CENTRO UNIVERSITÁRIO DA FEI

MARCELO FERREIRA

MODELAGEM DE DOMÍNIOS TEMPORAIS DE PLANEJAMENTO COM UML.P

São Bernardo do Campo

2009

MARCELO FERREIRA

MODELAGEM DE DOMÍNIOS TEMPORAIS DE PLANEJAMENTO COM UML.P

Dissertação de Mestrado apresentada ao Centro Universitário da FEI para obtenção do título de Mestre em Engenharia Elétrica.

Área de Concentração:
Inteligência Artificial Aplicada à Automação

Orientador:
Prof. Dr. Flavio Tonidandel

São Bernardo do Campo
2009

Ferreira, Marcelo

Modelagem de domínios temporais de planejamento com UML.P. / Marcelo Ferreira. – São Bernardo do Campo, 2009.

120 p.

Dissertação de Mestrado – Centro Universitário da FEI.
Orientador: Prof. Dr. Flavio Tonidandel.

1. Inteligência Artificial. 2. Engenharia do Conhecimento. I. Centro Universitário da FEI. II. Título.

Ao meu orientador Prof. Dr. Flavio Tonidandel e a todos os meus amigos, especialmente ao Marcelo Udo e aos meus familiares, especialmente à minha esposa Adriana, sempre presente em todos os momentos.

AGRADECIMENTOS

Primeiramente, agradeço a Deus pela força concedida e pela maravilhosa vida de aprendizado e amizade.

Agradeço ao meu orientador e professor Dr. Flavio Tonidandel, que com sua orientação e aconselhamento, tornou o desenvolvimento deste trabalho uma grande fonte de aprendizado.

Agradeço ao professor Dr. Paulo Sérgio e a professora Dra. Leliane Barros que participaram de minha qualificação e contribuíram significativamente com suas observações.

À minha esposa Adriana pelo apoio e paciência durante todo o desenvolvimento deste trabalho e à minha irmã Helaine e minha adorada mãe Dona Lourdes.

Ao meu amigo Marcelo Udo, que sempre compartilhou idéias e me apoiou muito ao longo de vários anos de amizade e ao amigo de trabalho Alessandre Galvão que também me apoiou no período de desenvolvimento deste trabalho.

A todos, meu sincero obrigado!

RESUMO

A necessidade de tratar domínios reais impulsionou fortemente o desenvolvimento dos sistemas de planejamento automático. Contudo, apesar deste avanço, existem lacunas entre o que se pode representar no mundo real e o que realmente pode ser interpretado pelos sistemas de planejamento. Uma destas lacunas é a modelagem de domínios temporais de planejamento. Para tratá-la, este trabalho apresenta uma proposta de uso da linguagem de modelagem UML (*Unified Modeling Language*) através do seu diagrama de tempo, que fora agregado como uma extensão à UML.P (*Unified Modeling Language in a Planning Approach)*. Com o resultado obtido, é possível modelar o contexto temporal do domínio, onde todos os elementos que participam do ciclo de vida estão presentes em um diagrama. O contexto temporal de uma ação é modelado a partir dos objetos e atributos que fazem parte do ciclo de vida da ação e a representação é realizada através do diagrama de tempo proposto.

Palavras-chave: Planejamento automático; Engenharia do Conhecimento; UML.P.

ABSTRACT

The need to treat real domains impelled the development of the automated planning systems strongly. However, in spite of this progress, there are some gaps between what can be represented in the real world and what can really be interpreted by the planning systems. One of these gaps is the modeling of temporal domains of planning. To treat it, this work presented a proposal of use of the modeling language UML (Unified Modeling Language) through its diagram of time that had been joined as an extension to UML.P (Unified Modeling Language in Planning Approach). With the obtained result, it is possible to model the temporary context of the domain, where all of the elements that participate in the life cycle are present in the diagram and the temporary context of an action, with the modeling of all of the objects and attributes of the action during its execution.

Keywords: Planning; Knowledge Engineering, UML.P.

SUMÁRIO

CAPÍTULO I 1
1. Introdução 1
1.1. Objetivos 3
1.2. Organização do Trabalho 3
CAPÍTULO II 4
2. Planejamento Temporal 4
2.1. Planejamento Automático 4
2.2. Tempo para planejamento 10
2.2.1. Instantes e intervalos 11
2.2.2. Álgebra de Ponto 13
2.2.3. Álgebra de Intervalo 16
2.3. Planejamento temporal 18
2.3.1. Planejamento com operadores temporais 19
2.3.2. Exemplo: Tratamento térmico de metais 19
2.3.3. Base e operador temporal 20
2.4. Domínios e Procedimentos Temporais de Planejamento 23
2.5. Resumo e conclusão do capítulo 24
CAPÍTULO III 25
3. Engenharia do Conhecimento para planejamento temporal 25
3.1. Engenharia do conhecimento 25
3.2. Representação do conhecimento 26
3.3. PDDL 27
3.3.1. Definição de domínios temporais em PDDL 2.1 28
3.3.2. PDDL para domínios temporais 32
3.3.3. Definição de ações temporais 35
3.4. UML 38
3.4.1. Diagrama de Casos de Uso 39
3.4.2. Diagrama de Classes 40
3.4.3. Diagrama de máquina de estados 42
3.4.4. Diagrama de tempo 43
3.4.5. Diagrama de Objetos 45
3.4.6. OCL 45
3.5. UML.P 47
3.6. Resumo e conclusão do capítulo 49
CAPÍTULO IV 50
4. Domínios temporais de planejamento com UML.P 50
4.1. Introdução 50
4.2. Representação de domínios temporais 51
4.3. Processo de definição de domínios 52
4.4. Elementos de um domínio temporal 55
4.5. O Diagrama de tempo da UML.P 56
4.6. Representação de ações temporais 61
4.6.1. Variação numérica contínua ao longo do tempo 65
4.6.2. Ações durativas de efeito contínuo 65
4.7. Tempo em estados ramificados 66
4.8. Abrangência do diagrama de tempo da UML.P 70
4.9. Interpretação do diagrama de tempo da UML.P em PDDL 2.1 79

4.9.1. Mapeamento para requerimento (:requirements) 81
4.9.2. Mapeamento para definição de ações com duração (:durative-action) 82
4.9.3. Mapeamento para definição da duração da ação (:duration) 82
4.9.4. Mapeamento para definição das condições (:condition)..... 84
4.9.5. Mapeamento para definição dos efeitos (:effects)..... 86
4.10. Resumo e conclusão do capítulo 87
CAPÍTULO V 89
5. Estudo de Caso..... 89
5.1. Exemplo: Tratamento térmico de metais..... 89
5.2. Resumo e conclusão do capítulo 112
CAPÍTULO VI 113
6. Conclusão e trabalhos futuros..... 113
6.1. Conclusão..... 113
6.2. Trabalhos futuros..... 114
BIBLIOGRAFIA 115

LISTA DE TABELAS

Tabela 2.1 - Relações de intervalo definida por pontos finais 11
Tabela 2.2 - Intervalos temporais de Allen (continua) 12
Tabela 2.3 - Tabela de composição para álgebra de ponto 14
Tabela 2.4 – Composição parcial para álgebra de intervalo 17
Tabela 4.1 Elementos do diagrama de tempo da UML.P 59
Tabela 4.2 Elementos para notação do diagrama para ações temporais na UML.P..... 63
Tabela 4.3 interpretação de aspectos temporais da PDDL e UML.P para efeitos de uma ação (continua) 86
Tabela 5.1 Tabela de casos de uso do domínio de tratamento térmico..... 90
Tabela 5.2 Descrição dos atributos e métodos das classes modeladas no diagrama de classes96

LISTA DE ILUSTRAÇÕES

Figura 2.1 - Exemplo de representação STRIPS 6
Figura 2.2 Estado após aplicação da ação Retirar 7
Figura 2.3 – Composição e conjunção transitiva..... 15
Figura 2.4 – Efeitos da composição de dois intervalos..... 18
Figura 2.5 – Exemplo de base de dados temporal para o domínio de tratamento térmico 20
Figura 2.6 – Exemplo de operador temporal de planejamento: domínio de tratamento térmico. 22
Figura 3.1 Ação durativa do domínio de exemplo Tratamento Térmico de Metais..... 28
Figura 3.2 Momentos em que as condições e efeitos são considerados em uma ação durativa na PDDL..... 29
Figura 3.3 Ação durativa discreta do domínio de exemplo Tratamento Térmico de Metais... 30
Figura 3.4 Ação durativa de efeito contínuo do domínio de tratamento térmico..... 32
Figura 3.5 Especificação de domínios em PDDL 2.1..... 33
Figura 3.6 Estrutura da ação temporal em PDDL 2.1..... 36
Figura 3.7 Definição BNF da duração de uma ação temporal em PDDL..... 36
Figura 3.8 Definição BNF dos qualificadores temporais em PDDL. 37
Figura 3.9 Definição BNF dos efeitos de uma ação temporal em PDDL. 37
Figura 3.10 Definição BNF dos qualificadores temporais em PDDL. 38
Figura 3.11 Aplicação da UML nas fases da modelagem de domínios..... 39
Figura 3.12 – Exemplo de Diagrama de Caso de Uso 40
Figura 3.13 Exemplo de diagrama de classes..... 41
Figura 3.14 Diagrama de máquina de estados..... 43
Figura 3.15 Elementos do Diagrama de tempo da UML 44
Figura 3.16 Exemplo de objetos em um diagrama 45
Figura 3.17 Esquema geral para um ambiente de planejamento 47
Figura 4.1 Fases no processo de definição de domínios de planejamento..... 53
Figura 4.2 Seqüência de definição de um domínio em UML.P 57
Figura 4.3 Diagrama de tempo da UML.P para os objetos do tipo Classe1 e Classe3..... 60
Figura 4.4 Diagrama para representação de ações temporais 64
Figura 4.5 Diagrama para representação de ações temporais com variação numérica contínua 65
Figura 4.6 Diagrama para representação de ações durativas de efeito contínuo..... 66
Figura 4.7 Exemplo de diagrama de máquina de estados e respectiva árvore de computação 67
Figura 4.8 Diagrama de tempo correspondente a uma ramificação de estados 68
Figura 4.9 Diagrama de tempo correspondente a várias ramificações de estados 69
Figura 4.10 Diagrama de tempo para representação da primeira/segunda relação temporal de Allen..... 72
Figura 4.11 Terceira relação temporal de Allen representada em um diagrama de tempo da UML.P..... 73
Figura 4.12 Quarta/quinta relação temporal de Allen representada no diagrama de tempo da UML.P..... 74
Figura 4.13 Sexta/sétima relação temporal de Allen representada no diagrama de tempo da UML.P..... 75
Figura 4.14 Oitava/Nona relação temporal de Allen representada pelo diagrama de tempo da UML.P..... 76
Figura 4.15 Décima/décima primeira relação temporal de Allen representada em UML.P 77

Figura 4.16 Décima segunda/décima terceira relação temporal de Allen representada em UML.P 78
Figura 4.17 Seqüência de modelagem em UML.P 80
Figura 4.18 Possibilidades para determinação da restrição de duração em PDDL. 83
Figura 4.19 Ilustração do mecanismo de interpretação para definição da duração de uma ação 84
Figura 5.1 Diagrama de caso de uso para domínio de tratamento térmico 89
Figura 5.2 Diagrama de classes do domínio de tratamento térmico de metais 96
Figura 5.3 Diagrama de máquina de estados do objeto peça 99
Figura 5.4 Diagramas de máquinas de estados para o objeto forno 100
Figura 5.5 Diagrama de máquina de estados do objeto recipiente 101
Figura 5.6 Diagrama de tempo para contexto global do domínio 102
Figura 5.7 Diagrama de tempo para ação temporal prepararParaTempera 104
Figura 5.8 Duração da ação prepararParaTempera definida em expressão e em OCL 105
Figura 5.9 Ação prepararParaTempera interpretada em PDDL 2.1 106
Figura 5.10 Diagrama de tempo para a ação temperar 107
Figura 5.11 Duração da ação temperar definida em expressão e em OCL 108
Figura 5.12 Ação temperar interpretada em PDDL 2.1 109
Figura 5.13 Diagrama de tempo para a ação prepararParaRevenimento 110
Figura 5.14 Diagrama de tempo para a ação prepararParaRevenimento 111

LISTA DE SÍMBOLOS

$\neg$ Negação em lógica de primeira ordem

^ Operador lógico ‘e’ em lógica de primeira ordem

γ Função de transição de estados

Σ Sistema de transição de estados

$\bullet$ Operador de composição da álgebra de ponto e álgebra de intervalo

$\varnothing$ Restrição vazia que não pode ser satisfeita em álgebra de ponto

ζ Constante ou variável de objeto em uma expressão temporalmente qualificada

@ Separador de variável de objeto e instantes em uma expressão temporalmente qualificada

Φ Base temporal

θ Conjunto de precondições de uma ação temporal

σ Equivalência dada pela substituição de um conjunto de expressões temporalmente qualificadas

ρ Conjunto de restrições de um objeto

Λ_{Φ} Conjunto de bases temporais

#t Variação contínua de valor ao longo do tempo na PDDL

α Relação de mapeamento de objetos

β Relação de mapeamento de ações

δ Relação de mapeamento de estados

π Plano

$\Rightarrow$ Operador condicional ‘se’ da lógica proposicional

CAPÍTULO I

1. Introdução

Planejamento Automático (PA) é uma área da Inteligência Artificial (IA) que estuda o processo automatizado de seqüenciamento e escolha de ações através de recursos computacionais. Um dos propósitos do PA é desenvolver ferramentas capazes de prover um planejamento otimizado e eficiente dos recursos diante das restrições impostas pelo ambiente (GHALLAB *et al*., 2004).

Um sistema de planejamento (ou planejador) utiliza um modelo conceitual para descrever os principais elementos de um problema a ser solucionado. Este modelo define um estado inicial, um estado objetivo e um conjunto de ações que modificam o mundo. A solução para o problema é uma combinação de ações que, quando aplicadas, transformam o estado inicial em um estado cujo objetivo esteja satisfeito. Todos os estados possíveis do mundo estão disponíveis em um espaço de estados e cada estado é definido por um conjunto de fórmulas atômicas. O planejador realiza uma busca neste espaço de estados para obter uma seqüência de ações que correspondam à solução. Tal seqüência é denominada plano.

A evolução dos planejadores possibilitou identificar abordagens específicas para alguns tipos de problemas. Desta forma, cada planejador possuía sua própria linguagem para especificar domínios e isto dificultava avaliar e identificar qual planejador era mais adequado para um determinado tipo de problema. Então, em 1998, foi criada pelo comitê internacional de competição de planejamento (*International Planning Competition* – AIPS 98) a PDDL (*Planning Domain Definition Language*) (McDERMOTT, 1998). O objetivo foi criar uma linguagem de definição de domínios de planejamento com o propósito de padronizar a especificação de domínios e, com isto, comparar o desempenho dos planejadores em diferentes tipos de problemas.

Mesmo com a possibilidade de tornar as técnicas de planejamento mais acessíveis e usuais, a PDDL foi criticada pela comunidade acadêmica por não ser uma linguagem formal. McCluskey (2003) aponta falhas na PDDL por ela não estar associada a um método de construção de modelos e possuir falhas na especificação de estados, ações e objetos. Além destas questões estruturais, são apontadas falhas na especificação de domínios reais pela falta de características que possibilitam estruturar os objetos e seus estados e, também, por ser baseada em LISP (*List Processing Language)* (McCARTHY,1960), a PDDL possui

características que a aproximam de uma linguagem de programação, o que dificulta a construção de modelos a partir dos domínios de planejamento.

Com o aumento da complexidade dos domínios, há uma tendência em manter uma separação lógica entre o mecanismo de planejamento e o conhecimento do domínio, pois ainda há problemas em adquirir, representar, construir, validar conhecimento e solucionar questões sobre quais elementos podem ser representados e como esta representação pode ser expressa (McCLUSKEY; SIMPSON, 2004). A partir destas necessidades, surgem ferramentas de engenharia do conhecimento que permitem representar os elementos, os aspectos estáticos e dinâmicos de um domínio sem a necessidade de especificar o problema em PDDL.

O GIPO (*Graphical Interface for Planning With Objects)* (SIMPSON *et al.*, 2001) é uma das ferramentas de engenharia do conhecimento pioneiras que combinam as técnicas de aquisição de conhecimento e planejamento utilizando uma abordagem orientada a objetos. O itSIMPLE (*Integrated Tool Software Interface for Planning Environment*) (VAQUERO *et al*, 2005) é uma ferramenta de engenharia do conhecimento também orientada a objetos e com uma linguagem baseada na UML (*Unified Modeling Language*) (OMG, 2003) denominada UML.P (UML com abordagem em planejamento automático) (VAQUERO *et al*, 2006).

Apesar do avanço obtido com o surgimento destas ferramentas, uma questão fundamental na modelagem de domínios reais de planejamento ainda não está devidamente tratada: a consideração dos aspectos temporais intrínsecos aos domínios de planejamento (CUSHING, *et al,2007)*. Em contrapartida, os estudos acerca do tratamento das questões temporais para os planejadores tiveram avanços expressivos a partir da Competição Internacional de Planejamento de 2002 (IPC – *International Planning Competition*), que incluiu uma linha de pesquisa específica para planejamento temporal (CUSHING, *et al,2007b)* e, com isto, novas abordagens surgiram para o tratamento das questões temporais nos planejadores, como o TGP (*Temporal Graph Plan*) (SMITH, WELD, 1999).

Sob este aspecto, há uma oportunidade de contribuição com o tratamento das questões temporais pela engenharia do conhecimento. Com o uso de uma linguagem de modelagem, é possível dispor de um diagrama capaz de auxiliar na representação e no entendimento de um contexto global, com todos os elementos do domínio ou em um contexto específico, apenas com os elementos que fazem parte de uma ação específica. Assim, a proposta deste trabalho visa apresentar o uso de uma abordagem para cada contexto a partir de trabalhos que foram realizados para representação temporal (ALLEN, 1984) e (PNUELLI, 1977).

1.1. Objetivos

O objetivo deste trabalho é propor uma abordagem para modelagem de domínios temporais de planejamento através do uso do diagrama de tempo da UML, estendendo-o para o contexto dos domínios de planejamento e agregando-o à UML.P.

1.2. Organização do Trabalho

Esta dissertação está organizada da seguinte forma:

CAPÍTULO 2: Apresenta uma visão geral dos sistemas de planejamento e a introdução sobre tempo para planejamento, com os trabalhos realizados sobre instantes e intervalos e planejamento temporal.

CAPÍTULO 3: É introduzido o conceito de Engenharia do Conhecimento para planejamento temporal e são analisados os pontos que estendem a PDDL para sua versão 2.1. Também neste capítulo são apresentados os diagramas da UML.P necessários para a modelagem de domínios de planejamento.

CAPÍTULO 4: Apresenta uma proposta de modelagem de domínios temporais de planejamento com o uso do diagrama de tempo da UML com uma abordagem específica para planejamento. Uma interpretação para PDDL 2.1 também será abordada neste capítulo.

CAPÍTULO 5: Apresenta um estudo de caso com a modelagem de um domínio em UML.P.

