



**UNIVERSIDADE FEDERAL DE ITAJUBÁ**

**Pró-Reitoria de Pesquisa e Pós-Graduação**

Programa de Pós-Graduação em Engenharia Elétrica

**Agente Inteligente para Planejamento  
de Produção em um Centro de Serviços de  
Corte e Solda Laser**

**Jayme Barg**

Dissertação submetida ao Programa de Pós-graduação em Engenharia Elétrica como requisito parcial à obtenção do título de **Mestre em Ciências em Engenharia Elétrica**

**Orientador: Luiz Edival de Souza, Dr.**

Itajubá – MG  
2005

# Dedicatória

A Maísa e Mauricio

## Agradecimentos

A minha querida esposa Maísa e meu querido filho Mauricio, pela compreensão e estímulo.

Ao amigo Onofre Bueno Filho, pelo companheirismo.

Ao professor Luiz Edival de Souza, por sua orientação.

Aos Professores Othon Guilherme Pinto Bravo, D. Sc. e Mario Aníbal Esteves, M.Sc., pelo apoio e incentivo.

Aos senhores Carsten Bosselmann e Ricardo de Mello Brito, D.Sc. , pela liberação e apoio para realização deste trabalho.

A Jörg Hoffmann pelo envio da rotina de gravação dos planos gerados no Metric-FF em arquivos.

*“A complexidade é a máscara da simplicidade”*  
Autor desconhecido

## Resumo

Este trabalho descreve a definição e implementação de um agente planejador inteligente e teve como principal fator motivador, a aplicação do mesmo a um problema real de planejamento de produção. O domínio escolhido para modelamento e implementação do agente inteligente é um centro de serviços de corte e solda Laser, para produção de peças de aço utilizadas na fabricação de carrocerias de automóveis.

Para representação do domínio de planejamento foi utilizada uma linguagem padrão de definição de domínios (PDDL 2.1), a mesma utilizada nas competições internacionais de planejamento, conhecidas como IPC (International Planning Competition) e executado pelo agente planejador independente do domínio Metric-FF.

Em um microcomputador com sistema operacional LINUX, foi implementada uma interface gráfica utilizando o MATLAB, para geração e visualização dos problemas, bem como a visualização dos planos gerados pelo agente. Ainda na interface gráfica, é possível executar um pós-processamento para visualização dos planos de produção gerados, agrupados por produtos ou por equipamentos.

## **Abstract**

This work describes the definition and implementation of an intelligent agent planner and had as the main motivator factor, the application in a real world problem of production planning. The domain for modeling and implementation of the intelligent agent is a laser welding and steel service center for production of car body parts applied in the automotive industry.

For the planning domain representation, a standard language for planning domain definition (PDDL 2.1) was used. This standard language is the same used during the International Planning Competition (IPC) and is executed by the independent domain planner Metric-FF.

In a microcomputer, based on LINUX operational system, a graphical interface developed in MATLAB was implemented to make possible the creation and visualization of the problems and generated plans, as well. From the graphical interface is possible to perform a pos-processing for visualization of the plans grouped by products or equipment.

# Índice

Dedicatória .....	i
Agradecimentos .....	ii
Resumo .....	iii
Abstract .....	iv
Índice .....	v
Lista de figuras .....	viii
Lista de tabelas e gráficos .....	xi
<b>1 Introdução .....</b>	<b>1</b>
1.1 Planejamento com Inteligência Artificial .....	4
<b>2 Identificação do problema .....</b>	<b>5</b>
2.1 Descrição do Domínio .....	6
2.2 Descrição do Problema .....	12
<b>3 Embasamento teórico .....</b>	<b>16</b>
3.1 O planejador STRIPS .....	16
3.2 A Linguagem de Definição de Domínios de Planejamento – PDDL .....	19
3.3 Definição Formal da Sintaxe .....	21
3.3.1 Forma de Backus-Naur .....	21
3.3.2 Forma de Backus-Naur Estendida .....	22
3.4 Sintaxe de definição de domínios .....	24
3.4.1 Sintaxe de definição de operadores .....	25

3.5	Sintaxe de definição de problemas .....	27
3.6	Metric-FF .....	28
3.6.1	Variáveis numéricas .....	29
3.6.2	Método de busca .....	30
3.6.3	Instalação do Metric-FF .....	31
<b>4</b>	<b>Proposta de solução do problema .....</b>	<b>32</b>
4.1	Visão geral sobre agentes inteligentes .....	32
4.1.1	Características do ambiente de atuação do agente .....	34
4.2	O agente planejador de produção .....	35
4.3	Módulo de definição dos estoques ideais .....	36
4.4	Módulo de definição dos fluxos de produção .....	37
4.5	Módulo de modelagem do estado inicial .....	39
4.6	Módulo de modelagem do estado objetivo .....	40
4.7	Módulo de definição dos operadores .....	41
4.7.1	Operador CORTAR .....	41
4.7.2	Operador SOLDAR .....	42
4.7.3	Operador VIRAR .....	44
4.7.4	Operador RESSALTAR .....	45
4.7.5	Operador APARAR_PRECISÃO e APARAR_SIMPLES .....	46
4.7.6	Operador LAVAR .....	47
4.7.7	Operador PRODUZIR .....	49
4.8	Exemplo de busca no domínio .....	53
4.9	Considerações gerais sobre o agente implementado .....	57
<b>5</b>	<b>Interface Gráfica .....</b>	<b>59</b>
5.1	Arquivos de entrada .....	60
5.2	Descrição das telas .....	63
<b>6</b>	<b>Resultados e conclusões .....</b>	<b>72</b>
6.1	Resultados obtidos .....	73
6.2	Perspectivas futuras .....	75
6.2.1	Restringir as pré-condições dos operadores .....	75

6.2.2 Interface gráfica mais eficiente .....	76
6.2.3 Instanciações numéricas pelo agente planejador .....	76
<b>Anexo A – Exemplo de definição de problema .....</b>	<b>78</b>
<b>Anexo B – Domínio de planejamento .....</b>	<b>79</b>
<b>Anexo C – Alteração no código fonte do Metric-FF .....</b>	<b>81</b>
<b>Referências .....</b>	<b>83</b>



## Lista de Figuras

Figura 2-1 – Linha de Corte Longitudinal, onde $M = F1 + F2 + 2.A$ .....	6
Figura 2-2 – Linha de corte transversal com empilhamento dos <i>blanks</i> em fardos	7
Figura 2-3 – Exemplo de <i>blank</i> reto e configurado .....	7
Figura 2-4 – Blank posicionados sob o feixe Laser para serem soldados .....	8
Figura 2-5 – Blank soldado. União entre dois <i>blanks</i> de espessuras E1 e E2 .....	8
Figura 2-6 – Pilha de <i>blanks</i> soldados sem ressaltos no lado de menor espessura	10
Figura 2-7 – Pilha de <i>blanks</i> soldados com ressaltos no lado de menor espessura	11
Figura 2-8 – (a) <i>Blank</i> produzido diretamente pela linha de Prensa .....	11
Figura 2-9 – (b) <i>blank</i> simétrico ao primeiro, virado no Virador de Fardos.....	11
Figura 2-10 – Fluxos de produção .....	15
Figura 3-1 – Descrição de um problema do mundo dos blocos representado em STRIPS .....	18

Figura 3-2 – Sintaxe de definição de domínios .....	24
Figura 3-3 - Sintaxe de definição de operadores .....	26
Figura 3-4 – Exemplo de representação de domínio e operador em PDDL 2.1 ..	27
Figura 3-5 – Sintaxe de definição de problemas .....	28
Figura 3-6 – Exemplo de representação de um problema em PDDL 2.1 .....	28
Figura 4-1 – Arquitetura do agente planejador de produção .....	35
Figura 4-2 – Exemplo de um fluxo de produção (F10) .....	38
Figura 4-3 – Representação gráfica do operador CORTAR .....	42
Figura 4-4 – Representação do operador CORTAR em PDDL .....	42
Figura 4-5 – Representação gráfica do operador SOLDAR .....	43
Figura 4-6 – Representação do operador SOLDAR em PDDL .....	43
Figura 4-7 – Representação gráfica do operador VIRAR .....	44
Figura 4-8 – Representação do operador VIRAR em PDDL .....	45
Figura 4-9 – Representação gráfica do operador RESSALTAR .....	45
Figura 4-10 – Representação do operador RESSALTAR em PDDL .....	46
Figura 4-11 – Representação gráfica dos operadores APARAR_PRECISÃO e APARAR_SIMPLES .....	47
Figura 4-12 – Representação dos operadores APARAR_PRECISÃO e APARAR_SIMPLES em PDDL .....	47
Figura 4-13 – Representação gráfica do operador LAVAR .....	48
Figura 4-14 – Representação do operador LAVAR em PDDL .....	49
Figura 4-15 – Plano de produção para o fluxo F10 .....	50
Figura 4-16 – Representação gráfica do operador PRODUZIR .....	52
Figura 4-17 – Representação do operador PRODUZIR em PDDL .....	52
Figura 4-18 – Exemplo de busca no domínio .....	56
Figura 4-19 – Arquitetura de pós-processamento .....	57
Figura 5-1 – Estrutura de pastas para a execução da interface gráfica .....	59
Figura 5-2 – Tela principal da interface gráfica .....	63
Figura 5-3 – Tela para entrada da quantidade de problemas a ser submetido ao agente planejador de produção .....	64
Figura 5-4 – Tela para visualização dos arquivos gerados (problemas) .....	65
Figura 5-5 – Exemplo de arquivo de fatos gerado .....	66
Figura 5-6 – Tela para seleção dos planos gerados .....	67

Figura 5-7 – Exemplo de plano de produção gerado .....	67
Figura 5-8 – Tela de pós-processamento .....	68
Figura 5-9 – Tela para seleção dos equipamentos .....	69
Figura 5-10 – Exemplo de plano de produção por equipamento .....	69
Figura 5-11 – Exemplo de plano de produção por produto .....	70
Figura 5-12 – Exemplo de um plano macro .....	71

## Lista de Tabelas e Gráficos

Tabela 4-1– Exemplo de definição de estoque ideal .....	37
Tabela 4-2– Exemplo de arquivo de entrada com estoques reais .....	39
Tabela 4-3 – Últimos equipamentos de processo para cada fluxo de produção .	51
Tabela 4-4 – Exemplos de valores de estoques reais e ideais .....	53
Tabela 4-5 – Exemplo de plano de produção gerado pelo agente planejador ....	55
Tabela 5-1 – Formato do arquivo de estoques reais por produto .....	61
Tabela 5-2 – Formato do arquivo de demandas previstas por produto .....	61
Tabela 6-1 – Nível de coincidência entre o agente planejador humano e o agente planejador de produção .....	74
Gráfico 6-1 – Nível de coincidência entre o agente planejador humano e o agente planejador de produção .....	74

# Capítulo 1

## 1. Introdução

O planejamento e o controle da produção estão entre os principais fatores que influenciam a produtividade industrial. As empresas devem se adaptar as condições de mercado, que mudam constantemente, afetando o tempo disponível para tomada de decisões [1].

Poucas áreas dentro da administração empresarial mudaram tanto como a administração da produção, nos últimos tempos. A produção, durante anos, foi considerada quase um mal necessário suportado por outros setores porque, afinal, uma empresa de manufatura não podia escapar de fazer seus produtos [2].

Entretanto nos últimos anos, é notório que se estabeleceu um novo paradigma revalorizando o papel da manufatura frente aos objetivos estratégicos da organização.

Esta revalorização da produção se deve a três razões básicas:

- a crescente pressão por competitividade que o mercado mundial tem demandado das empresas,
- o potencial competitivo que representa o recente desenvolvimento de novas tecnologias de processo e de gestão de manufatura,
- o desenvolvimento de um melhor entendimento do papel estratégico que a produção pode e deve ter na busca dos objetivos globais da organização.

O problema de planejamento da produção em um processo de manufatura vem sendo amplamente estudado e representa um desafio quanto à tomada de decisão para determinar o que produzir e quando produzir. Esta decisão deve satisfazer as restrições técnicas dos equipamentos envolvidos na cadeia produtiva, além de buscar uma seqüência ótima de produção visando uma minimização dos custos operacionais e de estoque. Em outras palavras, pode-se dividir o problema de planejamento de produção pelas atividades a serem desempenhadas até a obtenção do produto acabado e pelo seqüenciamento dessas atividades. Em muitas fábricas modernas, a atividade de planejamento é executada manualmente, através de um agente planejador humano e o seqüenciamento é feito de modo automático, com o uso de alguma ferramenta computacional.

Como exemplo de um sistema real de planejamento, podemos citar o TOSCA [3], utilizado pela empresa Hitachi para processos que tipicamente envolvem 350 produtos, com 35 diferentes máquinas e mais de 2000 operações. O plano gerado contempla 30 dias de produção, considerando três turnos de oito horas por dia. Os planos gerados pelo sistema TOSCA são capazes de determinar quais ações devem ser executadas para a produção de um determinado produto, indicando classes de equipamentos a serem utilizados, porém sem determinar exatamente qual deles devem ou podem ser utilizado.

Processos com uma menor diversidade de produtos, normalmente podem seguir um plano de produção fixo, mas ainda assim requerem um seqüenciamento automático. O sistema ISIS [4] foi desenvolvido especificamente para seqüenciamento de produção e foi utilizado pela primeira vez em uma fábrica de turbinas da empresa Westinghouse, onde eram produzidos diferentes tipos de lâminas. Para cada modelo de lâmina podia haver um ou mais planos, chamados de rotas. Quando uma ordem de serviço era gerada, um dos planos de produção era escolhido e seqüenciado de acordo com a necessidade e criticidade de produção, como por exemplo: produção de lâminas para formação de estoque ou para substituição de uma lâmina quebrada em uma turbina em serviço.

Métodos tradicionais de seqüenciamento de produção, como PERT [5], são capazes de gerar uma seqüência ordenada de passos que satisfaçam a um conjunto de restrições, porém demandam em torno de 80% a 90% do tempo dos planejadores humanos (usuários) para que todas as restrições sejam determinadas.

Embora existam dezenas de ferramentas que oferecem suporte para o planejamento de produção, a interferência humana normalmente se faz necessária para a validação final do plano, ou ainda, o planejamento é feito de forma puramente manual, acarretando em erros que embora possam não ser grosseiros, normalmente estão muito aquém do que se pode considerar como solução ótima.

Normalmente quando a diversidade de produtos envolvidos é pequena, a solução de planejamento de produção se torna mais fácil ou mesmo trivial para um planejador humano, pois todas as restrições envolvidas ficam evidenciadas e são contempladas e atendidas facilmente. Com o aumento da quantidade de diferentes produtos, a complexidade da solução também aumenta, o que leva ao fracionamento do problema em partes menores, e assim a visão global da cadeia produtiva é perdida.

O objetivo deste trabalho é de aplicar um planejador IA independente do domínio para solucionar um problema real de planejamento de produção, onde a partir da formulação do problema e da formulação dos objetivos, o agente planejador gere como saída uma seqüência de ações que represente o plano de produção global.

Como contribuição da implementação do agente planejador de produção, tem-se a redução do risco de erros de planejamento, em função da grande quantidade de variáveis envolvidas.

As ações (operadores) de planejamento relevantes para o domínio são descritas no formato STRIPS, com a definição dos operadores e variáveis, bem como as condições que devem ser satisfeitas para que a ação seja executada e também os efeitos gerados a partir da execução desta.

O domínio de planejamento escolhido foi um centro de serviços de corte e solda laser, que tem como principal função a produção de chapas de aço para atender a indústria automobilística.

As seções que se seguem apresentam a descrição detalhada do processo em questão, a definição do domínio e o planejador utilizado para a solução do problema.

## 1.1 Planejamento com Inteligência Artificial

Os primeiros sistemas desenvolvidos na área de Inteligência Artificial (IA) atendiam a uma classe de problemas que se caracterizavam, entre outros aspectos, por uma pequena interação com o ambiente onde eles eram utilizados e por não abordarem o tempo como um aspecto inerente aos problemas estudados.

Atualmente, existe uma área de pesquisa em IA que se preocupa em desenvolver sistemas que possuam, entre outras características, a capacidade de desenvolver suas atividades com alto grau de autonomia, imersos em ambientes reais, com os quais eles podem interagir fortemente, e executando atividades nas quais o aspecto tempo é importante, ou seja, podem existir restrições de tempo dentro das quais o sistema deve desenvolver suas atividades.

O desenvolvimento deste tipo de sistema, sobretudo no estudo de sistemas que possuem agentes que realizam ações num ambiente dinâmico, agregou, em torno de certos objetivos e propostas, um conjunto de pesquisadores que se dizem pesquisadores da sub-área de IA chamada “Artificial Intelligent Planning” (AIP) [6]. Esta sub-área tem crescido substancialmente nos principais encontros da área de IA. Os primeiros sistemas desenvolvidos, a partir do início da década de 70, são chamados por vários autores como “sistemas clássicos de planejamento”, embora não exista nenhum critério bem estabelecido que diferencie sistemas clássicos dos demais. Esta separação tem apenas um caráter histórico e procura colocar nesta classificação a maioria dos sistemas desenvolvidos durante a década de 70 e início da década de 80 .

Na era dos sistemas clássicos, o planejamento inteligente de atividades era entendido como sendo a determinação de uma seqüência de ações que leva o sistema do estado inicial ao estado final desejado (*goal state*).

Logo no início da década de 80, esta definição já não atendia mais às ambições dos sistemas que estavam sendo propostos na comunidade de *AI Planning* e, particularmente depois da metade da década, começaram a surgir novas definições , como por exemplo, sistemas capazes de lidar com replanejamento [7].



## Capítulo 2

### 2. Identificação do problema

O processo de fabricação de carrocerias de automóveis é dividido em uma série de etapas, como por exemplo: corte, estampagem, armação, grafagem, montagem, pintura, etc. Uma forte tendência, se não mesmo uma realidade, é a segmentação do processo de fabricação, onde a montadora exerce a função única de montar o automóvel, deixando as atividades de fabricação das partes integrantes sob responsabilidade de empresas especializadas em cada setor. No caso específico de carrocerias, a divisão é feita entre empresas especializadas em corte de chapas, empresas especializadas em estampagem (estamparias) e a própria montadora.

Como fator motivador, o contexto escolhido para ser especificado como o domínio do problema de planejamento é um centro de serviços de corte e solda Laser de chapas de aço galvanizadas, em função da grande quantidade de diferentes modelos de chapas a serem produzidas.

Estas chapas são enviadas a uma estamparia e após estampadas, a uma montadora, que então irá compor a carroceria do automóvel.

## 2.1 Descrição do Domínio

A primeira atividade desenvolvida em um centro de serviço de corte é a divisão longitudinal e aparada das bordas de uma tira de aço, enrolada em forma de bobina. O objetivo desta divisão é adequar a largura da bobina original (bobina mãe) com a largura desejada das peças a serem produzidas. O equipamento responsável pela divisão longitudinal e aparada das bordas das bobinas é uma linha de corte longitudinal ou *slitting line*. Com a divisão longitudinal da bobina mãe, novas bobinas são geradas (bobinas filhas). Note que a soma das larguras das bobinas filhas ( $F_1, F_2, \dots, F_n$ ), mais a largura das bordas aparadas ( $A$ ) será igual à largura da bobina mãe ( $M$ ), conforme indicado na Figura 2-1.

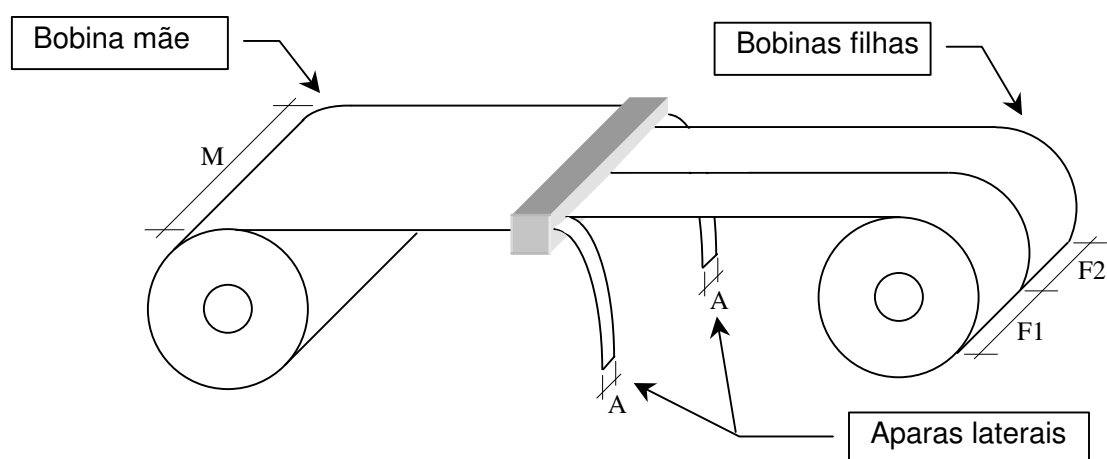


Figura 2-11 – Linha de Corte Longitudinal, onde  $M = F_1 + F_2 + 2A$

Uma vez que as bobinas filhas já se encontram produzidas com a largura específica de cada peça, estas são então desenroladas e cortadas em uma linha de prensa ou linha de corte transversal ou *blanking line*, obtendo-se assim as peças com largura e comprimento corretos. Obviamente a definição de largura e comprimento é feita previamente pelas empresas montadoras, e dependem do modelo final da peça, modelo do automóvel, etc. As peças geradas na saída da linha de corte transversal recebem o nome de *blanks*. Os *blanks* produzidos são

empilhados em *palletts* e embalados em fardos, estando assim prontos para serem enviados as empresas especializadas em estampagem. A quantidade de peças por pallett, limites máximos e mínimos de altura da pilha de *blanks* e peso total, são previamente definidos pelas empresas de estampagem. A Figura 2-2 ilustra o processo de corte de bobinas filhas em *blanks*.

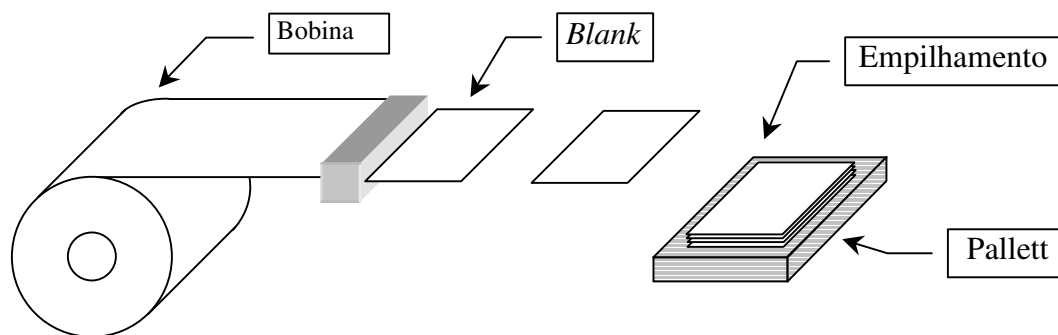


Figura 2-12 – Linha de corte transversal, com empilhamento dos *blanks* em fardos

Os *blanks* gerados podem ser classificados, basicamente, em dois tipos: retos, quando tem a forma de retângulos, trapézios ou paralelogramos (Figura 2-3 a), ou configurados, quando são gerados a partir de uma matriz ou ferramenta de corte e possuem as mais diversas formas (Figura 2-3 b).

