os índices dos estados intermediários e as abcissas as distâncias heurísticas desses estados ao estado objetivo do problema. As telas secundárias

apresentam as informações do problema e do domínio, o plano sendo gerado e a tabela de regressão.

Antes de entrarmos em detalhes sobre o protótipo, enumeraremos a lista de requisitos que orientaram a sua implementação e que consideramos gerais para projetos de planejadores baseados em busca heurística:

- Ferramenta portável aos principais sistemas operacional disponíveis: Windows e UNIX/Linux. Preferencialmente executável através da internet, dispensando instalação.
- Dados de entrada lidos de arquivos texto contendo as definições do domínio e do problema em formato PDDL.
- Capacidade de visualização dos planos gerados na tela ou gravação direta em arquivo texto. O formato desta visualização deve facilitar ao usuário identificar os estados intermediários visitados durante a busca e também os operadores disponíveis para a transição entre esses estados.
- Uso de Orientação a Objetos [19, 25] ou outro paradigma capaz de facilitar a manutenção e expansão da ferramenta frente a futuras inovações.
- Adoção do padrão MVC - *Modelo-Visão-Controle* [33, 119], permitindo a independência dos módulos responsáveis pelo tratamento dos dados, geração e visualização de planos.

3.4 Implementação do protótipo

A partir dos requisitos apresentados acima, foi implementado um protótipo na linguagem Java [25]. No fechamento desta dissertação, o protótipo era capaz de lidar com os problemas determinísticos com desempenho ligeiramente inferior ao GRT, e os primeiros experimentos com o uso de recursos estavam sendo desenvolvidos.

Apesar do código desenvolvido não apresentar a estabilidade de um produto, os experimentos permitidos com a ferramenta foram fundamentais na identificação dos conceitos apresentados neste capítulo e, principalmente, no estabelecimento dos rumos para novas pesquisas em planejamento automático - os trabalhos futuros previstos no Capítulo 5.

4

Planejamento automático aplicado à administração de um sistema de reservatórios de água - estudo de caso.

No capítulo anterior, o planejamento com suporte à recursos foi discutido e um respectivo projeto de desenvolvimento proposto. A motivação desse projeto foi originalmente a satisfação de problemas do mundo real, em especial um caso a ser descrito neste capítulo: a adminisração do sistema de reservatórios de água do Estado do Ceará.

4.1 Problemas de recursos hídricos no Estado do Ceará

O Ceará, pela sua localização geográfica no semi-árido nordestino, sofre de escassez de recursos hídricos em boa parte de seu território. A alternativa disponível para garantir a capacidade de desenvolvimento social e econômico do Estado é a transposição de águas. Atualmente, o Ceará dispõem de um sistema integrado de reservatórios e uma política de distribuição de água apresentado no Apêndice B. Essa política entretanto é definida a partir das considerações de governantes sobre os relatórios de vazão e demanda disponíveis. Tais relatórios são gerados a partir de estimativas matemáticas exatas traduzidas através do conhecimetno empírico compartilhado entre os engenheiros que administram cada um dos reservatórios. Além disso, a influência sócio-econômica de cada centro de demanda contribui para que as estimativas exatas iniciais sejam adequadas à realidade.

Um dos usos previstos para a ferramenta idealizada no Capítulo 3 é a simulação da distribuição de água a partir cenários plausíveis em relação à quantidade de chuva e o aumento da demanda esperados. Nesta seção, alguns destes cenários

são enumerados através da linguagem PDDL e ios resultados de simulações com o protótipo são analisadas em relação à sua aplicabilidade no mundo real.

4.2 Descrição do problema

O problema de distribuição de água pode ser dividido nos seguintes casos:

i Transferência de água entre um reservatório e um centro de demanda.

ii Compartilhamento de um reservatório por mais de um centro de demanda.

iii Suprimento de um centro de demanda por mais de um reservatório.

iv Sistema integrado, com vários centros de demanda e vários reservatórios.

Abaixo enumeramos as características destes casos, incluindo fragmentos de domínios em PDDL. No final deste capítulo, apresentamos uma versão completa de domínio e problema englobando todos os casos acima.

4.2.1 Transferência de água entre um reservatório e um centro de demanda

O caso mais simples de transferência de água em um sistema de reservatórios é quando temos apenas um reservatório de água e um centro de demanda conectado a esse reservatório. No caso do reservatório ter uma quantidade disponível de água e a demanda estar insatisfeita, basta transferirmos a quantidade requisitada ou a quantidade máxima disponível no o reservatório. Essa relação pode ser vista na Figura 4.1. A modelagem da transferência simples entre dois pontos de um sistema

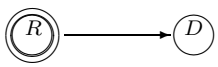

Figura 4.1. *Transferência entre um reservatório e um centro de demanda.*

de distribuição de água pode ser feita através do uma regra condicional simples: *se existe água disponível então transferir*. Traduzindo essa relação à linguagem PDDL, teremos o seguinte operador: Note que as relações de ordem *maior que*(]) e

```
(:action transferir
  :parameters   (#reservatorio #demanda)
  :precondition (and (#demanda [ 300) (#reservatorio ] 300)
  :effect (and (#demanda : #demanda + 300) (#reservatorio : #reservatorio - 300)
))
```

Exemplo 4.1. *Operador de transferência de água entre um reservatório e um centro de demanda.*

menor que([) impedem o operador de ser aplicado indefinidamente, tanto na busca progressiva quanto na regressiva.

4.2.2 Compartilhamento de um reservatório por mais de um centro de demanda

Reservatórios de água armazenam grandes volumes de água, e são geralmente abastecidos a partir de vários rios de uma determinada região. Isso causa o compartilhamento desse reservatórios por vários centros de demanda, conforme mostra a Figura 4.2. Naturalmente esse compartilhamento pressupõem uma determinada hierarquia na alocação da água, bem como uma política que garanta um suprimento mínimo a cada um dos centros de demanda dependentes do reservatório. Do ponto de vista

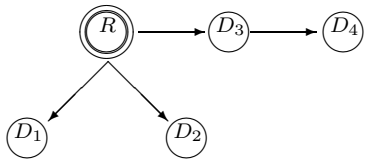

Figura 4.2. *Reservatório compartilhado entre dois ou mais centros de demanda.*

de representação do conhecimento temos algumas alternativas para erpresentar essa hierarquia:

i o uso de uma variável *booleana* ou numérica para o roteamente das alocações.

ii a divisão igual do volume disponível entre as demandas

iii a divisão proporcional do volume disponível entre as demandas

Na hipótese de uma variável numérica controlando o roteamento, cria-se uma lista circular composta pelos centros de demanda e volumes de água iguais ou proporcionais são distribuídos até qeu toda a demanda seja satisfeita, ou enquanto houver volume disponível no reservatório. Isso permite uma caso onde uma das demandas não seja atendida, ou seja, quando a água termina antes de todas as demandas receberem ao menos uma fração do volume inicial. Por esse motivo, o mais comum é a adoção de algum tipo de divisão prévia.

Note que para garantir proporções diferentes às diversas demandas pertencentes ao sistema, precisaríamos um planejador capaz de manipular funções matemáticas. Como o planejador aqui proposto suporta apenas a manipulação aritmética dos fluentes de um problema, devemos declarar um operador distinto para cada demanda, como descrito no Exemplo 4.2. Observe que o Exemplo 4.2 despreza aspectos como

```
; Demanda 4 com peso 2
(:action abastecerD4
   :parameters   (#reservatorio #demanda #atendimento)
   :precondition (and (#atendimentos = 4) (#demanda [ 300)
   :effect (and (#demanda1 : #demanda + 0.2 * #reservatorio)
      (#reservatorio : #reservatorio - 0.2 * #reservatorio) (#atendimento : 3)
))

; Demanda 3 com peso 3
(:action abastecerD3
   :parameters   (#reservatorio #demanda #atendimento)
   :precondition (and (#atendimentos = 3) (#demanda [ 400)
   :effect (and (#demanda1 : #demanda + 0.3 * #reservatorio)
      (#reservatorio : #reservatorio - 0.3 * #reservatorio) (#atendimento : 2)
))
```

Exemplo 4.2. *Operadores de transferência com distribuição proporcional de água.*

o limite da demanda e o fato de que o cálculo sequencial das proporções altera o volume final que cada demanda irá receber. Além disso, foi introduzido um fluent #atendimento que garante o roteamento na distribuição de água. No caso de uma descrição completa do domínio, a última demanda deveria reestabelecer o valor do atendimento para quatro.

4.2.3 Suprimento de um centro de demanda por mais de um reservatório

No caso de mais de um reservatório suprir uma mesma demanda, o raciocínio é análogo ao caso anterior só que as prioridade e proporções deveriam ser distribuídas sobre os reservatórios. A Figura 4.3 mostra também as duas formas de com-

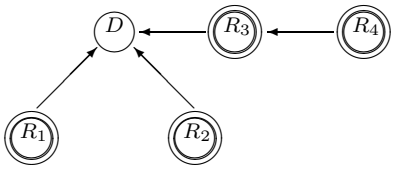

Figura 4.3. *Suprimento de uma demanda por mais de um reservatório.*

partilhamento de canal de transferência de água (adutora): com o uso de adutoras distintas ou uma única adutora compartilhada. Note que os reservatórios R_3 e R_4 aparecem encadeados na figura, o que representa o compartilhamento do mesmo canal de transmissão pelos duas fontes. Isso é freqüente no mundo real, onde temos um duto central compartilhado por vários pontos de suprimento e demanda. Esse compartilhamento agrega alta complexidade à política de alocação de água, pois a hierarquia de distribuição e as proporções previstas á cada demanda devem ser definidas dinamicamente.

4.2.4 Sistema integrado, com vários centros de demanda e vários reservatórios

No caso real do problema, temos diversos centros de demanda ligados a dutos supridos por diversos reservatórios, conforme mostra a Figura 4.4. Neste caso, o consumo e distribuição de água passa a ser tratado com um sistema integrado e não mais como pontos isolados. Ou seja, as decisões passam a ser tomadas do ponto de vista do reservatório e não mais pontulamente por demanda. O raciocínio utilizado na

prática para a distribuição de água pode ser resumido no seguinte parágrafo:

> Os reservatórios lançam quantidades de água conforme o somatório das demandas, que por sua vez consomem a água disponível a partir de uma hierarquia prévia.