CAPÍTULO 6: Apresenta a conclusão do trabalho bem como os trabalhos futuros.

CAPÍTULO II

2. Planejamento Temporal

2.1. Planejamento Automático

Planejamento automático, como dito anteriormente, é a área da Inteligência Artificial (IA) que tem como objetivo o estudo e a resolução de problemas cujas soluções possíveis estão relacionadas ao processo deliberativo de organização e escolha de ações de forma automatizada. A partir da descrição do estado atual de um ambiente e da descrição do estado objetivo, a aplicação deste processo possibilita identificar quais seqüências de ações são capazes de alcançar o objetivo, antecipando os resultados (GHALLAB; NAU; TRAVERSO, 2004).

Historicamente, as primeiras pesquisas na área de planejamento em IA ocorreram na década de 60, com os estudos das técnicas de busca e prova de teoremas. Em um problema real de busca, um agente necessita obter uma resolução percorrendo, no pior caso, todo o espaço de estados possíveis do domínio, com o risco de executar uma série de ações irrelevantes. A cada passo, o agente realiza uma escolha a respeito da transformação, ou ação, a ser aplicada. Caso a escolha não conduza a uma solução completa, o agente retorna ao estado de uma solução parcial e retoma a busca a partir deste ponto (RUSSEL; NORVIG, 2004).

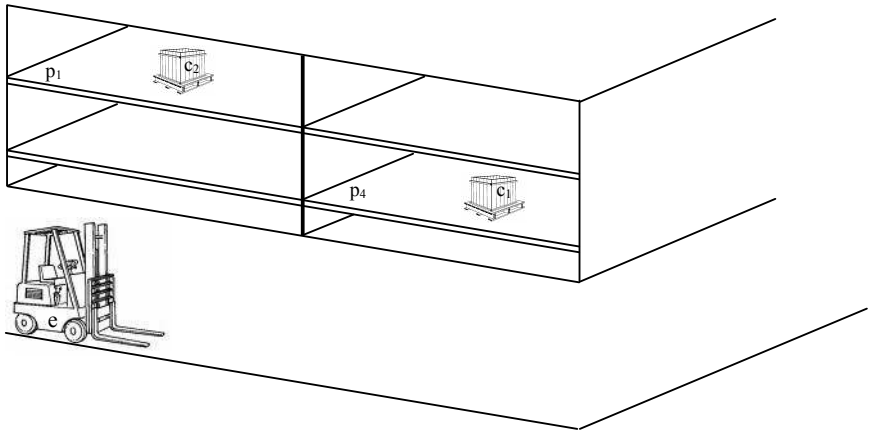
Newell, Shaw e Simon (1959) propuseram o primeiro sistema cujo princípio era solucionar qualquer problema de busca. O GPS (*General Problem Solver)* era um sistema que distinguia o conhecimento do problema e a estratégia para resolução. Em sua abordagem, o GPS lidava com um ambiente de tarefas, composto por objetos que podem ser transformados por operadores. Isto possibilita identificar a diferença entre os objetos e organizar as informações a respeito do ambiente e dos objetivos. Desta forma, o GPS possibilitava um modo de alcançar um objetivo final a partir do alcance de objetivos parciais. O GPS foi considerado o precursor dos sistemas de planejamento, pois, a princípio, ele poderia solucionar qualquer problema de busca, alguns de forma análoga aos problemas de planejamento (NEWELL; SIMON, 1963).

Embora o GPS pudesse ser aplicado a problemas simples de planejamento, sua abordagem conduzia ao pensamento de que esta seria a única forma de resolver problemas de

planejamento, quando na realidade esta abordagem poderia ser aplicada a apenas uma parte dos problemas (McDERMOTT; HENDLER, 1995). McCarthy e Hayes (1969) introduziram uma representação para domínios dinâmicos através de um formalismo lógico denominado cálculo situacional (*situation calculus*). Os principais elementos do cálculo situacional são as ações que podem ser executadas no mundo, os fluentes que descrevem o estado do mundo e as situações, que representam uma seqüência finita de ações. No cálculo situacional, o mundo é modelado progressivamente através de uma série de situações resultantes de ações executadas.

A primeira implementação do cálculo situacional foi o sistema QA3 desenvolvido por Green (1969), que resolvia uma variedade de problemas simples, expressos em cálculo de predicados e possuía interação através de perguntas e respostas. Neste trabalho, a aplicação do cálculo situacional deve-se aos axiomas sobre ações que conduzem a determinadas situações, de modo que uma seqüência de ações pudesse ser deduzida. Porém, a partir dos teoremas do cálculo situacional, os axiomas livres do trabalho de Green conduziam a problemas de busca de difícil solução. Então, em 1970, um grupo de pesquisadores do Instituto de Pesquisas de Stanford, Califórnia, desenvolveu um sistema que combinava o GPS e os métodos de provas de teoremas. Este sistema foi denominado STRIPS (*Stanford Research Institute Problem Solver)* (FIKES, NILSSON, 1970) e possibilitava a resolução de problemas maiores que os até então solucionados.

O sistema STRIPS busca a solução de problemas em um espaço baseado nos modelos do mundo representados por um conjunto de fórmulas de lógica de primeira ordem. A partir de um modelo atual, o sistema trabalha na busca de uma seqüência de operadores que produzam um estado do mundo no qual o objetivo (ou conjunto de fórmulas) seja verdadeiro. Os modelos do mundo são representados por estados e as mudanças ocorrem conforme a aplicação das ações. De forma geral, para que uma ação seja aplicada, é necessário que uma lista de pré-condições (fórmulas de lógica de primeira ordem) seja satisfeita no estado atual. A mudança ocorre em função de uma lista de adição (lista de fórmulas que devem ser adicionadas ao estado atual do mundo) e uma lista de remoção (lista de fórmulas que não serão mais verdadeiras após a aplicação do operador e que devem ser removidas) (FIKES, NILSSON, 1970).



Estado Inicial

Livre(e)	A empilhadeira está disponível
Em(c_1,p_4)	A caixa c_1 está localizada na prateleira p_4
Em(c_2,p_1)	A caixa c_2 está localizada na prateleira p1
PisoLivre	Nenhuma caixa está sobre o piso

Figura 2.1 - Exemplo de representação STRIPS
Fonte: autor

A figura 2.1 ilustra uma representação simples de STRIPS, cujo domínio refere-se a um armazém, onde uma empilhadeira necessita organizar os produtos com a troca de lugar entre duas caixas utilizando o piso como local intermediário. Um estado do mundo é representado por um conjunto de predicados. No estado inicial, a empilhadeira *'e'* está livre, a caixa 'c_1' está na prateleira 'p_4', a caixa 'c_2' está na prateleira 'p_1' e o piso está livre. Pela hipótese do mundo fechado de Reiter (1978), qualquer outra condição não mencionada no estado é considerada falsa.

A mudança do estado inicial ocorre pela aplicação dos operadores. Tais operadores possuem variáveis que, quando instanciadas, correspondem às ações. Prosseguindo com o exemplo acima, uma ação que compõe este domínio é:

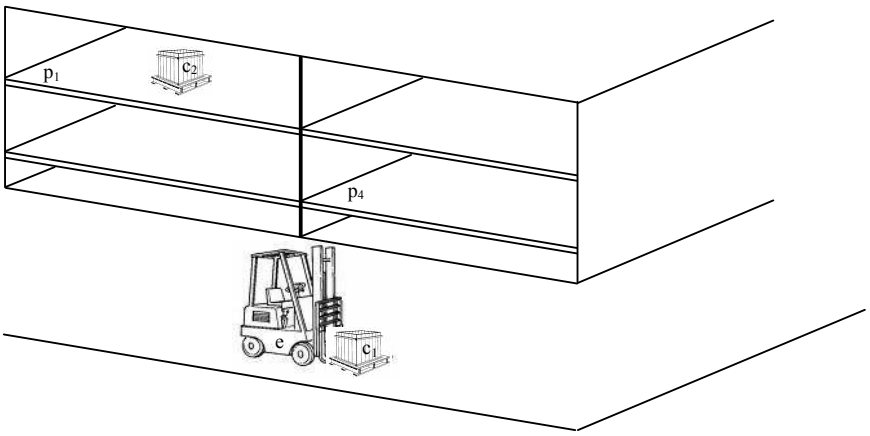
Ação(Retirar(e,c,p),

Pré-condição: Empilhadeira(e) $\wedge$ Caixa(c) $\wedge$ Prateleira(p) $\wedge$ Livre(e) $\wedge$ Em(c,p)

Lista de remoção: Livre(e), Em(c,p)

Lista de adição: Segurando(e,c), Vazio(p), $\neg$Em(c,p)).

A ação acima descreve a retirada de uma caixa de uma prateleira por uma empilhadeira. Todas as variáveis apresentadas na pré-condição devem constar na lista de parâmetros da ação e a execução da ação só será possível se a conjunção de literais for verdadeira: a empilhadeira 'e' deve estar livre para uso (Livre(e)), a caixa 'c' deve estar na prateleira 'p' (Em(c,p)).



Estado: segurando caixa	
Segurando(e, c_1)	A empilhadeira ‘e’ está segurando a caixa ‘c_1’
Vazio(p4)	A prateleira ‘p_4’está vazia
$\neg$Em(c_1,p_4)	A caixa ‘c_1’ não está localizada na prateleira ‘p_4’

Figura 2.2 Estado após aplicação da ação Retirar
Fonte: autor

Esta forma de representação refere-se a um **esquema de ação** (RUSSEL; NORVIG, 2004), que significa que várias ações podem ser derivadas pela instanciação das variáveis, como neste exemplo, a variável ‘e’ deve ser do tipo ‘Empilhadeira’, a variável ‘c’ do tipo ‘Caixa’ e a variável ‘p’ do tipo ‘Prateleira’.

Durante a execução da ação, os fatos Livre(e) e Em(c,p) descritos na lista de remoção são excluídos, pois não são mais verdadeiros. Os novos fatos descritos na lista de adição, Segurando(e,c), Vazio(p) e $\neg$Em(c,p) são adicionados e, com eles, um novo estado do mundo é alcançado. Pela hipótese STRIPS, todo literal não mencionado no efeito permanece inalterado. A figura 2.2, ilustra o estado após a aplicação da ação Retirar(e,c,p). Neste estado, a empilhadeira está segurando a caixa ‘c_1’, que não está mais localizada na prateira ‘p_4’, que está vazia.

Para encontrar uma seqüência de ações que solucione um problema, o sistema STRIPS utiliza a decomposição em subproblemas parciais, que são linearmente solucionados em relação à ordem de execução das ações do plano. Esta abordagem caracteriza um planejamento linear, com a resolução de todos os subproblemas de forma independente. Entretanto, a resolução pode não ser encontrada quando os subproblemas são dependentes e

devem ser solucionados em uma ordem específica, caracterizando a "Anomalia de Sussman" (SUSSMAN, 1975).

A busca por uma solução em um sistema simples de planejamento utiliza uma busca no espaço de estados. Composto por um grafo, cujos nós representam os estados do mundo e arcos que correspondem aos estados de transição, um plano corresponde ao caminho percorrido no espaço de busca. As duas abordagens mais comuns aos algoritmos de busca são a busca progressiva e regressiva. Na busca progressiva, ou para frente, o sistema procura uma solução a partir do estado inicial e, na busca regressiva, ou para trás, o início da procura é a partir do estado desejado retornando ao estado inicial. O sistema STRIPS utiliza uma abordagem semelhante à da busca regressiva, com enfoque na redução do espaço de estados (GHALLAB; NAU; TRAVERSO, 2004).

De modo geral, um sistema de planejamento é a implementação das técnicas de planejamento com o objetivo de encontrar uma seqüência de ações capaz de transformar um dado modelo de mundo em um modelo de mundo desejado. Estes sistemas caracterizam uma evolução dos sistemas solucionadores de problemas, pois além de aproveitar a estrutura do problema, controlam a explosão combinatória que existe em um mecanismo de busca.

Um sistema de planejamento possui elementos que possibilitam a utilização de um modelo como forma de representação. Um modelo define os principais componentes de um problema e é importante para esclarecer conceitos básicos, suposições restritivas e analisar a representação de domínios. Em um mundo real, existem diversos tipos de domínios que necessitam de uma abordagem específica, conseqüentemente, determinados planejadores têm um desempenho diferenciado conforme o tipo de problema.

Sacerdoti (1975) mostrou que os sistemas lineares baseados em STRIPS são incompletos e incapazes de solucionar problemas simples como a anomalia de Sussman e propôs uma abordagem não linear com o planejador NOAH (*Nets of Action Hierarchies*). Tate (1977) aprimorou esta abordagem com o sistema NONLIN, que utilizava busca regressiva (*backtracking*). Estes sistemas trabalham com a suposição de que o dinamismo do mundo é causado somente por suas ações. Esta característica foi denominada suposição-STRIPS (TONIDANDEL, 2003). Outras modalidades de planejamento foram propostas com a implementação dos sistemas SIPE (WILKINGS, 1988) que utilizava teoria causal e outras técnicas. Embora considerado avançado em sua época, este planejador ainda não resolvia uma série de problemas e era considerado lento para algumas aplicações.

Posteriormente, através de pesquisas em planejamento reativo, surgiram novos planejadores com capacidade de analisar um plano parcial e verificar se este é válido em um determinado instante ou se é necessário seu aprimoramento. O sistema TWEAK (CHAPMAN, 1987) foi o primeiro sistema com estas características, mas, mesmo assim, ainda era incapaz de solucionar alguns problemas em tempo aceitável. Com uma abordagem baseada em grafos, o sistema GRAPHPLAN (BLUM; FURST, 1997) analisa a relação entre as ações e os predicados do domínio e encontrava a solução resolvendo conflitos e inconsistências existentes no grafo.

A partir dos conceitos do sistema GRAPHPLAN e do uso da heurística, sistemas como FF (HOFFMANN; NEBEL, 2001) e LPG (GEREVINI; SERINA, 2002) aumentaram significativamente a gama de problemas de planejamento solucionados. Novos conceitos de raciocínio baseados em casos (TONIDANDEL, 2003) introduziram a reutilização de planos já existentes na definição de novos planos, possibilitando o aumento da assertividade e da velocidade na criação de planos.

Os sistemas de planejamento estão baseados em um sistema de transição de estados (HENZINGER; MANNA; PNUELI, 1992). As mudanças dos estados de um sistema causadas pela aplicação das ações configuram a abordagem dinâmica do planejamento em IA. Uma forma geral para representar estas características é um sistema de transição de estados. Formalmente, um sistema de transição de estados é descrito como uma tupla $\Sigma = (S, A, E, \gamma)$, onde:

- $S = \{s_1, s_2,...\}$ é um conjunto finito de estados;
- $A = \{a_1, a_2,...\}$ é um conjunto finito de ações;
- $E = \{e_1, e_2,...\}$ é um conjunto finito de eventos;
- $\gamma : S \times A \times E \rightarrow 2^S$ é uma função de transição de estados.

As ações e os eventos são elementos que estão associados ao dinamismo do sistema, entretanto, as ações são controladas pelo planejador e os eventos correspondem aos acontecimentos internos que podem ocorrer durante um estado. Se *a* é uma ação e $\gamma(s,a)$ não é vazio, então a ação *a* é aplicável no estado *s*. Se *e* é um evento e $\gamma(s,e)$ não é vazio, então *e* pode eventualmente ocorrer durante o estado *s* (GHALLAB; NAU; TRAVERSO, 2004).

Dado um sistema de transição de estados Σ, o objetivo do planejamento é encontrar uma seqüência de ações que atinja um objetivo a partir de uma determinada situação. O agrupamento desta seqüência caracteriza um plano. As formas como os planos são construídos caracterizam abordagens diferentes:

- Planejamento:
  - O objetivo é alcançado por qualquer seqüência de ações que alcance um estado e_g ou um conjunto de estados E_g.
  - O objetivo é satisfazer uma condição através da seqüência de estados do sistema, como um estado a ser alcançado em determinado ponto, um requisito a ser evitado ou um estado em que o sistema deve permanecer.
  - Indicadores positivos e negativos são adicionados aos estados, como forma de avaliar objetivos desejáveis e indesejáveis.
- Escalonamento: necessita de um controle para as restrições temporais e para a utilização de recursos, com a possibilidade de não ser possível escolher a seqüência de ações.

Além do sistema de transição de estados Σ, outros componentes de um sistema de planejamento são: Um controlador que, de acordo com o estado de entrada, provê uma ação conforme um plano e um planejador que, de acordo com a descrição do sistema Σ, uma situação inicial e um objetivo, sintetizam um plano para o controlador de modo a alcançar o objetivo (SMITH, FRANK, JÓNSSON, 2000).

Apesar do evidente avanço dos sistemas de planejamento, o quesito tempo ainda é um desafio para tratar domínios reais. Somente depois de alcançar certo grau de maturidade, alguns sistemas de planejamento começaram a considerar a duração das ações. Depois da concepção da Competição Internacional de Planejamento (IPC) em 1998 e da inclusão da tratativa de tempo na edição de 2002, com a PDDL 2.1, os sistemas de planejamento têm evoluído significativamente para tratar questões temporais simples e complexas, como as que envolvem concorrência. Nas seções seguintes, serão apresentados os tópicos que envolvem o planejamento temporal e o raciocínio temporal embarcado nos sistemas de planejamento.

2.2. Tempo para planejamento

A importância da representação temporal nos problemas de planejamento fomentou o desenvolvimento de diversos trabalhos na área de engenharia do conhecimento e lógica temporal. O trabalho de Allen (1983), que trata da manutenção do conhecimento sobre intervalos temporais, foi muito importante e tornou-se base para este e para diversos outros trabalhos, como o de Ladkin (1987) que propõe um sistema de intervalo de unidades de tempo denominado TUS (*Temporal Unit System*). Este sistema é um modelo canônico do Cálculo de

Intervalo de Allen para representar, de uma forma natural, todos os tipos de intervalos temporais padrões como anos, dias, horas, etc. Seguindo a mesma linha, o trabalho de Bittner (2001) sobre a partição da linha de tempo em diferentes granularidades (como dias, horas, minutos, etc.) e a localização dos intervalos em células dentro de cada partição.

A seguir serão abordados conceitos para representação e raciocínio temporal, como a referência temporal através de intervalos ou pontos e a relação entre estes pontos ou intervalos.

2.2.1. Instantes e intervalos

O dinamismo de um domínio pode ser analisado sob dois aspectos: "o que" está mudando e "quando" as mudanças ocorrem. "O que" representa o raciocínio vinculado a ações e eventos e "quando" trata as referências temporais, que são períodos de tempo em que as proposições permanecem verdadeiras ou pontos no tempo em que variáveis de estado têm seus valores alterados. Estes pontos são variáveis numéricas cujos valores estão limitados aos números reais R e representam os instantes. Segundo Allen (1983), um intervalo é um par (x,y) de números reais, onde $x \leq y$, representam o instante inicial e final.