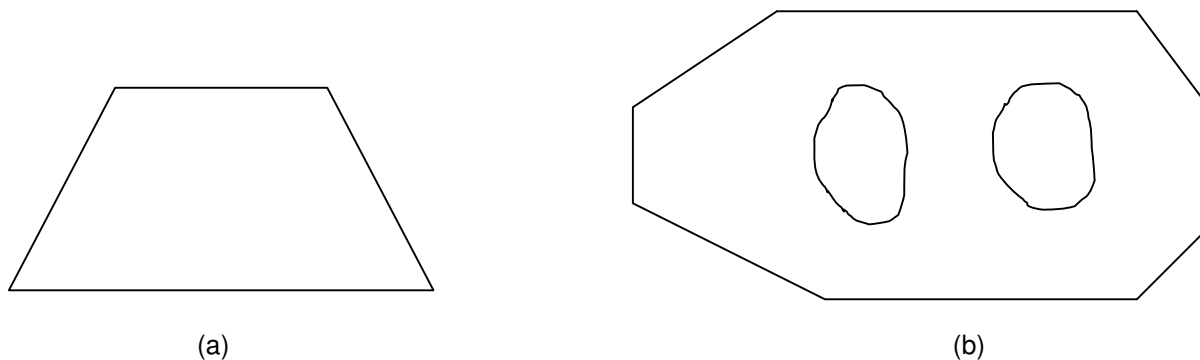


Figura 2-13 – Exemplo de *blank* reto e configurado

Um terceiro tipo de *blank* é o *blank* soldado. O *blank* soldado é produzido a partir da união de dois ou mais *blanks*, configurados ou retos, com espessuras diferentes.

A montagem de carrocerias de automóveis, utilizando-se *blanks* soldados, tem como objetivo um aumento da segurança dos passageiros em caso de colisão, menor peso da carroceria, (aumentando assim o rendimento do automóvel) e menor custo de produção.

O equipamento responsável pela solda dos *blanks* é uma linha de solda Laser ou *Laser welding machine*. Os *blanks* são posicionados lado a lado e transportados através de um feixe de Laser que une as peças por calor. Variáveis de processo como por exemplo, velocidade de transporte dos *blanks* sob o foco, ângulo de inclinação do feixe Laser, espessura dos *blanks*, refrigeração do cordão de solda, etc., exercem influência direta sobre a qualidade da solda. A Figura 2-4 mostra a seção transversal dos *blanks* de diferentes espessuras posicionados sob o feixe de Laser e a Figura 2-5 o aspecto da união entre os *blanks*, após a solda ter sido executada.

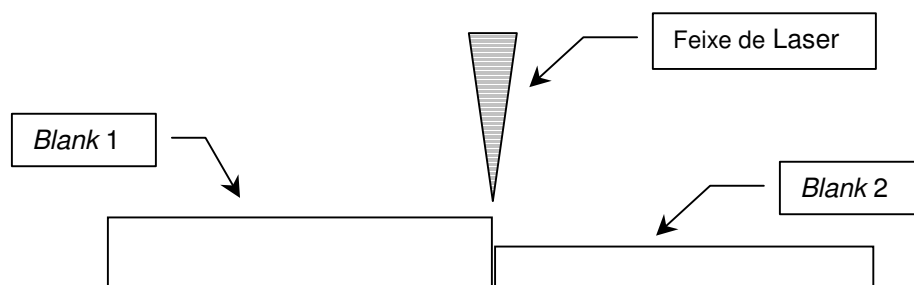


Figura 2-14 – Blank posicionados sob o feixe Laser para serem soldados

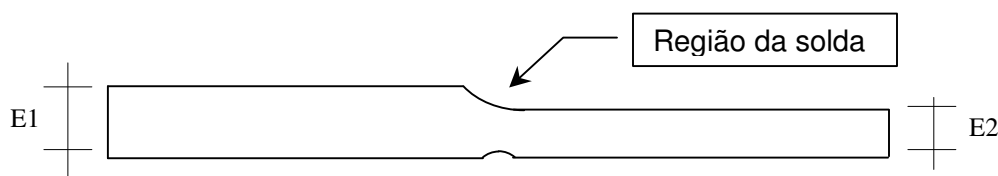


Figura 2-15 – Blank soldado. União entre dois *blanks* de espessuras E1 e E2

Além dos equipamentos básicos descritos acima, existem outros considerados como equipamentos auxiliares, porém também necessários para a produção em um centro de serviço de corte e solda Laser. São eles: guilhotinas convencionais e de precisão, lavadora de *blanks*, prensa de ressaltos e virador de fardos. A descrição funcional de cada equipamento auxiliar é feita a seguir.

- Guilhotina convencional – Guilhotinas convencionais são utilizadas para divisão de *blanks* retos, permitindo um aumento de produtividade na linha de prensa. Como exemplo, podemos citar um *blank* cujo comprimento especificado pela montadora é  $L$ . O *blank* é então produzido na linha de prensa com comprimento  $2L$ , e dividido ao meio em uma guilhotina convencional, reduzindo assim, o tempo de produção das peças na linha de prensa pela metade. É evidente que o tempo de processamento na guilhotina deve ser contabilizado no tempo total de produção do *blank*, porém a intenção é disponibilizar a linha de prensa para produção de outras peças. A decisão de se utilizar este recurso deve ser avaliada cuidadosamente antes da produção, considerando custos de mão-de-obra, ociosidade da linha de prensa e disponibilidade da guilhotina.
- Guilhotina de precisão – A tolerância referente à linearidade do corte da borda do *blank* que é submetido ao processo de solda Laser é da ordem de  $0,05$  mm. Este nível de linearidade é atingido pela Linha de Prensa somente em *blanks* cujo comprimento máximo da borda a ser soldada é de aproximadamente  $400$  mm. Para comprimentos superiores, o *blank* deve ser aparado em uma guilhotina de precisão para que a linearidade do corte seja garantida, permitindo assim a solda Laser.
- Lavadora de *blanks* – Para garantir um perfeito acabamento superficial em peças que compõem a parte externa da carroceria dos automóveis, os *blanks* devem estar totalmente livres de impurezas, pois caso contrário, essas impurezas acabam causando pequenas deformações superficiais durante o processo de estampagem. Estas deformações são percebidas mesmo após a

pintura das peças e são suficientes para que as mesmas sejam reprovadas pelo controle de qualidade.

Com o objetivo de eliminar totalmente as impurezas da superfície dos *blanks*, esses passam por um processo de lavagem e revestimento com uma camada de óleo para proteção contra oxidação. O equipamento que executa esse procedimento é a Lavadora de Blanks.

- Prensa de ressaltos – Como visto anteriormente, *blanks* soldados são compostos de dois ou mais *blanks* de diferentes espessuras. Quando este tipo de *blank* é empilhado, a altura final da pilha é maior no lado dos *blanks* de maior espessura. Obviamente no lado dos *blanks* de menor espessura a altura final da pilha é menor.

No processo de estampagem, o desempilhamento dos *blanks* soldados é realizado através de braços robotizados com ventosas, porém como existe um desnível entre as extremidades da pilha, as ventosas tocam o lado onde a altura é maior (lado dos *blanks* de maior espessura) e a sucção realizada pelas ventosas na direção cuja altura da pilha é menor (lado das *blanks* de menor espessura) não é possível, conforme mostra a Figura 2-6.

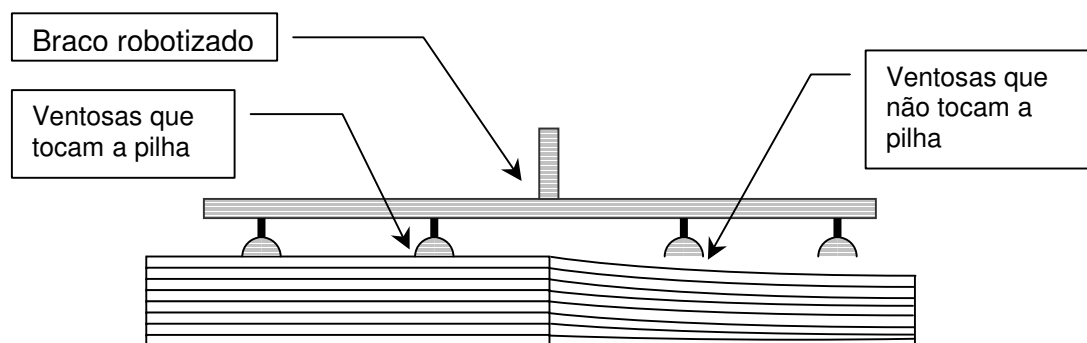


Figura 2-16 – Pilha de *blanks* soldados sem ressaltos no lado de menor espessura

Para solucionar este problema, são feitos pequenos ressaltos no lado do *blank* de menor espessura, de forma a igualar as alturas nas duas extremidades da pilha de *blanks* soldados e permitir o desempilhamento robotizado, conforme mostra a Figura 2-7.

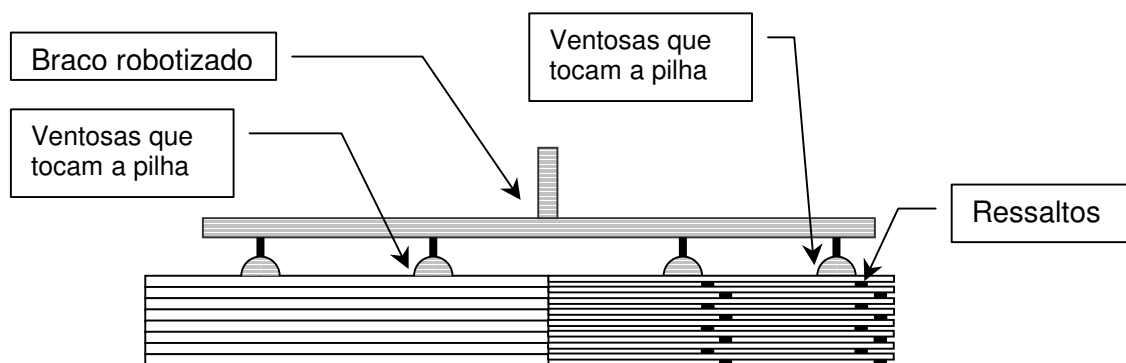


Figura 2-17 – Pilha de *blanks* soldados com ressaltos no lado de menor espessura

- Virador de fardos – Muitas peças que compõem as carrocerias de automóveis, embora sejam geometricamente idênticas, são fornecidas às estamparias em pares compostos por um *blank* que será utilizada um no lado esquerdo e um outro no lado direito da mesma. Exemplos típicos de peças fornecidas em pares são as laterais do automóvel, portas e longarinas.

Na prática, não existe diferença durante a produção dos *blanks* esquerdo e direito na Linha de Prensa. Somente depois de produzidos, metade da produção é virada no Virador de Fardos, ficando assim simétrica ao *blank* original e compondo então um par. A Figura 2-8 mostra um exemplo de um par de *blanks*, onde um é simétrico ao outro.

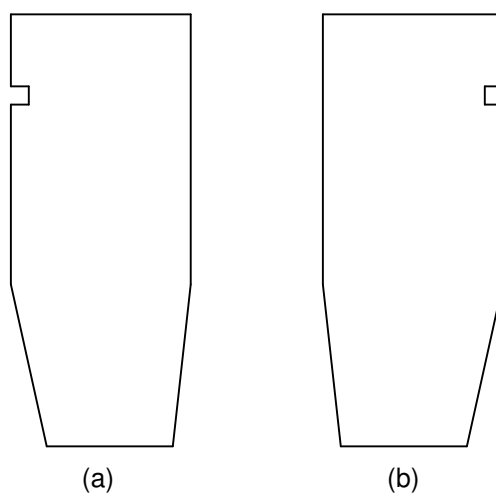


Figura 2-18 – (a) *Blank* produzido diretamente pela linha de Prensa e (b) *blank* simétrico ao primeiro, virado no Virador de Fardos.

## 2.2 Descrição do Problema

O planejamento da produção em centros de serviço de corte e solda Laser é feito de forma a se produzir para estoque, ou seja, existe uma previsão de demanda que é apresentada pelos clientes, onde são listadas as quantidades de consumo de cada *blank*, bem como a data de utilização dos mesmos. Uma vez conhecida a previsão de demanda de cada *blank*, estes são produzidos e enviados a um estoque de onde os mesmos são posteriormente retirados e entregues conforme a solicitação dos clientes e seguindo a previsão de demanda apresentada.

À medida que o estoque é consumido, novos *blanks* são produzidos para repor os níveis mínimos do estoque. Adotando-se esta estratégia de produção, é possível garantir uma entrega *just-in-time* aos clientes, minimizando os custos de estocagem dos produtos nas estamparias. É evidente que estes custos estão presentes nos centros de serviço.

A complexidade do planejamento da produção é diretamente proporcional a quantidade de diferentes produtos. Embora a solução do plano de produção seja trivial no caso de algumas poucas peças, normalmente é necessário acompanhar o nível de estoque para algumas dezenas de peças. Como a quantidade de peças que devem permanecer em estoque é dinâmica, pois a demanda varia ao longo do tempo, a complexidade de determinar o que deve ser produzido se torna ainda maior.

A proposta de solução apresentada no capítulo 4 é capaz de a partir de uma entrada que representa o estoque real dos mais diversos modelos de *blanks*, gerar uma lista de ações ordenadas que representa a seqüência de produção necessária para repor os níveis de estoque dentro de faixas consideradas como ideais. Os níveis ideais de estoque são previamente estabelecidos a partir da previsão de demanda e devem ser recalculados a cada alteração na previsão.

A lista de ações geradas, além de considerar a quantidade de *blanks* a serem produzidos, leva em conta também os passos necessários para a produção de cada item, evitando assim que seqüências impossíveis de serem realizadas sejam geradas.



Em função dos diferentes tipos e/ou modelos de produtos dentro de um centro de serviços de corte e solda Laser e também dos diversos equipamentos envolvidos no processo descrito na Seção 2.1, foram identificados e analisados 11 fluxos de produção possíveis. São eles:

- Fluxo de produção 1 (F1) – A bobina é processada na linha de corte longitudinal e entregue ao cliente ainda em forma de bobina, apenas com as bordas aparadas.
- Fluxo de produção 2 (F2) – Uma bobina que já se encontra com a largura correta para a produção do *blank* ao qual se destina, é processada na linha de corte transversal. Os *blanks* produzidos estão prontos para serem enviados aos clientes.
- Fluxo de produção 3 (F3) – A bobina é processada (aparada) na linha de corte longitudinal, os *blanks* são produzidos na linha de corte transversal e então enviados aos clientes.
- Fluxo de produção 4 (F4) – É o fluxo de produção de *blanks* soldados. A bobina é processada (aparada) na linha de corte longitudinal, os *blanks* são produzidos na linha de corte transversal e soldados uns aos outros, na máquina de solda Laser. Os *blanks* soldados são enviados aos clientes.
- Fluxo de produção 5 (F5) – É semelhante ao fluxo de produção F4, porém os *blanks* produzidos na linha de corte transversal ainda são divididos em uma guilhotina de precisão antes de serem processados na máquina de solda Laser.
- Fluxo de produção 6 (F6) – É semelhante ao fluxo de produção F5, porém os *blanks* soldados são processados na prensa de ressaltos antes de serem enviados aos clientes.

- Fluxo de produção 7 (F7) – É semelhante ao fluxo de produção F4, porém os *blanks* produzidos na linha de corte transversal ainda são divididos em uma guilhotina convencional antes de serem processados na máquina de solda Laser.
- Fluxo de produção 8 (F8) – É também um fluxo de produção de *blanks* soldados, onde os *blanks* produzidos na linha de corte transversal são virados no Virador de Fardos antes de serem soldados na máquina de solda Laser.
- Fluxo de produção 9 (F9) – É semelhante ao fluxo de produção F3, porém os *blanks* são virados no virador de fardos antes de serem enviados aos clientes.
- Fluxo de produção 10 (F10) - É semelhante ao fluxo de produção F3, porém os *blanks* são virados e lavados antes de serem enviados aos clientes.
- Fluxo de produção 11 (F11) - É semelhante ao fluxo de produção F3, porém os *blanks* são lavados na lavadora de *blanks* antes de serem enviados aos clientes.

Na Figura 2-9 estão indicados todos os fluxos de produção possíveis. Estes fluxos são os contemplados no centro de serviços de corte e solda Laser em questão, porém nada impede de novos fluxos serem criados em função da necessidade de produção, inclusive com a utilização de novos equipamentos para o atendimento dos requisitos de produção necessários e exigidos pelos clientes.

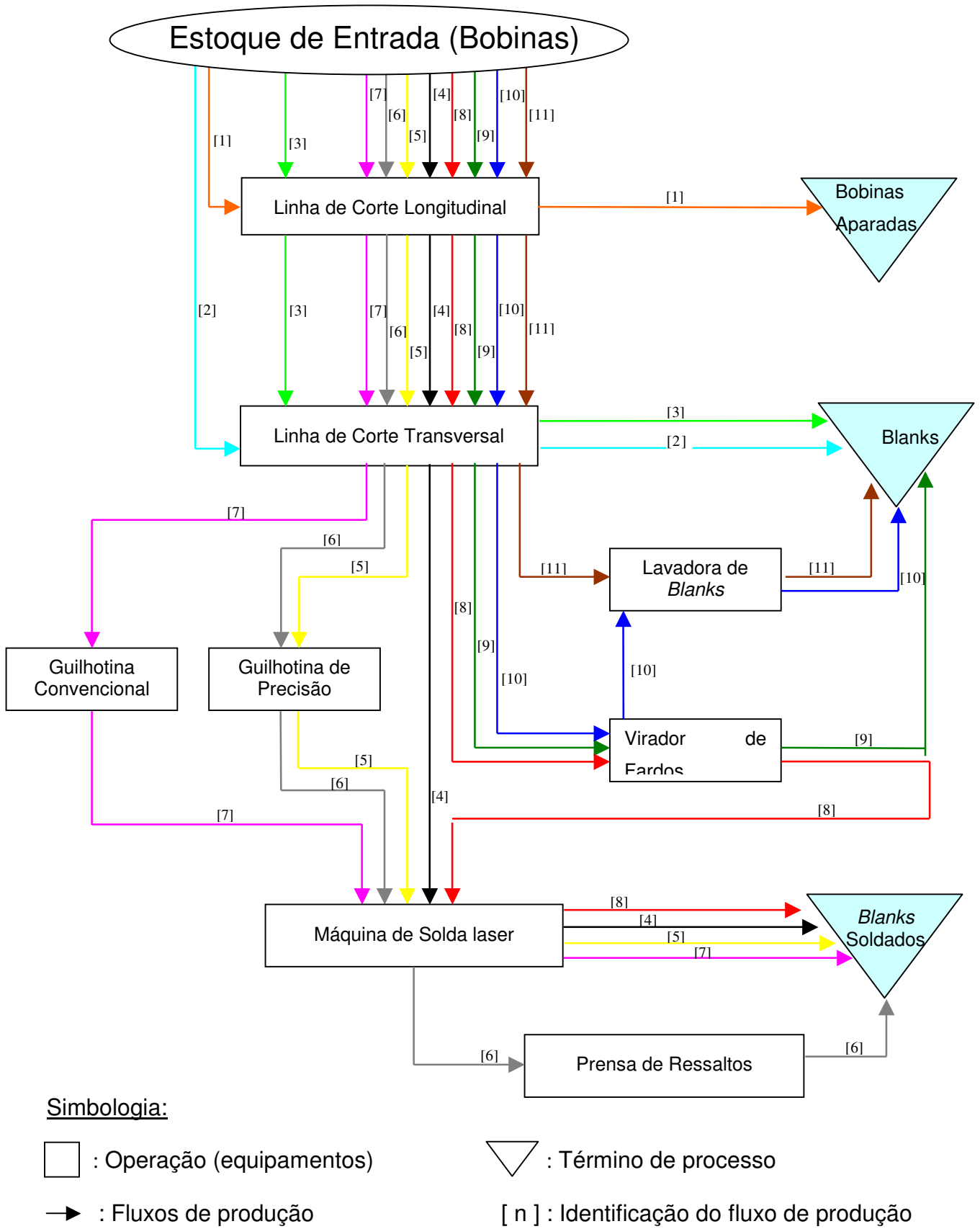


Figura 2-10 – Fluxos de produção

## Capítulo 3

### 3. Embasamento teórico

Neste capítulo é apresentado o planejador STRIPS que foi efetivamente, o primeiro planejador proposto para solução de problemas de planificação que se caracterizava por procurar uma seqüência de ações capaz de transformar o estado inicial do domínio, em um estado no qual o objetivo é satisfeito. Também é apresentada a linguagem PDDL, utilizada para a descrição de domínios de planejamento, bem como a descrição dos problemas no estilo proposto pelo planejador STRIPS. As características do planejador Metric-FF, utilizado para a solução do problema proposto no Capítulo 2, são apresentadas ao final desta seção.

#### 3.1 O planejador STRIPS

Em 1971 foi apresentado um planejador para representação de domínios de planejamento, conhecido como STRIPS (*STanford Research Institute Problem Solver*) [8]. Na representação proposta no planejador STRIPS, os estados inicial e final do mundo (domínio de planejamento) são representados por um conjunto de literais instanciadas, ou seja, predicados aplicados sobre constantes. Por exemplo, o estado inicial que representa a quantidade de 100 peças para um determinado produto\_1 e 80 peças para um produto\_N, pode ser representado como:

$$(\text{est\_ini produto\_1 } 100) \wedge \dots \wedge (\text{est\_ini produto\_N } 80)$$

Em uma descrição completa do estado inicial, todas as fórmulas atômicas que não são explicitamente declaradas, são assumidas como falsas. Esta premissa é conhecida como Hipótese do Mundo Fechado [9] e usada para tratar o Problema do Quadro [10].