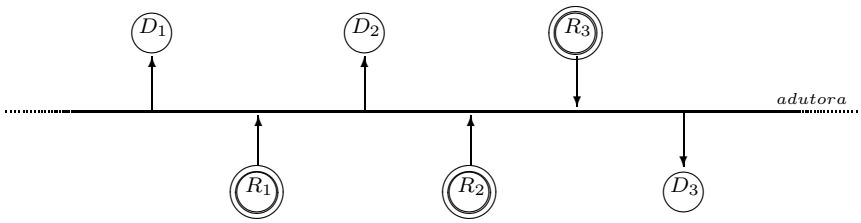

Figura 4.4. *Sistema integrado de reservatórios e centros de demanda.*

A partir deste contexto, a análise da distribuição de água deve sempre levar em conta o somatório de demandas versus o somatório de volumes disponíveis. A vazão do sistema depende de uma relação com a demanda, que por sua vez é feita de forma independente. Ou seja, para calcular a quantidade de água que um reservatório deve liberar no sistema é necessário a análise sobre as demandas pendentes e o consumo se dá em intervalos de tempo. Para que uma nova quantidade de água seja liberada no sistema, um número determinado de demandas deve ser suprida. Um exemplo de domínio e problema neste contexto é apresentado nas seções a seguir.

4.2.5 Versão PDDL do domínio

```
; Reservoir domain
; 2002 © Felipe Gaúcho

(define (domain reservoir)
(:predicates
(#volumeR1)
(#demandaD1)
(#volumeD1)

)

(:action transfer
:parameters (#demandaD1 #volumeR1)
:precondition
(and (#volumeR1 > 100)
(#volumeD1 < #demandaD1 ))
:effect
(and (#volumeD1 : #volumeD1 + %(#demandaD1 (#volumeR1-100)))
(#volumeR1 : #volumeR1 - %(#demandaD1 (#volumeR1-100)))
)
)
```

Exemplo 4.3. *O domínio de um sistema integrado de reservatórios.*

4.2.6 Versão PDDL do problema

```
(define (problem water_management)
   :domain reservoir
:objects a
   :init  (#volumeR1:500) (#volumeD1:0) (#limiarR1:50) (#minimoD1:300)
   :goal (#volumeD1>#minimoD1)

)
```

Exemplo 4.4. *O problema de um sistema integrado de reservatórios.*

4.2.7 Análise de simulações

A partir dos domínios e problemas descritos nas seções anteriores deste capítulo, realizamos uma série de simulações com o nosso protótipo e com o planejador GRT utilizado no AIPS 2000. Uma das dificuldades encontradas nestas simulações foi a falta de um mecanismo de inversão automática dos operadores contendo informações sobre recursos. No fechamento deste trabalho, o protótipo ainda não havia sido adaptado para tal função, nos obrigando a reverter manualmente os operadores dos domínios de testes. Isso fez com que o protótipo solucionasse alguns dos problemas testados, mas apresentasse dificuldade diante dos domínios mais complexos - com diversos reservatórios e centros de demanda. No caso do GRT, o planejador não possuía suporte à recursos, nos obrigando a comparações sem a aritmética de recursos.

No geral, podemos dizer que ainda não se conhece um planejador capaz de lidar com domínios e problemas dependentes de recursos. O protótipo desenvolvido e os planejadores baseados em heurística de busca revelaram um mecanismo pouco robusto diante de problemas do mundo real.

5

Conclusão

Desde a época em que os computadores eram apenas ficção científica até os dias atuais, o homem segue sonhando em criar uma máquina à sua semelhança, capaz de reações inteligentes aos estímulos do mundo real. A dificuldade em adquirir o poder da criação talvez revele ao homem a sua condição de criatura, mas de certa forma nos conforta por nos revelar também como criaturas inteligentes.

O ato em si de descrever o nosso comportamento em linguagem computacional se revela difícil diante do pouco que se tem certeza sobre o mecanismo responsável pelas nossas ações e sentimentos - nosso cérebro. Estima-se que a velocidade com que o cérebro processa as informações é muito superior a qualquer máquina já projetada pelo homem, mas nem isso podemos afirmar com certeza - talvez seja apenas uma melhor organização do conhecimento. Simular o raciocínio humano de forma geral nos parece ingênuo sob todos os aspectos. Mesmo implementando conceitos teóricos idealizados pelo homem, como dedução e inferência, a tarefa de definir como agimos e pensamos não parece integralmente passível de tratamento computacional. O que nos resta como comunidade científica é um senso comum sobre a necessidade de fragmentarmos essa abordagem através da construção de pequenos artefatos, sejam eles teóricos ou práticos. Cada um desses artefatos compõe a esperança na replicação da nossa inteligência e, independentemente do escopo e amplitude do artefato, a sua participação é fundamental para a realização da máquina pensante. O texto desta dissertação revela o entusiasmo de um estudante em contribuir na modelagem e programação de um desses artefatos: um planejador automático.

5.1 O artefato planejador

Como descrito no Capítulo 2, desde o início dos anos setenta, existe uma comunidade ativa pesquisando uma forma de criar planejadores automáticos. Independen-

temente dos experimentos registrados por esta comunidade e das diferentes tecnologias já conhecidas, algumas dificuldades parecem compartilhadas entre todos os que decidem desvendar a arte do planejamento. Analisando o discurso médio entre os pesquisadores de planejamento automático, encontramos a *linguagem* como principal geradora de complexidade. O problema reside não na linguagem PDDL em si, mas na inexistência de uma linguagem computacional ou teórica capaz de reproduzir a completude e flexibilidade da linguagem natural humana. Não porque as linguagens conhecidas sejam impotentes em expressar versões do discurso humano, mas pela alta complexidade que uma linguagem artificial exige para gerar tais versões. A mera verificação de uma referência a um objeto só é possível, para um computador, se essa referência for explícita. Mesmo considerando uma versão *bem comportada* de um discurso, onde todas as referências são préviamente estabelecidas, manipulações simples como cópias e comparações entre essas referências consomem eternos milisegundos de processamento.

Além da complexidade inerente aos problemas tratados por um planejador, como número de operadores e objetos pertencentes ao domínio de um problema, temos as limitações impostas pelos computadores atuais - memória, número de processadores, velocidade dos processadores, sistema operacional, etc. Abstraindo as questões tecnológicas, abaixo enumeramos algumas dificuldades que concluímos pertinentes a qualquer análise sobre a dificuldade na geração automática de planos:

- Linguagem
- Algoritmos de inferência
- Aplicabilidade dos planos e testes

As próximas seções discutem esses temas a partir do trabalho de pesquisa relatado nesta dissertação.

5.1.1 O idioma dos planejadores

Quando McDermoth formalizou a linguagem PDDL, o objetivo era padronizar um idioma para a pesquisa e o desenvolvimento de programas de computador capazes de gerar planos. Essa iniciativa gerou uma explosão demográfica na comunidade de

planejamento automático, ao permitir que grupos de pesquisa estabelecessem projetos de desenvolvimento de médio e curto prazo. Os departamentos de computação não precisavam mais discutir ou gerar tecnologia própria para a representação dos domínios e problemas de planejamento. Usando PDDL, vários *benchmarks* foram naturalmente estabelecidos nas primeiras competições de planejamento automático - tecnologias totalmente diferentes passaram a reconhecer o mesmo idioma.

Porém, apesar do sucesso das competições e da linguagem PDDL, não existe um consenso sobre a melhor linguagem para representar problemas de planejamento automático. De fato, inúmeros pesquisadores usam linguagens próprias ou dialetos não oficiais da PDDL. O próprio McDermoth reconhece, em seu artigo seminal, que a PDDL foi idealizada para minimizar o reflexo de linguagens predecessoras, impedindo que grupos de pesquisas tirassem proveito da estrutura das linguagens que utilizavam em seus planejadores. Esse requisito de imparcialidade imposto na geração da linguagem PDDL talvez tenha diminuido a eficiência dos planejadores em manipular os seus termos.

Embora esta dissertação trate os problemas de planejamento exclusivamente através da linguagem PDDL, deixamos como proposta de trabalho complementar a inspeção de novas linguagens ou a identificação de dialetos mais robustos da PDDL. Outro aspecto relevante é o tipo de conhecimento suportado pelos planejadores. PDDL é bem definida para problemas descritos através de lógicas monotônicas, ou seja, onde todos os objetos do domínio são previamente reconhecidos e o valor semântico dos fluents não muda com o tempo. Planejamento baseado em conhecimento incompleto ou em plausibilidade não é previsto para a PDDL padrão, forçando a extensão da linguagem. A expressividade das linguagens utilizáveis em planejamento automático aparece como fonte de pesquisa promissora e ainda em aberto na literatura internacional. Um sentimento se revela onipresente quando pensamos em definir uma linguagem de representação de problemas de planejamento: *quanto maior a expressividade da linguagem, maior o ônus sobre o algoritmo responsável pela sua manipulação.*

5.1.2 O algoritmo que pensa

Algoritmos que têm a pretensão de simular inferência a partir de um conjunto de fatos e um conjunto de regras herdam a NP-Completude de diversas fontes. Pri-

meiramente, temos a tarefa de instanciar o conjunto de esquemas de ações a partir de fluents armazenados em uma base de dados. A partr disto, precisamos decidir a ordem em que essas instâncias serão aplicadas - gerando um espaço denso de busca.

Talvez pelo alto custo de recursos computacionais nos anos oitenta, ou pela ausência de um mecanismo construtivo satisfatório até a chegada do Graphplan, os pesquisadores dedicaram a década ao estudo das propriedades matemáticas do planejamento automático. Esta análise revelou que a geração de planos para domínios genéricos é indecidível mesmo para problemas com o estado inicial finito [27, 34].

Atualmente só podemos conceber um uso real de um planejador sob um domínio específico, ou seja, podemos controlar a complexidade de um algoritmo apenas se soubermos de antemão os aspectos do problema que ele irá abordar. Alguns experimentos, como o estudo sobre o *mundo dos blocos* [43] realizado por John Slaney e Sylvie Thiébaux [116, 114], revelam uma distância abissal entre a capacidade de um planejador genérico e um específico em tomar decisões a partir de conjunto de fatos. O desempenho de planejadores específicos como o produzido pelos pesquisadores australianos[1] sugere uma alternativa híbrida, onde os planejadores possam reforçar a sua capacidade de busca com algum conhecimento prévio sobre um problema.