A descrição de um intervalo é obtida através de seus extremos, i^-, que representa seu instante inicial, e i^+, que representa seu instante final, ambos com a restrição $[i^- \leq i^+]$. Outra forma de descrição pode ser obtida quando o tempo é definido por uma seqüência de pontos $t_0, t_1,..., t_n$ tal que $t_0 < t_1 < ... < t_n$. Um intervalo é um conjunto de pontos ordenados e a relação entre dois intervalos pode ser representada pela equivalência entre os pontos iniciais e finais. A tabela 2.1 mostra que as relações entre dois intervalos t e s podem ser representadas pelas relações equivalentes entre seus pontos iniciais e finais, ou pela aplicação de operadores temporais diretamente sobre os intervalos. Nesta tabela, t^- corresponde ao ponto inicial do intervalo t e t^+ ao ponto final. Analogamente para s (ALLEN, 1983).

Tabela 2.1 - Relações de intervalo definida por pontos finais

Relação de Intervalo	Relação equivalente sobre ponto final
t < s	$t^+ < s^-$
t = s	$(t^- = s^-) \wedge (t^+ = s^+)$
t sobrepõe s	$(t^- < s^-) \wedge (t^+ > s^-) \wedge (t^+ < s^+)$
t encontra s	$t^+ = s^-$
t durante s	$((t^- > s^-) \wedge (t^+ <= s^+)) \vee ((t^- >= s^-) \wedge (t^+ < s^+))$

Fonte: (ALLEN, 1983)

Segundo Allen (1983), a relação *durante* da tabela anterior é importante para representação de uma relação de hierarquia entre intervalos, por exemplo, se uma condição P permanece em um intervalo T e se T ocorre em outro intervalo I e P permanece durante T, então P também permanece durante I.

Esta relação pode ser utilizada para definir uma hierarquia de intervalos cujas proposições podem ser herdadas, o que favorece o processo de raciocínio. Como exemplo, se um processo está relacionado ao dia de hoje, é necessário considerar os intervalos que também estão no dia de hoje, conforme a hierarquia. Se um fato está relacionado com "ontem", ele não pode afetar o que é verdadeiro agora.

Pela tabela 2.1 anterior, é possível observar cinco relações que podem ser representadas por meio de intervalos. Contudo, se a relação *durante* for separada em *durante*, *inicia* e *finaliza*, será possível representar qualquer relação temporal entre dois intervalos (ALLEN, 1983). A tabela 2.2 contém as possíveis relações entre dois intervalos: *antes de, encontra, sobrepõe, inicia, finaliza e durante (a, e, s, i, f, d)*; suas relações simétricas: *depois de, é encontrado por, é sobreposto por, é iniciado por, é finalizado por e contém (a', e', s', i', f', d')*; a relação *igual* (*ig*) é idêntica à sua relação simétrica, totalizando treze relações.

Tabela 2.2 - Intervalos temporais de Allen (continua)

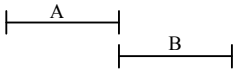
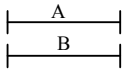
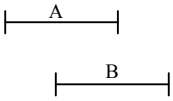
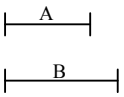
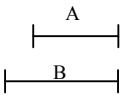
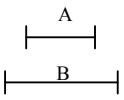
Relação	Relação Simétrica	Significado	Representação
A antes de B *a*	B depois de A *a'*	Existe um intervalo não vazio separando A e B. $(A^+ < B^-)$	
A encontra B *e*	B é encontrado por A *e'*	Não existe nenhum intervalo não vazio entre A e B. $(A^+ = B^-)$	
A é igual a B *ig*	B é igual a A *ig'*	Intervalos estritamente idênticos. $(A^- = B^-) \wedge (A^+ = B^+)$	
A sobrepõe B *s*	B é sobreposto por A *s'*	A é iniciado antes de B e B é finalizado após A. $(A^- < B^-) \wedge (A^+ < B^+)$	

Tabela 2.2 - Intervalos temporais de Allen (última)

Relação	Relação Simétrica	Significado	Representação
A inicia B *in*	B é iniciado por A *in'*	A possui o mesmo ponto inicial de B, mas está estritamente contido em B. $(A^- = B^-) \wedge (A^+ < B^+)$	
A finaliza B *f*	B é finalizado por A *f'*	A possui o mesmo ponto final de B e está estritamente contido em B. $(A^- > B^-) \wedge (A^+ = B^+)$	
A durante B *d*	B contém A *d'*	O ponto inicial de A ocorre depois do ponto inicial de B e seu ponto final ocorre antes do ponto final de B. $(A^- > B^-) \wedge (A^+ < B^+)$	

Fonte: (Allen, 1984)

Como as relações propostas por Allen são mutuamente exclusivas, não há ambigüidade nesta notação. Os relacionamentos entre intervalos são mantidos como uma rede cujos nós representam intervalos individuais. Cada arco entre estes nós é rotulado para representar um relacionamento possível. Assumindo que esta rede possui informações completas sobre os intervalos, quando um novo intervalo é adicionado, todas as conseqüências são computadas, inclusive o relacionamento que este novo intervalo tem com outros intervalos presentes na rede. Os relacionamentos entre os intervalos são redefinidos conforme uma relação de transitividade denominada por Allen como álgebra de intervalo (ALLEN, 1983).

Nos próximos tópicos serão abordadas duas formas importantes de relacionar tempo: a álgebra de ponto, que trabalha os possíveis relacionamentos entre pontos no tempo e a álgebra de intervalo, que, conforme dito anteriormente, trata a relação de transitividade entre intervalos.

2.2.2. Álgebra de Ponto

Álgebra de Ponto (AP) é um cálculo simbólico que possibilita associar ao tempo um conjunto de instantes com restrições qualitativas sem que estes estejam necessariamente ordenados. Considerando valores reais, existem somente três formas possíveis de relacionar dois instantes t_1 e t_2: $[t_1 < t_2]$, $[t_1 > t_2]$ ou $[t_1 = t_2]$. Se dois instantes possuem valores não

conhecidos e suas posições relativas não estão precisamente especificadas, seus relacionamentos podem ser determinados por: $[t_1 \leq t_2]$, $[t_1 \geq t_2]$ ou $[t_1 \neq t_2]$ (GHALLAB; NAU; TRAVERSO, 2004).

Seja P = {<, =, >} um conjunto de símbolos de relações primitivas entre instantes. Um conjunto de restrições qualitativas é dado por:

$$R = 2^P = \{\varnothing, \{<\}, \{=\}, \{>\}, \{<, =\}, \{>, =\}, \{<, >\}, P\}$$

Onde:

- "∅" significa uma restrição (vazia) que não pode ser satisfeita;
- "P" define a restrição universal;
- Cada elemento $r \in R$ é um conjunto de relações primitivas;
- r é uma restrição interpretada como uma disjunção destas relações primitivas, exemplo: $[t \neq t']$ é descrito como $[t\ r\ t']$ para $r = \{<, >\}$;
- Os operadores usuais ($\cup$, $\cap$, $\subset$, etc.) aplicados na teoria dos conjuntos podem ser aplicados em R, além do operador de composição $\bullet$, necessário para tratar transitividade. Exemplo:

  $$\forall\ r, q \in R: [(t_1\ r\ t_2) \wedge (t_2\ q\ t_3)] \Rightarrow (t_1\ r \bullet q\ t_3)$$

  A leitura é definida como: "Para todo r, q pertencente a R, se [(*a restrição entre* t_1 *e* t_2 *é dada por* r) e (*a restrição entre* t_2 e t_3 *é dada por* q)], então a *restrição entre* t_1 e t_3 *é dada pela composição* $r \bullet q$".

A tabela 2.3 define a composição entre as três relações primitivas (<, =, >) possíveis entre dois pontos. A primeira linha e a primeira coluna serão compostas pelo operador $\bullet$ e as demais células são o resultado desta composição, que podem ser visualizadas como:

Seja r = {<}, s = {=}, q = {>}, P = {<, =, >}
$r \bullet r = r;\ r \bullet s = r;\ r \bullet q$ = P;
$s \bullet r = r;\ s \bullet s = s;\ s \bullet q = q;$
$q \bullet r = P;\ q \bullet s = q;\ q \bullet q = q$

Tabela 2.3 - Tabela de composição para álgebra de ponto

•	<	=	>
<	<	<	P
=	<	=	>
>	P	>	>

Fonte: (GHALLAB, NAU; TRAVERSO, 2004)

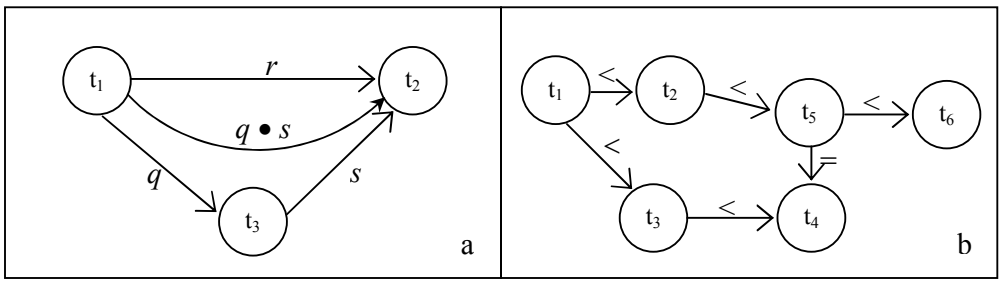


Figura 2.3 – Composição e conjunção transitiva
Fonte: (GHALLAB, NAU; TRAVERSO, 2004)

- Para três relações primitivas, a operação de composição é aplicada conforme a propriedade distributiva:

  $\forall\ r, s, q \in R$: $(r \cup q) \bullet s = (r \bullet s) \cup (q \bullet s)$

  $\forall\ r, s, q \in R$: $s \bullet (r \cup q) = (s \bullet r) \cup (s \bullet q)$

- A restrição simétrica de $[t_1\ r\ t_2]$ é r', de forma que $[t_2\ r\ t_1] \Leftrightarrow [t_1\ r\ t_2]$. Assim, r' é obtida pela substituição de "<" por ">" simetricamente no conjunto r de relações primitivas, enquanto que "=" permanece inalterado.

As operações que podem ser amplamente utilizadas para resolução de problemas de satisfação de restrições são $\cap$ e $\bullet$. Com estas operações, a verificação da consistência é obtida pela combinação de uma restrição r com a restrição vinculada à composição de transitividade (KROKHIN; JEAVONS, 2003). A figura 2.3(a) ilustra a relação de transitividade entre três pontos dado por:

$$([t_1\ r\ t_2] \wedge [t_1\ q\ t_3] \wedge [t_3\ s\ t_2]) \Rightarrow [t_1\ r \cap (q \bullet s)\ t_2]$$

Se $r \cap (q \bullet s) = \varnothing$ então as restrições r, q e s são inconsistentes.

A figura 2.3(b) ilustra algumas restrições em uma rede de álgebra de ponto vinculada por uma composição de transitividade:

$(t_4, t_6) : [t_4 = t_5] \bullet [t_5 < t_6] \Rightarrow [t_4 < t_6]$

$(t_2, t_3) : [t_2 > t_1] \bullet [t_1 \leq t_3] \Rightarrow [t_2\ P\ t_3]$

$(t_2, t_4) : [t_2 < t_5] \bullet [t_5 = t_4] \Rightarrow [t_2 < t_4]$

Outro caminho para (t_2, t_3, t_4) dado por (t_2, t_4): $[t_2\ P\ t_3] \bullet [t_3 < t_4] \Rightarrow [t_2\ P\ t_4]$ e a conjunção para as duas restrições sobre (t_2, t_4) é $P \cap \{<\} = \{<\}$.

Este tópico abordou a álgebra de ponto em um contexto suficiente para analisar os possíveis relacionamentos entre dois pontos. O próximo tópico abrange a álgebra de intervalo e como as relações entre intervalos podem ser aplicadas.

2.2.3. Álgebra de Intervalo

Álgebra de intervalo é o cálculo simbólico que possibilita relacionar um conjunto de intervalos no tempo com restrições qualitativas de forma similar à álgebra de ponto. A relação entre dois intervalos i e j, tal que seus pontos iniciais são dados por i^- e j^- e seus pontos finais por i^+ e j^+, cujas posições relativas estão dentro do conjunto $\Re$ (Reais) pode ser determinada pelas treze relações de Allen (1983) *antes de, encontra, sobrepõe, inicia, finaliza e durante (a, e, s, i, f, d)*; suas relações simétricas *depois de, é encontrado por, é sobreposto por, é iniciado por, é finalizado por e contém (a', e', s', i', f', d')* e a relação *igual* (*ig*), conforme observadas na tabela 2.2. As relações entre estes intervalos podem ser dadas por:

Seja P = {*a, e, ig, s, in, f, d, a', e', s', in', f', d'*} o conjunto de relações primitivas entre intervalos, o conjunto de restrições qualitativas pela álgebra de intervalo é dado por:

$$R = 2^{\mathrm{P}} = \{\varnothing;\ \{a\};\ \{e\};\ \{ig\};\ \{s\};\ \{in\};\ \{f\};\ \{d\};\ \{a, e\};\ \{a, ig\};\ ...;\ \{a, e, ig\};\ ...;\ \mathrm{P}\}$$

De forma análoga à álgebra de ponto, tem-se:

- "$\varnothing$" é uma restrição (vazia) que não pode ser satisfeita em P;
- "P" define a restrição universal que sempre é satisfeita;
- Cada elemento $r \in R$ é um conjunto de relações primitivas com 2^{13} restrições;
- *r* é uma restrição interpretada como uma disjunção destas relações primitivas, por exemplo, para o intervalo A da tabela 2.2 cuja relação [A {*a, e*} A'] significa [(A *a* A') $\vee$ (A *b* A')];
- Os operadores usuais ($\cup$, $\cap$, $\subset$, etc.) aplicados na teoria dos conjuntos podem ser aplicados em *R*, além do operador de composição $\bullet$, necessário para tratar transitividade:

  Para $r_1, r_2 \in$ R e intervalos A, B e C:

  $$([\mathrm{A}\ r_1\ \mathrm{B}]\ \text{e}\ [\mathrm{B}\ r_2\ \mathrm{C}]) \Rightarrow [\mathrm{A}\ (r_1 \bullet r_2)\ \mathrm{C}]$$

  A operação acima é computada a partir das 13 relações e da propriedade distributiva:

Para $r_1, r_2, r_3 \in$ R: $(r_1 \cup r_2) \bullet r_3 = (r_1 \bullet r_3) \cup (r_2 \bullet r_3)$

Igualmente para $r_3 \bullet (r_1 \cup r_2)$

- A restrição de simetria entre dois intervalos i_1 e i_2, [i_1 *r* i_2] é *r*' tal que [i_2 *r*' i_1] é obtido pela substituição de *r* por seu primitivo simétrico, ou seja, *a* por *a'*, *s* por *s'*, etc.

A tabela 2.4 exibe parcialmente as composições para as relações da álgebra de intervalo com base nas relações temporais de Allen da tabela 2.2. As restrições *u, v, w, x* e *y* referem-se aos conjuntos: $u = \{a, e, s\}$; $v = \{s, in, d\}$; $w = \{d, f\}$; $x = \{f, e, f'\}$ $y =$ P – {*a*, *e*, *a*', *e*'}. Algumas operações de composição não definidas nesta tabela podem ser completadas pelas restrições simétricas: $(r \cup q)' = q' \cup r'$ e que $\{ig\}$ é um elemento de igualdade na composição (linhas e colunas) (KROKHIN; JEAVONS, 2003).

Tabela 2.4 – Composição parcial para álgebra de intervalo

•	*a*	*e*	*s*	*in*	*d*	*f*	*b′*	*d′*	*f′*
a	*a*	*a*	*a*	*a*	$u \cup v$	$u \cup v$	*P*	*a*	*a*
e	*a*	*a*	*a*	*e*	*v*	*v*	$u' \cup v'$	*a*	*a*
s	*a*	*a*	*u*	*s*	*v*	*v*	$u' \cup v'$	$u \cup w'$	*u*
in	*a*	*a*	*u*	*in*	*d*	*d*	*a′*	$u \cup w'$	*u*
d	*a*	*a*	$u \cup v$	*d*	*d*	*d*	*a′*	*P*	$u \cup v$
f	*e*	*e*	*v*	*d*	*d*	*f*	*a′*	$u' \cup v'$	*x*
a′	*P*	$w \cup u$	$w \cup u'$	$w \cup u'$	$w \cup u'$	*a′*	*a′*	*a′*	*a′*
d′	$u \cup w'$	*v′*	$o' \cup w'$	$o \cup w'$	*y*	*v′*	$u' \cup v'$	*d′*	*d′*
f′	*a*	*e*	*o*	*o*	*v*	*x*	$u' \cup v'$	*d′*	*f′*
in′	$u \cup w'$	$o \cup w'$	$o \cup w'$	{*in,ig,in*'}	$o \cup w'$	*o′*	*a′*	*d′*	*d′*

Fonte: Adaptação de (ALLEN, 1983) e (GHALLAB; NAU; TRAVERSO, 2004)

É importante ressaltar que o operador de composição • não é comutativo na álgebra de intervalo, pois uma operação do tipo $\{d\} \cup \{a\}$ é igual a $\{d\}$ ao passo que $\{a\} \cup \{d\}$ é igual a $\{a, e, s, in, d\}$. Isto ocorre porque na operação entre intervalos, um intervalo *x* pode estar restrito a uma única posição em relação a outro intervalo *y*, porém, *y* pode ter várias possibilidades em relação à *x*.

A relação entre três intervalos também pode ser obtida através da tabela 2.4, pois cada célula possui uma relação entre dois intervalos e a transitividade entre os intervalos pode ser obtida com o auxílio de uma rede de pontos, semelhante à álgebra de pontos, através dos pontos limítrofes dos intervalos (LADKIN, 1987).

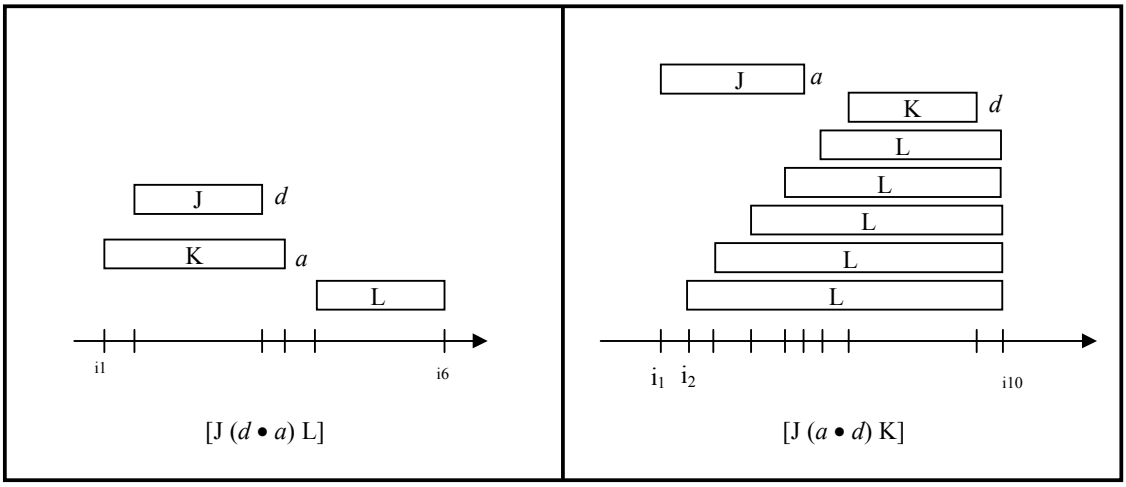


Figura 2.4 – Efeitos da composição de dois intervalos
Fonte: Adaptado de (GHALLAB; NAU; TRAVERSO, 2004) pag. 295

A figura 2.4 ilustra três intervalos J, K e L, onde [(J *d* K) ∧ (K *a* L)] restringe L a uma única posição em relação à J. Em contrapartida, considerando [(J *a* K) ∧ (K *d* L)], existem cinco possibilidades para L em relação à J. A partir da combinação possível entre os três intervalos J, K e L, é possível identificar as composições através da tabela 2.4. Também é possível identificar estas composições criando uma rede de intervalos e identificando as sub-redes válidas para cada uma das relações primitivas do conjunto P.