Da mesma forma, o estado objetivo ou estado final é representado. Ainda utilizando o exemplo acima, se for necessário que o estoque do produto\_1 seja elevado de 100 para 150 unidades, a representação do estado final será:

$$(\text{est\_final produto\_1 } \mathbf{150}) \wedge \dots \wedge (\text{est\_final produto\_N } 80)$$

Os conceitos de definição de estado inicial e estado final define parcialmente o problema de planejamento, pois ainda resta a definição dos operadores possíveis de serem executadas para que leve o estado do mundo para o estado objetivo. A teoria do domínio é a descrição formal dos operadores disponíveis para o agente planejador. Em STRIPS as ações são baseadas em operadores com pré-condições, efeitos de adição e efeitos de remoção. Os operadores contêm variáveis, que uma vez instanciadas definem as ações. A estrutura dos operadores em representação STRIPS pode ser definida como:

- A *descrição da ação*, um predicado que define o nome da ação e contém variáveis como parâmetros.
- A *pré-condição* (PC), um conjunto de literais que uma vez contidos no estado atual (verdadeiros), justificam a aplicação da ação  $\alpha$ .
- O *efeito de adição* (EF+), um conjunto de literais que são adicionados (tornam-se verdadeiros) após a aplicação da ação  $\alpha$  no estado atual.
- O *efeito de remoção* (EF-), um conjunto de literais que são removidos (tornam-se falsos) após a aplicação da ação  $\alpha$  no estado atual.

Para exemplificar o funcionamento da representação STRIPS, a Figura 3-1 apresenta a descrição parcial da teoria do domínio para o mundo dos blocos [11] [12].

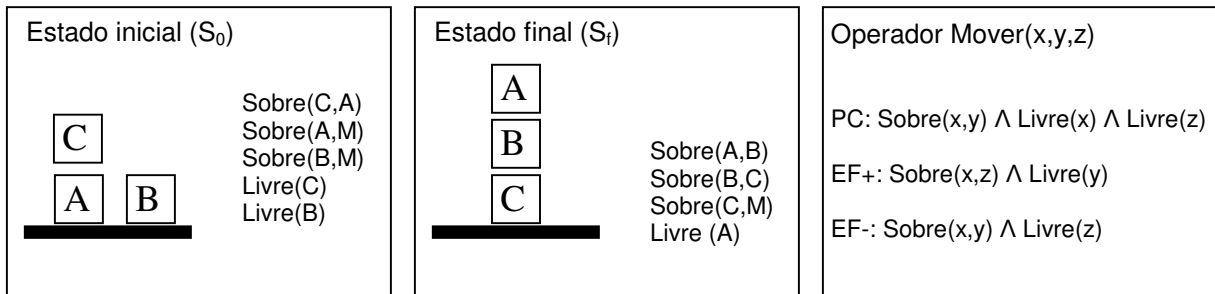


Figura 3-1 – Descrição de um problema do mundo dos blocos representado em STRIPS.

A linguagem de representação utilizada pelo planejador STRIPS utiliza a estratégia de divisão e conquista, ou seja, escolhe um dos literais do estado final e aplica ações ao estado atual até que seja atingido um estado que contenha esse literal. Depois seleciona o próximo literal da conjunção do estado final, e assim sucessivamente, até que alcance um estado que contenha todos os literais do estado objetivo satisfeitos. Esse processo pode ser descrito como:

- Seleciona um operador cujas pré-condições sejam satisfeitas no estado atual. A prioridade é para operadores em que os efeitos de adição contenham literais do estado final, de forma a torná-los verdadeiros.
- Armazena a ação aplicada em uma lista de ações aplicáveis.
- Gera um novo estado  $S_{n+1}$  a partir do estado  $S_n$ , adicionando os literais contidos nos efeitos de adição e removendo os literais contidos nos efeitos de remoção da ação aplicada.
- Se todos os literais contidos no estado final estiverem presentes no estado  $S_{n+1}$ , encerra a busca e retorna a lista de ações aplicadas como o plano. Caso contrário, retorna ao primeiro passo.

No exemplo do mundo dos blocos, existem três literais no estado final: *Sobre(A,B)*, *Sobre(B,C)* e *Sobre(C,M)*. O STRIPS tenta unificar as variáveis  $x$ ,  $y$  e  $z$  de modo que o conjunto de efeitos de adição contenha um ou mais literais do estado

objetivo. Uma instanciação possível, a partir do estado inicial, é  $mover(B,M,C)$ , satisfazendo o segundo literal do estado final e levando o estado do mundo para  $S_1 = \{Sobre(B,C), Sobre(C,A), Sobre(A,M), Livre(B)\}$ . Note que apesar de um dos literais do estado final ter se tornado verdadeiro, esta ação terá que necessariamente ser desfeita, pois nesse ponto apenas a ação  $mover(B,C,M)$  é aplicável, gerando um estado  $S_2 = \{Sobre(B,M), Sobre(C,A), Sobre(A,M), Livre(B), Livre(C)\}$ .

Uma solução possível para este problema é o plano de ações  $mover(B,M,C)$ ,  $mover(B,C,M)$ ,  $mover(C,A,M)$ ,  $mover(B,M,C)$  e  $mover(A,M,B)$ . Observe que esse plano apresenta uma repetição de ações, que pode levar o algoritmo a uma busca infinita de um plano consistente, conhecido como Anomalia de Sussman [13].

A estratégia de busca progressiva utilizada no planejador STRIPS também foi utilizada por muitos outros planejadores que foram desenvolvidos para tratar somente dos chamados domínios tipo STRIPS, ou seja, domínios cujos operadores são descritos segundo o formalismo desta linguagem. Dentre eles pode-se citar: NOAH [14], NONLIN [15], TWEAK [16] entre outros.

Outros planejadores utilizam linguagens para descrição de ações mais expressivas que o STRIPS, como a linguagem ADL (*Action Definition Language*) [17], permitindo o uso de efeitos condicionais. O planejador UCPOP [18] trabalha com um formalismo baseado na representação ADL, lidando com efeitos condicionais, objetivos universalmente quantificados e, efeitos e pré-condições universalmente quantificados. O planejador PRODIGY [19] também utiliza a mesma representação utilizada pelo STRIPS, porém executa uma busca com *backtracking*, realizada a partir de duas rotinas complementares: a simulação de planos e a regressão encadeada de estados. O PRODIGY é capaz de lidar com pré-condições negadas e disjuntas, quantificação existencial e universal e ainda efeitos condicionais.

## **3.2 A Linguagem de Definição de Domínios de Planejamento - PDDL**

A representação do mundo real por uma estrutura capaz de ser “entendida” e “interpretada” por um agente planejador é sem dúvida o primeiro passo para que um plano de ações possa ser gerado. Esta representação deve ser a mais fiel possível, e

conter informações sobre o estado atual do domínio, operadores que o alteram, condições para que um operador possa ser aplicado e também quais as conseqüências ou mudanças realizadas no domínio quando uma ação é executada.

A linguagem PDDL [20] [21] (*Planning Domain Definition Language*) foi desenvolvido com o intuito de definir uma linguagem padrão para representação de domínios e especificação de problemas para a competição de agentes planejadores AIPS-98 [22]. Como os operadores representados em PDDL são baseados nos operadores do modelo proposto no planejador STRIPS, as pré-condições e efeitos de um operador representam por si só a influência que a aplicação deste realiza no domínio. Outras características da linguagem PDDL são a representação de efeitos condicionais, definição de quantificadores universais, com a criação e destruição de objetos, especificação de restrições de segurança, especificação de operadores hierárquicos compostos por sub operadores e sub objetivos além de permitir que um mesmo problema representado em PDDL seja tratado por vários planejadores diferentes (portabilidade de problemas entre agentes planejadores).

A linguagem PDDL foi estruturada em subconjuntos de características, chamados *requirements*. A cada definição de um novo domínio, os *requirements* a serem utilizados devem ser declarados, desde que o agente planejador suporte-os. Por exemplo, uma vez que o *requirement* [:disjunctive-preconditions] seja declarado, é permitido que o operador lógico OU seja inserido na função objetivo.

Uma nova versão da linguagem para representação de domínios de planejamento foi desenvolvida como extensão da PDDL original. À linguagem PDDL 2.1 [23] foram incorporadas características, com a intenção de representar domínios de planejamento determinísticos que envolvam tempo e que necessitem de recursos de manipulação algébrica, incorporando características da linguagem ADL. A PDDL 2.1 está dividida em cinco níveis: o nível 1 é o próprio PDDL desenvolvido para a competição AIPS-98. O nível 2 é um suplemento ao nível 1, permitindo a comparação entre variáveis numéricas e atualização dos valores das mesmas. Os níveis 3 e 4 permitem a representação de ações onde o efeito das mesmas tem dependência temporal com efeitos não contínuos (nível 3) e efeitos contínuos (nível



4), facilitando assim a representação de domínios de tempo real. O nível 5 é uma extensão do nível 4 capaz de representar domínios contínuos e discretos em tempo real.

Embora existam dezenas de planejadores disponíveis, a natureza do domínio proposto nessa dissertação aponta para a necessidade de um agente capaz de realizar comparações (" $<$ ", " $<=$ ", " $=$ ", " $>$ " ou " $>=$ ") entre variáveis numéricas, além de cálculos matemáticos elementares (" $+$ ", " $-$ ", " $*$ " e " $/$ ").

Além de formalizar a definição de domínios de planejamento, a linguagem PDDL também define um formalismo para descrição dos problemas, que devem aparecer em arquivos distintos [24].

### **3.3 Definição Formal da Sintaxe**

No estudo formal das linguagens de programação a parte sintática deve ser respeitada. A definição formal da sintaxe de uma linguagem de programação é usualmente chamada de gramática em analogia com a terminologia usada nas linguagens naturais. Uma gramática consiste de um conjunto de definições (chamadas regras) que irão especificar as seqüências de caracteres (chamados itens léxicos) que irão formar programas válidos na linguagem. As duas classes de gramáticas úteis na definição formal de gramáticas são as gramáticas BNF (Forma de Backus-Naur) e as gramáticas regulares.

#### **3.3.1 Forma de Backus-Naur**

A Forma de Backus-Naur foi desenvolvida através de dois esforços de pesquisa paralelos conduzidas por dois homens, John Backus e Noam Chomsky. A nova notação foi ligeiramente alterada um pouco mais tarde para se descrever a linguagem ALGOL-60 por Peter Naur. Este método revisado para descrição de sintaxe passou a ser conhecido como Forma Backus-Naur ou, simplesmente, BNF (*Backus-Naur Form*).

Uma metalinguagem é uma linguagem que é usada para descrever uma outra linguagem. BNF é uma metalinguagem para linguagens de programação [25].

Uma gramática BNF é composta por um conjunto finito de regras que definem, em nosso caso, uma linguagem de programação. Além disto, BNF usa abstrações para representar estruturas sintáticas. As abstrações na descrição BNF são freqüentemente chamadas de símbolos não-terminais, ou simplesmente não-terminais. A representação para os nomes das abstrações (não-terminais) em BNF é um nome cercado pelos símbolos de menor e maior (< e >). Os itens léxicos da linguagem, também conhecidos como lexemas ou *tokens*, e são representados por nomes em negrito. Os itens léxicos são freqüentemente chamados terminais.

Uma regra BNF tem sempre um não-terminal em seu lado esquerdo e um conjunto composto por terminais e/ou não terminais em seu lado direito. O símbolo ::= é usado com o sentido de "é definido por" e une o lado esquerdo ao direito da regra. O símbolo | é usado com o significado de "ou" e é usado para não precisarmos escrever o lado esquerdo repetidas vezes.

### 3.3.2 Forma de Backus-Naur Estendida

Desde a sua introdução, muitas extensões tem sido propostas à BNF. Esta metalinguagem é chamada de EBNF (BNF Estendida) e difere da BNF em vários aspectos. Uma das principais alterações é o conjunto de metasímbolos. O conjunto de metasímbolos é formado pelos símbolos com propósitos especiais, como os símbolos BNF ::= e |. Os símbolos terminais são, na EBNF, cercados por haspas duplas e um ponto final é usado para terminar cada produção. O símbolo = é usado no lugar de ::= . Os símbolos não-terminais são escritos em itálico. As demais diferenças incluem:

- O uso de parêntesis para indicar agrupamento.
- O uso de colchetes para indicar 0 ou uma instância de um trecho X qualquer. Ou seja, X é opcional.

- O uso de chaves para indicar uma ou mais instâncias de um trecho X qualquer.

Por exemplo, um identificador pode ser definido em EBNF como:

*Identificador* = *Letra* {*Letra* | *Dígito*}

e em BNF poderia ser definido como:

Identificador ::= <Letra>

| <Identificador> <Letra>

| <Identificador> <Dígito>

O formalismo para descrição de domínios de planejamento na linguagem PDDL 2.1 segue a forma Backus-Naur estendida (EBNF):

- Cada regra é escrita no formato *<elemento sintático> ::= expansão*.
- Os símbolos menor e maior ( < e > ) delimitam os nomes dos elementos sintáticos.
- Elementos opcionais são delimitados por colchetes ( [ e ] ).
- O asterisco ( \* ) representa zero ou mais elementos, enquanto o sinal positivo (+) representa um ou mais elementos.
- Parêntesis são apenas delimitadores dos elementos sintáticos, não possuindo nenhuma interpretação semântica.
- Informações adicionais e regras de expansão aparecem sobrescritas por um marcador. A descrição do domínio de um problema deve definir a função de cada marcador que aparecer nos elementos sintáticos.

### 3.4 Sintaxe de definição de domínios

A Figura 3-2 apresenta a sintaxe de definição de domínios de planejamento em PDDL 2.1, onde as palavras chaves devem aparecer na ordem em que estão enumeradas.

<code>&lt; domain &gt;</code>	<code>::= (define (domain &lt; name &gt;)</code> <code>    [&lt; required – def &gt;]</code> <code>    [&lt; types – def &gt;]<sup>typing</sup></code> <code>    [&lt; constants – def &gt;]</code> <code>    [&lt; predicates – def &gt;]</code> <code>    &lt; action – def &gt;*)</code>
<code>&lt; require – def &gt;</code>	<code>::= (:requirements &lt; require – key &gt; +)</code>
<code>&lt; require – key &gt;</code>	<code>::= :strips</code>
<code>&lt; require – key &gt;</code>	<code>::= :adl</code>
<code>&lt; require – key &gt;</code>	<code>::= :typing</code>
<code>&lt; types – def &gt;</code>	<code>::= (:types &lt; typed list (name) &gt; )</code>
<code>&lt; typed? – list – of (t) &gt;</code>	<code>::=<sup>typing</sup> &lt; typed list (t) &gt;</code>
<code>&lt; typed? – list – of (t) &gt;</code>	<code>::= &lt; list (t) &gt;</code>
<code>&lt; constants – def &gt;</code>	<code>::= (:constants &lt; typed? list of (names) &gt;)</code>
<code>&lt; predicates – def &gt;</code>	<code>::= (:predicates &lt; atomic formula &gt; +)</code>
<code>&lt; atomic formula &gt;</code>	<code>::= (&lt; predicate &gt; &lt; typed? list of (variables)&gt;)</code>
<code>&lt; predicate &gt;</code>	<code>::= &lt; name &gt;</code>
<code>&lt; variable &gt;</code>	<code>::= ? &lt; name &gt;</code>

Figura 3-2 – Sintaxe de definição de domínios

**Name:** A categoria `< name >` consiste de uma palavra por uma palavra iniciada por uma letra, podendo conter caracteres alfanuméricos, além de hífen ( - ) e traços subscritos ( \_ ). Os nomes definidos em `< name >` devem ser únicos.

**Requirements:** Esta cláusula lista as características necessárias para que o planejador possa solucionar um determinado problema. PDDL é uma linguagem ampla e nem todos os planejadores têm a capacidade de tratar todas as capacidades da linguagem. Através da cláusula `:requirements` o planejador avalia se pode tratar o domínio ou não. As características mais comuns são:

- `:strips` – É o subconjunto básico da linguagem PDDL, permitindo apenas ações no estilo STRIPS. Quando não há cláusula `:requirements` especificada, assume-se somente o uso do subconjunto STRIPS da linguagem.

- *:adl* – É um super conjunto do formato STRIPS, incluindo declaração de tipos de variáveis (*:types*), pré-condições negadas (*:negative-preconditions*), pré-condições disjuntas (*:disjunctive-preconditions*), igualdades (*:equality*), pré-condições quantificadas (*:quantified-preconditions*) e efeitos condicionais (*:conditional-effects*).

**Types:** É uma lista usada para declarar tipos à uma lista de entidades. Os tipos são precedidos por um hífen ( – ) e são declarados ao final de uma lista de objetos. Um tipo atômico é apenas um predicado unário atemporal e pode ser usado em todo intervalo de tempo, enquanto este predicado não for negado.

**Constants:** Os nomes declarados neste campo, são tomados como constantes em todo o domínio. Tipos podem ser atribuídos a constantes, desde que a chave de requisitos *:types* seja especificada.

**Predicates:** Este campo consiste de uma lista de declarações de predicados, onde para cada um deles é especificada uma lista de variáveis, além de seus argumentos.

**Variable:** Este campo consiste de uma lista de declarações de variáveis, além de seus argumentos,

### 3.4.1 Sintaxe de definição de operadores

A sintaxe utilizada na declaração dos operadores de um domínio depende do conjunto de tipos de operadores declarados na cláusula *:requirements*. Como descrito no item 3.4, operadores declaradas no formato ADL são mais abrangentes, conforme mostrado na Figura 3-3

A aplicação de um operador ADL no formato  $\langle \textit{atomic} - \textit{eff} \rangle$  é executada a partir da instanciação de seus parâmetros. Uma vez que os predicados contidos em  $\langle \textit{atomic} - \textit{eff} \rangle$  estão totalmente instanciados, eles são adicionados ao estado  $S_n$ . Assim como os operadores de remoção em STRIPS, são retirados do estado  $S_n$  os predicados que aparecem negados em  $\langle \textit{atomic} - \textit{eff} \rangle$ .

Os operadores declarados com efeitos no formato (*when* *< formula >* *< atomic – eff >*) são aplicados a um estado  $S_n$  se o mesmo satisfizer a relação descrita em *< formula >*, fazendo com que os predicados *< atomic – eff >* sejam adicionados ao estado subsequente  $S_{n+1}$ .

Operadores que contenham efeitos declarados nos formatos (*forall* (*< typed? List of variables >*)<sup>+</sup> *< atomic – eff >*) e no formato (*forall* (*< typed? List of variables >*)<sup>+</sup> *< when < formula > < atomic – eff >*)) são aplicados a partir de todas as possíveis instâncias de *< typed? List of variables >*, considerando que as variáveis declaradas em *:types* são limitadas a instanciações com constantes do seu tipo compatível. As variáveis não declaradas em *:types* podem ser instanciadas com qualquer constante do domínio.

Os efeitos no formato (*and* *< eff – formula >* *< eff – formula >*<sup>+</sup>) são aplicados ao estado  $S_n$ , através da inclusão de cada um dos efeitos no estado  $S_{n+1}$ .

<i>&lt; action – def &gt;</i>	::= ( <i>:action</i> <i>&lt; name &gt;</i> <i>:parameters</i> ( <i>&lt; typed? List of (variable) &gt;</i> )
<i>&lt; action –de body &gt;</i>	::= <i>:precondition</i> <i>&lt; formula &gt;</i> <i>:effect</i> <i>&lt; eff – formula &gt;</i>
<i>&lt; formula &gt;</i>	::= <i>&lt; literal term &gt;</i>
<i>&lt; formula &gt;</i>	::= <i>not</i> <i>&lt; formula &gt;</i>
<i>&lt; formula &gt;</i>	::= ( <i>and</i> <i>&lt; formula &gt;</i> <i>&lt; formula &gt;</i> <sup>+</sup> )
<i>&lt; formula &gt;</i>	::= ( <i>or</i> <i>&lt; formula &gt;</i> <i>&lt; formula &gt;</i> <sup>+</sup> )
<i>&lt; formula &gt;</i>	::= ( <i>imply</i> <i>&lt; formula &gt;</i> <i>&lt; formula &gt;</i> )
<i>&lt; formula &gt;</i>	::= ( <i>exist</i> ( <i>&lt; typed? List of (variable) &gt;</i> <sup>+</sup> ) <i>&lt; formula &gt;</i> )
<i>&lt; atomic – eff &gt;</i>	::= <i>&lt; literal term &gt;</i>
<i>&lt; atomic – eff &gt;</i>	::= ( <i>and</i> <i>&lt; literal term &gt;</i> <i>&lt; literal term &gt;</i> <sup>+</sup> )
<i>&lt; eff – formula* &gt;</i>	::= <i>&lt; atomic –eff &gt;</i>
<i>&lt; eff – formula* &gt;</i>	::= ( <i>when</i> <i>&lt; formula &gt;</i> <i>&lt; atomic –eff &gt;</i> )
<i>&lt; eff – formula* &gt;</i>	::= ( <i>forall</i> ( <i>&lt; typed? List of (variable) &gt;</i> <sup>+</sup> ) <i>&lt; atomic –eff &gt;</i> )
<i>&lt; eff – formula* &gt;</i>	::= ( <i>forall</i> ( <i>&lt; typed? List of (variable) &gt;</i> <sup>+</sup> ) ( <i>when</i> <i>&lt; formula &gt;</i> <i>&lt; atomic –eff &gt;</i> ))
<i>&lt; eff – formula* &gt;</i>	::= <i>&lt; eff – formula* &gt;</i>
<i>&lt; eff – formula* &gt;</i>	::= ( <i>and</i> <i>&lt; eff – formula* &gt;</i> <i>&lt; eff – formula* &gt;</i> <sup>+</sup> )

Figura 3-3 - Sintaxe de definição de operadores

Na Figura 3-4 está descrito parte do exemplo de Drew McDermott [21] que descreve o domínio dos “Jarros” (*jug pouring domain*) em formato da linguagem PDDL 2.1.

```

(define (domain Jarros)
  (:requirements :typing :fluents)
  (:types jarro)
  (:functions
    (capacidade ?j – jarro)
    (quantidade ?j – jarro)
    :
    :
    :
  (: action transferir
    :parameters (?jarro1 ?jarro2 – jarro)
    :precondition (>= (capacidade ?jarro2) (+ (quantidade ?jarro1)
      (quantidade ?jarro2)))
    :effect (and (assign (quantidade ?jarro1) 0)
      (increase (quantidade ?jarro2) (quantidade ?jarro1)))
  )

```

Figura 3-4 – Exemplo de representação de domínio e operador em PDDL 2.1

### 3.5 Sintaxe de definição de problemas

O problema submetido ao agente planejador, também conhecido como arquivo de fatos, é definido em relação a um determinado domínio, descrevendo os fatos verdadeiros no estado inicial  $S_0$ , ao qual aplicado uma seqüência de ações tornam verdadeiros os fatos descritos como estado objetivo  $S_f$ , levando o mundo ao estado final  $S_f$ . A sintaxe para definição de problemas é definida conforme Figura 3-5.

O estado inicial *:init* de um domínio é uma lista de fórmulas atômicas consideradas verdadeiras no estado inicial  $S_0$ . A descrição do estado inicial de um domínio deve assumir um mundo fechado [9], ou seja, qualquer predicado não declarado em um estado é presumido como falso, conforme mencionado na Seção 3.1. Todos os objetos pertencentes ao domínio devem ser declarados no campo *:objects*, mesmo aqueles que por não serem verdadeiros, não são declarados no estado inicial  $S_0$ .