Durante nossas pesquisas implementamos vários programas específicos, pequenos e de testes, para a verificação da velocidade de um eventual sistema prático de auxílio à tomada de decisões. Outra abordagem foi o uso de planejadores, incluindo o nosso protótipo, a partir de descrições *viciadas* de problema da água. Ou seja, agregamos conhecimento embutido à descrição do domínio, ou condições fixas aos operadores. Tais condições foram definidas a partir de análises humanas sobre os resultados esperados, e dependem do conhecimento de algum especialista. Embora isso contrarie a meta de gerar um planejador específico, abre uma nova perspectiva de pesquisa, onde ferramentas adequadas ao mundo real podem herdar técnicas heurísticas de busca. A análise e projeto de tais ferramentas é um dos trabalhos em aberto reconhecidos no fechamento desta dissertação.

[1] John Slaney e Sylvie Thiébaux chefiam o centro de pesquisas tecnológicas da *Australian National University*/Sidney-AU.

5.1.3 A confiança nos planejadores

Outra vantagem de considerarmos a capacidade de planejadores genéricos manipulando informações específicas de um problema é resgatar a confiança nos planos gerados. Como vimos no Capítulo 3, os resultados dos planejadores atuais sofrem de baixa aplicabilidade no mundo real. Além da baixa expressividade da linguagem PDDL em relação a recursos, temos outros aspectos caóticos presentes nos sistemas do mundo real: mutabilidade dos fatos, conseqüências financeiras e sociais da tomada de decisões, prioridade do bem estar humano em relação a previsões matemáticas, etc.

Um sistema de administração de água, por exemplo, não permite que resultados de uma ferramenta sejam diretamente aplicados. Não que esses resultados não possam corresponder à realidade, mas as temíveis conseqüências de uma eventual falha impedem que os técnicos assumam o risco. Para que uma ferramenta possa efetivamente ser útil, seria necessário que a sua precisão fosse préviamente reconhecida. Seria necessário algum tipo de garantia vinculada aos planos.

Planejadores como o nosso protótipo apenas geram planos, mas não testam se esses planos são razoáveis em relação ao senso comum humano sobre as conseqüências de sua execução. Essa verificação, mesmo para domínios e problemas relativamente pequenos, é custosa e passível de falhas quando realizada manualmente. Alguns trabalhos já apontam a necessidade de um instrumento automático de validação de planos, incluindo a ferramenta utilizada no AIPS para testar os resultados das competições. Além da verificação sintática e semântica dos planos gerados, sugere-se uma verificação de *sensatez* de um determinado plano em relação à sua aplicabilidade ao mundo real. A criação de tal ferramenta de testes pressupõe um ampla discussão sobre quais aspectos esses testes deveriam verificar, sendo tarefa para um novo projeto.

5.2 A seqüência do trabalho

Como apresentado na seção anterior, muito do que se pesquisou neste trabalho suscita dúvidas e novos horizontes. Consideramos que o trabalho aqui relatado cumpriu a sua missão de introduzir uma nova área de pesquisa ao ambiente cearense e brasileiro. Não apenas pelos resultados obtidos, mas principalmente pela discussão

envolvendo diversas áreas de conhecimento - principalmente linguagens simbólicas, heurísticas de busca, grafos, algoritmos dedutivos e técnicas de programação. A troca de idéias durante nosso trabalho envolveu diversos grupos de pesquisa do Departamento de Computação da Universidade Federal do Ceará, técnicos da FUNCEME e a colaboração de pares internacionais através da Internet. Acreditamos que o conhecimento exigido para a implementação de ferramentas planejadoras complexas requerem o amadurecimento de uma visão holística sobre a ciência da computação. Não acreditamos suficiente o conhecimento isolado de qualquer uma das áreas citadas acima, mas necessária a presença de cada uma delas na formação de um ambiente de pesquisa proficiente em planejamento automático.

Concluímos esta etapa do trabalho conscientes do amplo desafio a ser ainda enfrentado. O desenvolvimento do protótipo e a análise de sua aplicabilidade no problema dos reservatórios de água nos permitiu identificar novos rumos em relação à pesquisa original - a maioria deles grandiosos em relação à suas dificuldades e objetivos. A despeito das gerações necessárias para a consolidação deste tema, esperamos ter contribuido na geração de uma trilha de pesquisa atraente para todos aqueles que sonham em dominar os computadores.

A

Projeto WAVES

WAVES - Water Availability and Vulnerability of Ecosystems of Northeastern of Brazil - é um projeto de cooperação Brasil-Alemanha que estuda a qualidade e disponibilidade de recursos hídricos em regiões de clima semi-árido, identificando a influência de fenômenos naturais e sociais sobre esses recursos.

A.1 Motivação

Como resultado do desenvolvimento econômico e industrial do mundo, e particularmente da utilização de fontes de energia fósseis, uma concentração crescente de gases tóxicos na atmosfera provocou mudanças no clima do planeta. Uma mudança climática global pode afetar regiões semi-áridas de maneira muito mais intensa. Essas áreas cobrem um terço do planeta e concentram aproximadamente 20% de sua população. Indicadores sócio-econômicos mostram que as condições de vida nesses locais estão muito abaixo da média mundial. Elevações da temperatura média, períodos prolongados de seca e aumento da variação climática têm um impacto nos processos hídricos, na vegetação e na utilização do solo e, conseqüentemente, na base das condições de vida humana. Se os sistemas de utilização do solo não forem adaptados, essas mudanças podem acelerar a desertificação e a degradação do solo, promovendo, ainda, uma migração massiva das áreas rurais para os centros urbanos ou outras áreas mais favoráveis. Se as conseqüências potenciais de uma mudança climática em regiões semi-áridas devem ser consideradas, também deve ser feita uma análise cruzada da interação entre o clima, a geosfera, o ciclo da água, a biosfera e, por último, mas não menos importante, as atividades humanas. Estudos integrados envolvendo hidrologia, ecologia, meteorologia, climatologia, pedologia, agronomia e ciências sociais e econômicas são necessários não apenas para melhorar as condições sócio-econômicas, mas também para tornar as regiões semi-áridas menos vulneráveis

a possíveis mudanças climáticas.

Como resultado do desenvolvimento econômico e industrial do mundo, o efeito estufa se impôs como agente responsável pelo desequilíbrio metereológico observado no final do século vinte. Enchentes, secas prolongadas e temperaturas desproporcionais não podem mais ser tratados como anormalidades climáticas, mas sim como uma adaptação do ecossistema Terrestre às ações de seu principal agente: o ser humano. Esse novo contexto torna premente a geração de tecnologias que permitam o desenvolvimento social em harmonia com o meio ambiente em que vivemos.

B

Contexto administrativo dos recursos hídricos no Estado do Ceará

Neste apêndice reunimos os aspectos políticos e sociais que regem a administração do sistema de reservatórios do Ceará, a começar pelo plano diretor do Estado em relação aos recursos hídricos.

B.1 Plano diretor

O Governo cearense, nos últimos doze anos vem implementando um importante esforço no sentido de promover um avanço significativo na política de desenvolvimento dos Recursos Hídricos, objeto central de um programa de convivência com a seca. Ao lado de um forte e planejado projeto de ampliação da infra-estrutura hídrica, o Estado estabeleceu um aparato jurídico - institucional para permitir a implantação de um ambicioso plano de gerenciamento da água para uso múltiplo no território estadual.

A Política Estadual de Recursos Hídricos, prevista no artigo 326 da Constituição Estadual, definida pela Lei Estadual de Recursos Hídricos Lei N° 11.996, de 24 de julho de 1992 visa proporcionar os meios para que a água, recurso essencial ao desenvolvimento sócio econômico, seja usada de forma racional e justa pelo conjunto da sociedade, em todo território do Ceará, tendo como objetivos principais:

- Compatibilizar a ação humana, em qualquer de suas manifestações, com a dinâmica do ciclo hidrólogo no Estado do Ceará, de forma a assegurar as condições para o desenvolvimento econômico e social, com melhoria da qualidade de vida e em equilíbrio com o meio ambiente;
- Assegurar que a água, recurso natural essencial à vida, ao desenvolvimento econômico e ao bem estar social possa ser controlada e utilizada, em padrões de

qualidade e quantidade satisfatórios, por seus usuários atuais e pelas gerações futuras, em todo o território do Estado do Ceará;

- Planejar e gerenciar, de forma integrada, descentralizada e participativa, o uso múltiplo, controle, conservação, proteção e preservação dos recursos hídricos.

Para implementação do modelo de gerenciamento dos recursos hídricos integrado, descentralizado e participativo sem a dissociação dos aspectos qualitativos e quantitativos, considerando as fases aérea, superficial e subterrânea do ciclo hidrológico, previsto na lei estadual de recursos hídricos, foi criada a Companhia de Gestão dos Recursos Hídricos do Estado do Ceará - COGERH. A Companhia das Águas, como vem sendo chamada, foi criada pela Lei nº 12.217, de 18 de novembro de 1993, com a missão de Gerenciar os Recursos Hídricos de Domínio do Estado do Ceará, e da União por delegação, de forma integrada, decentralizada e participativa, promovendo o seu uso racional, social e sustentado.

B.2 Planejamento do gerenciamento dos sistemas hídricos

A COGERH, já elaborou o Plano de Gerenciamento da Bacia do Curu, o Plano de Gerenciamento das Bacias Metropolitanas, o Plano de Gerenciamento da Bacia do Jaguaribe, o Estudo de Avaliação do Potencial de Águas Subterrâneas e de Lagoas das Bacias Hidrográficas da Região Metropolitana de Fortaleza, o Diagnóstico e Projeto de Recuperação da Infra - Estrutura Hídrica de Água Bruta do Sistema de Abastecimento D'Água da Região Metropolitana de Fortaleza, no âmbito do PROURB, com financiamento do BIRD, Os Planos de Gerenciamento das Bacias do Parnaíba, Acaraú e Coreaú e Litorâneas estão sendo estudados no âmbito do PROGERIRH, com financiamento do BIRD, através dos estudos referentes ao Eixo de Integração da Ibiapaba. Além dos Planos de Gerenciamento de Bacias, a COGERH está desenvolvendo um Sistema de Suporte a Decisão Espacial (SSDE), visando melhorar a qualidade do gerenciamento dos recursos hídricos no Estado. Este sistema é constituído de três partes: Banco de dados unificado com as informações hidrológicas históricas e as obtidas pelo monitoramento da COGERH, bem como os dados das demandas dos diversos usuários, conseguidas através de cadastros e atualizados com o auxílio de imagens de satélites; Modelos computacionais que possibilitem a avaliação da evolução dos estoques de água dos reservatórios em

diversos cenários hidrológicos e de demandas, e modelos computacionais que possibilitem o cálculo da demanda como função da área irrigada, do tipo de cultivo e da região do Estado; Interfaces gráficas, que possibilitem usuários, não especialistas, manipularem o sistema, fazendo perguntas sobre o comportamento histórico das demandas e ofertas, além de prever situações futuras do sistema sujeitas a vários cenários de oferta e demanda.