A álgebra de ponto e álgebra de intervalo apresentados neste tópico são importantes para a representação e raciocínio sobre tempo em planejamento. No próximo tópico, serão abordados conceitos e as formas gerais de planejamento temporal.

2.3. Planejamento temporal

Grande parte do avanço na área de planejamento deu-se pela necessidade de tratar domínios com características reais, cujas ações possuem duração ou sobreposição ao longo do tempo e/ou possuem restrições temporais. Sob o ponto de vista da representação do conhecimento, o raciocínio temporal em planejamento evoluiu a partir da lógica temporal com trabalhos de Allen (91), Pnuelli (1977) e McDermott (1982), entre outros. Sob o aspecto das técnicas de planejamento, a consideração do tempo de duração das ações e restrições temporais surgiu a partir dos sistemas clássicos de planejamento. Posteriormente, novas técnicas auxiliaram na resolução de problemas, como planejamento temporal baseado em grafos (SMITH; WELD, 1999) e heurística (KAMBHAMPATI; DO, 2001).

Nos próximos tópicos serão abordadas as técnicas atualmente aplicadas no desenvolvimento de sistemas temporais de planejamento.

2.3.1. Planejamento com operadores temporais

O planejamento com operadores temporais (GHALLAB; NAU; TRAVERSO, 2004) é baseado na álgebra de ponto. Cada proposição do domínio possui uma especificação do período em que serão verdadeiras. Os blocos de construção de um domínio temporal de planejamento são conjuntos finitos compostos por:

- Símbolos de constantes que correspondem aos objetos do domínio;
- Símbolos de variáveis;
- Símbolos que correspondem às relações rígidas, que não variam ao longo do tempo;
- Símbolos que correspondem às relações flexíveis, ou fluentes, que representam as relações entre as constantes do domínio que podem ou não ser verdadeiras em um dado instante;
- Restrições temporais baseadas na álgebra de ponto;
- Restrição de ligação entre as variáveis de objetos.

Um esquema de representação possui expressões invariantes e expressões dependentes do tempo, ambas relacionadas às restrições dos objetos. Estas relações e restrições especificam expressões temporalmente qualificadas (*etq*) e bases temporais. Uma *etq* é uma expressão na forma $p(\zeta_i,..., \zeta_k)@[t_s,t_e)$, onde p é uma relação flexível, $\zeta_i,..., \zeta_k$ são constantes ou variáveis de objetos e $@[t_s,t_e)$ são restrições temporais na forma $t_s < t_e$ que qualificam um intervalo semi-aberto, ou seja, @ define que esta restrição será verdadeira em t_s, mas não necessariamente em t_e. Uma *etq* $p(\zeta_i,..., \zeta_k)$ é verdadeira no intervalo definido por $@[t_s,t_e)$. Uma base temporal é um par $\Phi = (F, C)$, onde F é um conjunto finito de *etqs* que serão verdadeiras nas condições determinadas em C, que é um conjunto de restrições temporais e de objetos.

2.3.2. Exemplo: Tratamento térmico de metais

O seguinte exemplo mostra uma aplicação prática de um domínio de planejamento que será utilizado ao longo deste trabalho para ilustrar a forma de modelagem proposta.

Tratamento térmico é o processo que altera as propriedades físicas da estrutura molecular dos metais. Nos metais ferrosos, a têmpera é o tratamento que aumenta a dureza através do aquecimento do metal até uma temperatura crítica e um resfriamento brusco.

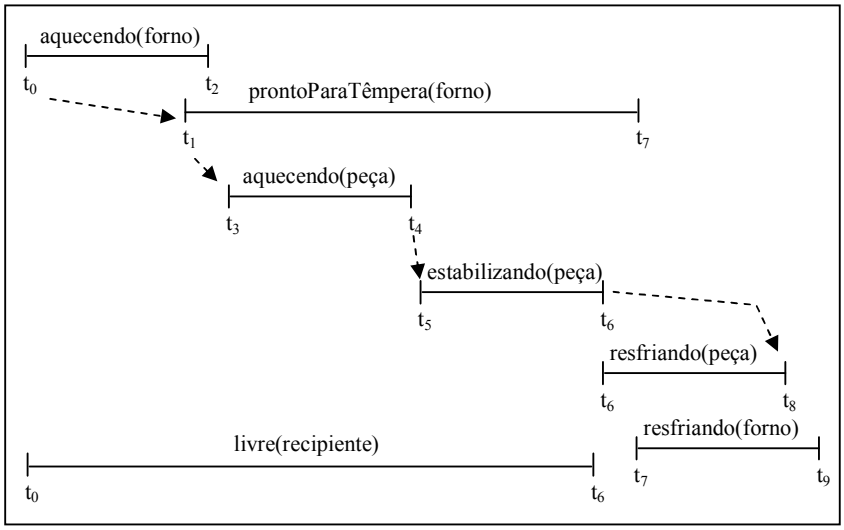


Figura 2.5 – Exemplo de base de dados temporal para o domínio de tratamento térmico
Fonte: autor

Quando o metal é submetido à têmpera, a estrutura interna de suas moléculas fica tensa e sem resistência a fortes choques físicos. Este problema é solucionado por outro tratamento térmico chamado revenimento, que consiste em elevar o metal até uma temperatura relativamente baixa em relação à têmpera e resfriá-lo bruscamente, como na têmpera. O resultado obtido por estes tratamentos é a combinação de dureza e resistência. Cada tipo de metal possui temperaturas específicas para têmpera e revenimento.

Neste exemplo, o forno a ser utilizado deve estar na temperatura desejada antes que o metal seja colocado e este deve permanecer dentro do forno por um período de aquecimento e um período de estabilização. Somente depois disto, o metal poderá ser retirado e depositado no recipiente para resfriamento e posteriormente colocado em uma bancada para ser disponibilizado. Neste exemplo, será considerado um único forno, um recipiente para resfriamento e uma bancada onde peças de metal serão acondicionadas. O trabalho é realizado por um operador de fornos.

2.3.3. Base e operador temporal

Um exemplo de base temporal pode ser obtido através do domínio de tratamento térmico de metais: para que uma peça seja tratada, existe um contexto que envolve diferentes objetos com ações e estados em momentos específicos. A figura 2.5 ilustra uma representação de uma base temporal Φ = ({aquecendo(forno)@[t_0,t_2), prontoParaTêmpera(forno)@[t_1,t_7), aquecendo(peça)@[t_3,t_4), estabilizando(peça)@[t_5,t_6), resfriando(peça)@[t_6,t_8), livre(recipiente)@[t_0,t_6), resfriando(forno)@[t_7,t_9)}, {adjacente(forno, recipiente), $t_0 < t_1 < t_3 < t_4 < t_5 < t_6 < t_8$}), que mostra que o forno deve estar aquecendo a partir do instante t_0 até t_2 e

permanece pronto para têmpera entre t_1 e t_7; a peça começa a ser aquecida no instante t_3 e permanece aquecendo até o instante t_4; logo em seguida, no instante t_5, a temperatura da peça mantêm-se estável até t_6, iniciando imediatamente o processo de resfriamento até o instante t_8; as restrições temporais determinam que o recipiente de resfriamento permaneça livre até o instante t_6, quando recebe a peça quente para resfriamento; logo após retirar a peça do forno, este inicia o processo de resfriamento a partir do instante t_7 até t_9.

Uma base temporal Φ define as mudanças do mundo considerando o tempo. Com base na suposição do mundo fechado, não há negação de fórmulas e conexões lógicas, pelo fato que estas fórmulas permanecem verdadeiras no tempo determinado pela *etq*. Uma relação rígida entre duas proposições deve estar especificada em Φ, como no exemplo anterior, a restrição do recipiente permanecer adjacente ao forno.

Outra característica inerente a uma base temporal é a capacidade de suportar uma *etq* cujo intervalo pode estar contido no intervalo de outra *etq*. Pela definição de Ghallab, Nau e Traverso (2004), um conjunto *F* de *etqs* suporta uma *etq* $e = p(\zeta_i,..., \zeta_k)@[t_1,t_2)$ se, e somente se, existir em *F* uma *etq* $p(\zeta'_i,..., \zeta'_k)@[\tau_1,\tau_2)$ e uma substituição σ tal que $\sigma\ (p(\zeta_i,..., \zeta_k)) = \sigma\ (p(\zeta'_i,..., \zeta'_k))$. Uma condição de ativação para *e* em *F* é a conjunção de duas restrições temporais $\tau_1 \leq t_1$ e $\tau_2 \leq t_2$, juntamente com uma condição de ligação de σ.

Uma condição de ativação para *e* em *F* pode exercer um efeito em outras *etqs*, pois *F* suporta um conjunto de *etqs* ε se, e somente se, existir uma substituição σ que unifique cada elemento de ε com um elemento de *F*. Uma condição de ativação para ε em *F* é a conjunção de condições de ativação para os elementos de ε. Uma base temporal $\Phi = (F, C)$ suporta um conjunto de *etqs* ε quando *F* suportar ε e existir uma condição de ativação c que seja consistente com *C*.

Além de compor a base temporal, as *etqs* possuem um papel fundamental na composição dos operadores (ou ações) temporais, que possuem a seguinte estrutura:

- Nome(*o*): expressão na forma $o(x_1,...,x_k, t_s,t_e)$, onde *o* é um operador e $x_1,...,x_k$ são variáveis de objetos e t_s,t_e são variáveis temporais. Variáveis sem restrições temporais são denominadas variáveis livres;
- Precondições(*o*) e Efeitos(*o*) são *etqs*;
- Constantes(*o*): é uma conjunção de restrições temporais e restrições de objetos, que são relações rígidas ou restrições de ligação na forma $x = y$, $x \neq y$, ou $x \in D$ sendo que este é um conjunto de constantes.

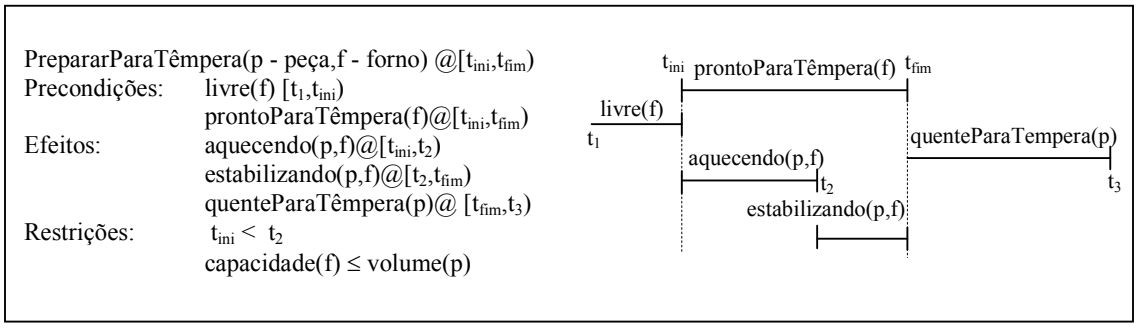


Figura 2.6 – Exemplo de operador temporal de planejamento: domínio de tratamento térmico. Fonte: Adaptado para este domínio a partir de (GHALLAB;NAU;TRAVERSO, 2004)

A figura 2.6 ilustra uma ação temporal a partir do exemplo do domínio de tratamento térmico. As variáveis t_{ini}, t_{fim} e t_2 são parâmetros implícitos da ação (observáveis por estarem nos parâmetros ou na lista de restrições) enquanto t_1 e t_3 são variáveis livres. Os intervalos descritos envolvem o tempo de execução da ação. As precondições requerem que o forno esteja livre e na temperatura necessária para têmpera no instante inicial. Ao iniciar a ação no instante t_{ini}, o forno recebe a peça e permanece na temperatura durante todo tempo de execução. São dois estágios para que a peça esteja suficientemente quente para ser temperada: no primeiro, há um aumento constante de temperatura e no segundo, um período de estabilização, quando a peça atinge a temperatura do forno. Ao término deste período, a peça estará suficientemente quente para ser temperada. Neste último período, entre t_2 e t_{fim} não há alteração de estados, somente o tempo em que o objeto necessita permanecer um determinado tempo em um estado.

Uma ação é um operador de planejamento parcialmente instanciado a = σ(*o*) para alguma substituição σ. Se houver uma precondição e uma restrição atendida por uma base temporal, então a ação é aplicável e será executada no tempo entre t_{ini} e t_{fim}. Novas *etqs* serão obtidas a partir do efeito da ação. De forma mais elaborada, uma base temporal Φ = (*F, C*) suporta a precondição de uma ação *a* se, e somente se, existir uma substituição de σ tal que, para cada p@[t_1,t_2) ∈ precondições(*a*) exista uma *etq* q@[τ_1,τ_2) ∈ *F* com σ(p) = σ(q). Uma condição de ativação para uma precondição(*a*) é uma conjunção, considerando todas as precondições de *a*, de duas restrições temporais $\tau_1 \leq t_1$ e $t_2 \leq \tau_2$ e a restrição de ligação em σ. O conjunto de todas as precondições(a) é definido como θ(a/*F*) (GHALLAB; NAU; TRAVERSO, 2004).

Quando as precondições de uma ação são suportadas pela base temporal, esta ação é aplicada. Apesar da semelhança com o planejamento clássico, em comparação com a função

de transição de estados, nada é removido da base temporal. O resultado da aplicação da ação *a* sobre uma base temporal Φ é uma nova base temporal estendida, com os efeitos de *a* e com outras restrições. Este resultado é semelhante ao obtido no planejamento clássico, porém, em comparação com uma função de transição de estado γ, nada é retirado da base temporal, pois não há uma lista de remoção como no planejamento clássico. Seja γ_0 um conjunto de possíveis bases temporais, *F* um conjunto finito de *etq* e *C* um conjunto de restrições temporais, o resultado da aplicação de uma ação *a* na base Φ pode ser observado por:

$$\gamma_0(\Phi, a) = \{(F \cup \text{efeitos}(a), C \cup \text{restrições}(a) \cup c) \mid c \}$$

Sempre que a ação *a* for aplicável, o conjunto $\gamma_0(\Phi, a)$ poderá conter várias bases temporais, pois $\theta(a/F)$ não é único. Cada uma destas bases pode ser obtida pela adição do conjunto de *etqs* de efeitos(*a*) e do conjunto de *etqs* de restrições(*a*) $\cup$ *c* para algum c $\in$ $\theta(a/F)$. Isto ocorre porque uma ação *a* pode ser aplicada de formas diferentes em Φ e em momentos diferentes e todos os conjuntos de possibilidades são considerados. Quando a ação não for aplicável, γ_0 será vazio. Para as variáveis de objetos, não há diferenças entre o planejamento clássico e o planejamento temporal.

No planejamento temporal, os efeitos negativos de uma ação não estão explícitos na definição do operador. Somente será definido o que for verdadeiro como o efeito da ação. No exemplo da figura 2.6 não é informado que o forno não está mais livre a partir de t_{ini}. Os efeitos serão resultantes do axioma ρ = condições(ρ) → disjunções(ρ), que define um conjunto de etqs e restrições temporais e de objetos. A partir deste axioma, é possível definir que para cada condição habilitada em condições(ρ), existe pelo menos uma disjunção em disjunções(ρ) que é consistente com Φ.

2.4. Domínios e Procedimentos Temporais de Planejamento

Um domínio temporal de planejamento é uma tupla $D = (\Lambda_\Phi, O, X)$ (GHALLAB; NAU; TRAVERSO, 2004), onde:

- Λ_Φ é o conjunto de todas as bases temporais que podem ser definidas com as restrições, constantes, variáveis e símbolos;
- *O* é um conjunto de operadores temporais de planejamento;
- *X* é um conjunto de axiomas do domínio.

Um problema temporal de planejamento em D é uma tupla $P = (D, \Phi_0, \Phi_g)$, onde:

- $\Phi_0 = (F, C)$ é uma base em Λ_Φ que satisfaz os axiomas de X. Φ_0 representa um cenário inicial que descreve além do estado inicial do domínio, a previsão de evolução independentemente das ações a serem planejadas;
- $\Phi_g = (G, C_g)$ é uma base que representa os objetivos como um conjunto G de *etqs* junto com um conjunto C_g de objetos e restrições temporais sobre as variáveis de G.

A descrição de um problema P é dada por $P = (O, X, \Phi_0, \Phi_g)$ e sua solução é um plano $\pi = \{a_1,..., a_k\}$ de ações, onde cada ação é uma instância parcial de um operador em O. Quando um plano π é aplicado em uma base Φ, a ordem das ações aparece situada no tempo, o que permite observar que um plano π não é apenas uma seqüência de ações, mas sim um conjunto. O resultado de π pode ser observado pelo conjunto de bases temporais γ em:

$$\gamma(\Phi, \{\}) = \{\Phi\}$$

$$\gamma(\Phi, \pi \cup \{a\}) = \cup_i \{\gamma(\Phi_i, a) \mid \Phi_i \in \gamma(\Phi_i, \pi)\}$$

Desta forma, π é uma solução para um problema $P = (O, X, \Phi_0,)$ se, e somente se, existe uma base $(F, C) \in \gamma(\Phi, \pi)$ vinculada a $\Phi_g = (G, C_g)$. As técnicas de planejamento aplicadas devem considerar tanto o conjunto π quanto a base (F, C) em $\gamma(\Phi_0, \pi)$ vinculada a Φ_g. As técnicas mais próximas são: espaço de busca, similar ao de planos parciais e a técnica de satisfação de restrições de problemas, que trata restrições temporais e do domínio.

2.5. Resumo e conclusão do capítulo

Neste capítulo foram abordados os conceitos fundamentais gerais do planejamento clássico e sobre planejamento temporal. As relações entre intervalos proposta por Allen(1983) serão fundamentais para a validação da proposta deste trabalho no capítulo 4. Além do presente trabalho, a álgebra de intervalos tem sido amplamente utilizada no embasamento de outros trabalhos referentes a raciocínio sobre tempo (KROKHIN; JEAVONS, 2003), lógica temporal (LADKIN, 1987) e planejamento temporal (GHALLAB; NAU; TRAVERSO, 2004) entre outros.

O próximo capítulo apresenta assuntos relacionados à Engenharia do Conhecimento com foco no aspecto temporal presente nos domínios de planejamento. Também será discutida a aplicação da UML que é uma linguagem de modelagem e representação do conhecimento e a PDDL, que é uma linguagem de definição de domínios de planejamento.