O estado objetivo *:goal* é uma fórmula atômica. A solução para um problema aplicado em um domínio, é um subconjunto do estado gerado pela aplicação de uma seqüência de ações ao estado inicial.

<code>&lt; problem &gt;</code>	<code>::= (define (problem &lt; name &gt;) (: domain &lt; name &gt;) [(: requirements :types)] &lt; object declaration &gt; [&lt; init &gt;] &lt; goal &gt;<sup>+</sup></code>
<code>&lt; object declaration &gt;</code>	<code>::= (:objects &lt; typed? List of (name) &gt;)</code>
<code>&lt; init &gt;</code>	<code>::= (:init &lt; atomic formula (name) &gt;<sup>+</sup>)</code>
<code>&lt; goal &gt;</code>	<code>::= (:goal &lt; formula &gt;)</code>

Figura 3-5 – Sintaxe de definição de problemas

A Figura 3-6 representa um exemplo de definição de problema para o domínio dos “Jarros” (*jug pouring domain*) em formato da linguagem PDDL 2.1.

```
(define (problem PROB_1)
  (:domain Jarros)
  (:objects JARRO_1 JARRO_2)
  (:init
    (= (capacidade JARRO_1) 5)
    (= (capacidade JARRO_2) 5)
    (= (quantidade JARRO_1) 2)
    (= (quantidade JARRO_2) 3)
  )
  (:goal
    (= (quantidade JARRO_1) 0)
  ))
```

Figura 3-6 – Exemplo de representação de um problema em PDDL 2.1

### 3.6 Metric-FF

O problema de planejamento considerando variáveis numéricas tem sido um desafio pois a representação proposta pelo planejador STRIPS no início da década de 70, suporta apenas representações proposicionais, onde comparações de grandezas e cálculos envolvendo variáveis numéricas não são consideradas.

Esse tipo de representação tem como grande desvantagem, a impossibilidade do modelamento de fenômenos essenciais para a representação de problemas reais. Durante a 3ª Competição Internacional de Planejamento (IPC-3) foi apresentado o



planejador Metric-FF [26] [27], capaz de lidar com variáveis numéricas e considerado um dos dois planejadores mais eficientes da competição junto com o LPG [28]. Embora existam outros planejadores capazes de lidar com variáveis numéricas, o Metric-FF foi escolhido, em função do seu desempenho na 3ª Competição Internacional de Planejamento (IPC-3)

### 3.6.1 Variáveis numéricas

Para atender a representação de ações capazes de lidar com variáveis numéricas, ou seja, representação de domínios na linguagem PDDL 2.1 (*i.e.* PDDL 2.1 nível 2) [29] [30], o planejador Metric-FF considera além do conjunto  $P$  de proposições, um conjunto de variáveis

$$V = \{v^1, \dots, v^n\}$$

onde  $n$  representa o número de variáveis.

Um estado  $s$  é representado por um par  $s = (p(s), v(s))$  onde  $p(s) \subseteq P$  e  $v(s) = (v^1(s), \dots, v^n(s)) \in \mathbf{Q}^n$  é um vetor de números racionais. Obviamente as proposições  $p(s)$  são verdadeiras e  $v^i(s)$  representa o valor da  $i$ -ésima variável numérica no estado  $s$ .

Uma *expressão* é definida como a correlação de uma variável  $v \in V$  e um número racional a partir do uso dos operadores aritméticos  $+$ ,  $-$ ,  $*$  e  $/$ . Uma *restrição numérica* é definida como uma tripla  $(exp, comp, exp')$ , onde  $exp$  e  $exp'$  são expressões e  $comp \in \{<, <=, =, =>, >\}$  são *comparadores*.

Um *efeito numérico* é uma tripla  $(v^j, atr, exp)$  onde  $v^j \in V$  é uma variável,  $atr \in \{:=, +=, -=, *=, /=\}$  é um *operador de atribuição*, e  $exp$  é uma expressão. Uma *condição* é um par  $(p(con), v(con))$ , onde  $p(con) \subseteq P$  e  $v(con)$  é um conjunto de restrições numéricas.

Um *efeito* é uma tripla  $(p(eff)^+, p(eff)^-, v(eff))$ , onde  $p(eff)^+ \subseteq P$ ,  $p(eff)^- \subseteq P$  são os próprios efeitos de adição e remoção, e  $v(eff)$  é um conjunto de efeitos numéricos tal que  $i \neq j$  para todo  $(v^i, atr, exp), (v^j, atr', exp') \in v(eff)$ .

Uma ação  $\alpha$  é um par  $(pre(\alpha), eff(\alpha))$ , onde  $pre(\alpha)$  são as pré-condições para execução da ação  $\alpha$ , e  $eff(\alpha)$  são os efeitos (tanto de adição como de remoção) decorrentes da execução da ação  $\alpha$ .

### 3.6.2 Método de busca

A estratégia de busca no espaço de estados utilizado pelo Metric-FF, é o A\* [31] [32]. A busca A\* utiliza uma heurística onde se um estado  $s'$  decorrente de um estado  $s$ , possui o valor de uma função custo superior ao do estado  $S$  ( $f(s') > f(s)$ ) o estado  $s'$  não é expandido. Em outras palavras, o algoritmo A\* expande todos os nós cujo custo  $f(s) < f^*$ , onde  $f^*$  é o custo da solução ótima.

O Metric-FF executa um passo de pré-processamento, onde as restrições numéricas, bem como os efeitos numéricos sejam transformados de modo a se tornarem *positivos monotônicos* [10] [26]. Uma restrição é positiva monotônica se seu valor numérico em um estado  $s'$ , decorrente de um estado  $s$ , tiver seu valor maior ou igual ao seu valor  $c$  no estado  $s$ . Um efeito é positivo monotônico se seu valor numérico em um estado  $s'$  decorrente de um estado  $s$ , for maior ou igual que seu valor  $c$  em  $s$ .

Deste modo, todos os efeitos de remoção e todos os efeitos numéricos os quais reduzem o valor da variável  $v^j$  afetada podem ser desprezados. Mais formalmente pode-se representar a verificação de uma variável como sendo positiva monotônica, como:

$$( \{ (v^j, comp, c) \mid \text{variável } v^j, comp \in \{>=, >\}, c \in \mathbf{Q}\}, \{c \mid c \in \mathbf{Q}, c > 0\} )$$

### **3.6.3 Instalação do Metric-FF**

O código fonte, escrito em linguagem C, está disponível gratuitamente para *download* na página oficial do Metric-FF [33].

A versão disponível na internet não está preparada para gerar um arquivo contendo o plano (solução) gerado. Para tal é necessário substituir a rotina “output\_planner\_info” do arquivo “main.c” pela rotina apresentada no Anexo C, e recompilar o programa, utilizando o comando “make ff”, na janela de comando do Linux Red Hat 9.0. Com esta modificação, o Metric-FF gera um arquivo contendo a solução, exatamente como foi usado na 3ª Competição Internacional de Planejamento (IPC-3) [34].

## **Capítulo 4**

### **4. Proposta de solução do problema**

Neste capítulo é apresentada uma visão geral sobre agentes inteligentes e feita uma descrição detalhada dos operadores implementados para solução do problema de planejamento da produção a partir da definição e implementação de um agente inteligente que tem como objetivo gerar uma seqüência de ações que representem um plano de produção em um Centro de Serviços de Corte e Solda Laser conforme apresentado no Capítulo 2. O agente é capaz de identificar as necessidades de produção a partir de um determinado estado e o plano gerado, uma vez executado, deve proporcionar o estado objetivo. Este agente será identificado como agente planejador de produção. A motivação deste projeto é proporcionar um ambiente de planejamento de produção seguro, livre de erros humanos, aplicável a situações reais.

#### **4.1 Visão geral sobre agentes inteligentes**

Atualmente, agentes inteligentes são objeto de pesquisa nos mais diversos campos como engenharia e informática. O termo agente é utilizado em muitas áreas

da ciência da computação, mas especialmente na área de inteligência artificial. Agentes são entidades capazes de receber estímulos de um meio externo e atuar de forma inteligente, fornecendo uma resposta a esse meio. Na verdade, existem inúmeras definições de agentes, com várias diferenças e particularidades entre elas. Franklin e Graesser [35] analisaram inúmeros conceitos e chegaram a uma definição bem abrangente: “agentes autônomos estão situados em um ambiente, no qual eles sentem e agem, sobre o tempo, seguindo sua própria lista de tarefas e sentem efeitos de suas ações no futuro”. Russell e Norvig [12] sugerem que: “um agente é algo capaz de perceber o ambiente no qual está inserido, através de sensores e atuar nele, através de atuadores”.

Apesar de não existir um consenso sobre uma definição formal do que seja um agente, algumas características esperadas são estabelecidas a partir da analogia com agentes do mundo real, de modo que é possível conceituar um agente como uma entidade ativa, capaz de atuar em um determinado domínio em função do conhecimento específico que possui. Um agente autônomo inteligente deve apresentar como principais características, a capacidade de incorporar conhecimento, tomar decisões e atuar no ambiente de acordo com as decisões tomadas.

Embora o sucesso de um agente esteja intimamente ligado ao nível de conhecimento do domínio no qual está inserido, adquirido a partir de uma seqüência de percepções, espera-se que o agente atue corretamente, de modo a cumprir seu objetivo. Pode-se definir como agente racional, o agente que toma a decisão correta [12]. Sendo assim, é necessário estabelecer uma medida de desempenho, com objetivo de medir o quanto de sucesso o agente obtém, a partir de sua atuação.

Russell e Norvig [12] definem como agente racional ideal como: “para cada seqüência de percepção possível, um agente racional ideal deve agir de modo a maximizar sua medida de desempenho, baseado nas evidências estabelecidas pela seqüência de percepções e pelo conhecimento do próprio agente”.

### 4.1.1 Características do ambiente de atuação do agente

Em função das características do ambiente de atuação do agente, as definições das ações a serem executadas e a base de conhecimento do agente são afetadas. Russell e Norvig [12] sugerem a seguinte classificação para ambientes de atuação de agentes:

- **Acessível x inacessível**

Um ambiente acessível é aquele no qual o agente pode obter informação completa, precisa e atualizada a respeito do estado em que o ambiente se encontra em um determinado momento, fazendo com que todo o conhecimento necessário para a tomada de decisão esteja disponível. Quanto mais acessível um ambiente, maior a chance de um agente maximizar sua desempenho.

- **Determinístico X não determinístico**

Um ambiente é determinístico quando o efeito decorrente da execução de uma ação é totalmente conhecido, em outras palavras, o estado alcançado a partir da execução de uma ação é certo e definido pelo efeito da ação.

- **Episódico X não episódico**

Em um ambiente episódico, o agente delibera o que fazer baseado apenas na percepção do episódio corrente, ou seja, a atuação do agente se dá por episódios discretos. Não há interação entre o estado corrente e o estado alcançado.

- **Estático x dinâmico**

Um ambiente estático só é alterado pela própria atuação do agente, ou seja, durante o processo de tomada de decisão, não é necessário que o agente continue a observar o ambiente.

- **Discreto X contínuo**

Um ambiente é discreto se a partir de uma quantidade finita de ações que podem ser executadas, o ambiente é levado a uma quantidade também finita de estados.

## 4.2 O agente planejador de produção

A arquitetura proposta para o agente planejador de produção de modo a representar o domínio descrito no Capítulo 2 está representada na Figura 4-1 e é composta das seguintes partes:

1. Módulo de definição dos estoques ideais;
2. Módulo de definição dos fluxos de produção;
3. Módulo de modelagem do estado inicial;
4. Módulo de modelagem do estado objetivo;
5. Módulo de definição dos operadores, e
6. Módulo Planejador.

As quantidades reais dos estoques de cada produto são informadas ao agente planejador através de um arquivo, em formato texto, (TXT). As informações contidas neste arquivo, são retiradas do sistema de controle de estoque corporativo. Um outro arquivo, também em formato texto, contém as demandas previstas para cada produto e é criado a partir das previsões de consumo, enviadas pelos clientes.

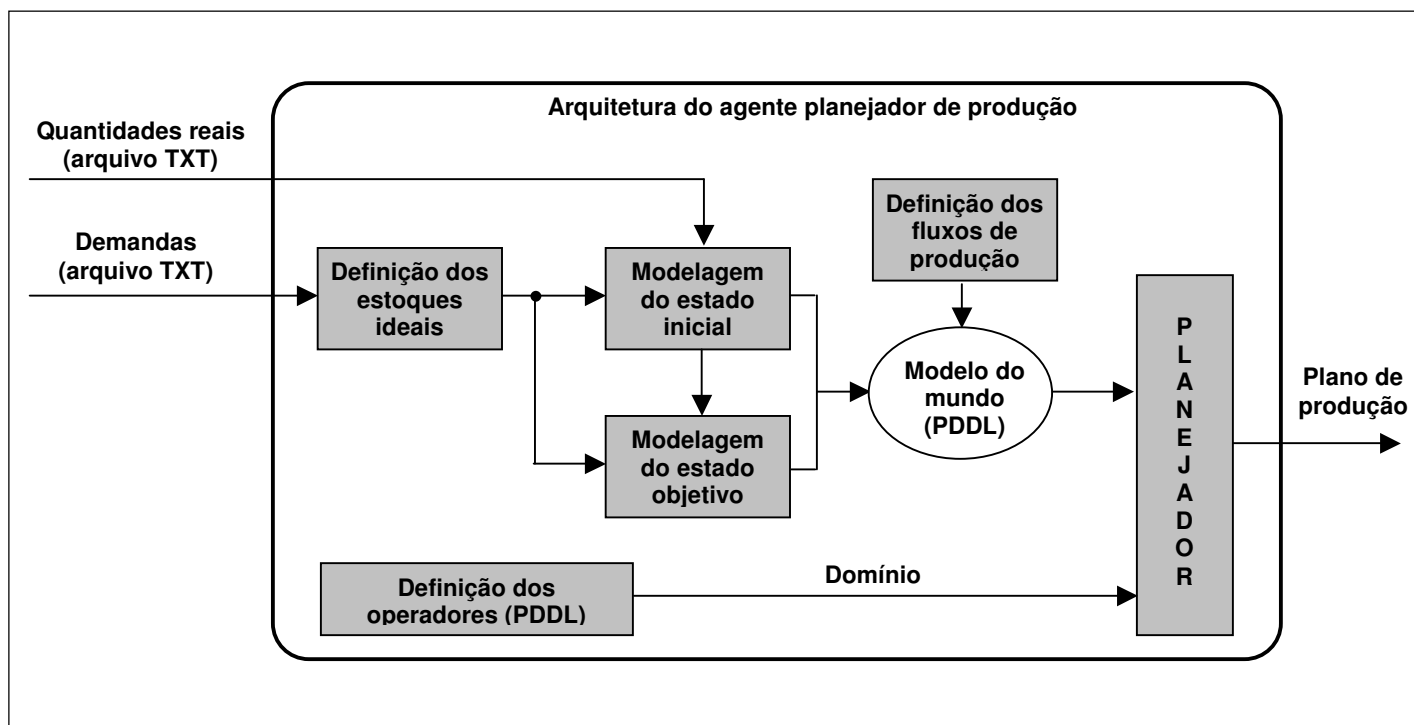


Figura 4-1 – Arquitetura do agente planejador de produção

Cada um dos módulos de compõem o agente planejador de produção será apresentado nas seções que se seguem.

### 4.3 Módulo de definição dos estoques ideais

O agente planejador de produção utiliza como parte dos dados de entrada, o estoque considerado ideal para cada um dos produtos considerados no domínio de planejamento. Como este trabalho trata de um problema real, por questões de ética, os nomes verdadeiros (códigos de identificação) dos produtos serão mantidos em sigilo, adotando-se nomes fictícios no formato <PRODUTO\_N>, onde o número N diferencia um produto de outro.

Como base para definição dos estoques ideais de cada produto, foram utilizadas as previsões de demanda de cada um deles, informadas pelos clientes. Considerando que o envio das peças para montadoras de automóveis é feito na modalidade *just-in-time*, e que em alguns casos o tempo máximo de entrega não pode ultrapassar 18 horas, se faz necessária a formação de um estoque de segurança, levando-se em conta possíveis indisponibilidades de equipamentos. Em resumo, podemos considerar que a decisão de produzir ou não um determinado produto é tomada a partir da verificação dos níveis de estoque de segurança definido como “confortáveis”.

As montadoras de automóveis informam suas demandas previstas para cada peça considerando entregas semanais. Sendo assim, baseado em experiência, foi determinado como nível de estoque seguro, uma quantidade de peças suficiente para atender quatro entregas, ou seja, quatro semanas de produção. Podemos definir o estoque ideal de um produto, como sendo:

$$EI(P)_n = \sum_{i=n}^{n+3} DS(i)$$

Onde:

$EI(P)_n$  = Estoque ideal do produto P na semana n

$DS(i)$  = Demanda prevista para semana i

n = Semana



Considerando a variação de  $i$  entre  $n$  e  $n+3$ , temos o valor acumulado do estoque ideal de um produto  $P$ , como sendo a soma da demanda prevista para um período de quatro semanas, conforme exemplificado na Tabela 4-1

Produto_1						
Semana (n)	1	2	3	4	5	6
Demanda (DS)	100	120	80	100	150	90
Estoque ideal (EI)	400	450	420	.....	.....	.....

Tabela 4-1– Exemplo de definição de estoque ideal

$$EI(\text{Produto}_1)_1 = DS(1) + DS(2) + DS(3) + DS(4) = 100 + 120 + 80 + 100 = 400$$

$$EI(\text{Produto}_1)_2 = DS(2) + DS(3) + DS(4) + DS(5) = 120 + 80 + 100 + 150 = 450$$

$$EI(\text{Produto}_1)_3 = DS(3) + DS(4) + DS(5) + DS(6) = 80 + 100 + 150 + 90 = 420$$

⋮  
⋮  
⋮

$$EI(\text{Produto}_1)_n = DS(n) + DS(n+1) + DS(n+2) + DS(n+3)$$

A representação em PDDL utilizada para o estoque ideal de um produto e listada no arquivo de fatos (problema) submetido ao agente planejador de produção, onde é atribuída ao estoque ideal do <PRODUTO\_N>, a quantidade definida como ideal  $EI(P)$ , através do somatório das demandas previstas em quatro semanas é:

$$(= (\text{est\_ideal } \langle \text{PRODUTO\_N} \rangle) EI(P))$$

#### 4.4 Módulo de definição dos fluxos de produção

As ações geradas como plano de produção dependem única e exclusivamente dos fluxos de produção, conforme definidos no Capítulo 2. Desta forma, a definição dos fluxos de produção deve ser feita para cada um dos produtos, uma vez que a

pré-condição para escolha de uma ação disponível no domínio só é satisfeita a partir do efeito de adição gerado pela execução de uma outra ação, respeitando o fluxo.

Tomando-se como exemplo, um produto cujo fluxo de produção deve seguir as etapas de corte na Linha de Corte Transversal, inversão de face no Virador de Fardos e processo de lavagem na Lavadora de Blanks (fluxo número 10, apresentado na Figura 2-9), é possível verificar que a pré-condição para que seja possível lavar o *blank*, é que o mesmo tenha sido virado no Virador de Fardo e para isso, ele obviamente deve estar cortado, conforme Figura 4-2.

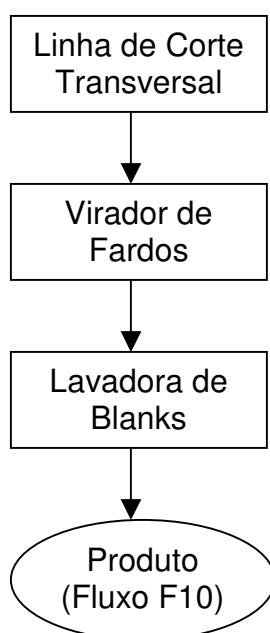


Figura 4-2 – Exemplo de um fluxo de produção (F10)

A determinação do fluxo de produção para cada produto é feita a partir das necessidades exigidas pelas características geométricas, requisitos de qualidade e conveniência de processo. O fluxo de cada produto é listado internamente na geração dos problemas pela interface gráfica descrita no Capítulo 5. Caso um novo produto seja considerado, o fluxo necessário para produção do mesmo, deve ser previamente definido e informado ao agente planejador de produção.

A representação em PDDL, do fluxo de produção de cada produto é declarada no arquivo de fatos no formato abaixo, onde Fj representa o fluxo de produção definido para o <PRODUTO\_N>.

(fluxo <PRODUTO\_N> Fj)

## 4.5 Módulo de modelagem do estado inicial

A definição do estado inicial de cada problema resume-se basicamente em conhecer as quantidades reais em estoque para cada produto. Como as demandas para cada produto são definidas semanalmente pelos clientes, os estoques reais também são monitorados semanalmente, para serem comparados com as demandas e a partir dessa comparação o agente planejador define a necessidade de produção.

Um arquivo em forma de tabela, contendo o estoque real de cada produto é utilizado como dado de entrada para o agente planejador.

	Estoque real (ER) – Semana n
Produto_1	380
Produto_2	520
⋮	⋮
⋮	⋮
Produto_N	95

Tabela 4-2– Exemplo de arquivo de entrada com estoques reais

A representação em PDDL do estoque real de cada produto é similar a representação do estoque ideal, onde ER(P) representa a quantidade real, em estoque, do PRODUTO\_N:

(= (est\_real <PRODUTO\_N> ER(P))

## 4.6 Módulo de modelagem do estado objetivo

Embora o objetivo do agente planejador seja gerar uma seqüência de ações de produção, a necessidade de produzir ou não, um determinado produto é identificada a partir da comparação do nível de estoque real com o nível de estoque ideal. Inicialmente pode-se considerar que o objetivo é manter o estoque real igual ao estoque ideal, porém esta regra faria com que o planejador gerasse um plano a cada vez que o estoque real caísse abaixo do estoque ideal, mesmo que por uma pequena quantidade, desencadeando assim produções de pequenos lotes, que do ponto de vista prático são inviáveis de serem produzidos.

Para resolver esse problema foi definido, além do estoque ideal, um estoque mínimo ainda considerado seguro e que a partir do qual é necessário o disparo do plano de produção. Esse estoque mínimo é o ponto a partir do qual o estoque deve ser repostado, para que não haja risco de desabastecimento dos clientes e foi definido como 50% do estoque ideal, representando assim o estoque suficiente para fornecimento de produtos por duas semanas.