B.2.1 Organização dos usuários

Na Política de Águas do Ceará, é dado ênfase à organização dos usuários como forma de garantir a participação destes no gerenciamento dos recursos hídricos. A COGERH vem desenvolvendo um trabalho de conscientização e educação para a gestão das águas dos açudes estratégicos dos municípios, dos vales perenizados e das bacias hidrográficas, em especial nas bacias do Curu, Alto, Médio e Baixo Jaguaribe, Banabuiú, Metropolitanas e mais recentemente do Salgado e Acaraú Os canais de participação no processo de gestão das águas são garantidos em cada um dos níveis (açudes, municípios, vales perenizados e bacias hidrográficas) onde são constituídas comissões de usuários, sendo que o Comitê de Bacia Hidrográfica, previsto pela lei estadual n.º 11.996 de 1992, com poder consultivo e deliberativo, é a instância mais importante de participação dos usuários e demais setores (sociedade civil, poder público etc) e de planejamento e ações na área dos recursos hídricos. Em outubro de 1997, foi constituído o Comitê da Bacia Hidrográfica do Curu (primeiro comitê de bacia do Ceará e do Nordeste) que se encontra em processo de renovação de diretoria para o terceiro mandato. Em abril de 1999, foram instalados os Comitês das Bacias do Baixo e Médio Jaguaribe, que contam com a participação de expressivos setores dos usuários, da sociedade civil e dos poderes públicos que atuam na região. Em 2001, foi dado posse à diretoria do Comitê da Sub-Bacia Hidrográfica do rio Banabuiú. Em 2002, foi homologado pelo Conselho Estadual de Recursos Hídricos (CONERH), a criação dos comitês da Sub-Bacia do Alto Jaguaribe e da Sub-Bacia do Salgado. Na bacia do rio Acaraú, foi constituída a Comissão de Usuários do Vale do Acaraú, onde foram realizados três encontros regionais para discussão do processo de organização do Comitê de Bacia. O Fórum das Águas das Bacias Metropolitanas, que vem funcionando desde março de 1997, marcou para julho deste ano a realização do Congresso de Constituição do Comitê de Bacia. Está previsto,

para o segundo semestre deste ano, o início dos trabalhos na bacia do Parnaíba, com o diagnóstico institucional das organizações e entidades que atuam na bacia. A COGERH, em articulação com outros órgãos e entidades que atuam nas bacias hidrográficas realizou durante o ano de 2001, 181 eventos entre seminários, cursos, reuniões de operação de açudes e de negociação de conflitos, as quais contaram com 7.251 participantes

B.2.2 Monitoramento e operação dos sistemas hídricos

O monitoramento como instrumento da gestão dos recursos hídricos tem a função de realizar as macromedições e o acompanhamento dos aspectos qualitativos e quantitativos da água, no que diz respeito aos níveis dos açudes, vazões liberadas, consumo dos usuários, vazões nos rios perenizados e os níveis de contaminação química e biológica, servindo de informação para auxiliar a tomada de decisão da operação. No Ceará, do regime de chuvas concentrado em quatro meses, associado a uma formação geológica com predominância de rochas cristalinas (70% do território é formado por rochas cristalinas), resultam rios intermitentes que permanecem secos cerca de seis meses por ano e não raro o ano inteiro. Nestas condições, o fornecimento de água para os mais diversos usos, deve provir do armazenamento em reservatórios superficiais e em menor escala dos poços perfurados. A adoção da bacia hidrográfica como unidade de planejamento figura como um dos princípios fundamentais do gerenciamento dos recursos hídricos. No Ceará foram delineadas 11 bacias. No Estado do Ceará foram cadastrados pela Secretaria dos Recursos Hídricos mais de 7.200 açudes, com um potencial de acumulação estimado em 12 bilhões de metros cúbicos. Com uma reserva explorável estimada em 1,2 bilhão de metros cúbicos por ano, o Estado tem hoje cadastrados mais de 13.000 poços. A operação objetiva principalmente no caso dos açudes, definir a liberação de águas de forma a atender a demanda (os usos), levando em consideração a oferta disponível e as características do próprio açude. Atualmente a COGERH gerencia em convênio com o DNOCS, 108 açudes públicos, com capacidade total de acumulação de 10,3 bilhões de metros cúbicos. Além do monitoramento quantitativo dos níveis d'água dos reservatórios e das vazões de perenização dos leitos naturais e vazões transferidas por canais e adutoras, a COGERH, vem mais recentemente exercendo o monitoramento qualitativo dos recursos hídricos das bacias do Médio e Baixo Jaguaribe,

Banabuiú, Curu, Acaraú e Metropolitanas. Os parâmetros por enquanto analisados são: concentração de cloretos, condutividade elétrica, oxigênio dissolvido, ph e turbidez. Para o desenvolvimento do monitoramento dos recursos hídricos estaduais, a COGERH recuperou e ampliou a rede de estações limnimétricas dos 110 maiores açudes do Estado, e instalou em convênio com o Ministério da Ciência e Tecnologia, um conjunto de 7 (sete) plataformas de coletas de dados em tempo real (PCD) nos sistemas hídricos do Jaguaribe e Metropolitano. A COGERH, vem implantando desde 1998 um projeto piloto de macro hidrometação. O plano se iniciou com a contratação de dois consultores e prosseguiu com a aquisição de equipamentos de última geração, como medidores de vazão eletromagnética e ultra-sônica, que chegam a ter imprecisão em torno de 2,5% da vazão medida. Encontra-se em fase de implantação 16 medidores dos mais diversos tipos na Bacia Metropolitana. Procurou-se utilizar equipamentos de alta precisão para a medição de indústrias como Antártica, Kaiser, Bermas e Marisol.

C

Linguagem de definição de domínios de planejamento - PDDL

Essa seção descreve a parte da linguagem PDDL - *Planning Domain Definition Language* - que permite a descrição de domínio e problemas em formato compatível com STRIPS. Embora a linguagem PDDL permita a descrição de problemas no formato ADL [95], descrevemos apenas o subconjunto da linguagem utilizada pelo planejador aqui proposto.

C.1 PDDL

A linguagem PDDL [82] foi criada por Drew McDermott em 1998 para a primeira versão da competição mundial de planejamento automático - AI Planning Systems 98 [84]. O objetivo dessa linguagem é estabelecer um padrão de notação para a descrição de domínios usados como referência(*benchmarks*) na análise do desempenho dos planejadores durante competições.

Desde a sua criação, algumas extensões vem sendo adotadas para melhorar a sua expressividade. No texto que segue, apresentamos a versão utilizada no AIPS 2000 [4, 7].

C.2 Sintaxe

A sintaxe da linguagem PDDL segue o seguinte formato:

i Cada regra é escrita no formato <*elemento sintático*> ::= *expansão*.

ii Os símbolos de maior e menor (< e >) delimitam os nomes dos elementos sintáticos.

iii Colchetes ([e]) delimitam informações opcionais.

iv O asterisco ($*$) representa *zero ou mais elementos*, enquanto o sinal positivo ($+$) representa *um ou mais elementos*.

v Alguns elementos sintáticos são parametrizados. Exemplo: *<list (symbol)>* denota uma lista de símbolos, onde há uma definição EBNF para $<list\ x>$ e uma definição para *<symbol>*:

- $<list\ x>$ é definido como $<list\ x> ::= (x^*)$
- $<list\ x>$ pode ser definido apenas como $< symbol >^*$

vi Parêntesis são apenas delimitadores dos elementos sintático, não possuindo nenhuma interpretação semãntica na meta linguagem EBNF.

vii Informação adicional e regras de expansão podem aparecer sobrescritas por um marcador, tal como:

- $[(: types...)]^{:typing}$
- $< atomic\ formula >::=^{:typing} (< predicate > < typed\ list\ (variable) >)$

A descrição do domínio de um problema deve definir a função de cada marcador que aparecer sobrescrito nos elementos sintáticos.

C.3 Descrição de domínios

A EBNF para a definição de domínios é apresentada na Figura C.1. Além disso, nas competições do AIPS[1] são consideradas as seguintes restrições:

i As palavras chaves devem aparecem na ordem em que estão enumeradas na Figura C.1.

ii A definição do domínio e a definição dos problemas devem aparecer completamente definidos em arquivos separados. Não existem recursos para compartilhamento de definições distribuídos em dois ou mais arquivos.

[1] O planejador apresentado nessa dissertação adota os padrões usados no AIPS, permitindo assim a comparação dos resultados obtidos frente ao estado da arte.

Names: a categoria $< name >$ consiste de uma palavra iniciada por uma letra e contendo: letras, dígitos, hifens $(-)$ e subescritos $(_)$. Letras maiúsculas e minúsculas não são diferenciadas. Os nomes definidos em $(< name >)$ devem ser únicos.

$$
\begin{array}{lll}
< domain > & ::= & (define\ (domain\ < name >) \\
 & & \quad [< required - def >] \\
 & & \quad [< types - def >]^{:typing} \\
 & & \quad [< constants - def >] \\
 & & \quad [< predicates - def >] \\
 & & \quad < action - def >^{*}) \\
< require - def > & ::= & (: requirements\ < require - key >^{+}) \\
< require - key > & ::= & : strips \\
< require - key > & ::= & : adl \\
< require - key > & ::= & : typing \\
< types - def > & ::= & (: types\ < typed\ list\ (name) >) \\
< typed? - list - of\ (t) > & ::=^{:typing} & < typed\ list\ (t) > \\
< typed? - list - of\ (t) > & ::= & < list\ (t) > \\
< constants - def > & ::= & (: constants\ < typed?\ list\ of\ (names) >) \\
< predicates - def > & ::= & (: predicates\ < atomic\ formula >^{+}) \\
< atomic\ formula > & ::= & (< predicate > < typed?\ list\ of\ (variables) >) \\
< predicate > & ::= & < name > \\
< variable > & ::= & ? < name >
\end{array}
$$

Figura C.1. *Sintaxe de definição de domínios.*

Requirements: existem dois possíveis tipos de domínios: STRIPS e ADL. O formato padrão de descrição de domínios é o STRIPS, não necessitando de declaração explícita. O formato ADL é um superconjunto do formato STRIPS, portanto não há necessidade de declaração do formato STRIPS quando usarmos o formato ADL. Os possíveis valores para as chave : *requirement* são:

: *strips*	domínios com ações no formato STRIPS
: *adl*	domínios com ações no formato ADL

Types: os argumentos de tipagem (: *types*) são definidos a partir da seguinte sintaxe:

$$
\begin{array}{ll}
< \textit{typed list } (x) > & ::= \ x^{+} \\
< \textit{typed list } (x) > & ::= \ x^{+} - \ < \textit{typed list } (x) > \\
< \textit{type} > & ::= < \textit{name} > \\
< \textit{type} > & ::= \ (\textit{either} \ < \textit{name} >^{+})
\end{array}
$$

Uma lista tipada é usada para declarar os tipos de uma lista de entidades.