CAPÍTULO III

3. Engenharia do Conhecimento para planejamento temporal

Neste capítulo, serão abordados conceitos a respeito da Engenharia do Conhecimento (EC) e como são tratadas as características temporais nos domínios através de uma linguagem de definição de domínios.

3.1. Engenharia do conhecimento

EC em Planejamento Automático (PA) é o processo que trata a aquisição, validação e manutenção do conhecimento acerca dos domínios de planejamento. Planejamento em IA é por natureza baseado em conhecimento. Um conjunto de conhecimento associado ao planejamento é denominado domínio e a representação simbólica deste conjunto é denominada *modelo do domínio*. Os sistemas de planejamento dependem de um modelo de domínio, pois, para atingir um dado objetivo, utilizam este modelo para obter uma seqüência de ações. Da mesma forma que os sistemas de planejamento necessitam desenvolver técnicas robustas de solução de problemas, a EC necessita desenvolver técnicas que aprimorem a aquisição e representação do conhecimento (BIUNDO, 2003).

Os sistemas de planejamento são aplicáveis a inúmeros tipos de domínios. Conseqüentemente, uma pessoa que realiza o papel de engenheiro do conhecimento não possui informações suficientes sobre todos os domínios, então, torna-se necessária a intervenção de um especialista no domínio para auxiliar a correta identificação dos elementos e seus aspectos estáticos e dinâmicos. As técnicas de EC permitem realizar as atividades de obtenção, representação e armazenamento de informações de forma padronizada e independente de qualquer mecanismo de planejamento, pois, a partir das características do domínio, pode existir um sistema de planejamento mais apropriado para resolução do problema (MOTTA, 2000).

Hetzberg (1996) propõe que a aquisição do conhecimento em um domínio de planejamento passa por um processo que envolve suposições fundamentais sobre o ambiente, como a necessidade de considerar escalonamento ou simplesmente planejamento, modelagem determinística de ações, duração e recursos consumidos. As suposições pragmáticas sobre o ambiente referem-se ao tipo de plano necessário e critérios de otimização. Depois de tratadas estas questões sobre o ambiente, o processo de modelagem é iniciado com a identificação dos

objetos relevantes e classes de objetos do domínio, suas propriedades, relacionamentos (predicados) e restrições sobre estes predicados. Por fim, a heurística define as regras gerais de aproximação e aplicabilidade para auxiliar na geração dos planos.

A consistência interna e precisão de um modelo são obtidas através de uma validação que envolve um processo de identificação e remoção de erros. Uma vez que as origens de um modelo não são objetos matemáticos, este não pode ser provado corretamente (McCLUSKEY, 2001), diferentemente dos sistemas de planejamento. Um modelo suporta um sistema de planejamento e ocorre muito antes deste. Conseqüentemente, as validações de um modelo são diferentes das validações dos sistemas de planejamento, podendo até utilizar lógicas diferentes.

Além dos pontos supracitados, existem diferentes papéis envolvidos na modelagem de domínios, realizados por diferentes pessoas, como especialistas em planejamento, engenheiros de domínios, especialistas em domínios, especialistas em ferramentas e usuários finais. Embora existam estes diferentes papéis, muitas vezes eles são realizados pela mesma pessoa. As atividade beneficiadas pelo processo determinado EC, segundo (McCLUSKEY, 2003b) são:

- Criar e manter uma documentação sobre o domínio e sua dinâmica aos desenvolvedores;
- Auxiliar na identificação e correção de erros no início do desenvolvimento;
- Possibilitar a análise do modelo de forma aprofundada e auxiliar a transcrição do domínio.

3.2. Representação do conhecimento

Um modelo é uma simplificação da realidade que descreve um sistema a partir de uma perspectiva e, sem um modelo, não somos capazes de entender domínios complexos na sua totalidade. Para suportar o processo de modelagem do domínio e compartilhar o conhecimento adquirido nesta fase, é necessária a adoção de uma forma de comunicação e representação (KRUCHTEN, 2000).

Na representação do conhecimento destacam-se dois paradigmas. Um de representação proposicional (LARKING; SIMON, 1987) e outro de representação analógica, baseado em modelo (KULPA, 1994). As linguagens proposicionais descrevem um domínio através de um conjunto de sentenças (ou lógica de predicados) e devido sua natureza descritiva, todas as propriedades e restrições do mundo são necessárias.

Conseqüentemente, surgem alguns problemas como o *frame problem* (McCARTHY; HAYES, 1969) (para cada ação, quais propriedades do mundo permanecem inalteradas por sua execução), o *qualification problem* (McCARTHY, 1977) (para cada ação, o número de precondições é imenso) e o *ramification problem* (POLLOCK, 1998) (todas as ações possuem infinitas ramificações, traçando um futuro indefinido) e teorias para resolvê-los, como a Teoria Causal (SHANAHAM, 1997) que propõe a resolução do *frame problem*. Por serem puramente descritivas, as linguagens proposicionais possuem grande capacidade de ser flexíveis e expressivas (McCLUSKEY, 2003), mas se tornam ineficientes na medida em que os domínios atingem grandes proporções, com muitas entidades e restrições. Uma delas é a PDDL que será abordada no próximo tópico.

3.3. PDDL

Conforme dito anteriormente, a PDDL (*Planning Domain Definition Language*) foi inicialmente desenvolvida para competição internacional de planejamento e escalonamento (IPC-98) com o propósito de padronizar a especificação de domínios e problemas de planejamento e avaliar o desempenho e o comportamento de diferentes planejadores para um mesmo domínio. A sintaxe das expressões da PDDL está baseada na linguagem LISP (*List Processing Language*) (McCARTHY, 1960), cujo conceito principal é o uso de estruturas simples entre parênteses, denominadas listas, que são aplicadas para dados e códigos. Com uma semântica baseada em STRIPS, a PDDL representa os objetos de um domínio, as ações, o estado inicial do mundo e o estado objetivo.

A partir de características herdadas da linguagem ADL (*Action Description Language*) (PEDNAULT, 1989), a PDDL é capaz de representar efeitos condicionais nas ações, quantificação universal, qualificação existencial e ações hierárquicas compostas de sub-ações e sub-objetivos (McDERMOTT, 1998). A PDDL possui uma lista de requerimentos para cada tipo específico de domínio. Esta lista é importante para permitir que um planejador trate de forma específica cada tipo de requerimento.

Em sua versão inicial (McDERMOTT, 1998), a PDDL ainda não tinha uma representação consolidada para expressões numéricas e atualização de seus valores. Esta deficiência fora corrigida por McDermott (2000) em um artigo da "*AI Magazine*". Posteriormente, Maria Fox e Derek Long (2003) apresentam uma extensão da versão 2.1 com estruturas específicas para tratar ações com dependência temporal, cujos efeitos são discretos e contínuos. Ainda nesta versão, uma nova extensão (nível 5) é proposta para tratar efeitos contínuos e discretos em tempo real (FOX; LONG, 2003).

```
(:durative-action entregarPeca
  :parameters (?p - Peca ?r - Recipiente ?bs - Bancada)
  :duration (= ?duration (tempoEntrega ?p))
  :condition
     (and
        (at start (temperada ?p))
        (at start (revenida ?p))
        (at start (imersa ?p ?r))
        (at start (= (tempPeca ?p) (tempLiq ?r)))
        (at start (not (livre ?r)))
        (over all (not (livre ?r)))
    )

  :effect
    (and
        (at end (not (imersa ?p ?r)))
        (at end (sobre ?p ?bs))
        (at end (livre ?r))
      )
)
```

Figura 3.1 Ação durativa do domínio de exemplo Tratamento Térmico de Metais
Fonte: autor

3.3.1. Definição de domínios temporais em PDDL 2.1

Por ser uma linguagem centrada em ações, a PDDL trata os aspectos temporais através de dois tipos diferentes de ações. Apesar de existirem diversos tipos de ações temporais (DO; KAMBHAMPATI, 2001), segundo Maria Fox e Derek Long (2003), ações durativas[1] discretas e ações durativas com efeito contínuo, que serão apresentadas posteriormente, oferecem maior capacidade de expressão e possibilitam que os planejadores explorem a concorrência dos objetos envolvidos no domínio, justificativas pelas quais a versão 2.1 foi desenvolvida. A base para este tipo de ação ainda está calcada nas mudanças lógicas dos fluentes, causadas pela aplicação da ação. Estas mudanças são consideradas instantâneas e somente as variações numéricas são consideradas no intervalo de execução da ação.

No domínio de exemplo tratamento térmico, a figura 3.1 ilustra a ação temporal entregarPeça, que ocorre imediatamente após o tratamento de revenimento, que, como dito anteriormente, é necessário para eliminar as tensões internas do metal causadas pela têmpera e evitar que este se quebre. A ação recebe três objetos como parâmetro: uma peça (p), um recipiente (r) e uma bancada de saída (bs). A duração da ação é determinada pela função de tempo tempoEntrega, cujo valor será determinado no problema a ser solucionado. A condição da ação determina que a peça esteja temperada e revenida no início, bem como deve estar imersa no recipiente com a temperatura igual à do líquido contido. O recipiente não deve estar livre no início *(at start (not (livre ?r))),* devendo permanecer assim ao longo da ação *(over all (not (livre ?r)))*, tornando-se livre somente na conclusão *(at end (livre ?r))* e está definido nos efeitos quando a peça (p) ficará sobre a bancada (bs).

---

[1] O termo 'ação durativa' fora traduzido livremente pelo autor a partir de (FOX; LONG, 2003) para designar ações executadas em um determinado intervalo de tempo.

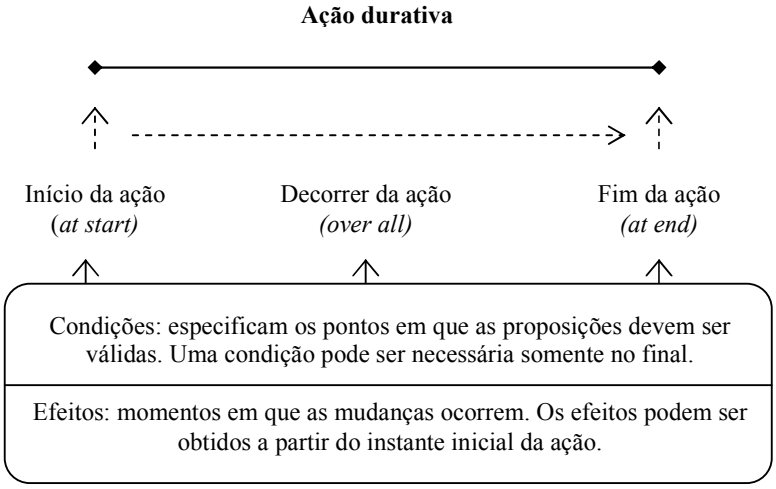


Figura 3.2 Momentos em que as condições e efeitos são considerados em uma ação durativa na PDDL
Fonte: autor

Durante o intervalo de execução, são considerados alguns pontos específicos em que as proposições serão exclusivas para a ação em questão. No início do intervalo (definido por *at start*), ou seja, no momento em que a ação é aplicada. No fim do intervalo (definido por *at end*), ou seja, no momento em que os efeitos finais da ação são alcançados, e durante todo o intervalo de execução da ação (definido por *over all*). Estes pontos possibilitam determinar se os efeitos da ação são imediatos a partir do momento em que esta é executada ou se ocorrerão imediatamente após a sua conclusão. Esta definição permite que o planejador explore a concorrência das ações na geração de planos.

Para possibilitar que ações que não são mutuamente exclusivas (utilizam os mesmos objetos) possam ser executadas compartilhando um mesmo instante, ponto em que ocorre o final de uma e o início da outra, a definição de *over all* implica na condição de intervalo aberto nas extremidades, ou seja, está livre no início e no final da ação. Se uma proposição deve permanecer com um valor durante todo o intervalo, sem considerar intervalo aberto, esta proposição deve ser restringida por *at start, over all e at end*. Nesta condição, uma ação só será executada somente se a outra estiver completamente finalizada.

Os pontos específicos, ou instantes, definidos na condição da ação, demonstram os momentos da ação em que as proposições devem estar disponíveis. Pela figura 3.2, é possível observar que a ação poderá ser aplicada se no início (*at start*) a proposição estiver disponível, se durante a execução ela estiver disponível (*over all*), ou e se no final ela estiver disponível (*at end*). Analogamente, nos efeitos da ação, estas mesmas restrições definem os momentos em que os efeitos da ação começam a ser concretizados (logo no início, durante ou no final).

```
(:durative-action aquecerFornoParaTempera
 :parameters (?p - Peca ?f - Forno)
 :duration (= ?duration (/ (- (/ (+ (maxGrausTemp ?p) (minGrausTemp ?p)) 2)
           (tempForno ?f)) (taxaAquecimento ?f)))
 :condition
     (and
       (at start (ligado ?f))
       (at start (vazio ?f))
       (at start (< (tempForno ?f) (minGrausTemp ?p)))
       (over all (not (temperada ?p)))
       (over all (vazio ?f))
     )
 :effect
       (at end(increase (tempForno ?f)
       (- (/ (+ (minGrausTemp ?p) (maxGrausTemp ?p)) 2) (tempForno ?f))))
)
```

Figura 3.3 Ação durativa discreta do domínio de exemplo Tratamento Térmico de Metais
Fonte: Autor

A partir da especificação dos instantes em que as proposições são necessárias para as ações, os planejadores podem se tornar capazes de explorar a concorrência entre as ações no processo de definição do plano. Esta abordagem é facilitada com uma visão de tempo baseadas em pontos (inicial e final), pois a precisão pode ser significativamente aumentada, de modo que nenhuma condição lógica seja definida como verdadeira e falsa no mesmo instante, o que tornaria qualquer plano inválido.

Além da determinação dos instantes em que as proposições estão disponíveis ou que são verdadeiras, as ações temporais na PDDL também possibilitam a modelagem de situações em que ocorre atualização de uma variável na execução da ação. Esta definição pode ser aplicada a recursos consumidos e reabastecidos por ações diferentes na mesma intersecção de tempo. Um exemplo de produção e consumo de recursos pode ser obtido em (FOX; LONG, 2003). A figura 3.3 ilustra uma ação (discreta) de aquecer o forno para temperar uma peça, onde a temperatura do forno é incrementada ao final da ação (pela construção *increase*).

A especificação da duração da ação segue o padrão de notação prefixada, com a definição de uma equação ou valor definido, que determinam uma duração fixa. Na figura 3.3, a especificação da duração da ação refere-se à equação:

maxGrausTemp = temperatura máxima para têmpera da peça em ºC.

minGrausTemp = temperatura mínima para têmpera da peça em ºC.

tempForno = temperatura do forno em ºC.

taxaAquecimento = taxa de aquecimento do forno em ºC/min.

$$duração = \frac{\left(\frac{temperaturaMáxima + temperaturaMínima}{2}\right) - temperaturaDoForno}{taxaDeAquecimento}$$

Equação 3.1 Duração da ação para aquecer forno
Fonte: autor

A equação 3.1 define o tempo de duração da ação de aquecer o forno para têmpera. A taxa de aquecimento, a temperatura atual do forno e a temperatura necessária para realizar a têmpera determinam o tempo necessário para aquecer o forno.

Para permitir que a duração seja influenciada por fatores externos, a PDDL 2.1 permite que a duração seja expressa através de uma inequação, o que oferece grande flexibilidade (exemplo: *duration (and (< ?duration x) (> ?duration y)*. A segurança pode ser aumentada com o acréscimo de uma condição que restrinja a variável que faça parte da especificação da duração, pois fatores externos que influenciarem esta variável, conseqüentemente afetarão a duração da ação. Se diversas ações concorrentes incrementam uma variável deste tipo, a duração da ação torna-se reduzida. Este controle não é possível sem o uso de uma inequação (FOX; LONG, 2003).

As ações discretas, conforme exemplo da figura 3.3, abstraem as mudanças contínuas e focam no resultado obtido no ponto em que a mudança ocorre. A sintaxe da PDDL 2.1 possibilita especificar estes pontos. Entretanto, quando um domínio exige variação contínua dos valores, bem como mudanças discretas, é necessária a utilização de recursos adicionais na especificação das ações. Para especificar que uma variável numérica será incrementada ou decrementada a uma determinada taxa ao longo da execução da ação e, também, acessada por outras ações neste período, a PDDL utiliza o símbolo #t para especificar mudança contínua a partir do início da execução até o final.

As variáveis numéricas cujos valores terão efeito contínuo ao longo da ação isentam a especificação do momento (início, durante ou final) de ocorrência, pois quando o símbolo #t é utilizado, a variável é atualizada continuamente ao longo da execução da ação e pode ser avaliada e modificada por outras ações em qualquer instante.

```
(:durative-action aquecerFornoParaTempera
 :parameters (?p - Peca ?f - Forno)
 :duration (= ?duration
          (/ (- (/ (+ (maxGrausTemp ?p) (minGrausTemp ?p)) 2)
          (tempForno ?f)) (taxaAquecimento ?f)))
 :condition
   (and
    (at start (ligado ?f))
    (at start (vazio ?f))
    (over all (not (temperada ?p)))
    (over all (vazio ?f))
   )

 :effect
     (increase (tempForno ?f) (* #t (taxaAquecimento ?f))))
)
```

Figura 3.4 Ação durativa de efeito contínuo do domínio de tratamento térmico
Fonte: autor

A figura 3.4 a ilustra uma ação durativa de efeito contínuo para o domínio de exemplo de tratamento térmico de metais. Na ação que aquece o forno para têmpera, a temperatura (tempForno) é aumentada gradualmente ao longo do tempo, conforme a capacidade de aquecimento do forno (taxaAquecimento) e a qualquer momento em que for avaliada, terá um valor diferente, modelado por: (increase (tempForno ?f) (* #t (taxaAquecimento ?f)))). A duração da ação não foi alterada em relação à modelagem anterior.

A PDDL realiza um papel extremamente importante no processo de engenharia do conhecimento para planejamento, pois está diretamente ligada aos planejadores. Desta forma, para poder atingir o objetivo deste trabalho e completar o ciclo de vida da modelagem de um domínio de planejamento, é necessário abordar as partes principais de sua sintaxe no que se referem ao tempo. No próximo tópico, a sintaxe da PDDL será brevemente apresentada para que seja possível propor uma tradução para a linguagem de modelagem proposta no capítulo 4.