Na representação da função objetivo em PDDL abaixo, o estoque real é comparado com 50% do estoque ideal (ponto de reposição) onde caso o estoque real seja maior que o ideal multiplicado por 0.5, o objetivo está atendido, não sendo gerado plano algum de produção é:

$$(>= (\text{est\_real } <\text{PRODUTO\_N}>) (* (\text{est\_ideal } <\text{PRODUTO\_N}>) 0.5))$$

Esta regra é aplicada individualmente para cada produto e pode ser mudada a qualquer momento de acordo com a necessidade de formação de estoque, como por exemplo, o aumento das quantidades estocadas para um ou mais produtos em função de longas paradas programadas dos equipamentos para manutenção.

O Anexo A apresenta um exemplo de definição de problema na linguagem PDDL, generalizado para N produtos, contendo a declaração dos estados iniciais e função objetivo.

## 4.7 Módulo de definição dos operadores

Os operadores desenvolvidos para o domínio de planejamento Centro de Serviços de Corte e Solda Laser apresentado no Capítulo 2, foram estabelecidos de modo que cada um deles represente a produção em um determinado equipamento do fluxo produtivo. A partir deste princípio, foram definidos sete operadores da seguinte forma:

- Operador CORTAR – Representa o corte de material na Linha de Corte Transversal.
- Operador SOLDAR – Representa a solda de dois *blanks* na Máquina de Solda Laser.
- Operador VIRAR – Representa a inversão de face no Virador de Fardos.
- Operador RESSALTAR – Representa a execução de ressaltos em um *blank* soldado, na Prensa de Ressaltos.
- Operador APARAR\_PRECISÃO – Representa o processo de apara na Guilhotina de Precisão.
- Operador APARAR\_SIMPLES – Representa o processo de apara na Guilhotina Convencional.
- Operador LAVAR – Representa a lavagem de *blanks* na Lavadora de *Blanks*.

### 4.7.1 Operador CORTAR

O operador CORTAR representa o processo de produção de *blanks* na Linha de Corte Transversal e é incluída no plano de ações gerado pelo agente planejador de produção, cada vez que o estoque real de um determinado produto pertencente aos fluxos de produção F2 à F11, for inferior ao estoque ideal. Considerando que o primeiro processo de produção é o corte dos *blanks*, podemos considerar que a ação CORTAR é a primeira ação a ser executada. O efeito da execução desta ação é a própria declaração que um determinado produto foi cortado na Linha de Corte Transversal.

A Figura 4-3 representa uma notação gráfica do operador CORTAR, onde as pré-condições necessárias para que a ação seja executada são apresentadas na parte superior do retângulo e os efeitos resultantes da execução da ação são apresentadas na parte inferior do mesmo. O nome do operador é apresentado no interior do retângulo. Esta representação será utilizada para os demais operadores.

Estoque real (Produto\_N) < Estoque ideal (Produto\_N)

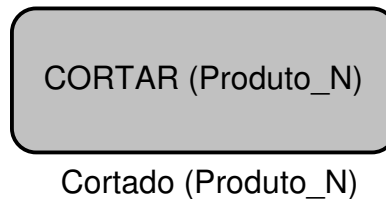


Figura 4-3 – Representação gráfica do operador CORTAR

A Figura 4-4 mostra a representação do operador CORTAR em PDDL.

```
(:action CORTAR
 :parameters (?p)
 :precondition (< (est_real ?p) (est_ideal ?p))
 :effect (cortado ?p)
)
```

Figura 4-4 – Representação do operador CORTAR em PDDL

## 4.7.2 Operador SOLDAR

Conforme descrito na Seção 2.1, a produção de *blanks* soldados consiste em soldar dois *blanks*, normalmente com espessuras distintas (ver Figura 2-4 e Figura 2-5). Desta forma, a condição mínima para que um *blank* seja soldado a outro, gerando assim um *blank* soldado, é que cada uma das partes que o compõe estejam disponíveis para a solda.

O operador SOLDAR foi implementado para representar o processo na máquina de solda laser, levando em consideração que as quantidades de partes que compõe o *blank* soldado estão disponíveis. Esta é a condição mínima que permite o processo de solda. A Figura 4-5 apresenta a notação gráfica da ação SOLDAR.

(Blank\_Soldado Produto\_N P1 P2)  
 Estoque real (P1) >= Estoque ideal (P1)  
 Estoque real (P2) >= Estoque ideal (P2)

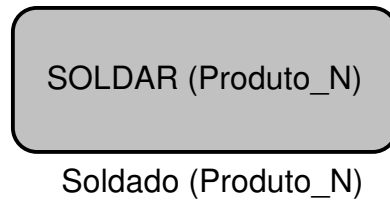


Figura 4-5 – Representação gráfica do operador SOLDAR

Considerando um determinado *blank* soldado P composto pelas partes P1 e P2, podemos assumir que o *blank* soldado N poderá ser produzido se as quantidades reais das peças P1 e P2 em estoque forem maiores que os estoques ideais. Sendo assim, a pré-condição para execução da ação SOLDAR pode ser expressa como a conjunção de três fatos:

- N é um *blank* soldado composto por P1 e P2;
- Existem peças P1 suficientes para execução da solda, e
- Existem peças P2 suficientes para execução da solda.

O efeito adicionado com a execução da ação SOLDAR é que o produto N encontra-se soldado. A Figura 4-6 mostra a representação do operador SOLDAR em PDDL.

```
(:action SOLDAR
  :parameters (?p ?p1 ?p2)
  :precondition (and (blank_soldado ?p ?p1 ?p2)
                    (>= (est_real ?p1) (est_ideal ?p1))
                    (>= (est_real ?p2) (est_ideal ?p2))
                  )
  :effect (soldado ?p)
)
```

Figura 4-6 – Representação do operador SOLDAR em PDDL

### 4.7.3 Operador VIRAR

O processo de virar uma determinada peça é executado em um equipamento chamado virador de fardos. Este processo é necessário cada vez que uma determinada peça é produzida e há necessidade de entregá-las aos clientes em pares, compostos por uma peça representando o lado esquerdo e a outra o lado direito. Como exemplo, podemos citar as portas de um automóvel, onde os *blanks* são cortados de forma idêntica, e 50% deles são virados de modo a compor um par de portas, por exemplo, a porta dianteira esquerda em par com a porta dianteira direita.

Conforme indicado na Figura 2-9, o processo de virar uma peça deverá ser incluída no plano de ação gerado pelo agente planejador, cada vez que a produção de *blanks* pertencentes aos fluxos de produção F8, F9 e F10 for necessária.

o processo que antecede ao Virador de Fardo para qualquer um dos fluxos F8, F9 ou F10 é o corte na Linha de Corte Transversal. Desta maneira, a pré-condição para que um Produto\_N seja virado é que o mesmo tenha sido cortado (ação CORTAR(Produto\_N) executada e efeito Cortado (Produto\_N) gerado). Como efeito de adição, é declarado que o Produto\_N está virado. A Figura 4-7 mostra a representação gráfica do operador VIRAR.

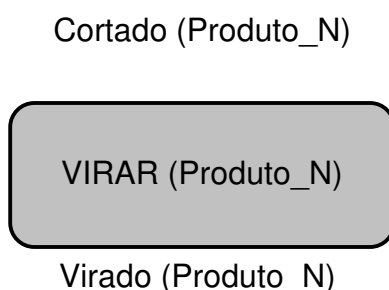


Figura 4-7 – Representação gráfica do operador VIRAR

A representação do operador VIRAR em PDDL, conforme foi implementado, é apresentado na Figura 4-8.



```

(:action VIRAR
 :parameters (?p)
 :precondition (cortado ?p)
 :effect (virado ?p)
)

```

Figura 4-8 – Representação do operador VIRAR em PDDL

#### 4.7.4 Operador RESSALTAR

O processo de ressaltos é executado por um equipamento chamado Prensa de Ressaltos e tem como objetivo nivelar as pilhas de *blanks* soldados. Este nivelamento é necessário, pois em função da diferença de espessura das partes que compõem os *blanks* soldados quando os mesmos são empilhados uns sobre os outros, a altura final da pilha no lado das partes de menor espessura é inferior ao lado em que as partes de maior espessura estão empilhadas. Normalmente, o desempilhamento dos *blanks* é feito por braços robotizados e esta desnivelamento impede que as ventosas dos manipuladores dos robôs toquem a pilha de *blanks* de modo uniforme, causando risco de queda do *blank* ou mesmo inviabilizando o desempilhamento (ver Figura 2-6 e Figura 2-7). O operador RESSALTAR, representa o processamento de *blanks* soldados na Prensa de Ressaltos.

Conforme a Figura 2-9, os únicos produtos que são processados na Prensa de Ressaltos são *blanks* soldados (fluxo de produção F6). Deste modo, a condição para que um determinado Produto\_N seja ressaltado, é que o mesmo já tenha sido soldado. A representação gráfica do operador RESSALTAR apresentada na Figura 4-9, mostra, além da pré-condição para execução da ação, o efeito de adição que nada mais é do que a declaração do Produto\_N como ressaltado. A Figura 4-10 mostra o operador RESSALTAR implementada em PDDL.

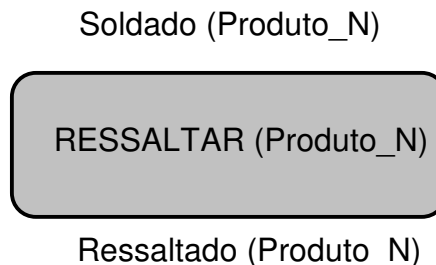


Figura 4-9 – Representação gráfica do operador RESSALTAR

```
(:action RESSALTAR
:parameters (?p)
:precondition (soldado ?p)
:effect (ressaltado ?p)
)
```

Figura 4-10 – Representação do operador RESSALTAR em PDDL

#### 4.7.5 Operadores APARAR\_PRECISÃO e APARAR\_SIMPLES

O processo de aparar de *blanks* em guilhotinas é executado por dois motivos. O primeiro deles é que para que seja possível efetuar a solda de *blanks*, é necessário que a borda a ser soldada possua uma linearidade de corte menor que 0,05 mm. No caso onde o comprimento da borda é inferior a 600 mm, esta linearidade é facilmente alcançada pelo próprio corte do *blank* na Linha de Corte Transversal. Porém nos casos onde o comprimento da solda for superior a esta medida, é necessária a preparação da borda a ser soldada, de modo a garantir a linearidade necessária. Desta forma, *blanks* com comprimento de solda maior que 600 mm, têm suas bordas aparadas em uma guilhotina de precisão.

O segundo motivo para que um *blank* seja processado em uma guilhotina, se deve ao fato de que por conveniências de processo, buscando uma melhor produtividade, um determinado *blank* de comprimento “L”, é cortado com um comprimento múltiplo de “L” na Linha de Corte Transversal, e então redividido em uma guilhotina. Para esse processo, não é necessário que a guilhotina seja de precisão, ou seja, é possível utilizar uma guilhotina convencional.

Tanto o processo de aparar de *blanks* na Guilhotina de Precisão como na Guilhotina Convencional são precedidos pelo processo de corte na Linha de Corte Transversal (fluxos de produção F5, F6 e F7), conforme Figura 2-9. Desta maneira, foi considerado como pré-condição para execução destas ações que os *blanks* tenham sido cortados, tendo como única diferença os efeitos, que Aparado (Produto\_N) representa o produto N aparado na Guilhotina de Precisão e Aparado\_Simples (Produto\_N) o produto N aparado na Guilhotina Convencional. A

Figura 4-11 mostra a representação gráfica dos operadores APARAR\_PRECISÃO e APARAR\_SIMPLES e a Figura 4-12 a representação dos mesmos em PDDL.

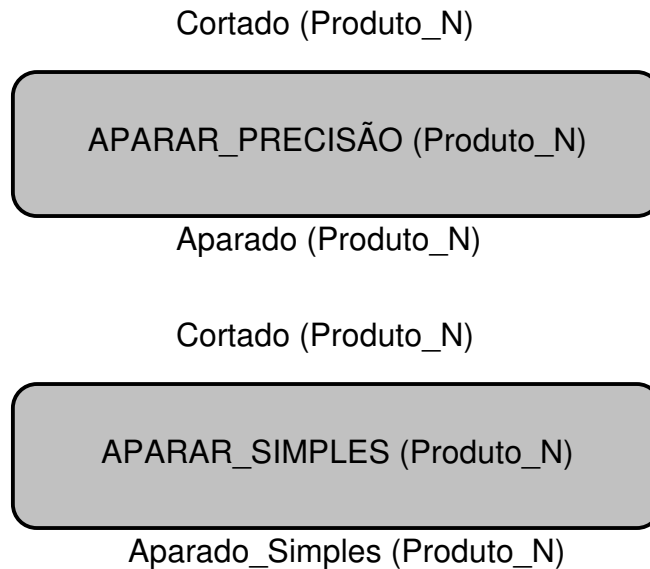


Figura 4-11 – Representação gráfica dos operadores APARAR\_PRECISÃO e

#### APARAR\_SIMPLES

```
(:action APARAR_PRECISAO
:parameters (?p)
:precondition (cortado ?p)
:effect (aparado ?p)
)

(:action APARAR_SIMPLES
:parameters (?p)
:precondition (cortado ?p)
:effect (apara_simples ?p)
)
```

Figura 4-12 – Representação dos operadores APARAR\_PRECISÃO e APARAR\_SIMPLES em PDDL

### 4.7.6 Operador LAVAR

Para garantir um perfeito acabamento superficial em peças que compõem a parte externa da carroceria dos automóveis, os *blanks* devem estar totalmente livres de impurezas, pois caso contrário, essas impurezas acabam causando pequenas

deformações superficiais durante o processo de estampagem. Estas deformações são percebidas mesmo após a pintura das peças e são suficientes para que as mesmas sejam desviadas pelo controle de qualidade.

Com o objetivo de eliminar totalmente as impurezas da superfície dos *blanks*, é necessário que os mesmos passem por um processo de lavagem e revestimento com uma camada de óleo para proteção contra oxidação. O equipamento que executa esse procedimento é a Lavadora de Blanks.

O processo de lavagem de *blanks* pode ser executado com *blanks* provenientes diretamente da Linha de Corte Transversal (fluxo de produção F11) ou ainda com *blanks* virados no Virador de Fardos (fluxo de produção F10), conforme mostrado na Figura 2-9. Desta maneira foram considerados como pré-condições para execução da ação LAVAR, que um determinado produto N esteja cortado, desde que o mesmo pertença ao fluxo F11, ou ainda que o produto N esteja virado se pertencer ao fluxo F10. Como efeito de adição, é declarado que o produto N está lavado. A Figura 4-13 mostra a representação gráfica do operador LAVAR e a Figura 4-14 a representação do operador implementada em PDDL.

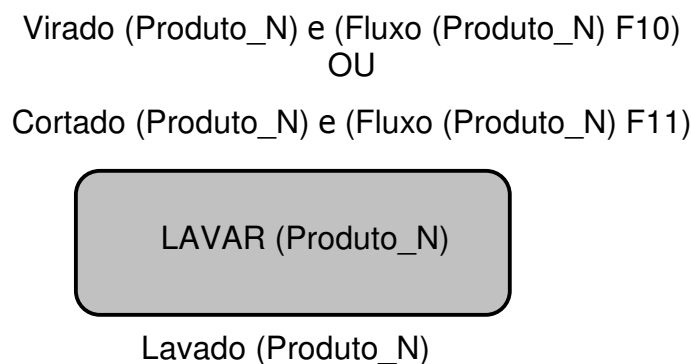


Figura 4-13 – Representação gráfica do operador LAVAR

```

(:action LAVAR
 :parameters (?p)
 :precondition (or (and (fluxo ?p F10) (virado ?p))
                  (and (fluxo ?p F11) (cortado ?p)))
 :effect (lavado ?p)
)

```

Figura 4-14 – Representação do operador LAVAR em PDDL

### 4.7.7 Operador PRODUZIR

O operador PRODUZIR não representa o processamento de peças em nenhum equipamento específico e tem como único objetivo tornar o estoque real de cada produto igual ao estoque ideal, conforme definido nas Seções 4.3 e 4.5 de modelagem de definição dos estoques ideais e de modelagem do estado inicial, respectivamente.

Uma vez que o objetivo do agente planejador de produção é gerar um plano de ações que leve o estoque real de cada produto à níveis considerados ideais, é necessário que a ação PRODUZIR seja executada cada vez que for necessário repor o estoque de qualquer produto. Para isso, este operador tem como pré-condições para sua execução, o último processo de cada fluxo de produção, pois uma vez que cada produto for processado no último equipamento referente ao seu fluxo, significa que o mesmo foi produzido e o nível de estoque repostado ao nível ideal.

Tomando como exemplo um determinado produto N cujo fluxo é o processamento seqüencial na Linha de Corte Transversal → Virador de Fardos → Lavadora de *Blanks* (fluxo de produção F10), podemos considerar, em última análise que para produzir uma peça que pertença a este fluxo, é necessário que a mesma tenha sido processada na Lavadora de *Blanks*. Considerando Produto\_N a identificação de um determinado produto pertencente ao fluxo F10 e que foi processado na Lavadora de *Blanks*, analisando a Figura 4-15, pode-se notar que se o Produto\_N foi lavado, necessariamente ele deve ter sido virado e para isso, cortado. Sendo assim, é possível estabelecer como pré-condição para a ação PRODUZIR, a expressão:

(Lavado (Produto\_N))  $\wedge$  (Fluxo (Produto\_N) F10)

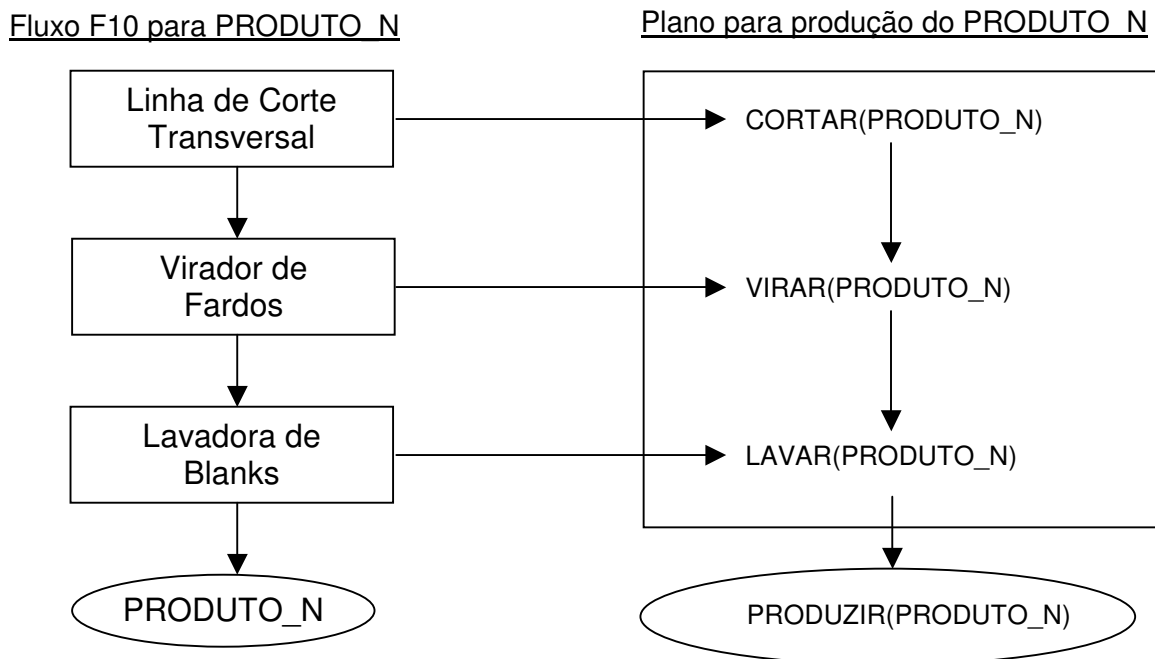


Figura 4-15 – Plano de produção para o fluxo F10

O exemplo acima, representa apenas a pré-condição estabelecida para produtos pertencentes ao fluxo F10. Evidentemente este operador deve contemplar todos os fluxos de produção possíveis, que uma vez seguidos, tornam o estoque real de cada produto igual ao estoque definido como ideal. Para definição de todas as pré condições necessária para a execução da ação PRODUZIR, foi identificado o último equipamento de processo para cada fluxo possível. A Tabela 4-3 mostra o último equipamento de cada fluxo de produção possível, e o efeito de adição da execução de cada operador que representa o processamento em cada equipamento:

<b>Fluxo</b>	<b>Efeito de adição</b>
F3	Cortado (Produto_N)
F4	Soldado (Produto_N)
F5	Aparado (Produto_N)
F6	Ressaltado (Produto_N)
F7	Aparado_Simples (Produto_N)
F8	Virado (Produto_N)
F9	Virado (Produto_N)
F10	Lavado (Produto_N)
F11	Lavado (Produto_N)

Tabela 4-3 – Últimos equipamentos de processo para cada fluxo de produção

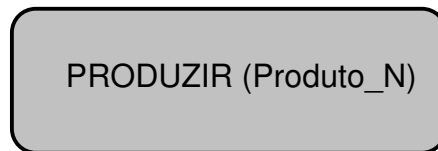
O operador PRODUZIR gera como efeito de adição o aumento da quantidade de peças do estoque real, igualando-o igual ao estoque ideal. Note que a ação não determina a quantidade de peças que devem ser produzidas, mas sim quais peças devem ser.

Sempre que o estoque de uma determinada peça estiver abaixo de 50% do estoque ideal, conforme descrito na seção 4.6, o plano gerado pelo agente indicará os processos pelo qual aquele produto é produzido, e será finalizado pela ação PRODUZIR, fazendo com que o estoque seja regularizado.

A definição deste operador permitiu que os planos de produção gerados fossem apresentados de forma similar aos planos gerados pelo planejador de produção humano, enquanto que os operadores que representam cada equipamento dos fluxos produtivos permitiu um detalhamento dos planos de produção, que não eram gerados pelo agente humano.

A representação gráfica do operador PRODUZIR é apresentada na Figura 4-16 e a representação em PDDL na Figura 4-17.

Cortado (Produto_N) e (Fluxo (Produto_N) F3)	OU
Soldado (Produto_N) e (Fluxo (Produto_N) F4)	OU
Aparado (Produto_N) e (Fluxo (Produto_N) F5)	OU
Ressaltado (Produto_N) e (Fluxo (Produto_N) F6)	OU
Aparado_Simples (Produto_N) e (Fluxo (Produto_N) F7)	OU
Virado (Produto_N) e (Fluxo (Produto_N) F8)	OU
Virado (Produto_N) e (Fluxo (Produto_N) F9)	OU
Lavado (Produto_N) e (Fluxo (Produto_N) F10)	OU
Lavado (Produto_N) e (Fluxo (Produto_N) F11)	OU



Estoque\_real (Produto\_N) = Estoque\_Ideal (Produto\_N)

Figura 4-16 – Representação gráfica do operador PRODUZIR

```
(:action PRODUZIR
:parameters (?p)
:precondition (or (and (cortado ?p) (fluxo ?p F3))
                 (and (soldado ?p) (fluxo ?p F4))
                 (and (aparado ?p) (fluxo ?p F5))
                 (and (ressaltado ?p) (fluxo ?p F6))
                 (and (apara_simples ?p) (fluxo ?p F7))
                 (and (virado ?p) (fluxo ?p F8))
                 (and (virado ?p) (fluxo ?p F9))
                 (and (lavado ?p) (fluxo ?p F10))
                 (and (lavado ?p) (fluxo ?p F11)))
:effect (increase (est_real ?p) (- (est_ideal ?p
(est_real ?p)))
)
```

Figura 4-17 – Representação do operador PRODUZIR em PDDL



O Anexo B apresenta o domínio de planejamento, representado em PDDL e utilizado para a solução do problema proposto. Como os operadores estão no formato STRIPS, é possível verificar as pré-condições e efeitos de cada uma das ações.