Os tipos são precedidos por um sinal de menos ($-$), e são declarados ao final de uma lista de objetos. Existem dois tipos pré-definidos: o *object* e o *number*.

Exemplo de um $<$ *typed list* (*name*) $>$:

volume distancia $-$*number* reservatorio

Se uma definição de domínio, ocorrer a linha acima como argumento de : *types*, são declarados três novos tipos: volume, distância e reservatório. Volume e distancia são declarados tipos numéricos, enquanto o reservatório é declarado apenas como *objetc*. Ou seja, cada objeto insanciado como volume ou distancia representará um número, enquanto objetos instanciados como reservatorio são apenas estruturas padrões *object*. Um tipo atômico é apenas um predicado unário atemporal e pode ser usado em todo intervalo de tempo onde esse predicado não seja negado. Além de nomes de tipos atômicos, também existem tipos agregados de dados: (*either* $t_1 ... t_k$) representa a união dos tipos $t_1 + t_2 + ... + t_k$.

Constants: o campo : *constants* é simplesmente uma lista de nomes (esses nomes podem ser tipados se a chave de requisito : *typing* for especificada). Os nomes da lista são tomadas como novas constantes do domínio (talvez com tipos especificados). Por exemplo:

(: *constants* Taua $-$*reservatorio*
distTauaOroz distTauaMachadinho $-$*distancia*)

Essa declaração indica que nesse domínio existem três constantes distintas, Taua, do tipo reservatorio, e distTauaOroz e distTauaMachadinho do tipo distancia. Se os tipos não são necessários, a seguinte declaração pode ser feita:

(: *constants* Taua distTauaOroz distTauaMachadinho)

Predicates: os campos : *predicate* consistem de uma lista de declarações de predicados. Para cada predicado são especificados uma lista de variávies (possivelmente tipadas) e a aridade do predicado (e, se necessário, os tipos de seus argumentos). Igualdade (=) é um predicado binário predefinido.

C.4 Ações

C.4.1 Ações em formato STRIPS

Se a definição do domínio especifica ações no estilo STRIPS (não possui a chave : *adl*), então a definição das ações válidas nesse domínio seguem o formato contido na Figura C.2.

$$
\begin{array}{lcl}
< action - def > & ::= & (: action \ \ < name > \\
 & & \ : parameters \ (< typed?\ list\ of\ (variables) >) \\
 & & \ < action - def\ body >) \\
< action - def\ body > & ::= & : precondition \ < POS - CONJUNCTION > \\
 & & \ : effect \ < CONJUNCTION > \\
< term > & ::= & < name > \\
< term > & ::= & < variable > \\
< atomic\ formula\ (t) > & ::= & (< predicate > \ t^{*}) \\
< literal\ (t) > & ::= & < atomic\ formula\ (t) >) \\
< literal\ (t) > & ::= & (not \ < atomic\ formula\ (t) >) \\
< POS - CONJUNCTION > & ::= & < atomic\ formula\ (term) > \\
< POS - CONJUNCTION > & ::= & (and \ < atomic\ formula\ (term) > \\
 & & \ < atomic\ formula\ (term) >^{+}) \\
< CONJUNCTION > & ::= & < literal\ (term) > \\
< CONJUNCTION > & ::= & (and \ < literal\ (term) > < literal\ (term) >^{+})
\end{array}
$$

Figura C.2. *Sintaxe de ações no formato STRIPS.*

C.4.2 Ações em formato ADL

Se o domínio declara o uso de ações no formato ADL, utilizamos a sintaxe de definição de ações que aparece na Figura C.3.

Todas as variáveis livres em : $precondition$ $< FORMULA >$ e em : $effects$ $< EFF - FORMULAS >$ devem aparecer em : $parameters$.

Os efeitos de uma ação ADL são especificados de maneira a evitar certas construções cujas semânticas seriam difíceis de ser interpretadas. Em particular, quando usamos $when$ (diferente de implicação) temos uma EFF-FORMULA, a qual não pode mais aparecer como antecedente de outro $when$. Ou seja, não podemos aninhar $whens$.

Ações ADL tem semântica semelhante às suas contrapartes STRIPS. Novamente, cada instância de uma variável em : $parameters$, digamos σ, gera uma instância particular do esquema que representa uma ação. A instância Δ, de um esquema que representa uma ação, é aplicável a um estado $\mathcal{S}$ se e somente se este estado satisfizer as precondições de Δ considerando o valor de σ. Se $\mathcal{S}$ satisfizer as pré-condições de um esquema de ação, instanciados com os valores de σ, ess ação irá mapear $\mathcal{S}$ para um novo estado $\mathcal{S}'$. $\mathcal{S}'$ pode ser computada a partir de $\mathcal{S}$ através do conjunto : $effect$ da ação aplicada(deixando inalteradas todas as outras fórmulas atômicas verdadeiras em $\mathcal{S}$ e que não aparecem negadas em : $effect$).

A aplicação de uma ação ADL em um estado $\mathcal{S}$ depende do formato em que os efeitos são declarados:

i Efeitos declarados no formato $< ATOMIC - EFFS >$ são aplicados a partir da instanciação de seus parâmetros. Uma vez que os predicados contidos em $< ATOMIC - EFFS >$ estão totalmente instanciados, eles são adicionados ao estado $\mathcal{S}$. Tal qual os operadores de deleção em STRIPS, são retirados de $\mathcal{S}$ os predicados que aparecem negados em $< ATOMIC - EFFS >$.

ii Efeitos declarados no formato ($when$ $< FORMULA >$ $< ATOMIC - EFFS >$) são aplicados se $\mathcal{S} \models < FORMULA >$. Se o estado $\mathcal{S}$ satisfizer a relação descrita em $< FORMULA >$, os predicados de $< ATOMIC - EFFS >$ são adicionados a $\mathcal{S}$.

iii Efeitos declarados no formato ($forall$ ($<$ *typed*? $list\ of\ variables$ $>)^+$ $< ATOMIC - EFFS >$) e no formato ($forall$ ($<$ *typed*? $list\ of\ variables$ $>$

$$
\begin{array}{lcl}
< action - def > & ::= & (: action \ \ < name > \\
 & & \ : parameters \ (< typed? \ list \ of \ (variable) >) \\
 & & \ < action - def \ body >) \\
< action - def \ body > & ::= & : precondition \ < FORMULA > \\
 & & \ : effect \ < EFF - FORMULA > \\
< FORMULA > & ::= & < literal \ term) > \\
< FORMULA > & ::= & not \ < FORMULA > \\
< FORMULA > & ::= & (and \ < FORMULA > < FORMULA >^{+}) \\
< FORMULA > & ::= & (or \ < FORMULA > < FORMULA >^{+}) \\
< FORMULA > & ::= & (imply \ < FORMULA > < FORMULA >) \\
< FORMULA > & ::= & (exists \ (< typed \ list \ of \ (variable) >^{+}) \\
 & & \ < FORMULA >) \\
< FORMULA > & ::= & (exists \ (< typed \ list \ of \ (variable) >^{+}) \\
 & & \ < FORMULA >) \\
< ATOMIC - EFFS > & ::= & < literal \ term) > \\
< ATOMIC - EFFS > & ::= & (and \ < literal \ term) > < literal \ term) >^{+}) \\
< EFF - FORMULA^{*} > & ::= & < ATOMIC - EFFS > \\
< EFF - FORMULA^{*} > & ::= & (when \ < FORMULA > \\
 & & \ < ATOMIC - EFFS >) \\
< EFF - FORMULA^{*} > & ::= & (forall \ (< typed? \ list \ of \ (variable) >^{+}) \\
 & & \ < ATOMIC - EFFS >)) \\
< EFF - FORMULA^{*} > & ::= & (forall \ (< typed? \ list \ of \ (variable) >^{+}) \\
 & & \ (when \ < FORMULA > \\
 & & \ < ATOMIC - EFFS >)) \\
< EFF - FORMULA > & ::= & < EFF - FORMULA^{*} > \\
< EFF - FORMULA > & ::= & (and \ < EFF - FORMULA^{*} > \\
 & & \ < EFF - FORMULA^{*} >^{+})
\end{array}
$$

Figura C.3. *Sintaxe de ações no formato ADL.*

$)^+ < when\ < FORMULA >\ ATOMIC-EFFS >)$) são aplicados a partir de todas as possíveis instâncias de $< typed?\ list\ of\ variables >$[2]

iv Efeitos declarados no formato ($and\ < ONE-EFF-FORMULA > < ONE-EFF-FORMULA >^+$) são aplicados através da inclusão de cada uma das $< ONE-EFF-FORMULA >$ em $\mathcal{S}$.

C.5 Representação dos problemas

Um problema é o que o planejador tenta resolver, e é definido em relação ao domínio em questão. Um problema enumera o estado inicial e o estado objetivo através do formato demonstrado na Figura C.4.

$$
\begin{array}{lll}
< action-def > & ::= & (:action\ < name > \\
 & & :parameters\ (< typed?\ list\ of\ (variable) >) \\
 & & < action-def\ body >) \\
< problem > & ::= & (define\ (problem\ < name >) \\
 & & (:domain\ < name >) \\
 & & [(:requirements\ :typing)] \\
 & & < object\ declaration > \\
 & & [< init >] \\
 & & < goal >^+ \\
< objectdeclaration > & ::= & (:objects\ < typed?\ list\ of\ (name) >) \\
< init > & ::= & (:init\ < atomic\ formula\ (name) >^+ \\
< goal > & ::= & (:goal\ < FORMULA >)
\end{array}
$$

Figura C.4. *Sintaxe da definição de problemas.*

O estado inicial ($:init$) de um problema é uma lista de fórmulas atômicas consideradas verdades no instante inicial, normalmente chamado de S_0.

A descrição dos estados em um arquivo contendo a definição de um problem deve assumir um mundo fechado, ou seja, todo o predicado não declarado em um estado é automaticamente considerado falso [103].

[2]Variáveis tipadas são limitadas a constantes de seu tipo compatível. Variáveis não tipadas podem assumir o valor de qualquer constante do domínio.