3.3.2. PDDL para domínios temporais

A sintaxe da PDDL (McDERMOTT, 1998) segue a convenção:

- Cada regra é definida no formato <elemento sintático> **: :** = expansão;
- Os elementos sintáticos são delimitados pelos símbolos de maior e menor (<,>);
- As informações opcionais são representadas por colchetes ([,]). Esta informação pode estar acompanhada de um requisito sobrescrito entre parênteses, que significa que o material só será incluído para o domínio que estiver sendo modelado se este requisito for declarado: ([:types ...]) $^{(:typing)}$;

```
<domain>                ::= (define (domain <name>)
                              [<require-def>]
                              [<types-def>]:typing
                              [<constants-def>]
                              [<predicates-def>]
                              [<functions-def>]:fluents
                             <structure-def>_)
<require-def>           ::= (:requirements <require-key>+)
<require-key>           ::= <requirement list>
<types-def>             ::= (:types <typed list (name)>)
<constants-def>         ::= (:constants <typed list (name)>)
<predicates-def>        ::= (:predicates <atomic formula skeleton>+)
<atomic formula skeleton>
                        ::= (<predicate> <typed list (variable)>)
<predicate>             ::= <name>
<variable>              ::= ?<name>
<atomic function skeleton>
                        ::= (<function-symbol>
                             <typed list (variable)>)
<function-symbol>       ::= <name>
<functions-def>         ::=:fluents (:functions <function typed list
                                    (atomic function skeleton)>)
<structure-def>         ::= <action-def>
<structure-def>         ::=:durative-actions <durative-action-def>
```

Figura 3.5 Especificação de domínios em PDDL 2.1
Fonte: (FOX; LONG, 2003)

- O símbolo **: :=** pode ser sobrescrito por um identificador de marcador, indicando que a expansão é possível somente se houver uma declaração para o marcador;
- Um asterisco (*) indica “zero ou mais de “ e (+) indica “um ou mais”;
- Alguns elementos sintáticos são parametrizados. Ex. <list (symbol)>
- Os parênteses não possuem interpretação e apenas auxiliam a delimitação

Com base nestas convenções, a sintaxe da PDDL 2.1 especificada em (FOX; LONG, 2003) no formato *Backus Naur Form* (BNF) para definição de domínios de planejamento é definida pela figura 3.5. Um domínio é identificado por um nome único (cadeia de tipos alfanuméricos) e é constituído de seis partes: requerimentos, tipos, constantes, predicados, funções e ações. Cada parte descrita na figura 3.5 pode ser descrita como:

- `<require-key>`

Define os requerimentos necessários para o domínio. Esta informação é fundamental ao planejador, de modo que este possa tratar características específicas do domínio. No caso dos domínios temporais, são necessários três requerimentos mínimos:

`:durative-actions`	Permite ações durativas
`:duration-inequalities`	Permite restrições nas ações durativas com uso de inequações.
`:continuous-effects`	Permite que as ações durativas afetem continuamente os fluentes durante sua execução.

- `<types-def>`

  Define os tipos de objetos que representam as entidades do domínio. Também é possível especificar vários objetos do mesmo tipo. Exemplo:

  (:types
  Bancada - object
  BancadaEntrada BancadaSaida- Bancada
  Peça - object
  Forno - object
  Operador - object
  Recipiente - object
)

- `<constants-def>`

  Define padrões de nomes constantes em todo domínio e extensível aos problemas relacionados ao domínio. Exemplo:

  (:constants
  Engrenagem - Peça
  Polia - Peça
  Eixo - Peça
)

- `<predicates-def>`

  Define os predicados do domínio e as variáveis que os compõem. Exemplo:

  (:predicates

```
    (imersa ?pec - Peca ?rec - Recipiente)
    (dentro ?pec - Peca ?for - Forno)
    (sobre ?pec - Peca ?ban - Bancada)
    (temperada ?pec - Peca)
    (revenida ?pec - Peca)
)
```

- `<functions-def>`

  São as variáveis de função ou predicados que estarão associadas a valores numéricos. Para cada predicado, são definidas as variáveis e seus argumentos. Exemplo:

```
(:functions
  (minGrausTemp ?pec - Peca)
  (maxGrausTemp ?pec - Peca)
  (minGrausRev ?pec - Peca)
  (maxGrausRev ?pec - Peca)
  (tempPeca ?pec - Peca)
  (calEspecPeca ?pec - Peca)
)
```

- `<structure-def>`

  Engloba a definição da estrutura das ações. A PDDL define uma estrutura específica para ações atemporais e outra para ações temporais, que necessitam de elementos adicionais para especificar duração e os momentos em que as proposições terão valores definidos. A seguir, serão apresentados os elementos que compõem as ações temporais.

3.3.3. Definição de ações temporais

Na PDDL 2.1, a especificação de ações temporais é a principal parte na definição de um domínio temporal de planejamento. A sintaxe destas ações, definidas em (FOX; LONG 2003), influencia os tipos presentes na cláusula :*requirements* que abrange todo o domínio.

A definição de uma ação temporal exige critérios pré-estabelecidos de duração, restrição temporal dos instantes em que as proposições devem ser verdadeiras (no início, durante e no final da ação) para as condições de execução e para os efeitos da ação.

```
<durative-action-def> ::= (:durative-action <da-symbol>
                            :parameters ( <typed list (variable)> )
                            <da-def body>)
<da-symbol>           ::= <name>
<da-def body>         ::= :duration <duration-constraint>
                          :condition <da-GD>
                          :effect <da-effect>
```

Figura 3.6 Estrutura da ação temporal em PDDL 2.1
Fonte: (FOX; LONG, 2003)

A figura 3.6 ilustra a definição dos elementos de uma ação temporal em BNF. A ação é composta por três partes: identificação da ação *:durative-action <da-symbol>*, parâmetros recebidos *:parameters (<typed list (variable)>)* e o corpo da ação *<da-def body>*, sendo este subdividido em duração da ação *:duration <duration-constraint>*, condições *:condition <da-gd>* e efeitos *:effect <da-effect>*.

```
<duration-constraint>           ::= :duration-inequalities
                                    (and <simple-duration-constraint>+)
<duration-constraint>           ::= ()
<duration-constraint>           ::= <simple-duration-constraint>
<simple-duration-constraint>    ::= (<d-op> ?duration <d-value>)
<simple-duration-constraint>    ::= (at <time-specifier>
                                     <simple-duration-constraint>)
<d-op>                          ::=:duration-inequalities <=
<d-op>                          ::=:duration-inequalities >=
<d-op>                          ::= =
<d-value>                       ::= <number>
<d-value>                       ::=:fluents <f-exp>
```

Figura 3.7 Definição BNF da duração de uma ação temporal em PDDL.
Fonte: (FOX; LONG, 2003)

A figura 3.7 ilustra a especificação em BNF das restrições de duração das ações temporais. O tempo total da ação pode ser restringido por igualdade (=) ou por inequação (<= ou >=). A duração da ação pode ser definida por um valor, uma equação ou inequação. A duração da ação também pode ser suprimida quando for muito difícil de ser determinada. Quando determinado um ponto na duração da ação (*at start, over all, at end*), significa que a duração será avaliada neste ponto. Se nada for informado, o padrão é a avaliação no início (at start).

```
<da-GD>                 ::= ()
<da-GD>                 ::= <timed-GD>
<da-GD>                 ::= (and <timed-GD>+)
<timed-GD>              ::= (at <time-specifier> <GD>)
<timed-GD>              ::= (over <interval> <GD>)
<time-specifier>        ::= start
<time-specifier>        ::= end
<interval>              ::= all
```

Figura 3.8 Definição BNF dos qualificadores temporais em PDDL.
Fonte: (FOX; LONG, 2003)

A condição para que uma ação seja executada é definida em *<da-GD>*. Nesta cláusula, são definidos os momentos em que os predicados são instanciados ao longo da ação: no início (at start, over all, at end). A figura 3.8 ilustra a especificação destes momentos. Esta especificação está relacionada à figura 3.6.

As possibilidades para especificação dos efeitos da ação estão apresentados na figura 3.9, que complementa a figura 3.6 anterior. Da mesma forma que as condições devem ser válidas em um determinado instante da execução da ação, os efeitos também devem ser relacionados a um instante da ação (*<timed-effect>*). Os efeitos condicionais definidos por *(forall (<variable>*) <da-effect>)* e *(when <da-GD> <timed-effect>)* determinam que em determinado instante (*at start, over all, at end*) todas as instâncias de variáveis (*<variable>*) serão afetadas se (*when*) as expressões forem válidas neste instante.

```
<da-effect>      ::= ()
<da-effect>      ::= (and <da-effect>*)
<da-effect>      ::= <timed-effect>
<da-effect>      ::=:conditional-effects (forall (<variable>*) <da-effect>)
<da-effect>      ::=:conditional-effects (when <da-GD> <timed-effect>)
<da-effect>      ::=:fluents (<assign-op> <f-head> <f-exp-da>)
<timed-effect>   ::= (at <time-specifier> <a-effect>)
<timed-effect>   ::= (at <time-specifier> <f-assign-da>)
<timed-effect>   ::=:continuous-effects (<assign-op-t> <f-head> <f-exp-t>)
<f-assign-da>    ::= (<assign-op> <f-head> <f-exp-da>)
<f-exp-da>       ::= (<binary-op> <f-exp-da> <f-exp-da>)
<f-exp-da>       ::= (- <f-exp-da>)
<f-exp-da>       ::=:duration-inequalities ?duration
<f-exp-da>       ::= <f-exp>
```

Figura 3.9 Definição BNF dos efeitos de uma ação temporal em PDDL.
Fonte: (FOX; LONG, 2003)

Além dos efeitos lógicos ocorridos no início, ao longo duração ou no final da ação, é possível determinar quais efeitos numéricos influenciam a duração da ação. A figura 3.9 ilustra a possibilidade de associar uma inequação à duração (*?duration*).

```
<assign-op-t>       ::= increase
<assign-op-t>       ::= decrease
<f-exp-t>           ::= (* <f-exp> #t)
<f-exp-t>           ::= (* #t <f-exp>)
<f-exp-t>           ::= #t
```

Figura 3.10 Definição BNF dos qualificadores temporais em PDDL.
Fonte: (FOX; LONG, 2003)

O símbolo #t representa o período que uma determinada ação permanece ativa. A proposta desta representação é especificar efeito contínuo a uma taxa de variação constante ao longo do ciclo de vida da ação. A figura 3.10 define a sintaxe de aplicação deste símbolo em ações com efeitos contínuos. A associação de #t a uma expressão (<*f-exp*>) está relacionada a um operador de incremento ou decremento.

Neste tópico foi apresentada a PDDL para os aspectos temporais. No próximo tópico, será abordada a UML no intuito de apresentar uma linguagem de modelagem para suportar a engenharia do conhecimento.

3.4. UML

A UML (*Unified Modeling Language*) é uma linguagem de modelagem que surgiu em meados dos anos 90 com o propósito de unificar métodos para análise e projetos orientados a objetos. Segundo os precursores da linguagem, Grady Booch, Ivar Jacobson e James Rumbaugh, nesta época havia mais de cinqüenta métodos diferentes e poucos eram capazes de suprir todas as necessidades. Então, a partir de 1994, reuniram as principais características dos métodos Booch, OOSE (*Objectory Process*) e OMT (Object Modeling Techinique) em uma linguagem unificada (BOOCH; JACOBSON, RUMBAUGH, 2000). Com a participação de um consórcio de empresas e da comunidade internacional, a UML em sua versão 1.0 fora oferecida à OMG (*Object Management Group*) para padronização em janeiro de 1997 e após diversas revisões, a versão 1.1 fora adotada pela OMG em novembro deste mesmo ano. Desde então, a manutenção da UML foi assumida pela RTF (*Revision Task Force*) da OMG e novas revisões foram efetuadas até a 2.1.1, que é a versão oficial disponibilizada pela OMG no período em que este trabalho fora desenvolvido.

O objetivo principal da UML é representar conceitualmente e fisicamente um domínio e possibilitar o entendimento de sua estrutura e dinâmica. Com a utilização de uma simbologia gráfica com uma semântica bem definida, permite a comunicação entre diferentes especialistas envolvidos na construção de arquiteturas de sistemas, processos de negócio, aplicações de infra-estrutura e estrutura de dados (OMG, 2008).

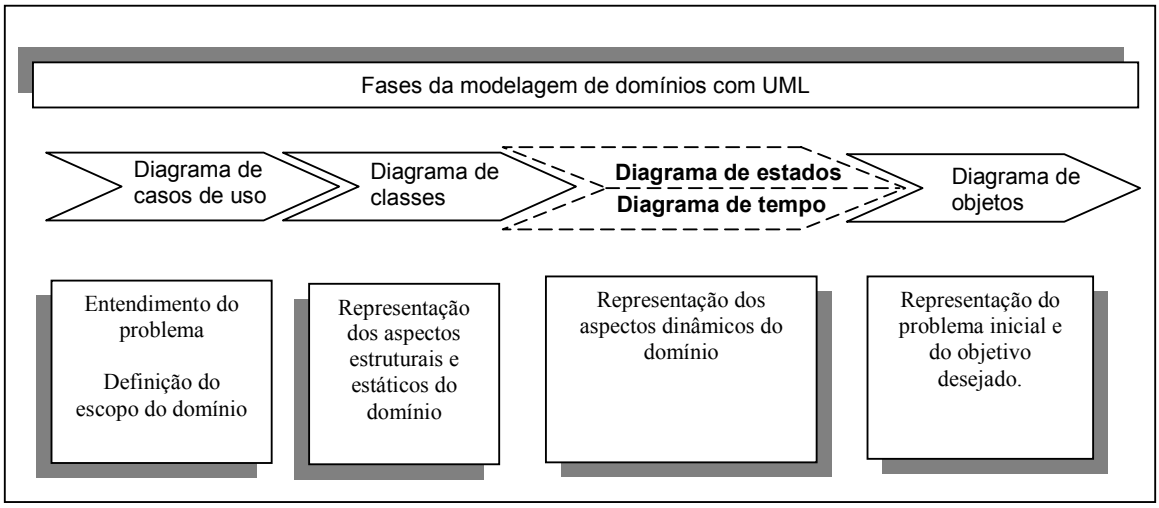


Figura 3.11 Aplicação da UML nas fases da modelagem de domínios
Fonte: autor

Por ser independente de linguagem de programação, suporta níveis mais elevados de abstração, como a definição de componentes, padrões e colaborações (BOOCH, RUMBAUGH; JACOBSON, 2000).

A simbologia gráfica da UML consiste em diagramas que especificam os aspectos estáticos e dinâmicos do domínio. A figura 3.11 ilustra os diagramas que fazem parte do escopo deste trabalho: *Diagrama de Casos de Uso*, que define um comportamento geral e a interação de um agente com o sistema; *Diagrama de Classes*, que define a estrutura do sistema, através da representação das entidades e seus relacionamentos; *Diagrama de Máquina de Estados*, que define o aspecto comportamental dos objetos ao longo do ciclo de vida do domínio; *Diagrama de Tempo*, que proporciona o raciocínio sobre tempo; *Diagrama de Objetos*, que define o comportamento de um conjunto de instâncias das classes.

Os demais diagramas estão disponíveis em (OMG, 2008). Para auxiliar a especificação das restrições dos objetos do domínio, a UML utiliza uma linguagem textual denominada OCL (*Object Constraint Language*) que será apresentada posteriormente.

3.4.1. Diagrama de Casos de Uso

Um caso de uso representa uma interação entre um ator do sistema e uma funcionalidade, com um resultado valoroso a este. Um diagrama de caso de uso representa o conjunto de cenários representativos do domínio a um observador externo. Cada caso de uso especifica o comportamento do domínio ou parte dele e é uma descrição de um conjunto de seqüência de ações que interagem com um agente. Desta forma, são úteis para fornecer uma maneira de promover o entendimento sobre o domínio. Além de uma interação com um agente, um caso de uso pode ser dependente ou estar incluso em outro caso de uso.

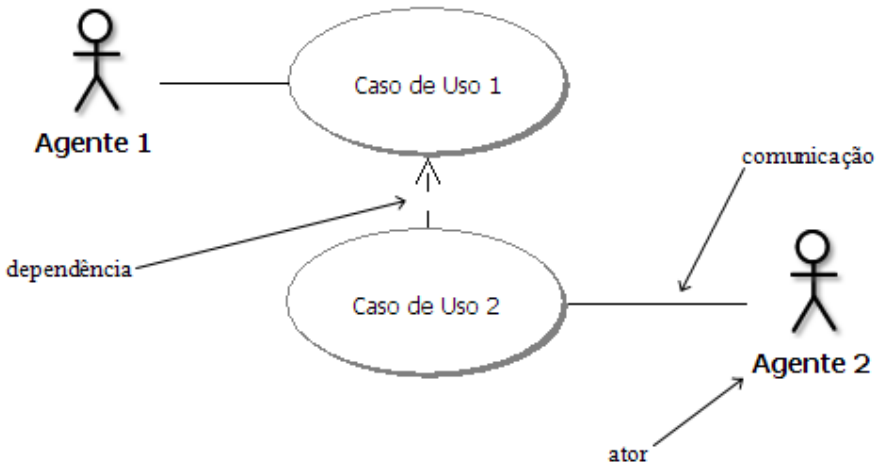


Figura 3.12 – Exemplo de Diagrama de Caso de Uso
Fonte: Autor

A figura 3.12 ilustra um diagrama de caso de uso. O ‘Caso de Uso 2’ é dependente do ‘Caso de Uso 1’, ou seja, o primeiro precisa ser executado antes do segundo. Neste exemplo, o ator ‘Agente 1’ comunica-se somente com o ‘Caso de Uso 1’, mas é possível que um ator se comunique com diversos casos de uso e vice-versa.

3.4.2. Diagrama de Classes

O diagrama de classes representa o conjunto de entidades que fazem parte do contexto de um domínio. As classes e seus relacionamentos são os elementos mais importantes em um domínio orientado a objetos. Segundo (BOOCH, RUMBAUGH; JACOBSON, 2000), uma classe é uma descrição de um conjunto de objetos que compartilham os mesmos atributos, operações, relacionamentos e semântica. Um diagrama de classes define um contexto estático, exibindo o que interage, mas não como esta interação ocorre. Os elementos do mundo real são representados por classes e estas traduzem com precisão seus atributos e comportamentos.

Uma classe é representada graficamente por um retângulo dividido em três partes: nome da classe, atributos da classe e operadores. Os atributos são propriedades nomeadas de uma classe que descreve um intervalo de valores que as instâncias da propriedade podem apresentar. Uma classe pode ter qualquer número de atributos ou até mesmo não ter nenhum. Um operador é a implementação de um serviço que pode ser solicitado por alguns objetos da classe para modificar seu comportamento. Uma classe pode ter qualquer número de operadores ou até mesmo não ter nenhum (BOOCH, RUMBAUGH; JACOBSON, 2000).