## 4.8 Exemplo de busca no domínio

Para exemplificar a busca de uma sequência de ações que satisfaçam a função objetivo, será considerado que o estoque de um determinado produto <PROD\_1> é inferior a 50% do estoque considerado ideal. O produto <PROD\_1> é um blank soldado composto das partes <PROD\_1\_1> e <PROD\_1\_2>, onde o estoque de <PROD\_1\_1> é maior que o estoque ideal, porém o estoque do <PROD\_1\_2> também é inferior a 50% do estoque ideal. Será considerado também, que a parte referente a <PROD\_1\_2> deve ser aparada na Guilhotina de Precisão. Para facilitar a compreensão do exemplo, serão atribuídos valores para os estoques reais e ideais, conforme Tabela 4-4 abaixo:

Produto	Estoque real	Estoque ideal
PROD_1	30	100
PROD_1_1	115	110
PROD_1_2	50	120

Tabela 4-4 – Exemplos de valores de estoques reais e ideais

O estado inicial (base de conhecimento) para o exemplo proposto, pode ser representada em PDDL da seguinte maneira:

```
; ----- Definição dos fluxos de produção -----
(fluxo PROD_1 F4)
(fluxo PROD_1_2 F3)
(fluxo PROD_1_3 F5)
; ----- Definição dos estoques reais -----
(= (est_real PROD_1) 30)
(= (est_real PROD_1_1) 115)
(= (est_real PROD_1_2) 50)
```

```

; ----- Definição dos estoques ideais -----
(= (est_ideal PROD_1) 100)
(= (est_ideal PROD_1_1) 110)
(= (est_ideal PROD_1_2) 120)
; ----- Declaração dos blanks soldados -----
(blank_soldado PROD_1 PROD_1_1 PROD_1_2)
; ----- Função objetivo -----
(>= (est_real PROD_1) (*(est_ideal PROD_1) 0.5))

```

Como o operador PRODUZIR é o único que faz com que o estoque real seja igual ao estoque ideal, ou seja, o próprio efeito de adição do operador, ele será sempre o ponto de partida da busca no espaço de operadores descritos no domínio de planejamento. A pré-condição que deve ser satisfeita para que o estoque real seja igualado ao estoque ideal é  $(and (soldado ?p) (fluxo ?p F4))$ , que uma vez instanciada a partir das declarações contidas no estado inicial pode ser escrita como  $(and (soldado PROD_1) (fluxo PROD_1 F4))$ , onde o segundo termo da expressão é verdadeiro.

Para que o primeiro termo da expressão torne-se verdadeiro e a ação PRODUZIR (PROD\_1) possa ser incluída no plano de ação, o agente planejador deve buscar entre as ações definidas no domínio, uma ação que tenha como efeito de adição, a expressão  $(soldado PROD_1)$ .

Para que a ação SOLDAR seja incluída no plano de ações, tornando assim a expressão  $(soldado PROD_1)$  verdadeira, é necessário que a pré-condição  $(and (blank_soldado ?p ?p1 ?p2) (>= (est_real ?p1) (est_ideal ?p1)) (>= (est_real ?p2) (est_ideal ?p2)))$  seja satisfeita. Fazendo a instanciação das variáveis com os dados declarados no estado inicial temos  $(and (blank_soldado PROD_1 PROD_1_1 PROD_1_2) (>= (est_real PROD_1_1) (est_ideal PROD_1_1)) (>= (est_real PROD_1_2) (est_ideal PROD_1_2)))$ . O primeiro termo da expressão instanciada é verdadeiro, pois está declarado no estado inicial. Como o estoque real de <PROD\_1\_1> é 115 e o estoque ideal 110, o segundo termo da expressão também é verdadeiro. O terceiro termo é falso, pois o estoque real de <PROD\_1\_2> é 50 e o estoque ideal 120.

Neste ponto, mais uma vez o agente planejador busca uma ação onde o efeito de adição seja o de igualar o estoque real ao ideal, ou seja, executar a ação PRODUZIR. Como a instanciação deve ser feita com o produto <PROD\_1\_2> e o

mesmo é processado através do fluxo F5, a pré-condição que deve ser satisfeita para execução da ação PRODUIR (PROD\_1\_2) é *(and (aparado ?p) (fluxo ?p F5))*, instanciada com a variável <PROD\_1\_2>, gerando assim a expressão *(and (aparado PROD\_1\_2) (fluxo PROD\_1\_2 F5))*. O segundo termo da expressão é verdadeiro, pois está declarado no estado inicial.

Para que a expressão *(aparado PROD\_1\_2)* torne-se verdadeira, é necessária a execução da ação APARAR\_PRECISAO (PROD\_1\_2) que deve ter a pré-condição *(cortado PROD\_1\_2)* satisfeita. Da mesma forma, a execução da ação CORTAR (PROD\_1\_2) que tem como efeito de adição a expressão *(cortado PROD\_1\_2)*, deve ter a sua pré-condição *(< (est\_real ?p) (est\_ideal ?p))* satisfeita. Como a instanciação é feita com a variável <PROD\_1\_2>, temos a expressão *(< (est\_real PROD\_1\_2) (est\_ideal PROD\_1\_2))* verdadeira, pois o estoque real do produto PROD\_1\_2 é 50 e o ideal 120.

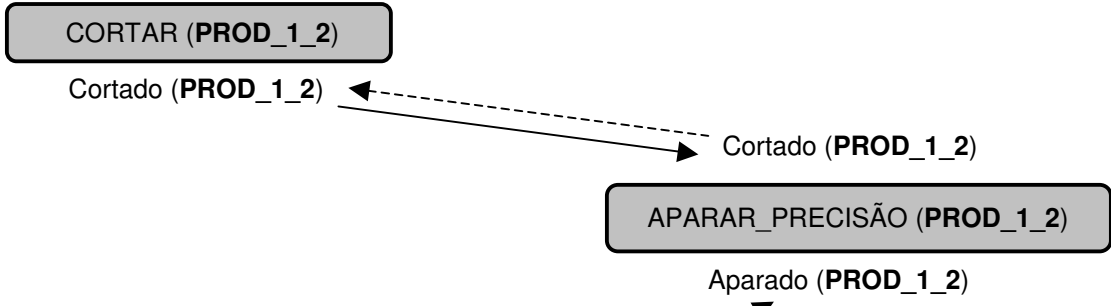
De forma resumida, temos um plano gerado de modo que a execução da ação CORTAR (PROD\_1\_2) satisfaça a pré-condição da APARAR\_PRECISAO (PROD\_1\_2) que por sua vez satisfaz a pré-condição da ação PRODUIR (PROD\_1\_2). Uma vez produzido o produto <PROD\_1\_2>, a pré-condição da ação SOLDAR (PROD\_1 PROD\_1\_1 PROD\_1\_2) é satisfeita, que por sua vez também satisfaz a pré-condição da ação PRODUIR (PROD\_1), conforme indicado na Figura 4-12. A Tabela 4-5 mostra o plano de ações gerado para atender a função objetivo.

<b>Plano de produção</b>
CORTAR (PROD_1_2)
APARAR_PRECISAO (PROD_1_2)
PRODUIR (PROD_1_2)
SOLDAR (PROD_1 PROD_1_1 PROD_1_2)
PRODUIR (PROD_1)

Tabela 4-5 – Exemplo de plano de produção gerado pelo agente planejador

A Figura 4-18 mostra a busca através da satisfação das pré-condições das ações geradas no plano de produção, de modo a atender ao objetivo de tornar o estoque real do produto (PROD\_1) igual ao estoque ideal.

Estoque real (**PROD\_1\_2**) < Estoque ideal (**PROD\_1\_2**)



Cortado (Produto\_N) e (Fluxo (Produto\_N) F3) OU  
 Soldado (Produto\_N) e (Fluxo (Produto\_N) F4) OU  
 Aparado (**PROD\_1\_2**) e (Fluxo (**PROD\_1\_2**) F5) OU  
 Ressaltado (Produto\_N) e (Fluxo (Produto\_N) F6) OU  
 Aparado\_Simples (Produto\_N) e (Fluxo (Produto\_N) F7) OU  
 Virado (Produto\_N) e (Fluxo (Produto\_N) F8) OU  
 Virado (Produto\_N) e (Fluxo (Produto\_N) F9) OU  
 Lavado (Produto\_N) e (Fluxo (Produto\_N) F10) OU  
 Lavado (Produto\_N) e (Fluxo (Produto\_N) F11)



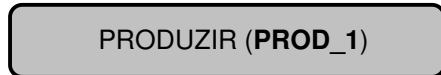
Estoque\_real (**PROD\_1\_2**) = Estoque\_Ideal (**PROD\_1\_2**)

**PROD\_1** é um *blank* soldado composto **PROD\_1\_1** e **PROD\_1\_2**  
 Estoque real (**PROD\_1\_1**) >= Estoque ideal (**PROD\_1\_1**)  
 Estoque real (**PROD\_1\_2**) >= Estoque ideal (**PROD\_1\_2**)



Soldado (**PROD\_1**)

Cortado (Produto\_N) e (Fluxo (Produto\_N) F3) OU  
 Soldado (**PROD\_1**) e (Fluxo (**PROD\_1**) F4) OU  
 Aparado (Produto\_N) e (Fluxo (Produto\_N) F5) OU  
 Ressaltado (Produto\_N) e (Fluxo (Produto\_N) F6) OU  
 Aparado\_Simples (Produto\_N) e (Fluxo (Produto\_N) F7) OU  
 Virado (Produto\_N) e (Fluxo (Produto\_N) F8) OU  
 Virado (Produto\_N) e (Fluxo (Produto\_N) F9) OU  
 Lavado (Produto\_N) e (Fluxo (Produto\_N) F10) OU  
 Lavado (Produto\_N) e (Fluxo (Produto\_N) F11)



Estoque\_real (**PROD\_1**) = Estoque\_Ideal (**PROD\_1**)

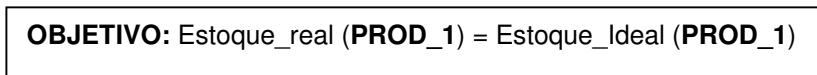


Figura 4-18 – Exemplo de busca no domínio

## 4.9 Considerações gerais sobre o agente implementado

O agente foi totalmente implementado e é capaz de gerar os planos de produção a partir da criação dos problemas de planejamento, considerando os estoques reais e as demandas de cada produto, ou seja, alterações de estoque e demandas são previstas, podendo-se solicitar ao agente a execução de um novo plano a qualquer momento. Alterações no domínio de planejamento, como por exemplo, novos fluxos de produção ou indisponibilidade de equipamentos do processo produtivo não foram considerados. Desta forma podemos caracterizar o domínio de atuação do agente planejador de produção como sendo acessível, determinístico e estático.

Os planos de produção gerados, contendo a lista de ações necessárias para que o estado objetivo seja alcançado ainda podem ser pós-processados, de forma que seja possível a visualização de planos de produção parciais, separados por produtos ou por equipamentos. O objetivo deste pós-processamento é permitir a visualização do planejamento individual de cada equipamento descrito no Capítulo 2, facilitando a alocação de recursos de mão-de-obra convenientemente. A Figura 4-19 descreve a arquitetura do pós-processamento, onde a partir do plano gerado pelo agente, é feita a separação das ações.

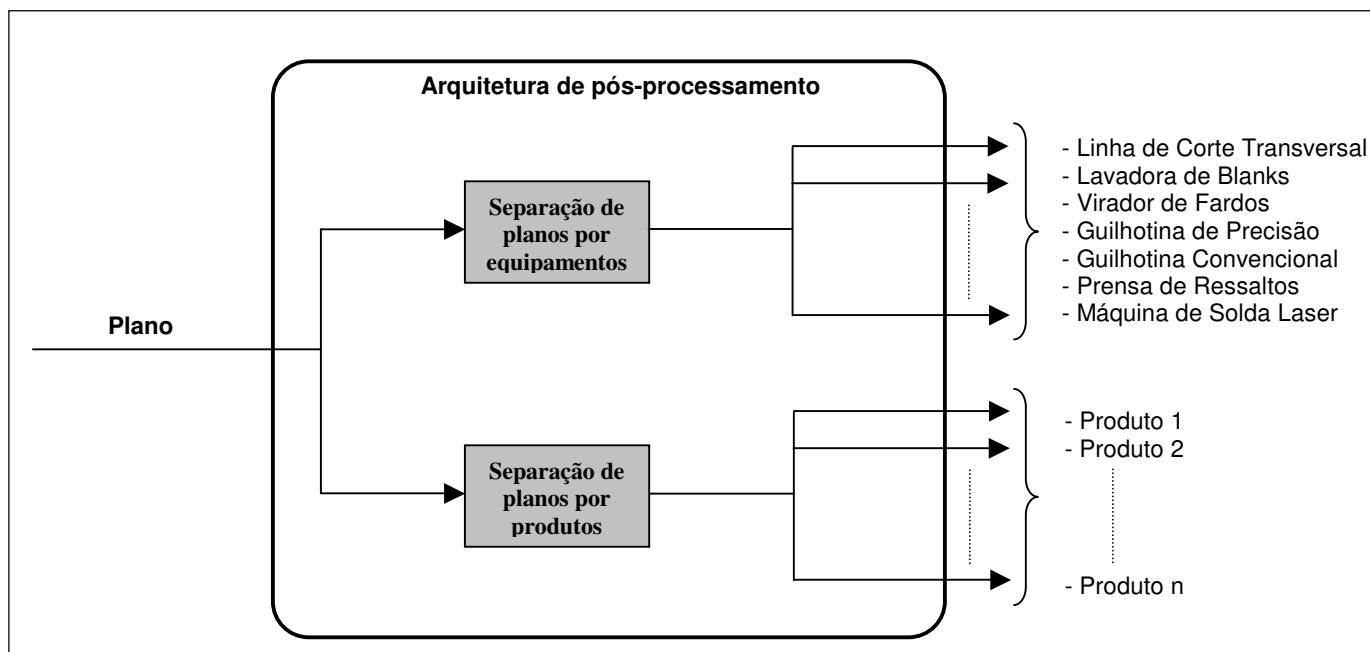


Figura 4-19 – Arquitetura de pós-processamento

O agente planejador de produção é executado a partir de uma interface gráfica desenvolvida em MATLAB que é executada sobre o sistema operacional LINUX Red Hat 9.0. A partir da interface gráfica é possível ler os arquivos de entrada, executar o agente IA e visualizar o plano gerado. A descrição detalhada da interface é apresentada no capítulo 5.

## Capítulo 5

### 5. Interface Gráfica

Uma interface gráfica foi totalmente desenvolvida, com o objetivo de operacionalizar a criação dos problemas de planejamento, bem como o processamento dos mesmos pelo Metric-FF e a visualização das soluções geradas pelo agente planejador da produção. A interface está dividida em dois módulos, sendo o primeiro, responsável pela criação dos problemas, a partir dos dados de entrada (estoques reais e demandas dos produtos) e o segundo, a criação das telas de operação e consulta dos resultados, ambas implementadas em Matlab V6.5 para Linux.

A estrutura de pastas criada para para a perfeita execução da interface gráfica é fixa e deve respeitar o modelo representado na Figura 5-1.

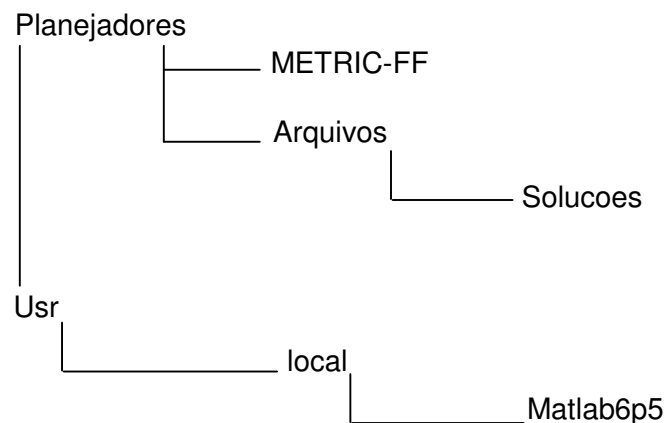


Figura 5-1 – Estrutura de pastas para a execução da interface gráfica

Após a criação da estrutura acima, o Metric-FF deve ser instalado na pasta /Planejadores/METRIC-FF/, e o Matlab instalado na pasta Usr/local/Matlab6p5/.

O arquivo de dados contendo as quantidades reais de cada produto, o arquivo de demandas e o arquivo com a descrição do domínio de planejamento (Anexo B) devem ser copiados para pasta Planejadores/Arquivos/. A estrutura dos arquivos de entrada de dados (estoques reais e demandas) serão apresentadas na Seção 5.1 a seguir. As soluções gerada pelo agente planejador de produção são gravadas na pasta Planejadores/Arquivos/Solucoes/ (\*).

Uma vez que a interface foi criada para comparação de soluções reais propostas por um planejador de produção humano com as soluções geradas pelo agente planejador de produção artificial, foram tratados como base de dados de entrada 70 produtos diferentes, durante um período de 17 semanas. De acordo com a natureza do problema proposto (Centro de Serviços de Corte e Solda Laser) este período é mais do que suficiente para análise, uma vez que as demandas previstas são atualizadas semanalmente, permitindo que a cada novo período, uma nova simulação seja feita, gerando assim um novo plano de produção mais atualizado. Por exemplo, o plano de produção para semana 10, gerado na semana 1 leva em consideração as demandas previstas para as semanas 10, 11, 12 e 13, conforme descrito no Seção 4.1.1. Como as demandas são dinâmicas e corrigidas semanalmente, os planos de produção gerados nas semanas subsequentes até a semana 10, podem sofrer alterações.

## 5.1 Arquivos de entrada

Tanto o arquivo contendo as quantidades reais do estoque de cada produto, como o arquivo de demandas, devem ser do tipo “texto” e copiados para a pasta “/Planejadores/Arquivos/”. O nome do arquivo com as quantidades reais do estoque deve ser <real.txt>, respeitando o seguinte formato, mostrado na Tabela 5-1:

(\*) O nome da pasta “Soluções” deve ser escrito sem o cedilha e o til, conforme indicado no texto.



Real	1	2	3	.....	j	.....	17
Prod_1	ER11	ER12	ER13	.....	ER1j	.....	ER117
Prod_2	ER21	ER22	ER23	.....	ER2j	.....	ER217
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Prod_i	ERi1	ERi2	ERi3	.....	ERij	.....	ERi17
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Prod_70	ER701	ER702	ER703	.....	ER70j	.....	ER7017

Tabela 5-1 – Formato do arquivo de estoques reais por produto

Onde:

{1, 2, 3, ..., j, ..., 17} = Número inteiro que representa a 1ª semana até a 17ª semana.

{Prod\_1, Prod\_2, ..., Prod\_i, ..., Prod\_70} = Campo alfanumérico, contendo o nome de cada produto.

ERij = Número inteiro que representa o estoque real do produto Prod\_i na semana j.

Exemplo: ER703 = 1000, significa que o estoque real do produto Prod\_70 na semana 3 é igual a 1000 peças.

O nome do arquivo contendo as demandas previstas para cada produto deve ser <previsao.txt>, respeitando o seguinte formato, mostrado na Tabela 5-2:

Previsao	1	2	3	.....	j	.....	20
Prod_1	DS11	DS12	DS13	.....	DS1j	.....	DS120
Prod_2	DS21	DS22	DS23	.....	DS2j	.....	DS220
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Prod_i	DSi1	DSi2	DSi3	.....	DSij	.....	DSi20
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Prod_70	DS701	DS702	DS703	.....	DS70j	.....	DS7020

Tabela 5-2 – Formato do arquivo de demandas previstas por produto

Onde:

{1, 2, 3, ..., j, ..., 20} = Número inteiro que representa a 1ª semana até a 20ª semana.

{Prod\_1, Prod\_2, ..., Prod\_i, ..., Prod\_70} = Campo alfanumérico, contendo o nome de cada produto.

DS<sub>ij</sub> = Número inteiro que representa a demanda prevista do produto Prod<sub>i</sub> na semana j.

Exemplo: DP703 = 1500, significa que a demanda prevista do produto Prod<sub>70</sub> na semana 3 é igual a 1500 peças.

O arquivo de demandas previstas contempla um período de 20 semanas para permitir o cálculo do estoque ideal para a semana 17, pois o mesmo é calculado a partir do somatório das demandas previstas para as semanas 17, 18, 19 e 20, conforme descrito na Seção 4.1.1.

Comentários:

- O limite para o tamanho do nome dos produtos é de 18 caracteres.
- O caracter separador de cada campo deve ser um espaço em branco.
- A ordem dos produtos listados no arquivo de demandas previstas deve ser exatamente a mesma ordem dos produtos listados no arquivo de estoques reais.

## 5.2 Descrição das telas

A tela principal da interface gráfica está representada na Figura 5-2 e contém cinco botões: Gerar arquivos, Visualizar arquivo, Metric-FF, Visualizar Solução e Pós-processamento.