O campo : *objects* é obrigatório e enumera os objetos pertencentes ao domínio. Note que todos os objeto presentes no domínio devem ser declarados, mesmo aqueles cujo o status não é declarado no estado inicial.

O estado objetivo (: *goal*) de um problema é uma fórmula. Uma solução para um problema é uma seqüência de ações tal que: (a) a sequência de ações é aplicável a partir do estado inicial do problema; (b) o estado objetivo é um subconjunto do estado gerado pela aplicação da seqüência de ações ao estado inicial.

Por convenção, a representação dos estados se limita a conjunções de fórmulas atômicas.

C.6 Exemplos de definições PDDL

Para exemplificarmos o uso da linguagem PDDL, utilizaremos domínios que foram usados como referências (*benchmarks*) na comparação do desempenho dos planejadores durante o AIPS 2000.

C.6.1 Logistics (STRIPS)

```
;; logistics domain
;;

(define (domain logistics)
  (:requirements :strips)
  (:predicates  (package ?obj)
        (truck ?truck)
(airplane ?airplane)
                (airport ?airport)
                (location ?loc)
(in-city ?obj ?city)
                (city ?city)
(at ?obj ?loc)
(in ?obj ?obj))
```

```
(:action load-truck
  :parameters
   (?obj
    ?truck
    ?loc)
  :precondition
   (and (package ?obj) (truck ?truck) (location ?loc)
   (at ?truck ?loc) (at ?obj ?loc))
  :effect
   (and (not (at ?obj ?loc)) (in ?obj ?truck)))

(:action load-airplane
  :parameters
   (?obj
    ?airplane
    ?loc)
  :precondition
   (and (package ?obj) (airplane ?airplane) (location ?loc)
   (at ?obj ?loc) (at ?airplane ?loc))
  :effect
   (and (not (at ?obj ?loc)) (in ?obj ?airplane)))

(:action unload-truck
  :parameters
   (?obj
    ?truck
    ?loc)
  :precondition
   (and (package ?obj) (truck ?truck) (location ?loc)
        (at ?truck ?loc) (in ?obj ?truck))
  :effect
   (and (not (in ?obj ?truck)) (at ?obj ?loc)))
```

```
(:action unload-airplane
  :parameters
   (?obj
    ?airplane
    ?loc)
  :precondition
   (and (package ?obj) (airplane ?airplane) (location ?loc)
        (in ?obj ?airplane) (at ?airplane ?loc))
  :effect
   (and (not (in ?obj ?airplane)) (at ?obj ?loc)))

(:action drive-truck
  :parameters
   (?truck
    ?loc-from
    ?loc-to
    ?city)
  :precondition
   (and (truck ?truck) (location ?loc-from) (location ?loc-to) (city ?city)
   (at ?truck ?loc-from)
   (in-city ?loc-from ?city)
   (in-city ?loc-to ?city))
  :effect
   (and (not (at ?truck ?loc-from)) (at ?truck ?loc-to)))

(:action fly-airplane
  :parameters
   (?airplane
    ?loc-from
    ?loc-to)
  :precondition
   (and (airplane ?airplane) (airport ?loc-from) (airport ?loc-to)
(at ?airplane ?loc-from))
```

```
  :effect
   (and (not (at ?airplane ?loc-from)) (at ?airplane ?loc-to)))
)

;; Problem probLOGISTICS-10-0
;;

(define (problem logistics-10-0)
(:domain logistics)
(:objects apn1 apt1 apt2 apt3 apt4 pos1 pos2 pos3 pos4 pos5
cit4 cit3 cit2 cit1 tru4 tru3 tru2 tru1 obj43 obj42
obj41 obj33 obj32 obj31 obj23 obj22 obj21 obj13 obj12 obj11 )

(:init (package obj11) (package obj12) (package obj13) (package obj21)
  (package obj22) (package obj23) (package obj31) (package obj32)
(package obj33) (package obj41) (package obj42) (package obj43)
(truck tru1) (truck tru2) (truck tru3) (truck tru4) (city cit1)
(city cit2) (city cit3) (city cit4) (location pos1)
(location apt1) (location pos2) (location apt2) (location pos3)
(location apt3) (location pos4) (location apt4) (airport apt1)
(airport apt2) (airport apt3) (airport apt4) (airplane apn1)
(at apn1 apt1) (at tru1 pos1) (at obj11 pos1) (at obj12 pos1)
(at obj13 pos1) (at tru2 pos2) (at obj21 pos2) (at obj22 pos2)
(at obj23 pos2) (at tru3 pos3) (at obj31 pos3) (at obj32 pos3)
(at obj33 pos3) (at tru4 pos4) (at obj41 pos4) (at obj42 pos4)
(at obj43 pos4) (in-city pos1 cit1) (in-city apt1 cit1)
(in-city pos2 cit2) (in-city apt2 cit2) (in-city pos3 cit3)
(in-city apt3 cit3) (in-city pos4 cit4) (in-city apt4 cit4))

(:goal (and (at obj31 pos3) (at obj33 apt3) (at obj41 apt3)
(at obj23 pos4) (at obj11 pos3) (at obj22 apt2)
(at obj12 apt1) (at obj21 pos4) (at obj42 pos4)
(at obj32 pos1)))
)
```

C.6.2 Logistics (ADL)

```
; Revised by DVM, 6/3/1998 so that all domains have consistent names.
; The only difference in the logic is the use of conditional effects
; in the adl version, so that an object is at a location before it is
; unloaded.

;;; these domains engineered from the att satplan encodings
;;; which were derived from the graphplan domains
;;; which I think were derived from Veloso's prodigy domains
;;; - DSW 1/97

(define (domain logistics-strips)
  (:requirements :strips)
  (:predicates  (OBJ ?obj)
        (TRUCK ?truck)
                (LOCATION ?loc)
(AIRPLANE ?airplane)
                (CITY ?city)
                (AIRPORT ?airport)
(at ?obj ?loc)
(in ?obj1 ?obj2)
(in-city ?obj ?city))

  ; (:types ) ; default object

(: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 LOAD-AIRPLANE
  :parameters
   (?obj
    ?airplane
    ?loc)
  :precondition
   (and (OBJ ?obj) (AIRPLANE ?airplane) (LOCATION ?loc)
   (at ?obj ?loc) (at ?airplane ?loc))
  :effect
   (and (not (at ?obj ?loc)) (in ?obj ?airplane)))

(: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 UNLOAD-AIRPLANE
  :parameters
   (?obj
    ?airplane
    ?loc)
  :precondition
   (and (OBJ ?obj) (AIRPLANE ?airplane) (LOCATION ?loc)
        (in ?obj ?airplane) (at ?airplane ?loc))
```

```
  :effect
   (and (not (in ?obj ?airplane)) (at ?obj ?loc)))

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

(:action FLY-AIRPLANE
  :parameters
   (?airplane
    ?loc-from
    ?loc-to)
  :precondition
   (and (AIRPLANE ?airplane) (AIRPORT ?loc-from) (AIRPORT ?loc-to)
(at ?airplane ?loc-from))
  :effect
   (and (not (at ?airplane ?loc-from)) (at ?airplane ?loc-to)))
)


;; Problem att-log0
;;
```

```
(define (problem att-log-a)
  (:domain logistics-strips)
  (:objects package1 package2 package3 package4
    package5 package6 package7 package8
    pgh-truck bos-truck la-truck airplane1 airplane2
    bos-po pgh-po la-po bos-airport pgh-airport la-airport
    pgh bos la)
  (:init (OBJ package1) ; statis predicates
 (OBJ package2)
 (OBJ package3)
 (OBJ package4)
 (OBJ package5)
 (OBJ package6)
 (OBJ package7)
 (OBJ package8)
 (TRUCK pgh-truck)
 (TRUCK bos-truck)
 (TRUCK la-truck)
 (AIRPLANE airplane1)
 (AIRPLANE airplane2)
 (LOCATION bos-po)
 (LOCATION la-po)
 (LOCATION pgh-po)
 (LOCATION bos-airport)
 (LOCATION la-airport)
 (LOCATION pgh-airport)
 (AIRPORT bos-airport)
 (AIRPORT pgh-airport)
 (AIRPORT la-airport)
 (CITY pgh)
 (CITY bos)
 (CITY la)
```

```
(IN-CITY pgh-po pgh)
(IN-CITY pgh-airport pgh)
(IN-CITY bos-po bos)
(IN-CITY bos-airport bos)
(IN-CITY la-po la)
(IN-CITY la-airport la)
(at package1 pgh-po);; dynamic predicates
(at package2 pgh-po)
(at package3 pgh-po)
(at package4 pgh-po)
(at package5 bos-po)
(at package6 bos-po)
(at package7 bos-po)
(at package8 la-po)
(at airplane1 pgh-airport)
(at airplane2 pgh-airport)
(at bos-truck bos-po)
(at pgh-truck pgh-po)
(at la-truck la-po))
 (:goal (and (at package1 bos-po)
     (at package2 bos-airport)
     (at package3 la-po)
     (at package4 la-airport)
     (at package5 pgh-po)
     (at package6 pgh-airport)
     (at package7 pgh-po)
     (at package8 pgh-po)))
 )
```

Referências Bibliográficas

[1] Aho, A. V., Sethi, R., e Ullman, J. D. *Compiladores - princípios, técnicas e ferramentas*. LTC - Livros Técnicos e Científicos Editora Ltda., 1995.

[2] Ali, S. A., Linninger, A. A., e Stephanopoulos, G. *Synthesis of Batch Processing Schemes as Synthesis of Operating Procedures : A Means-Ends Analysis and Non-Monotonic Planning Approach*. AIChE Annual Meeting, 1998.

[3] Atkins, E. M., Abdelzaher, T. F., Shin, K. G., e Durfee, E. H. Planning and resource allocation for hard real-time, fault-tolerant plan execution. *Journal of the Association for Computing Machinery - ACM* (1999).

[4] Bacchus, F. Subset of PDDL for the AIPS2000 planning competition. The AIPS-00 planning competition comitee, 2000.

[5] Bacchus, F., e Ady, M. Planning with resources and concurrency – a forward chaining approach. In *IJCAI* (2001).

[6] Bacchus, F., e Kabanza, F. Using temporal logics to express search control knowledge for planning. *Journal of Artificial Intelligence Research - JAIR 116* (2000).

[7] Bacchus, F., e Nau, D. The 2000 AI planning systems competition. The AI magazine, 2001.

[8] Barto, A. G., Bradtke, S. J., e Singh, S. P. Learning to act using real-time dynamic programming. Tech. Rep. UM-CS-1993-002, University of Massachusetts, 1993. citeseer.nj.nec.com/article/barto93learning.html.