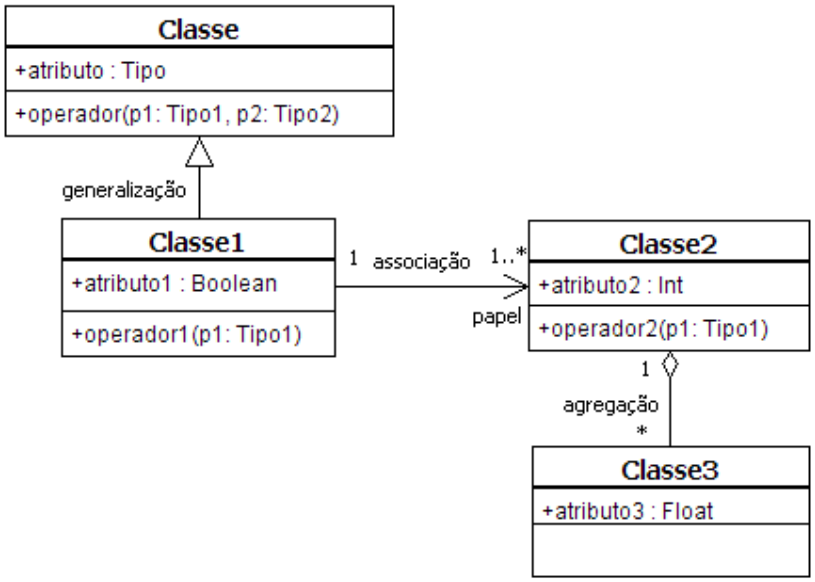


Figura 3.13 Exemplo de diagrama de classes
Fonte: Autor

A figura 3.13 ilustra um diagrama de classes com três tipos de relacionamentos: a generalização, a associação e a agregação. A generalização define subtipos de classes que herdam os atributos e métodos da classe superior, mas não o contrário. Neste exemplo, 'Classe1' é um tipo de 'Classe'. A associação especifica objetos de um item conectados a objetos de outros itens, com a possibilidade de navegação entre os objetos. Quando uma classe participa de uma associação, ela pode executar um papel específico no relacionamento. A seta indica o sentido de navegação. A agregação indica que um objeto é composto por outros objetos (um objeto maior composto por objetos menores). Em um relacionamento entre duas classes pode ser necessário especificar a quantidade de objetos que podem ser conectados pela instância de uma associação. Esta quantidade é denominada multiplicidade. Neste exemplo, uma instância da 'Classe1' pode estar conectada a uma ou mais instâncias da 'Classe2' e uma 'Classe2' tem várias instâncias da 'Classe3'.

Os relacionamentos entre as classes são:

- Generalização: uma classe inferior herda características e métodos de uma classe superior em um relacionamento "é um tipo de". Este relacionamento pode utilizar o tipo de visibilidade 'protegido';
- Associação simples: é um relacionamento estrutural que especifica que objetos de um item estão conectados a objetos de outro item. Este relacionamento

permite o acesso ao objeto em um determinado sentido. A seta indica o sentido de navegação para acessar o outro objeto.

- Agregação: é um relacionamento conceitual que determina um objeto é o "todo" e vários outros objetos do mesmo tipo relacionados definem a "parte".
- Composição: é uma variação da agregação que define que um objeto é composto por outros objetos. A existência da parte está condicionada à existência do todo.

Os atributos e métodos de uma classe possuem uma característica denominada visibilidade, que define como outra classe pode acessar seus atributos e métodos. As formas possíveis são:

- Público: o atributo/método pode ser acessado por qualquer classificador externo. Na representação pode ser utilizado o símbolo + antes do nome.
- Protegido: o atributo/método pode ser acessado pelos descendentes da classe. Na representação pode ser utilizado o símbolo # antes do nome.
- Privado: somente o próprio classificador pode acessar o atributo/método. Na representação pode ser utilizado o símbolo - antes do nome.

3.4.3. Diagrama de máquina de estados

O diagrama de máquina de estados define o comportamento de um elemento do domínio. Este comportamento é caracterizado pelos estados em que o elemento passa durante seu ciclo de vida. Um estado é definido por um conjunto de valores dos atributos do objeto que satisfazem uma determinada condição. Relacionado aos estados, os eventos estão localizados no tempo e no espaço e são capazes de disparar uma transição de estados, que levam o objeto de um estado a outro. A representação gráfica de um estado é um retângulo de cantos arredondados, com o nome do estado.

Segundo (BOOCH, RUMBAUGH; JACOBSON, 2000), a máquina de estados é capaz de especificar o comportamento de objetos que devem responder a estímulos assíncronos. Um objeto que recebe um sinal e não possui uma máquina de estados, não terá nenhuma reação ao estímulo recebido. Para que uma transição ocorra, é necessário que o objeto esteja em um estado de origem. Se ocorrer um evento de ativação que avalie uma condição e esta for satisfeita, então é disparada uma ação que altera os valores das variáveis do objeto, levando-o a um estado de destino.

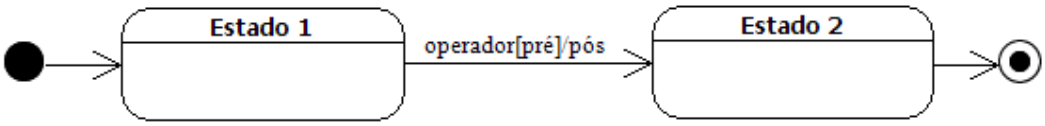


Figura 3.14 Diagrama de máquina de estados
Fonte: Autor

A figura 3.14 ilustra um simples diagrama de máquina de estados cujo estado inicial é definido pelo círculo preenchido e o estado 1 conduz ao estado 2 através do operador. O estado final é um círculo cheio envolto de um círculo vazio.

3.4.4. Diagrama de tempo

O diagrama de tempo explora o raciocínio sobre tempo em um domínio. Seu principal direcionamento está nas mudanças das condições ou estados ao longo do ciclo de vida do domínio ou de um determinado contexto dentro do domínio, que é representado por uma linha particionada em intervalos, através de instantes iniciais e finais. De acordo com a definição da OMG (2007), o diagrama de tempo descreve o comportamento tanto de classificadores individuais quanto de interação de classificadores, com foco no tempo de ocorrência dos eventos causando mudanças nas condições modeladas durante o ciclo de vida.

Pelo padrão da UML, o diagrama de tempo é uma representação especial da interação entre objetos com foco no tempo específico em que as mensagens são trocadas por estes objetos. A leitura do diagrama segue uma linha de tempo que se refere ao ciclo de vida do objeto ou contexto modelado. A leitura desta linha é realizada da esquerda para a direita e pode conter quantos pontos forem necessários para realizar a modelagem do tempo em que cada objeto permanece em um determinado estado.

O diagrama de tempo permite incluir múltiplos objetos que fazem parte do contexto modelado. Cada objeto possui seu próprio ciclo de vida e estes são empilhados no diagrama, de modo que seja possível a troca de mensagens entre os objetos ao longo dos respectivos ciclos de vida. Esta troca de mensagem é representada por uma seta com a direção da mensagem e um rótulo que representa o conteúdo da mensagem (PILONE; PITMAN, 2005).

Além da especificação dos instantes ao longo do tempo, o diagrama permite também a especificação da duração entre dois intervalos e a determinação de critérios de restrição temporal atrelados aos instantes.

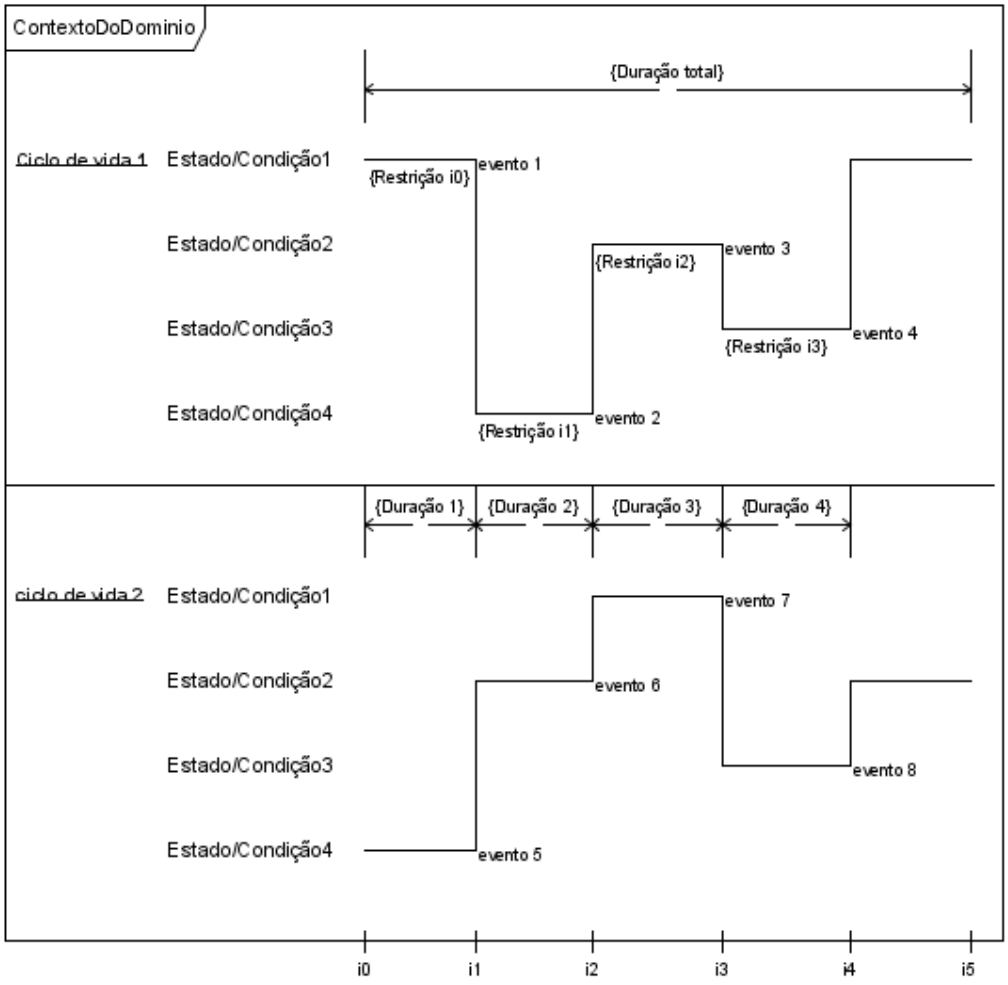


Figura 3.15 Elementos do Diagrama de tempo da UML
Fonte: Modelo realizado pelo autor com base em (OMG, 2007)

Os elementos que compõem o diagrama de tempo estão apresentados na figura 3.15. Nesta figura, é possível observar que o quadro representa o ciclo de vida de um contexto do domínio em questão. Para este contexto, são determinados todos os instantes relevantes. Neste exemplo, o domínio possui instantes crescentes de i0 a i5. Dentro deste contexto, é possível especificar os ciclos de vida subjacentes e para cada um destes, os estados/condições que serão alcançados ao longo do contexto. A mudança de um estado é determinada por um evento e a permanência em um estado pode ser especificada por uma restrição de duração, que é determinada para um intervalo (especificado entre chaves: *{Duração}*), através da especificação do instante inicial e final. Nesta figura, o ciclo de vida 1 permanece no estado/condição 1 do instante i0 ao instante i1, respectivamente para os demais estados/condições.

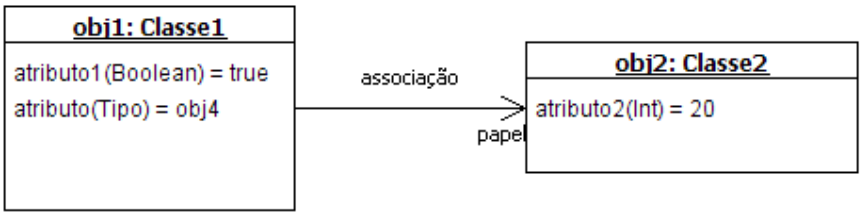


Figura 3.16 Exemplo de objetos em um diagrama
Fonte: autor

3.4.5. Diagrama de Objetos

O diagrama de objetos define a modelagem de uma visão estática de um conjunto de objetos e seus relacionamentos em um determinado ponto no tempo. Cada objeto representa a instância de uma classe com valores específicos para os atributos desta classe. Quando um diagrama é modelado, os objetos devem respeitar os relacionamentos definidos no diagrama de classes. Na modelagem de domínios, este diagrama auxilia na visualização, especificação, documentação e construção dos aspectos estáticos de sistemas através de engenharia de produção e reversa.

A figura 3.16 ilustra dois objetos participantes de um diagrama de objetos. Cada objeto é representado com por um retângulo com o nome do objeto e a classe a partir da qual o objeto fora instanciado. Logo abaixo, são listados todos os atributos da classe com os valores no instante. Os objetos são relacionados conforme especificação do diagrama de classes. Este relacionamento é denominado vínculo. Neste exemplo, os objetos foram instanciados a partir do exemplo na figura 3.13.

3.4.6. OCL

Embora os diagramas da UML representem contextos estáticos e dinâmicos em um domínio, estes não são suficientemente completos e refinados para prover todos os aspectos restritivos para os elementos e seus relacionamentos. Geralmente, as restrições são definidas utilizando a linguagem natural, de forma descritiva, o que pode gerar interpretações ambíguas. Para aprimorar a especificação e reduzir ambigüidades, foi desenvolvida uma linguagem de restrição de objetos, ou *Object Constraint Language* – OCL (OMG, 2003). Como seu objetivo é ser uma linguagem de modelagem, não tem capacidade de especificar uma lógica de programação e suas expressões não são executáveis.

A OCL pode ser utilizada para os seguintes propósitos: especificar invariantes sobre classes e tipos; especificar pré e pós-condições das operações e métodos; descrever restrições de estados e operações; especificar condições de parada e proteção e descrever regras de derivação para atributos em qualquer expressão sobre um modelo UML. Estereótipos de restrições são definidos pelas palavras chaves:

- *context* (contexto): define uma expressão OCL dentro do contexto de um modelo UML;
- *inv* (invariante): define que a expressão deve ser verdadeira em todas as instâncias do classificador (ex. uma classe);
- *pre* (pré-condição): especifica condições associadas a uma operação que são definidas como pré-condição para que esta seja disparada;
- *pos* (pós-condição): especifica o resultado de uma operação;
- *self* (próprio): referencia uma instância de uma classe;
- *let* (seja): atribui um valor a um atributo/variável;
- *if-then-else-endif:* condicionais se, então senão e fim-se.

Como exemplo, a partir do diagrama de classes especificado na figura 3.13, além das restrições definidas nos relacionamentos entre as classes “Classe2” e “Classe3” em que uma classe possui uma agregação de várias instâncias da segunda classe, uma especificação em OCL pode ser descrita como:

context Classe2 ***inv****:*

if forAll(c2:Classe2 | c2.atributo2 > 0) then

let self.atributo1 = ***true***

endif

A classe Classe2 terá o atributo1 verdadeiro se todas as instâncias da classe Classe2 tiverem o valor do atributo2 maior que zero.

A OCL, de fato, aumenta muito a capacidade de expressar restrições em modelos definidos em UML. No contexto temporal proposto neste trabalho, a OCL será extremamente importante para definir restrições de duração das ações e condições temporais. Existem mais especificações acerca da OCL, que estão disponíveis em (OMG, 2003) e estão fora do escopo deste trabalho. A seguir, será apresentada a UML com enfoque em domínios de planejamento.

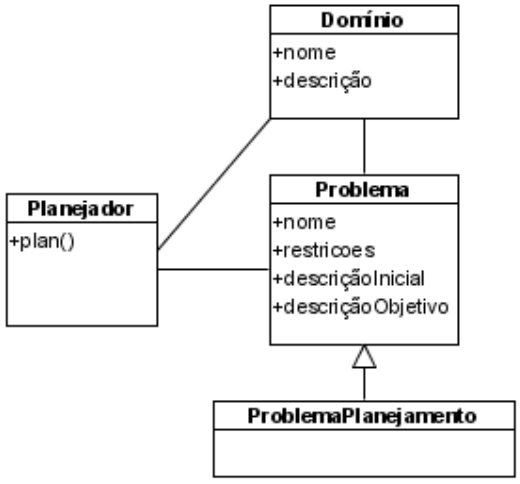


Figura 3.17 Esquema geral para um ambiente de planejamento
Fonte: (VAQUERO *et al* 2006)

3.5. UML.P

A UML é uma linguagem de propósito geral e, conseqüentemente, possui características intrínsecas aos domínios de planejamento. Por esta razão, a UML.P (VAQUERO; *et al* 2005) foi definida como uma aplicação da UML com abordagem em planejamento automático. Desta forma, torna-se possível aplicar uma metodologia de Engenharia do Conhecimento na modelagem de domínios e problemas de planejamento. Esta questão é de grande interesse da comunidade acadêmica, pois há a necessidade de modelar domínios para resolução de problemas reais.

As considerações iniciais para esta abordagem estão em torno de um esquema geral de um ambiente de planejamento. A figura 3.17 ilustra sintaticamente os elementos deste esquema e seus relacionamentos: um domínio, identificado por um nome, possui sua própria descrição, que engloba a especificação de todos os elementos, seus relacionamentos e as restrições gerais do domínio; um problema está associado a um domínio, mas um domínio pode estar associado a vários problemas. Cada um possui suas próprias restrições, uma descrição inicial do problema e uma descrição do objetivo a ser alcançado. Esta configuração de elementos é congruente com as regras definidas para o planejador, que está associado a estes elementos, mas possui um tratamento específico para cada um (VAQUERO *et al*, 2006). Depois destas considerações preliminares, é possível observar quais são as diferenças existentes entre a UML e a UML.P.

Os casos de uso na UML descrevem uma seqüência de ações que resultam em algo valioso para um ator do domínio. A especificação de um caso de uso é realizada em linguagem natural e são consideras as pré e pós-condições, as dependências e as ações são descritas por meio de um fluxo básico e para cada ação neste fluxo, pode haver fluxos alternativos que tratam os desvios. Na UML.P um caso de uso é estruturado e não considera a linguagem natural de forma que não surjam entendimentos dúbios. Entretanto, o diagrama de casos de uso segue a mesma especificação da UML.

Na UML.P, as classes permitem visibilidade pública aos atributos e não utilizam os critérios protegido e privado, portanto, não é necessário utilizar o símbolo '+' antecedendo o atributo na representação. Da mesma forma, os métodos não possuem restrição de visibilidade e também não são especificadas as assinaturas nos métodos, pois em planejamento as ações nunca retornam valores. Na UML.P o relacionamento entre as classes são do tipo Generalização, Associação Simples, Agregação e Composição.

Qualquer classe na UML.P pode possuir um diagrama de máquina de estados, principalmente aquelas que executam ações. O objetivo principal do diagrama de máquina de estados é analisar as mudanças que afetam o objeto quando uma ação é disparada. Uma ação pode afetar diversos objetos, porém, o objeto é único no diagrama. Conseqüentemente, a ação deve estar disponível em outros diagramas simultaneamente. Na representação de um estado, é possível visualizar os atributos válidos.

Os objetos na UML.P são necessário para determinar os problemas a serem solucionados. Em domínios de planejamento, os problemas são definidos por uma circunstância em que apenas dois pontos são conhecidos: uma situação inicial e um objetivo. Estas situações são definidas por um diagrama de objetos para o estado inicial e um diagrama de objetos para o estado objetivo. Tanto na UML quanto na UML.P este diagrama é composto por objetos e vínculos, que referem-se aos relacionamentos especificados no diagrama de classes.