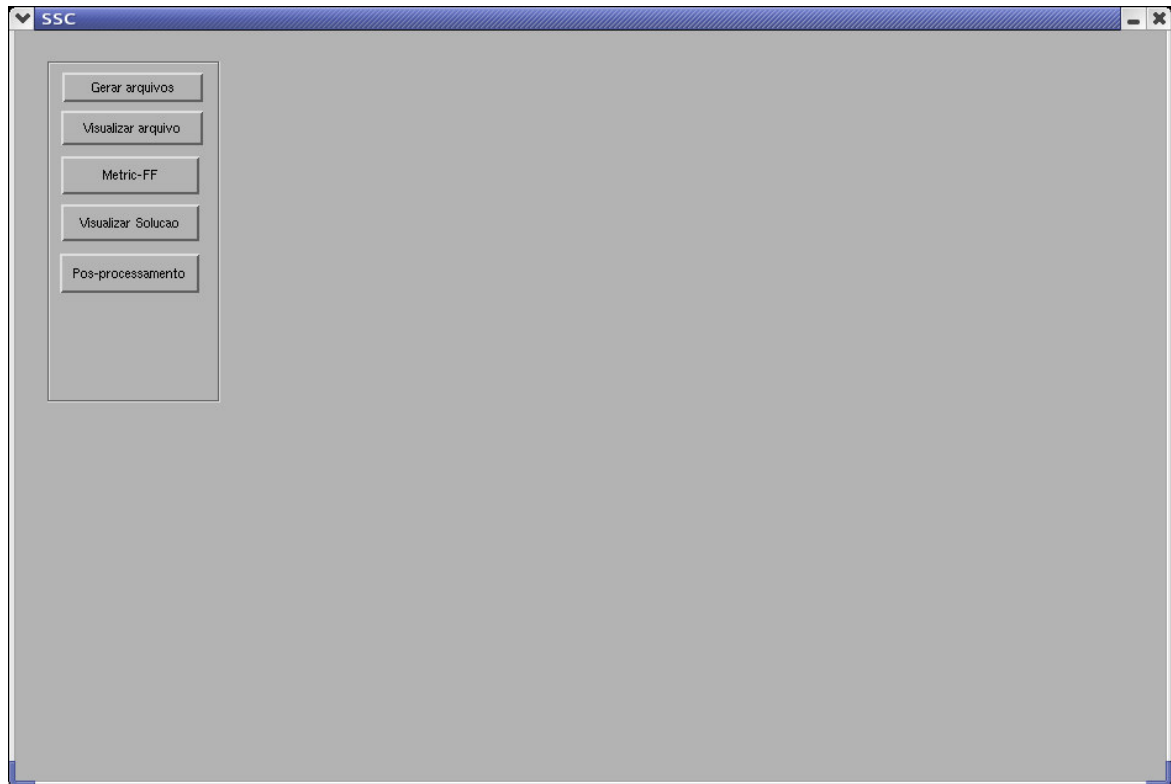


Figura 5-2 – Tela principal da interface gráfica

O início da operação da interface gráfica deve ser feito a partir do acionamento do botão "Gerar arquivos". Ao se pressionar este botão é criado um arquivo de fluxos de produção chamado "fluxos" na pasta /Planejadores/Arquivos/. Esse arquivo contém o fluxo de produção para cada um dos 70 produtos contemplados pela interface gráfica. Após a criação do arquivo de fluxos é aberta uma nova janela, conforme Figura 5-3, para a entrada da quantidade de problemas que se deseja submeter ao agente planejador de produção. O número digitado deve ser um inteiro positivo entre 1 e 17, representando assim as 17 semanas avaliadas. Após a

digitação da quantidade de arquivos que se deseja gerar o botão “OK” deve ser pressionado.

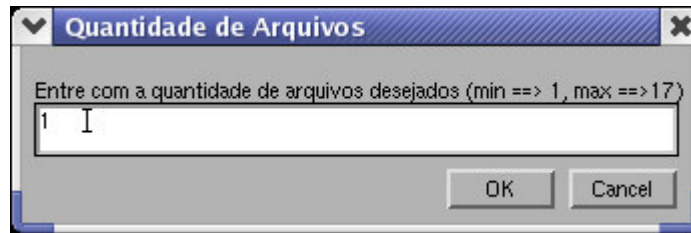


Figura 5-3 – Tela para entrada da quantidade de problemas a ser submetido ao agente planejador de produção

Após o botão “OK” ser pressionado, uma tela de espera é aberta para que os arquivos contendo as quantidades reais em estoque de cada produto (*real.txt*), bem como o arquivo contendo as demandas previstas (*previsão.txt*) sejam consultados, e um novo arquivo, conforme modelo apresentado no Anexo A, seja criado na pasta */Planejadores/Arquivos/*. Serão criados tantos arquivos quanto o número digitado na tela apresentada na Figura 5-3. Os arquivos criados são gravados com o seguinte formato de nome: *S<j>.pddl*, onde:

*S* = Representa a palavra “Semana”.

*<j>* = Número inteiro que representa a 1ª até a 17ª semana (conforme número digitado na tela da Figura 5-3).

*.pddl* = Extensão do arquivo que indica um arquivo de fatos no formato PDDL.

Após a criação dos arquivos de fatos, é possível a visualização dos mesmos, através do acionamento do botão “Visualizar arquivos” na tela principal. Quando este botão é pressionado, é aberta uma tela onde aparecem listados todos os arquivos no formato PDDL existentes na pasta */Planejadores/Arquivos/*. Para a visualização do arquivo desejado, basta selecioná-lo e pressionar o botão “OK”. A Figura 5-4 exemplifica a tela para seleção do arquivo que se deseja visualizar e a Figura 5-5 exemplifica o modo como o arquivo é exibido.

Comentários:

- A Figura 5-4 mostra 17 arquivos criados, onde a 1ª semana é a semana 36 e a 17ª semana é a semana 52.
- Para “rolar” a tela de visualização do arquivo de fatos, basta pressionar as teclas “PageUp” ou “PageDown” do teclado do computador, uma vez que o “ScrollLock” esteja ativo.

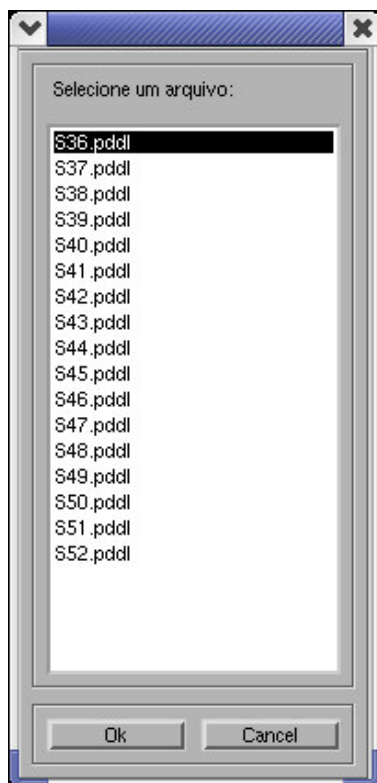
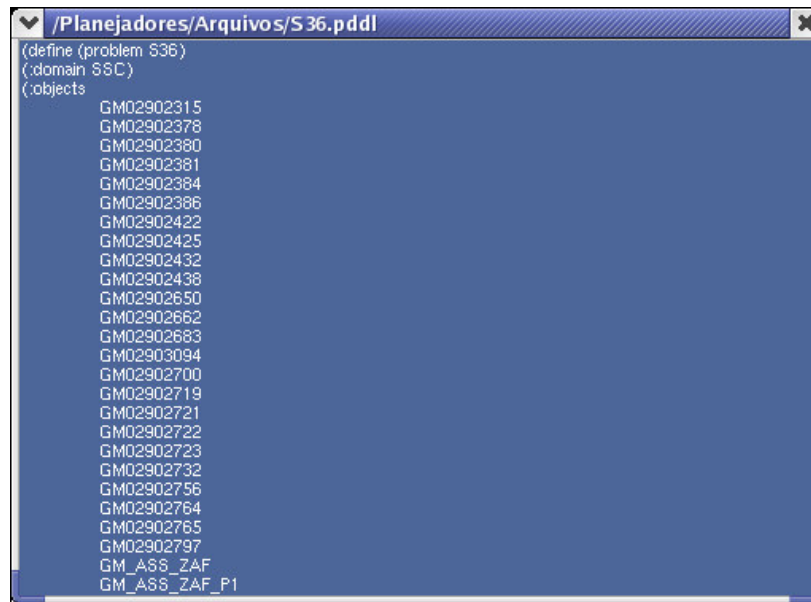


Figura 5-4 – Tela para visualização dos arquivos gerados (problemas)

A screenshot of a file editor window titled "/Planejadores/Arquivos/S36.pddl". The window contains a list of objects in a Prolog-like format. The text is as follows:

```
(define (problem S36)
  (:domain SSC)
  (:objects
    GM02902315
    GM02902378
    GM02902380
    GM02902381
    GM02902384
    GM02902386
    GM02902422
    GM02902425
    GM02902432
    GM02902438
    GM02902650
    GM02902662
    GM02902683
    GM02903094
    GM02902700
    GM02902719
    GM02902721
    GM02902722
    GM02902723
    GM02902732
    GM02902756
    GM02902764
    GM02902765
    GM02902797
    GM_ASS_ZAF
    GM_ASS_ZAF_P1
```

Figura 5-5 – Exemplo de arquivo de fatos gerado

O botão “Metric-FF” é utilizado para submeter o arquivo de fatos gerado ao planejador METRIC-FF. Ao se pressionar este botão, a tela de seleção de arquivos (Figura 5-4) é aberta mais uma vez para que se possa selecionar o arquivo de fatos para o qual se deseja obter o plano de produção. Após a seleção do arquivo de fatos, um arquivo contendo o plano de ação gerado pelo agente planejador de produção é gravado na pasta /Planejadores/Arquivos/Solucoes/, com o nome no formato S<j>.soln, onde:

S = Representa a palavra “Semana”.

<j> = Número inteiro que representa a 1ª até a 17ª semana (conforme número digitado na tela da Figura 5-3).

.soln = Extensão do arquivo que indica uma solução (plano gerado).

A visualização do arquivo de solução gerado é possível a partir do acionamento do botão “Visualizar solução” na tela principal da interface gráfica. Ao se pressionar este botão é aberta uma tela para seleção dos arquivos que contém os planos gerados, conforme Figura 5-6.

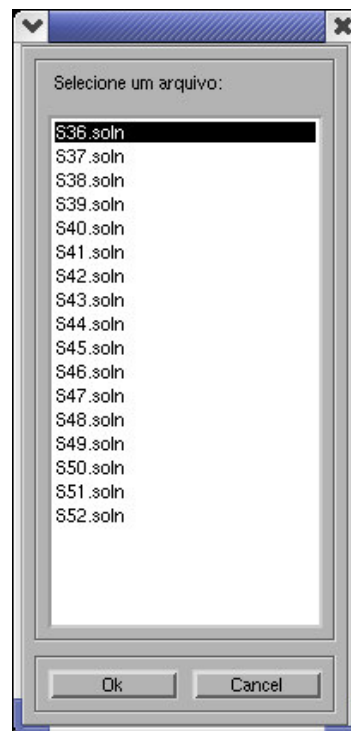


Figura 5-6 – Tela para seleção dos planos gerados

Uma vez selecionado um arquivo de planos gerados, é aberta a tela de visualização do plano, conforme Figura 5-7.

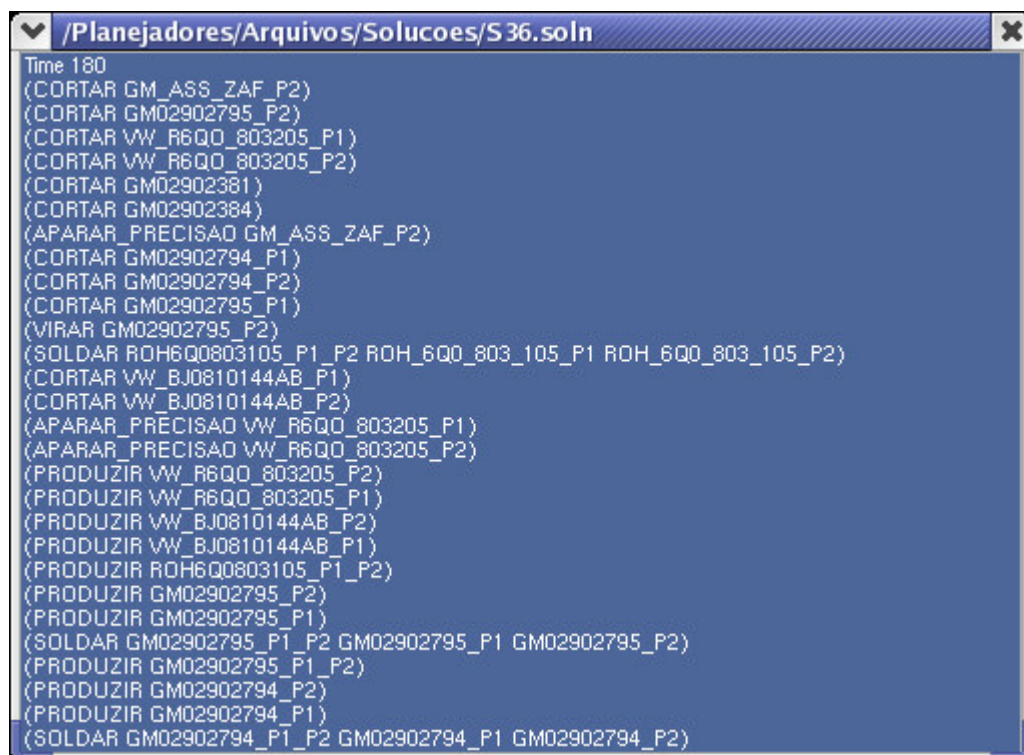


Figura 5-7 – Exemplo de plano de produção gerado

Comentários:

- Para “rolar” a tela de visualização do plano de produção, basta pressionar as teclas “PageUp” ou “PageDown” do teclado do computador, uma vez que o “ScrollLock” esteja ativo.
- O plano mostrado nesta tela inclui todas as ações necessárias para que os estoques ideais sejam atingidos na semana j.

O botão “Pós-processamento” na tela principal da interface gráfica, permite a visualização da solução gerada de forma agrupada. Ao se pressionar este botão, a tela para seleção dos planos gerados (Figura 5-6) é novamente exibida. Uma vez selecionado o plano gerado, uma nova tela, conforme mostrada na Figura 5-8, é aberta para a seleção de um dos seguintes tipos de agrupamento (visualização):

- Visualização por equipamento,
- Visualização por produto, ou
- Plano macro.



Figura 5-8 – Tela de pós-processamento

No caso da seleção de visualização por equipamento, a lista dos equipamentos considerados no arquivo de operadores é exibida, de modo que o plano de produção específico para cada equipamento possa ser visualizado.



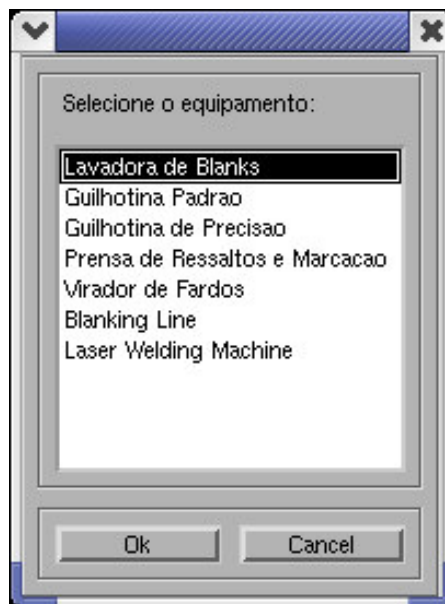


Figura 5-9 – Tela para seleção dos equipamentos

A Figura 5-9 mostra a tela para seleção do equipamento ao qual se deseja ver o plano de produção gerado.

- Uma vez selecionado o equipamento, o botão “OK” deve ser pressionado para visualização do plano de produção gerado. A Figura 5-10 mostra um exemplo de plano de produção para a Linha de Corte Transversal (*Blanking Line*).

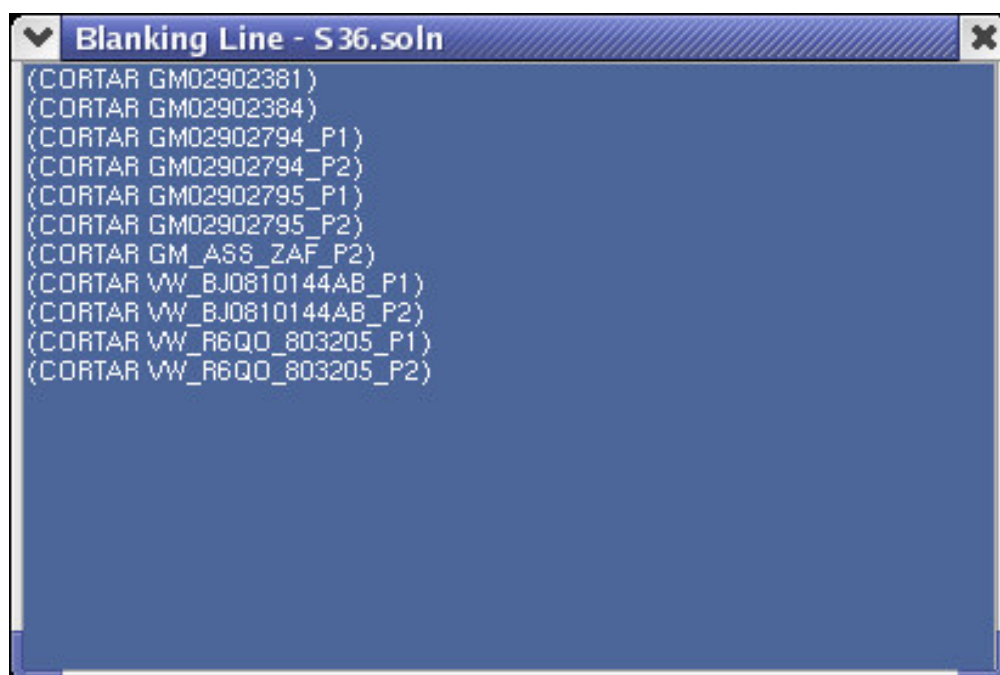


Figura 5-10 – Exemplo de plano de produção por equipamento

Comentários:

- No caso de não existir plano de produção para o equipamento selecionado, a seguinte mensagem é exibida: “Sem plano de produção para esse equipamento !”.

Caso seja selecionada a opção de visualização por produto na tela de pós-processamento (Figura 5-8), é aberta uma tela para seleção do produto desejado. Uma vez selecionado o produto, é possível ver as ações que devem ser executadas para que o produto selecionado seja produzido.

A Figura 5-11 mostra um exemplo de plano de produção selecionado para um determinado produto.

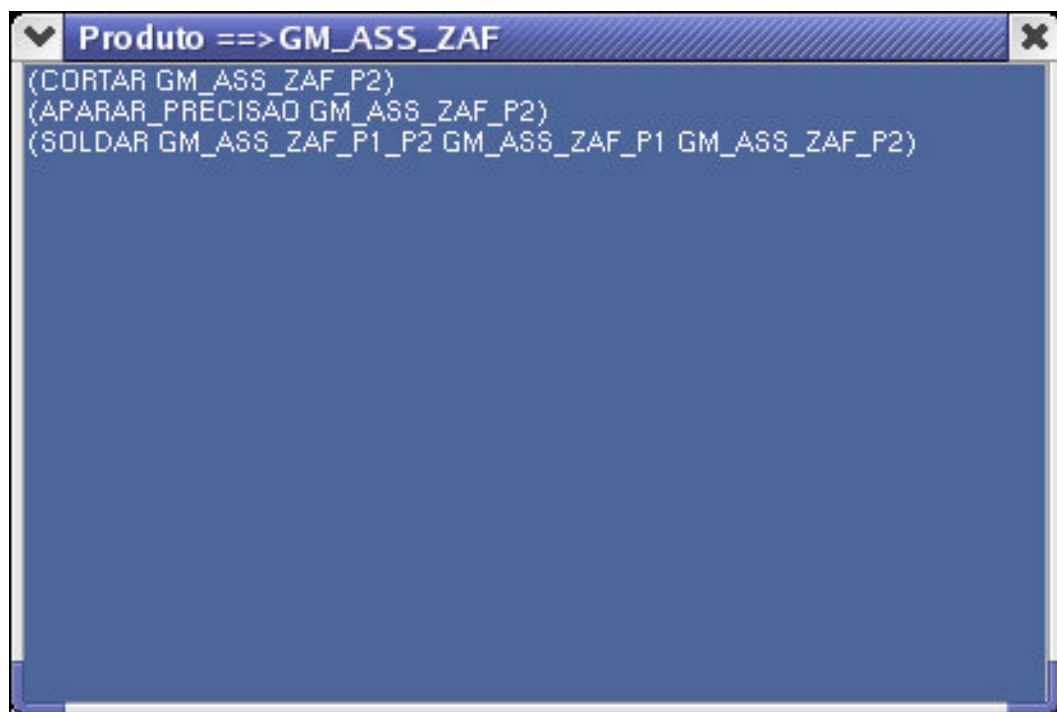


Figura 5-11 – Exemplo de plano de produção por produto

Comentários:

- No caso de não existir plano de produção para o produto selecionado, a seguinte mensagem é exibida: “Sem plano de produção para esse produto !”.

A última alternativa de visualização é feita a partir da seleção da opção “Plano macro” na tela de pós-processamento (Figura 5-8). Caso essa alternativa seja selecionada, uma tela contendo as ações <PRODUZIR> é exibida, de modo que se possa ter uma visão geral de quais produtos devem ser produzidos para que os níveis de estoque reais possam ser aumentados para os níveis considerados ideais no arquivo de fatos. A Figura 5-12 mostra um exemplo de um plano macro.

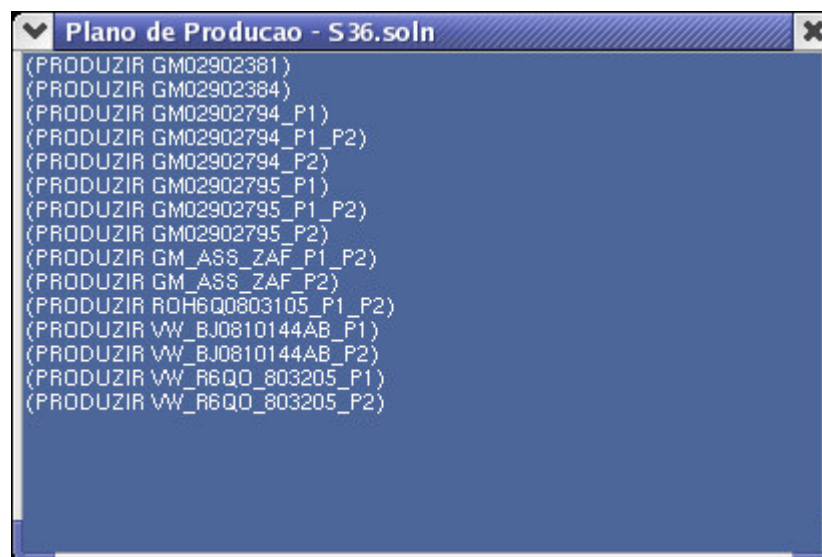


Figura 5-12 – Exemplo de um plano macro

Por esta tela podemos ter uma visão geral de quais produtos devem ser produzidos na semana 36.

## **Capítulo 6**

### **6. Resultados e conclusões**

Neste trabalho foi definido e implementado um agente planejador inteligente capaz de gerar um plano de ações que represente um plano de produção em um centro de serviços de corte e solda laser. Foi considerado um conjunto de 70 produtos diferentes, processados em 7 equipamentos distintos e respeitando os fluxos de produções reais. Os dados primários, utilizados para simulação do agente implementado, foram apurados durante 20 semanas consecutivas, no período da semana 36 de 2004 até a semana 3 de 2005, e também representam dados reais de produção.