[9] Blum, A., e Furst, M. Fast planning through planning graph analysis. In *IJCAI* (1995), pp. 1636–1642.

[10] Bonet, B., e Geffner, H. *HSP: heuristic search planner*. Universidad Simón Bolívar, Caracas/Venezuela, 1999.

[11] Bonet, B., e Geffner, H. Planning and control in artificial intelligence: a unifying perspective. Universidad Simón Bolivar, 1999.

[12] Bonet, B., e Geffner, H. Planning with incomplete information as heuristic search in belief space. *American Association for Artificial Intelligence - AAAI* (2000).

[13] Bonet, B., Loerincs, G., e Geffner, H. *A robust and fast action selection mechanism for planning*. American Association for Artificial Intelligence - AAAI, 1997.

[14] Bruynooghe, M. *Logic programming for describing and solving planning problems*. American Association for Artificial Intelligence - AAAI, 2000.

[15] Campos, N., e Studart, T. *Gestão de águas - princípios e práticas*. Associação Brasileira de Recursos Hídricos, 2001.

[16] Chien, S., Rabideau, G., Knight, R., Sherwood, R., Engelhardt, B., Mutz, D., Estlin, T., Smith, B., Fisher, F., Barrett, T., Stebbins, G., e Tran, D. ASPEN - Automating space mission operations using automated planning and scheduling. In *SpaceOps* (Toulouse, France, 2000). http://www-aig.jpl.nasa.gov/public/planning/aspen/aspen_index.html/.

[17] Cimatti, A., Roveri, M., e Traverso, P. Strong planning in non-deterministic domains via model checking. In *Artificial Intelligence Planning Systems* (1998), pp. 36–43.

[18] Cormen, T. H., Leiserson, C. E., e Rivest, R. L. *Introduction to algorithms*. The MIT Press, 1990.

[19] Correia, C., e Tafner, M. *Análise orientada a objetos*. Visual Books, 2001.

[20] Crawford, J., e Auton, L. Experimental results on the cross-over point in satisfatibility problems. In *11th National Conference of AI* (1993), pp. 21–27.

[21] da Rocha, H. V., e Baranauska, M. C. C. *Design e avaliação de interfaces humano-computador*. UNICAMP, 2000. Escola de computação.

[22] Davis, M., Logemann, G., e Loveland, D. A machine program for theorem proving. *Journal of the Association for Computing Machinery - ACM 5* (1962), 394–397.

[23] de Araújo, J. C., e Santaela, S. T. Gestão da qualidade. In *Gestão de águas - princípios e práticas* (2001), Associação Brasileira de Recursos Hídricos, pp. 139–157.

[24] de Assis de Souza Filho, F. Notas sobre planejamento de recursos hídricos no ceará. In *Experiências de gestão de recursos hídricos* (2001), Agência Nacional de Águas - Ministério do Meio Ambiente, pp. 13–37. http://www.cogerh.com.br.

[25] de Souza, E. C. *Programação Orientada a Objetos com Java*. Editora Relativa, 2002.

[26] Ernst, M. D., Millstein, T. D., e Weld, D. S. Automatic SAT-Compilation of Planning Problems. *IJCAI–97* (Agosto 1998), 1169–1176.

[27] Erol, K., Nau, D. S., e Subrahmanian, V. S. Complexity, Decidability and Undecidability Results for Domain-Independent Planning. *Elsevier Science* (Setembro 1994).

[28] Ferraris, P., e Giunchiglia, E. Planning as satisfiability in nondeterministic domains. *American Association for Artificial Intelligence - AAAI* (2000).

[29] Fikes, R. E., e Nilsson, N. J. STRIPS: A new aproach to the application of the theorem proving to problem solving. *Artificial Intelligence 2* (1971), 27–120.

[30] Fink, E., e Veloso, M. Prodigy 4.0 planning algorithm. University of Toronto, Canada, 1994.

[31] Fitting, M. *First-Order logic and automated theorem proving*. Imperial College of Science, Technology & Medicine, 1989.

[32] Friedman, M., e Weld, D. S. Least commitment action selection. *University of Washington* (1995).

[33] Gamma, E., Helm, R., Johnson, R., e Vlissides, J. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.

[34] Garey, M. R., e Johnson, D. S. *Computers and intractability. A guide to the theory of NP-Completeness*. Freeman and Company Publishing, 1979.

[35] Gelder, A. V., e Tsuji, Y. K. Satisfiability testing with more reasoning and less guessing. In *Cliques, Coloring and Satisfiability: Second DIMACS Implementation Challenge* (1996), D. S. Johnson and M. Trick, Eds., DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society.

[36] Gent, I., e Walsh, T. Towards an understanding of hill-climbing procedures for sat. In *American Association for Artificial Intelligence - AAAI* (Julho 1993), vol. 11, MIT Press, pp. 28–33.

[37] Ginsberg, M. L. *Readings in nonmonotonic reasoning*. Stanford University, UK, 1987.

[38] Giunchiglia, E. *Planning as Satisfiability with Expressive Action Languages: Concurrency, Constraints and Nondeterminism*. International Conference on Principles of Knowledge Representation and reasoning - Morgan Kaufmann Publishers, 2000.

[39] Giunchiglia, E., e Lifschitz, V. *Dependet Fluents*. 14th International Joint Conference on Artificial Intelligence, 1995.

[40] Giunchiglia, E., e Lifschitz, V. Action Languages, Temporal Action Logics and the Situation Calculus. *American Association for Artificial Intelligence - AAAI* (1998).

[41] Giunchiglia, E., Massaroto, A., e Sebastiani, R. *Act, and the Rest Will Follow : Exploiting Determinism in Planning as Satisfiability*. Instituto Trentino di Cultura, Itália, 1998.

[42] Green, C. Application of theorem proving to problem solving. In *IJCAI* (1969).

[43] Gupta, N., e Nau, D. S. Complexity Results for Blocks-World Planning. *American Association for Artificial Intelligence - AAAI* (1991).

[44] Haas, A. The case for domain-specific frame axioms. In *The frame Problem in Artificial Intelligence* (1987), Morgan Kaufmann.

[45] Haigh, K. Z., e Veloso, M. Route Planning by Analogy. *International Conference on Case-Based Reasoning, Portugal* (1995).

[46] Helmert, M. Complexity results for standard benchmark domains in planning. *Journal of Artificial Intelligence Research - JAIR* (2002).

[47] Hoffman, J., e Nebel, B. The FF planning system: fast plan generation trough heuristic search. In *12th International Symposium on Methodologies for Intelligent Systems - ISMIS* (2000), Springer-Verlag, pp. 216–227.

[48] Hundal, S. S., e Brown, F. M. A theory of nonmonotonic planning. *Journal of the Association for Computing Machinery - ACM* (1991).

[49] Ismail, H. O., e Shapiro, S. C. Two Problems with Reasoning and Acting in Time. *International Conference on Principles of Knowledge Representation and reasoning - Morgan Kaufmann Publishers* (2000).

[50] Kakas, A., Miller, R., e Toni, F. Planning with incomplete information. *American Association for Artificial Intelligence - AAAI* (2000).

[51] Kambhampati, S. *AI Planning : A Prospectus on Theory and Applications.* ACM Computing Surveys, 1992.

[52] Kambhampati, S., Parker, E., e Lambrecht, E. Understanding and extending Graphplan. In *Steel & Alami 1997* (1997), Steel and Alami, Eds., pp. 260–272.

[53] Kartha, G. N., e Lifschitz, V. A Simple Formalization of Actions Using Circumscription. University of Texas, USA, 1996.

[54] Kautz, H., e Cohen, B. Noise strategies for improving local search. In *12th National Conference in AI* (Julho 1994), pp. 337–343.

[55] Kautz, H., Cohen, B., e Selman, B. Local search strategies for satisfiability testing. In *DIMACS Series in Discrete Nathematics and Theoretical Computer Science* (Julho 1996), vol. 26, pp. 521–532.

[56] Kautz, H., McAllester, D., e Selman, B. Encoding Plans in Propositional Logic. *International Conference on Principles of Knowledge Representation and reasoning - Morgan Kaufmann Publishers* (1996).

[57] Kautz, H., e Selman, B. Planning as satisfiability. *in Proceedings of ECAI92* (Agosto 1992).

[58] Kautz, H., e Selman, B. Computational challenges in propositional reasoning and search. In *Proceedings of 15th Int. Joint Conference on Artificial Intelligence - IJCAI* (1997).

[59] Kautz, H., e Selman, B. Blackbox: A new aproach to the application of theorem proving to problem solving. *American Association for Artificial Intelligence - AAAI* (Junho 1998), 58–60.

[60] Kautz, H., e Selman, B. Unifying SAT-based and graph-based planning. In *16th International Joint Conference on Artificial Intelligence - IJCAI* (1999), M. Kaufmann, Ed., pp. 318–325.

[61] Kautz, H., Selman, B., e McAllester, D. Pushing the envelope: planning, propositional logic and stochastic search. In *Proceedings of 13th National Conference on Artificial Intelligence - IJCAI* (1996), M. Press, Ed., pp. 1194–1201.

[62] Kautz, H., Selman, B., e McAllester, D. Evidence for invariants in local search. In *14th National Conference in AI* (1997), pp. 321–326.

[63] Khardon, R. Learning action strategies for planning domains. *Artificial Intelligence 113* (1990), 125–148. Elsevier.

[64] Koehler, J. Planning under resource constraints. *ECAI* (1998).

[65] Koehler, J., e Hoffmann, J. On reasonable and forced goal orderings and their use in an agenda-driven planning algorithm. *Journal of Artificial Intelligence Research - JAIR 12* (Junho 2000), 338–386.

[66] Koehler, J., e Schuster, K. Elevator control as a planning problem. *Journal of the Association for Computing Machinery - ACM 31*, 3 (Setembro 2000).

[67] Lamport, L. *LaTeX: A Document Preparation System*. Addison-Wesley, 1986.

[68] Levesque, H. What is planning in the presence of sensing. *American Association for Artificial Intelligence - AAAI* (1996).

[69] Li, C., e Anbulagan. Heuristics based on unit propagation for satisfatibility problems. In *15th International Joint Conference on AI* (Agosto 1997).

[70] Liaw, W.-B., e Brown, F. M. *Knowledge Representing Schemes for Planning*. ACM Computing Surveys, 1992.

[71] Liaw, W.-B., Brown, F. M., e Park, S. S. Towards commonsense planning. *ACM Computing Surveys* (1990).

[72] Lifschitz, V. *The Logic of Common Sense*. ACM, 1995.

[73] Lifschitz, V. Missionaries and Cannibals in the Causal Calculator. University of Austin, 1999.

[74] Lin, F. An Ordering on Subgoals for Planning. Hong Kong/China, 1997.

[75] Lin, F., e Shoham, Y. Provably Correct Theories of Action. *Journal of the Association for Computing Machinery - ACM 42* (Março 1995), 293–320.

[76] Major, D. C., e Lenton, R. L. *Applied water resource systems planning*. Prentice-Hall, 2001.

[77] McAllester, D., e Rosenblitt, D. Systematic Nonlinear Planning. *American Association for Artificial Intelligence - AAAI* (1991).

[78] McCain, N., Remolina, E., Tacchella, A., e Lifschitz, V. Getting to the airport: the oldest planning problem in AI. University of Texas, USA, 1999.

[79] McCarthy, J. Situations, actions and causal laws. Tech. rep., Stanford University, 1963.

[80] McCarthy, J., e Hayes, P. *Some Philosophical Problems from the Standpoint of Artificial Intelligence*. Edinburgh University Press, 1969.

[81] McDermott, D. The Current State of AI Planning Research. Yale University/USA, 1995.

[82] McDermott, D. PDDL - the planning domain definition language. Yale University/USA, 1998.

[83] McDermott, D. Using Regression-Match Graphs to Control Search in Planning. Yale University/USA, 1999.

[84] McDermott, D. The 1998 AI Planning Systems Competition. Yale University/USA, apr 2000.

[85] McIlraith, S. A. A Closed-Form Solution to the Ramification Problem (Sometimes). não publicado, Outubro 1999.

[86] Minton, S. *Learning effective search control knowledge: an explanation approach*. Kluwer Academic Publishers, 1988.

[87] Minton, S. *Machine Learning Methods for Planning*. Morgan Kaufmann Publishers, 1992.

[88] Minton, S., Bresina, J., e Drummond, M. *Total-Order and Partial-Order Planning : A Comparative Analysis*. Morgan Kaufmann Publishers, 1994.

[89] Minton, S., knoblock, C., Kuokka, D., Gil, Y., Joseph, R., e Carbonell, J. Prodigy 2.0: the manual and tutorial. Tech. Rep. CMU-CS-89-146, University of Carnegie Mellon, 1989.

[90] Nau, D. S., Cao, Y., Lotem, A., e Muñoz-Avila, H. SHOP: Simple Hierarchical Ordered Planner. Tech. Rep. CS–TR–3981, UMIACS-TR-9904, University of Maryland (IJCAI–99), MD/USA, Abril. 1999.

[91] Newell, A., e Simon, H. *Human Problem Solving*. Prentice-Hall, 1972.

[92] Nilsson, N. J. *Principles of Artificial Intelligence*. Tioga, 1980.

[93] Nilsson, N. J. *Artificial Intelligence: Synthesis*. Oxford Press, 1995.

[94] Pearl, J. *Heuristics*. Morgan Kaufmann, 1983.

[95] Pednault, E. P. D. Adl: Exploring the middle ground between strips and the situation calculus. In *KR'89: Proc. of the First International Conference on Principles of Knowledge Representation and Reasoning*, R. J. Brachman, H. J. Levesque, and R. Reiter, Eds. Morgan Kaufmann, San Mateo, CA, 1989, pp. 324–332.

[96] Penberthy, J. S., e Weld, D. S. Ucpop: A sound, complete, partial order planner for ADL. *In Proceedings, Conf. on Knowledge representation and reasoning 3* (1992).

[97] Refanidis, I., e Vlahavas, I. GRT: a domain independent heuristic for STRIPS world based on greedy regression tables. In *5th European Conference on Planning* (Durham/UK, 1999), pp. 346–358.

[98] Refanidis, I., e Vlahavas, I. SSPOP: a state space partial-order planner. Aristotle University of Thessaloniki- Greece, 1999.

[99] Refanidis, I., e Vlahavas, I. Exploiting state constraints in heuristic state-space planning. In *5th International Conference on Artificial Intelligence Planning Systems - AIPS* (AAAI Press, Menlo Park, 2000).

[100] Refanidis, I., e Vlahavas, I. Heuristic planning with resources. Aristotle University of Thessaloniki- Greece, 2000.

[101] Refanidis, I., e Vlahavas, I. Multiobjective Heuristic State-Space Planning. Tech. rep., Aristotle University of Thessaloniki- Greece, 2001.

[102] Reiter, R. *The Frame Problem in the Situation Calculus (Sometimes) and a Completness Result for Goal Regression*. University of Toronto, Canada, 1991.

[103] Reiter, R. *Knowledge In Action - Logical Foundations for Describing and Implementing Dynamical Systems*. Book Draft. University of Toronto, Canada, 2000.

[104] Reiter, R., Finzi, A., e Pirri, F. Open World Planning in the Situation Calculus. *American Association for Artificial Intelligence - AAAI* (2000). http://www.aaai.org.

[105] R.M.Simpson, e T.L.McCluskey. An object-graph planning algorithm. University of Huddersfield, UK, Setembro 1999.

[106] Schubert, L. Monotonic Solution of the frame problem in the situation calculus: an efficient method for world with fully specified actions. In *Knowledge Representation and Defeasible Reasoning* (1989), H. Kyburg, R. Loui, and G. Carlson, Eds.

[107] Selman, B., Levesque, H., e Mitchel, D. A new method for solving hard satisfiability problems. In *American Association for Artificial Intelligence - AAAI* (Julho 1994), vol. 10, pp. 440–446.

[108] Shanahan, M. *Solving the Frame Problem – A Mathematical Investigation of the common Sense Law of Inertia*. The MIT Press, Londres/Inglaterra, 1997.

[109] Shapiro, S., Pagnucco, M., Lespérance, Y., e Levesque, H. Iterated Belief Change in the Situation Calculus. unpublished, 1999.

[110] Shapiro, S. C. *Encyclopedia of Artificial Intelligence, Second Edition*. John Wiley & Sons, Inc, 1992.

[111] Shapiro, S. C. Belief Spaces as Sets of Propositions. *Journal of Experimental and Theoretical Artificial Intelligence 2&3*, 5 (Abril. 1993), 225–235.

[112] Silvestre, R. S. Um tratamento formal para o raciocínio sobre ação em inteligência artificial. Tese de Mestrado, Universidade Federal do Ceará, Ceará/Brasil, 1998.

[113] Simpson, R. M., McCluskey, T. L., Zhao, W., Aylett, R. S., e Doniat, C. An integrated graphical tool to support knowledge engineering in AI Planning. In *European Conference on Planning* (Toledo, Spain, 1991), American Association for Artificial Intelligence - AAAI.

[114] Slaney, J., e Thiébaux, S. Blocks world tamed - ten thousand blocks in under a second. Tech. Rep. TR-ARP-17-95, Australian National University, Outubro 1995.

[115] Slaney, J., e Thiébaux, S. How best to put things on top of other things. Tech. Rep. TR-ARP-6-96, Australian National University, Novembro 1996.

[116] Slaney, J., e Thiébaux, S. Linear time near-optimal planning in the Blocks World. Australian National University, 1996.

[117] Smith, D. E., e Weld, D. S. Conformant Graphplan. *15th National Conference on AI* (1998).

[118] Smith, S. J. J., Nau, D. S., e Throop, T. Computer Bridge: A Big Win for AI Planning. *AI Magazine 19* (Junho 1998), 93–105.

[119] Stelting, S. A., e Maassen, O. *Applied Java Patterns*. Prentice Hall PTR / Sun Microsystems Press, 2002.

[120] Stone, P., Blythe, J., e Veloso, M. The Need for Different Domain-Independent Heuristics. *In Proceedings of The Second International Conference on AI Planning Systems* (Junho 1994).

[121] Sussman, G. *A computer model of skill acquisition*. Elsevier/North-Holland, 1975.

[122] Sutton, R. S. Planning by incremental dynamic programming. In *Proceedings of the Eighth International Workshop on Machine Learning*, Morgan Kaufmann, pp. 353–357.

[123] Tate, A., Drabble, B., e Dalton, J. O-Plan: a knowledge-based planner and its application to logistics. University of Edinburgh, UK, Setembro 1996.

[124] Ternovskaia, E. Automata theory for reasoning about actions. University of Toronto, Canada, 1999.

[125] Tsuneto, R., Erol, K., Hendler, J., e Nau, D. Commitment strategies in hierarchical task network planning. *American Association for Artificial Intelligence - AAAI* (Agosto 1996).

[126] Turner, C. H. *Causal Action Theories and Satisfiability Planning*. Tese de Doutorado, University of Austin, USA, 1998.

[127] Vrakas, D., Refanidis, I., e Vlahavas, I. An operator distribution method for parallel planning. Aristotle University of Thessaloniki- Greece, 2000.

[128] Weld, D. S. An Introduction to Least Commitment Planning. *AI Magazine Summer/Fall* (1994).

[129] Weld, D. S. Recent Advances in AI Planning. Tech. Rep. UW–CSE–98–10–01, University of Washington, Seattle/USA, Outubro 1999.

[130] Wurbs, R. A. *Water management models: a guide to software*. Prentice-Hall, 1995.

Endereços: a maior parte do material utilizado para a pesquisa contida nessa dissertação foi conseguido gratuitamente na rede mundial de computadores (*World Wide Web - www*). Esta constante busca por publicações relacionadas com o planejamento automático gerou uma grande lista de endereços, que é enumerada abaixo. Acreditamos que uma visita a essa lista[3] seja um bom ponto de partida, um estímulo aos que desejam ingressar no estudo de planejamento automático. Boa viagem.

[3]Lista ativa em 2001 - apesar da dinâmica da rede mundial de computadores, os endereços das instituições devem permanecer ativos por longos períodos. Endereços pessoais e de assuntos muito específicos foram omitidos.

American Association for Artificial Intelligence	www.aaai.org
Association for Computing Machinery	www.acm.org
Aristotle University of Thessaloniki	www.csd.auth.gr
Durham University	www.dur.ac.uk
Elsevier	www.elsevier.com
Jet Propulsion Laboratory	www-aig.jpl.nasa.gov
Universidad Simón Bolivar	www.ldc.usb.ve
Universidade Federal do Ceará	www.lia.ufc.br
University of Austin	www.cs.utexas.edu
University of Toronto	www.cs.toronto.edu/cogrobo
University of Washington	www.cs.washington.edu
Yale University	www.cs.yale.edu/homes/dvm/