A análise comparativa entre os planos gerados pelo agente planejador inteligente e o agente planejador humano, está descrita da seção 6.1 e comprova a eficácia do agente planejador implementado.

A interface gráfica desenvolvida, permitiu a visualização dos planos gerados de forma agrupada por produtos e por equipamentos, o que caracteriza a primeira contribuição do trabalho, uma vez que os planos gerados pelo agente planejador humano consideravam apenas o planejamento macro, ou seja, determinando quais produtos deveriam ser produzidos no período (semanas), mas não contemplava de forma detalhada, quais fluxos deveriam ser seguidos. Este acréscimo de informação nos programas de produção, permitiu uma melhor alocação dos recursos humanos

por equipamento e permitirá ainda, a identificação de “gargalos” de produção podendo auxiliar, no futuro, em uma tomada de decisão para investimentos em contratação de mão-de-obra ou aquisição de novos equipamentos.

## 6.1 Resultados obtidos

Para análise dos resultados, foi considerado como medida de desempenho do agente implementado, o nível de coincidência entre os planos gerados pelo agente planejador inteligente e o agente planejador humano. A análise através do nível de coincidência, consiste em comparar os produtos indicados para produção pelo agente planejador humano e os produtos indicados para produção nos planos de produção gerados pelo agente planejador inteligente.

Cada um dos planos gerados pelo agente planejador de produção foi pós-processado, de modo que os planos fossem separados por produtos, conforme descrito na Seção 4.9. Desta forma foi possível verificar quais produtos deveriam ser produzidos em cada uma das semanas, segundo o agente implementado.

Após o pós-processamento, as ações Produzir (Produto\_N) foram comparadas com as ordens de produção reais, geradas pelo agente planejador humano, e então verificado o nível de coincidência entre os planos. Para a realização da comparação, foram calculadas as relações percentuais de quais produtos deveriam ser produzidos, segundo o agente planejador de produção e que também foram recomendados para produção de acordo com agente planejador humano.

O objetivo desta comparação, foi de verificar se o agente planejador de produção foi capaz de gerar planos que realmente poderiam manter os estoque de cada produto, dentro dos níveis considerados ideais e assim garantir a continuidade operacional dos clientes, sem causar paradas de produção por problemas de desabastecimento das peças.

No período de comparação foi das 17 semanas consecutivas (semana 36 de 2004 até a semana 52 de 2004), o índice de coincidência médio superou 94% [36]. A Tabela 6-1 mostra os resultados obtidos pelas comparações efetuadas, indicando para cada um dos planos gerados pelo agente planejador de produção, quantos planos coincidiram com os gerados pelo agente planejador humano. Os mesmos resultados são apresentados em forma de gráfico, no Gráfico 6-1.

Plano	Produtos planejados pelo agente humano	Produtos recomendados pelo a agente planejador	Nível de coincidência
1	15	14	93,3%
2	18	17	94,4%
3	19	18	94,7%
4	20	19	95,0%
5	15	14	93,3%
6	11	10	90,9%
7	15	15	100,0%
8	20	19	95,0%
9	22	21	95,5%
10	22	22	100,0%
11	21	20	95,2%
12	18	17	94,4%
13	13	13	100,0%
14	11	10	90,9%
15	8	7	87,5%
16	9	8	88,9%
17	14	13	92,9%
		<b>Média →</b>	<b>94,2%</b>

Tabela 6-1 – Nível de coincidência entre o agente planejador humano e o agente planejador de produção

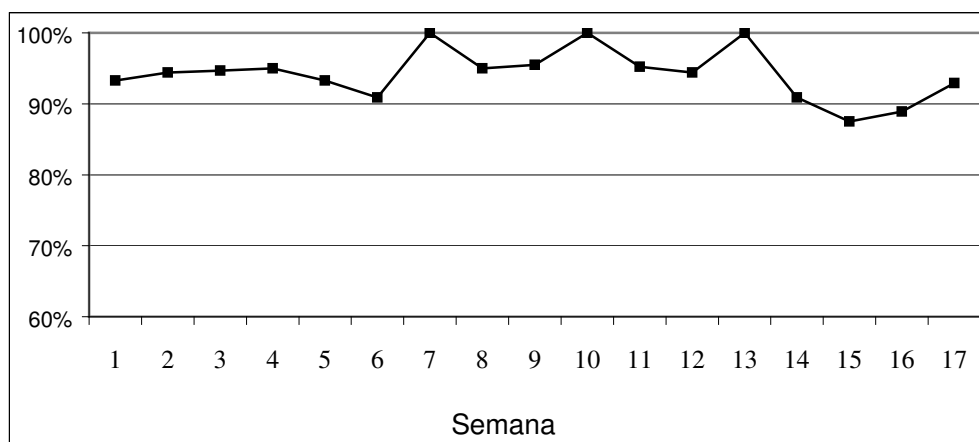


Gráfico 6-1 – Nível de coincidência entre o agente planejador humano e o agente planejador de produção

As divergências encontradas entre os planos foram identificadas como conveniência de programação em função de solicitações de priorização de fornecimento solicitadas pelos próprios clientes ou ainda por contingências em função da indisponibilidade de equipamentos.

Como extensão da análise, foi feita a comparação dos planos, considerando períodos de duas semanas, ou seja, as ações de produção geradas pelo agente planejador inteligente para uma determinada semana, foram comparadas com os planos de produção gerados pelo agente planejador humano naquela semana e na semana subsequente. Neste caso, o índice de coincidência foi de 100% para as 17 semanas avaliadas.

A análise comparativa dos planos permitiu concluir que o agente planejador inteligente foi capaz de gerar planos de produção consistentes, sendo capaz de manter os estoques nos níveis considerados ideais, atendendo assim ao objetivo proposto neste trabalho.

Um outro resultado interessante, e que pela análise do nível de coincidência, pode-se perceber que das 17 semanas avaliadas, o agente planejador humano programou um produto a mais por 14 delas. Esta variação pode ser causada por uma antecipação da produção real, permitindo que produtos programados para a semana subsequente fossem antecipados, ou ainda, pelo motivo do agente planejador humano tender a ser mais conservador, programando produções desnecessária, de modo a reforçar a garantia do atendimento de prazos de entrega dos produtos aos clientes.

As seções seguintes apresentam as propostas de extensão ao atual projeto.

## **6.2 Perspectivas futuras**

### **6.2.1 Restringir as pré-condições dos operadores**

O agente planejador inteligente tem como pré-condições dos operadores possíveis de serem executadas no domínio de planejamento, apenas predicados que representam o processamento em um equipamento prévio, que permite o andamento necessário dentro do fluxo de produção de um determinado produto.

As pré-condições dos operadores podem ser restringidas, com a inclusão de predicados que considerem a disponibilidade de matéria-prima, componentes de embalagem (p.ex o modelo de *pallet* necessário para acondicionamento de um determinado produto) ou ainda a disponibilidade dos equipamentos. Por exemplo, no domínio “Centro de Serviços” , a pré-condição que permite que a ação RESSALTAR seja executada, é  $(soldado\ ?p)$ . A proposta é que a pré-condição seja composta pela conjunção de fórmulas atômicas, conforme abaixo:

$$(soldado\ ?p) \wedge (pallet\ ?n\ ?p) \wedge (disponível\ ?e)$$

onde  $?n$  representa o modelo do *pallet* utilizado para acondicionar o produto  $?p$  e  $?e$  o equipamento onde o produto  $?p$  deve ser processado, neste caso, a própria prensa de ressaltos.

## 6.2.2 Interface gráfica mais eficiente

A interface gráfica desenvolvida, é capaz de gerar os problemas automaticamente, a partir da leitura dos arquivos de estoques reais e demanda dos clientes. Como extensão deste trabalho, a interface gráfica pode ser melhorada para que seja possível o cadastramento de novas peças e também de novos fluxos de produção que por ventura possam surgir.

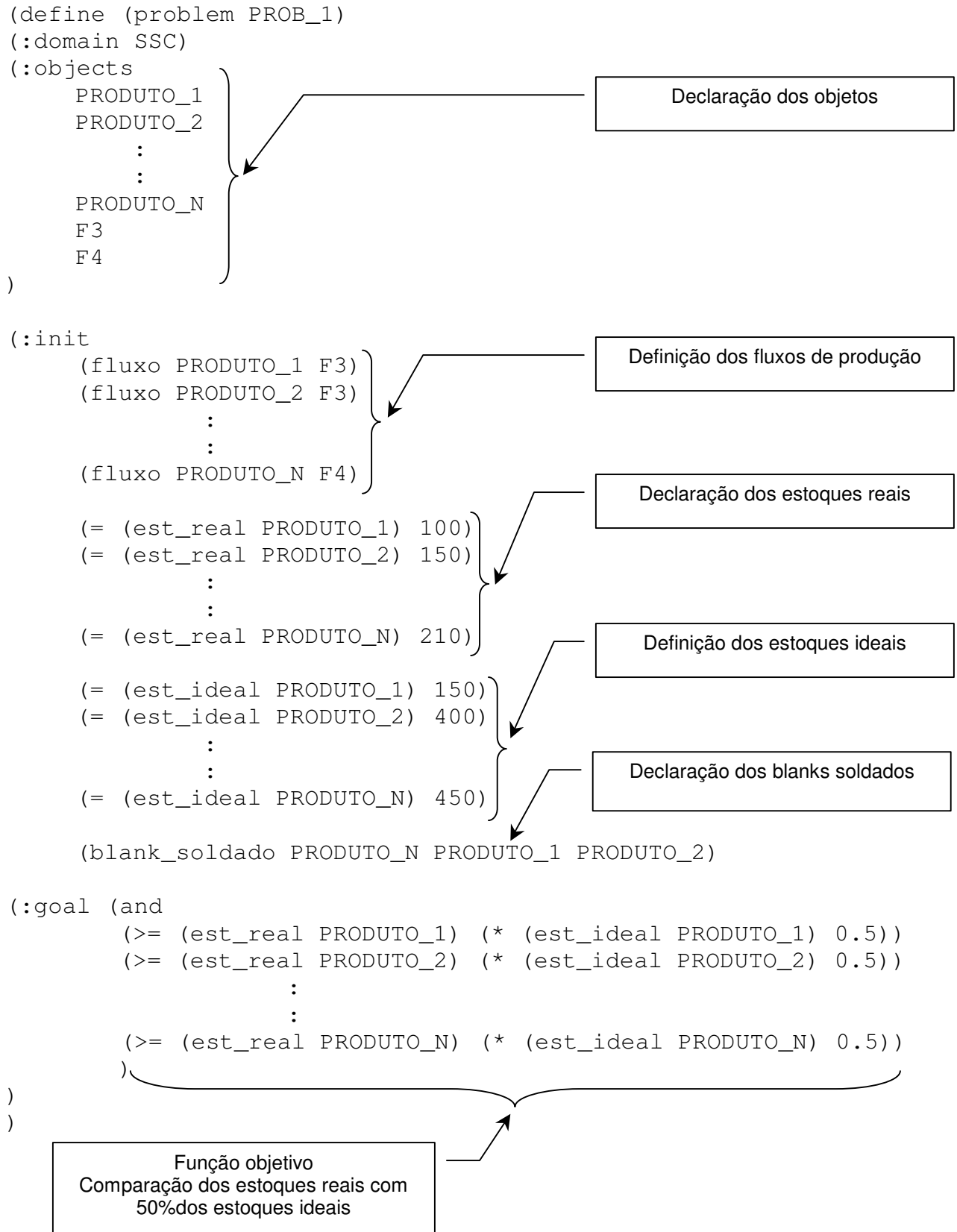
## 6.2.3 Instanciações numéricas pelo agente planejador

Conforme descrito na Seção 3.6.1, o planejador Metric-FF é capaz de lidar com variáveis numéricas, para solução de problemas com representação de domínios na linguagem PDDL 2.1, onde operadores aritméticos (+, -, / e \*) e operadores de comparação (=, >, >=, < e <=) são utilizados. Embora o planejador seja capaz de executar cálculos e comparações, não é possível executar instanciações de variáveis com valores numéricos. Desta forma, as ações descritas nos planos gerados, apenas indicam quais produtos devem ser produzidos.



Com o desenvolvimento de um planejador que permita instanciações de variáveis numéricas, será possível gerar planos de ações que indiquem, além de quais produtos devem ser produzidos, quantas unidades devem ser produzidas.

## ANEXO A - Exemplo de definição de problema



**ANEXO B - Domínio de planejamento**

```

(define (domain SSC)

  (:requirements :typing :fluents)

  (:predicates
    (cortado ?p)
    (soldado ?p)
    (virado ?p)
    (ressaltado ?p)
    (aparado ?p)
    (apara_simples ?p)
    (lavado ?p)
    (blank_soldado ?p ?p1 ?p2)
    (fluxo ?p ?f))

  (:functions
    (est_real ?p)
    (est_ideal ?p))

;----- AÇÕES -----

(:action PRODUZIR
  :parameters (?p)
  :precondition (or (and (cortado ?p) (fluxo ?p F3))
    (and (soldado ?p) (fluxo ?p F4))
    (and (aparado ?p) (fluxo ?p F5))
    (and (ressaltado ?p) (fluxo ?p F6))
    (and (apara_simples ?p) (fluxo ?p F7))
    (and (virado ?p) (fluxo ?p F8))
    (and (virado ?p) (fluxo ?p F9))
    (and (lavado ?p) (fluxo ?p F10))
    (and (lavado ?p) (fluxo ?p F11)))
  :effect (increase (est_real ?p) (- (est_ideal ?p) (est_real
?p))))

)

(:action CORTAR
  :parameters (?p)
  :precondition (< (est_real ?p) (est_ideal ?p))
  :effect (cortado ?p)
)

```

```
(:action SOLDAR
:parameters (?p ?p1 ?p2)
:precondition (and (blank_soldado ?p ?p1 ?p2)
                  (>= (est_real ?p1) (est_ideal ?p1))
                  (>= (est_real ?p2) (est_ideal ?p2))
                )
:effect (soldado ?p)
)

(:action VIRAR
:parameters (?p)
:precondition (cortado ?p)
:effect (virado ?p)
)

(:action RESSALTAR
:parameters (?p)
:precondition (soldado ?p)
:effect (ressaltado ?p)
)

(:action APARAR_PRECISAO
:parameters (?p)
:precondition (cortado ?p)
:effect (aparado ?p)
)

(:action APARAR_SIMPLES
:parameters (?p)
:precondition (cortado ?p)
:effect (apara_simples ?p)
)

(:action LAVAR
:parameters (?p)
:precondition (or (and (fluxo ?p F10) (virado ?p))
                 (and (fluxo ?p F11) (cortado ?p)))
:effect (lavado ?p)
)
)
```

## ANEXO C - Alteração no código fonte do Metric-FF

Para que a solução gerada pelo planejador possa ser salva em um arquivo, que será utilizado pela interface gráfica, é necessário que a rotina “output\_planner\_info” do arquivo “main.c” seja substituída pela rotina abaixo. Após a substituição, é necessário que o programa seja recompilado, a partir do comando “make ff” na janela de comando do Linux Red Hat 9.0.

```
void output_planner_info( void )

{
    printf( "\n\ntime spent: %7.2f seconds instantiating %d easy, %d hard
action templates",
           gtempl_time, gnum_easy_templates, gnum_hard_mixed_operators );
    printf( "\n          %7.2f seconds reachability analysis, yielding
%d facts and %d actions",
           greach_time, gnum_pp_facts, gnum_actions );
    printf( "\n          %7.2f seconds creating final representation
with %d relevant facts, %d relevant fluents",
           grelev_time, gnum_relevant_facts, gnum_relevant_fluents );
    printf( "\n          %7.2f seconds computing LNF",
           gLNF_time );
    printf( "\n          %7.2f seconds building connectivity graph",
           gconn_time );
    printf( "\n          %7.2f seconds searching, evaluating %d states,
to a max depth of %d",
           gsearch_time, gevaluated_states, gmax_search_depth );
    printf( "\n          %7.2f seconds total time",
           gtempl_time + greach_time + grelev_time + gLNF_time + gconn_time +
gsearch_time );

    printf("\n\n");

    if ( lfound_plan ) {
        print_official_result();
    }
    exit( 0 );
}

FILE *out;
void print_official_result( void )
{
    int i;
    char name[MAX_LENGTH];
```

```
printf( name, "%s.soln", gcmd_line.fct_file_name );
if ( (out = fopen( name, "w")) == NULL ) {
    printf("\n\nCan't open official output file!\n\n");
    return;
}
times( &lend );
fprintf(out, "Time %d\n",
        (int) ((lend.tms_utime - lstart.tms_utime + lend.tms_stime -
lstart.tms_stime) * 10.0));
for ( i = 0; i < gnum_plan_ops; i++ ) {
    print_official_op_name( gplan_ops[i] );
    fprintf(out, "\n");
}
fclose( out );
}
void print_official_op_name( int index )
{
    int i;
    Action *a = gop_conn[index].action;
    if ( a->norm_operator ||
        a->pseudo_action ) {
        fprintf(out, "(%s", a->name );
        for ( i = 0; i < a->num_name_vars; i++ ) {
            fprintf(out, " %s", gconstants[a->name_inst_table[i]]);
        }
        fprintf(out, ")");
    }
}
```

## Referências

- [1] N. Slack, S. Chambers and R. Johnston, *Administração da Produção*. Ed. Atlas, 2002
- [2] F. A. Serra, M. C. S. Torres and A. P. Torres, *Administração Estratégica*. Reichmann & Affonso Editores - Rio de Janeiro, 2003
- [3] Hitachi Reseach Group, *The Management of Job-Shop Scheduling Constraints*, (AIAI Technical Report N° 121) NSF Dynamic Scheduling Workshop Paper, Jan 1993.
- [4] M. S. Fox and S. F. Smith, *ISIS: a knowledge-based system for factory scheduling*. Expert Systems, 1(1):25-49 (1984).
- [5] H. Kerzner, *Project Management - A Systems Approach to Planning, Scheduling and Controlling* (1979)
- [6] J. F. Allen, J. A. Hendler, and A. Tate, *Readings in planning*. San Mateo, Morgan Kaufmann, 1990.
- [7] R. A. C. Bianchi, M. V. T Santos and M. Rillo, *Integração de um sistema de planejamento de atividades numa célula de montagem robotizada*. SBAI, 1995.

- [8] R. Fikes and N. Nilsson, *STRIPS: A new approach to the application of theorem proving to problem solving*. *Journal of Artificial Intelligence*, 2(3-4), 1971.
- [9] R. Reiter, *On closed world data bases*. In H. Gallaire and J. Minker, editors. *Logic and data bases*. Plenum press, 1978.
- [10] J. Mc Carthy and P. Hayes. *Some philosophical problems from the standpoint of artificial intelligence*. *Machine Intelligence* 4, 1969.
- [11] E. V. N. Lorenci, *Definição e implementação de uma arquitetura deliberativa para um agente inteligente robótico*, Tese de mestrado, UNIFEI, 2004.
- [12] S. Russel and P. Norvig, *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, 1995.
- [13] G. Sussmann, *A computer model of skill acquisition*. Elsevier, North-Holland, 1975.
- [14] E. D. Sacerdoti, *The nonlinear nature of plans*. *IJCAI-75*, 206, 1975.
- [15] A. Tate, *Generating project networks*. *Proceedings of the international joint conference on artificial intelligence*, 3, 1977.
- [16] D. Chapman, *Planning for conjunctive goals*. *Artificial intelligence*, 32(3), 333, 1987.
- [17] E. P. D. Pednault, *ADL: Exploring the middle ground between strips and the situation calculus*. *Proceedings of the knowledge representation conference*, 1989.



- [18] J. S. Penbethy and D.S. Weld, *UCPOP: A sound, complete, partial order plan for ADL*. Proceedings of course 3 of the international conference on principles of knowledge representation and reasoning, 1992.
- [19] M. M. Veloso, J. Carbonell, M. Alicia Pérez, D. Borrajo, E. Fink and J. Blythe, *Integrating planning and learning: The PRODIGY architecture*. Journal of experimental and theoretical artificial intelligence 7, 81 (1995).
- [20] D. Mc Dermott, *PDDL – The planning domain definition language*. Yale University, USA, 1998.
- [21] D. Mc Dermott, *The 1998 AI planning systems competition*, AI magazine, 21 (2), 2000.
- [22] M. Fox and D. Long, *The 3<sup>rd</sup> International Planning Competition: results and analysis*. Journal of Artificial Intelligence Research, 20, 1, 2003.
- [23] M. Fox and D. Long, *PDDL 2.1: An extension to PDDL for expressing temporal planning domains*. Journal of artificial intelligence research, 20-61, 2003.
- [24] M. Fox and D. Long, *The 3<sup>rd</sup> International Planning Competition: temporal and metric planning*. Proceedings of course of The International Conference on AI Planning and Scheduling, Toulouse, França, 2002.
- [25] J. J. Neto, *Introdução a Compilação*, Rio de Janeiro, Livros Técnico Científicos, 1987.
- [26] J. Hoffmann, *The Metric-FF planning system: Translating “ignoring delete list” to numeric state variables*. Journal of artificial intelligence research, 20, 291-341, (2003).

- [27] J. Hoffmann and B. Nebel, *The FF planning system: fast plan generation through heuristic search*. Journal of Artificial Intelligence, 14(3), 237, 2001.
- [28] A. Gerevini, A. Saetti, and I. Serina, *Planning through stochastic local search and temporal action graphs in LPG*. Journal of artificial intelligence reaserch. Special issue on the 3th international planning competition, 2003.
- [29] J. Rintanen and J. Hoffmann, *An overview of recent algorithms for AI planning*. Künstliche Intelligenz, 2, 5, 2001.
- [30] I. Refanidis and I. Vlahavas, *The GRT planner*, AI-magazine, outono, 2001.
- [31] P. E. Hart, N. J. Nilsson and B. Raphael. *Correction to "A formal basis for the heuristic determination of minimum cost paths*. IEEE Transactions on systems science and cybernetics, SSC-4(2):100-107, (1968).
- [32] A. Blum and M. Furst, *Fast planning through planning graph analysis*, Proceedings of course 14 of The International Joint Conference on Artificial Intelligence, 1995.
- [33] <http://www.informatik.uni-freiburg.de/~hofmann/ff.html>, acessado em 14 de novembro de 2005.
- [34] J. Hoffmann, *Extending FF to numerical state variables*. Proceedings of course 15 of The European Conference on Artificial Intelligence, Lyon, France, 2002.
- [35] S. Franklin and A. Graesser, *Is it an agent, or just a program?: A taxonomy for autonomous agents*, Proceedings of the 3<sup>rd</sup> international workshop on agent theoretics, architectures and languages, Springer – Verlag, 1996.
- [36] J. Barg e L. E. Souza, *Agente planejador inteligente para planejamento de produção em um centro de serviços de corte de aço*, VII SBAI, 2005.