Para atingir a proposta deste trabalho é necessário estender a UML.P, agregando a esta o diagrama de tempo. Este diagrama possui elementos suficientes para uma proposta geral de modelagem de tempo a partir dos elementos trabalhados nos diagrama de classes e máquina de estados. Para que este diagrama atenda os requisitos mínimos de um domínio temporal de planejamento, serão consideradas as seguintes adaptações:

- Dependência obrigatória do diagrama de máquina de estados: pelo menos um diagrama de máquina de estados deve estar modelado para que seja construído um diagrama de tempo;
- Quadro: o contexto temporal pode envolver um único objeto ou mais, chegando ao limite de todos os objetos. Desta forma, um quadro pode conter somente um diagrama de máquina de estados ou pode consolidar até o limite de todos os diagramas de máquina de estados em um único diagrama de tempo. Neste caso, todos os objetos participarão da linha de tempo do domínio;
- Ciclo de vida: cada ciclo de vida representa exclusivamente um objeto modelado no diagrama de máquina de estados;
- Estado/Condição: Cada estado configurado no diagrama de máquina de estados será tratado em um determinado intervalo de tempo;
- Eventos: os eventos serão definidos como as ações. As pré e pós-condições das ações serão oriundas do diagrama de máquina de estados (origem e destino);
- Unidade de tempo: as unidades de tempo serão mantidas da mesma forma;
- Restrição temporal: o uso da restrição temporal fica facultativo, pois depende do domínio a ser modelado;
- Restrição de duração: será obrigatoriamente relacionado à duração da ação. A determinação da duração da ação será realizada em OCL, pois pode depender de expressões cujas variáveis podem estar em diversas classes ou funções;
- Mensagens: não serão utilizadas.

3.6. Resumo e conclusão do capítulo

Neste capítulo foram abordados os temas relevantes para engenharia do conhecimento e modelagem de domínios de planejamento com a PDDL e modelagem de domínios com a UML. Estes temas são fundamentais para a elaboração da proposta de modelagem de domínios temporais com UML.P, que será apresentado no próximo capítulo.

Será possível analisar o diagrama de tempo no contexto de planejamento sob a ótica da álgebra de intervalo de Allen e também como as ações temporais com efeito contínuo podem ser modeladas neste diagrama. A validação será realizada com o auxílio do cálculo de intervalo de Allen, que permitirá analisar se o diagrama será eficaz para modelar as diferentes relações temporais entre dois intervalos.

CAPÍTULO IV

4. Domínios temporais de planejamento com UML.P

4.1. Introdução

O contexto apresentado anteriormente mostrou conceitos sobre planejamento e algumas aplicações da engenharia do conhecimento para modelagem e definição de domínios. Contudo, é necessária uma abordagem específica na engenharia do conhecimento capaz de representar os aspectos temporais inerentes ao dinamismo do domínio. Embora a percepção do tempo seja contínua, existem diferentes níveis de abstração para analisar tempo em um domínio, ou seja, analisar o ciclo de vida do domínio pode ser diferente de analisar o ciclo de vida de uma ação, pois o primeiro ciclo abrange os objetos do domínio e o segundo somente os objetos que participam da ação e seus fluentes.

Um domínio de planejamento é composto por elementos estáticos, como os objetos e por elementos dinâmicos como os fluentes, ou atributos, que estão relacionados aos objetos. A modelagem temporal em um domínio trata a indexação do dinamismo a determinados instantes no tempo e trata também os intervalos, que correspondem à duração das ações e permanência dos objetos nos estados. Existem diferentes fatores que determinam a duração de uma ação, como concorrência de recursos compartilhados ou uma função de duração entre outros que serão analisados posteriormente.

Desta forma, neste capítulo será apresentada uma proposta de modelagem de domínios temporais através da extensão da UML.P com a adaptação do diagrama de tempo da UML para domínios temporais de planejamento. Este diagrama deve ser capaz de representar o ciclo de vida do domínio, de modo que seja possível avaliar o tempo em que os objetos permanecem nos estados e o ciclo de vida das ações, possibilitando o mapeamento dos instantes em que as proposições são válidas. Sem o recurso de um diagrama de tempo, a modelagem de domínios de planejamento em UML.P fica limitada na medida em que os aspectos temporais não são representados.

Contudo, para que o diagrama proposto seja aplicável na prática, será necessário verificar se este é suficientemente abrangente para representar todas as relações entre dois intervalos. Esta verificação será realizada pela análise da correspondência com as relações temporais de Allen (1983), que, como visto no capítulo 2, descrevem qualquer relação entre dois intervalos. Se isto não for possível, o diagrama terá uma abrangência limitada.

4.2. Representação de domínios temporais

Os estudos sobre raciocínio temporal em PA surgiram com a necessidade de tratar domínios reais. Allen e Koomen (1983) foram os precursores das formas atuais de tratar tempo com a proposta de um conjunto de relações, conforme apresentado no capítulo 2, que define qualquer relação entre dois intervalos. A partir desta relação, Allen propõe uma álgebra de intervalos para garantir consistência na especificação de domínios. Ainda nesta linha de trabalho, Allen (1984) apresenta a modelagem de domínios temporais com o uso da representação de ações baseadas em STRIPS, porém, com condições e efeitos temporalmente qualificados. Também define que as ações possuem um tempo de execução que pode afetar a relação entre dois intervalos. De fato, este trabalho baseou a extensão da PDDL para a versão 2.1 com o trabalho de (FOX; LONG, 2003) para ações durativas.

Com uma proposta de lógica temporal de intervalos, Allen (1991) define a indexação dos predicados a intervalos indivisíveis, denominados instantes. Estes predicados devem ser verdadeiros nos instantes determinados para que uma ação seja aplicável. Allen foi motivado pela necessidade de tratar a execução simultânea de ações e por dois tipos de raciocínio: predição baseada em um conjunto de suposições e em planejamento, quais suposições um agente deve fazer sobre seu comportamento futuro e sobre cenários futuros. Além destes trabalhos, outras contribuições de Allen (1990) sobre lógica temporal de intervalo e raciocínio temporal foram muito importantes para evolução deste assunto na área.

De forma semelhante, o planejamento com base temporal (DEAN; McDERMOTT, 1987), conforme observado no capítulo 2, possui uma abordagem em planejamento com a indexação das proposições em intervalos, definindo assim quando estas devem ser válidas. Na representação do domínio, este intervalo é definido por um instante inicial e um instante final. Outra abordagem é o planejamento cronológico (GHALLAB; NAU; TRAVERSO, 2004) que utiliza funções de tempo representadas por variáveis de estado, pelas quais um domínio é descrito. Alguns planejadores foram desenvolvidos para esta abordagem, como o IxTeT System (GHALLAB; LARUELLE, 1994) que possui um gerenciador de álgebra de ponto e restrições temporais e o ParcPlan (EL-KHOLY; RICHARD, 1996) que tem o objetivo de tratar a execução de ações paralelas e recursos limitados.

Apesar dos avanços obtidos, dentro do contexto da engenharia do conhecimento, estes trabalhos estão mais próximos dos sistemas de planejamento e ainda trazem à tona os pontos levantados por McCluskey(2003), como a necessidade de um mecanismo claro para

raciocínio. Desta forma, poucos trabalhos voltados para o tratamento de aspectos temporais foram realizados em engenharia do conhecimento.

O GIPO (*Graphical Interface for Planning with Objects)* (SIMPSON, 2005) é um dos trabalhos em engenharia do conhecimento que está em desenvolvimento para utilizar uma modelagem baseada na PDDL + Nível 5 (FOX; LONG, 2001) com ações instantâneas e duração dos acontecimentos definida por processos e controlada por eventos. Diferentemente das ações, os processos não causam transição de estados, apenas mantêm os aspectos lógicos e alteram aspectos numéricos. Os eventos são funções de transição de estado que podem ter pré e pós-condições numéricas. São instantâneos e divergem das ações pelo fato de não integrarem uma solução gerada pelo planejador.

Calcada na UML, a proposta do presente trabalho é definir uma abordagem independente de uma linguagem de planejamento e focada em um processo de engenharia do conhecimento com diferentes níveis de abstração, mas baseada nos fundamentos de ações temporais, instantes e intervalos. Entretanto, a modelagem dos aspectos temporais faz parte de uma etapa do processo de definição de domínios que acontece depois de etapas preliminares que identificam os objetos, seus atributos e as ações que estão inseridas no contexto dinâmico do domínio.

4.3. Processo de definição de domínios

Para alcançar todos os detalhes temporais de um domínio, primeiramente é necessário realizar todo um processo de engenharia do conhecimento para identificar a estrutura que suporta o domínio e o dinamismo intrínseco aos seus elementos. Toda complexidade envolvida neste processo pode ser adequadamente tratada com o auxílio de modelos. Da mesma forma que o modelo de um sistema computacional é menos complexo e mais acessível que seus códigos e componentes (ARRINGTON, 2001), um modelo de domínio de planejamento também propicia a redução da complexidade em relação à especificação direta em uma linguagem de definição de domínios, como a PDDL (McCLUSKEY, 2003).

Durante o levantamento de informações a respeito do domínio, várias pessoas podem estar envolvidas e, não necessariamente, os especialistas no domínio devem conhecer como especificar o domínio em uma linguagem como a PDDL. Com a modelagem, e especialmente a modelagem visual, o modelo pode ser revisto com facilidade e as decisões podem ser tomadas rapidamente (ARRINGTON, 2001). Com o auxílio de uma ferramenta de engenharia do conhecimento como o itSIMPLE (*Integrated Tool Software Interface for Modeling Planning Environment*) (VAQUERO *et al,* 2006), a especificação de um domínio é realizada

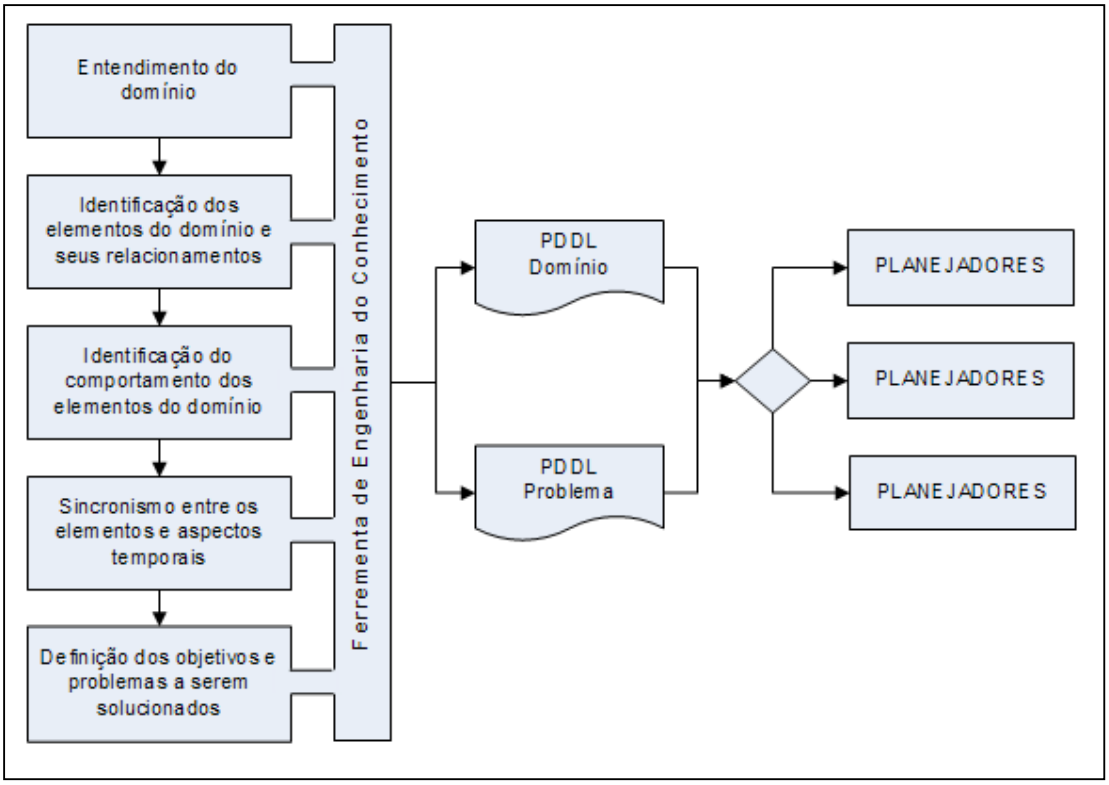


Figura 4.1 Fases no processo de definição de domínios de planejamento
Fonte: autor

por um processo com fases específicas, partindo de uma abstração maior, com a identificação do comportamento geral do domínio, posteriormente com a definição da estrutura e finalmente com os aspectos dinâmicos.

Na modelagem com UML.P, que é a linguagem utilizada pelo itSIMPLE, é possível visualizar o modelo a partir de diferentes perspectivas, uma delas é como a maior parte do sistema interage e coopera (ARRINGTON, 2001), outra é a visão que um especialista tem a partir do entendimento do comportamento geral do domínio através do diagrama de casos de uso. Com uma ferramenta de engenharia do conhecimento estas perspectivas são ampliadas, pois com uma integração com sistemas de planejamento e a partir da modelagem realizada, a solução para os problemas podem ser obtidas através dos planos gerados pelos planejadores, suprindo a visão de um executor que necessita destes planos para realizar tarefas.

A figura 4.1 ilustra separadamente cada etapa do processo de engenharia do conhecimento para domínios de planejamento. Na fase inicial de entendimento, os agentes do domínio são identificados bem como os cenários em que atuam. Cada cenário é representado por um caso de uso na UML.P e este possui as condições necessárias para sua execução e os

efeitos que serão obtidos após sua execução. Nesta fase, o comportamento geral do sistema é determinado pelo diagrama de casos de uso, que mostra toda interação do agente com os elementos que deverão compor o domínio.

A próxima fase é a identificação dos elementos estruturais do domínio. A partir dos casos de uso são identificadas as entidades do domínio e seus relacionamentos. A representação é realizada no diagrama de classes da UML.P e as classes referem-se aos objetos em um domínio de planejamento e seus atributos e relacionamentos referem-se aos predicados (VAQUERO, 2007). O diagrama de classes é fundamental para a fase posterior, que definirá a dinâmica do domínio. Cada classe relevante será uma entidade com estados definidos em um ciclo de vida.

A identificação do comportamento dos elementos do domínio é realizada pelo diagrama de máquina de estados, que define o comportamento de cada classe. Neste diagrama, para cada classe relevante, são definidos os estados e as mudanças destes estados são realizadas pelas ações. Uma ação está entre dois estados de uma classe, sendo que o estado de origem armazena as condições para a execução da ação e o estado de destino os efeitos após a conclusão. Como uma ação pode possuir diversos objetos como parâmetros, é possível que a mesma ação esteja presente em mais de um diagrama de máquina de estados.

A próxima etapa é a definição do sincronismo entre os elementos e os aspectos temporais e é realizada a partir dos estados definidos no diagrama de máquina de estados, que são transferidos para o diagrama de tempo. Assim, é possível relacionar instantes aos acontecimentos, determinando o início e fim de um evento ou a duração de uma ação. A partir deste diagrama, é possível visualizar o tempo de permanência dos objetos nos estados em função do tempo de execução das ações que possuem recursos compartilhados.

A definição dos objetivos e problemas a serem solucionados refere-se à última etapa do processo. Através do diagrama de objetos, são definidos os estados objetivos a serem alcançados a partir dos estados origem. Com a conclusão de todas as etapas, a ferramenta de engenharia do conhecimento que dá suporte ao processo possui parâmetros suficientes para realizar a tradução para PDDL, submetê-la aos planejadores e dispor uma análise do plano gerado.

De uma forma abrangente, o processo de engenharia do conhecimento para planejamento fora apresentado e a partir deste ponto é possível observar onde a modelagem dos aspectos temporais está localizada no processo de engenharia do conhecimento. Nas

seções seguintes, a proposta de uso do diagrama de tempo e sua integração à UML.P será detalhada.

4.4. Elementos de um domínio temporal

Um domínio de planejamento é composto por características temporais e atemporais. As características atemporais estão baseadas no modelo conceitual de planejamento de um sistema de transição de estados $\Sigma = (S, A, E, \gamma)$ (DEAN; WELLMAN, 1991) conforme apresentado no capítulo 2, onde:

- $S = \{s_1, s_2,...\}$ é um conjunto finito de estados;
- $A = \{a_1, a_2,...\}$ é um conjunto finito de ações;
- $E = \{e_1, e_2,...\}$ é um conjunto finito de eventos;
- $\gamma : S \times A \times E \rightarrow 2^S$ é uma função de transição de estados.

Uma generalização deste modelo para a especificação de seqüências temporizadas fora proposta em (HENZINGER; MANNA; PNUELI, 1992) com a indexação de intervalos mínimos e máximos para cada transição de estado. Este modelo generalizado, denominado sistema temporizado de transição de estados (*timed state transition system*) é descrito como $\Sigma = (S, A, E, \gamma, l, u)$ onde:

- $l_\gamma \in N$ (conjunto dos números inteiros) é um intervalo mínimo para cada transição $_\gamma \in \gamma$ e $l_{\gamma l} = 0$;
- $u_\gamma \in N \cup \{\infty\}$ é um intervalo máximo para cada transição $_\gamma \in \gamma$ tal que $u_\gamma \geq l_\gamma$ para todo $_\gamma \in \gamma$.

Embora os sistemas de planejamento sejam baseados nos sistemas de transição de estados (GHALLAB; NAU; TRAVERSO, 2004), um sistema de planejamento com características temporais baseado em um sistema temporizado de transição de estados pode ter uma precisão maior devido a indexação de instantes às proposições para determinar quando estas devem ser válidas (DEAN; McDERMOTT, 1987), além da especificação de um intervalo mínimo e máximo para cada transição.

Com base nestas afirmações, faz-se necessária a complementação da definição de um domínio temporal de planejamento que permita uma modelagem abrangente, flexível e suportada por uma ferramenta de engenharia do conhecimento. Além dos pontos temporais,

os elementos estruturais como objetos e variáveis necessitam ser considerados nesta definição. A próxima seção apresentará esta definição e também o diagrama de tempo proposto.

4.5. O Diagrama de tempo da UML.P

No processo de modelagem de domínios, a fase de aquisição de conhecimento é responsável pelo levantamento das informações que futuramente irão direcionar o melhor tipo de solução, considerando questões como escalonamento, tempo, consumo de recursos e outros aspectos descritos em (HETZBERG, 1996). Como a UML possui um propósito geral de modelagem, o uso de seu diagrama de tempo com enfoque em planejamento proporciona um suporte a esta fase inicial.

Considerando a representação de tempo através de uma linha com os momentos indexados, o sincronismo do domínio é definido por momentos comuns a diferentes objetos onde estes permanecem em um determinado estado respeitando as relações temporais definidas por Allen (1983). Além do tempo de permanência de um objeto em um determinado estado, o tempo de execução das ações que alteram estes estados também deve ser considerado.

O diagrama de tempo proposto para UML.P deve manter a referência temporal entre os estados e as classes do domínio. Através de uma linha de tempo válida para todas as classes, os instantes são indexados aos acontecimentos e os intervalos são definidos pelo tempo de permanência nos estados e pelo tempo de execução das ações. Com esta abordagem, as dependências entre os objetos são facilmente visualizadas. Para uma abordagem mais específica, posteriormente será apresentado o diagrama de tempo capaz de modelar o contexto de uma ação temporal.

A montagem do diagrama de tempo é feita a partir do diagrama de classes e dos diagramas de máquina de estados, pois todas as classes modeladas, seus estados e ações podem participar de um contexto temporal em um diagrama. Depois de analisado o domínio em um nível mais elevado de abstração com o auxílio dos casos de uso, as entidades do domínio são identificadas no diagrama de classes e o diagrama de máquina de estados define os estados para cada classe relevante no modelo. A partir deste ponto o diagrama de tempo pode ser modelado.

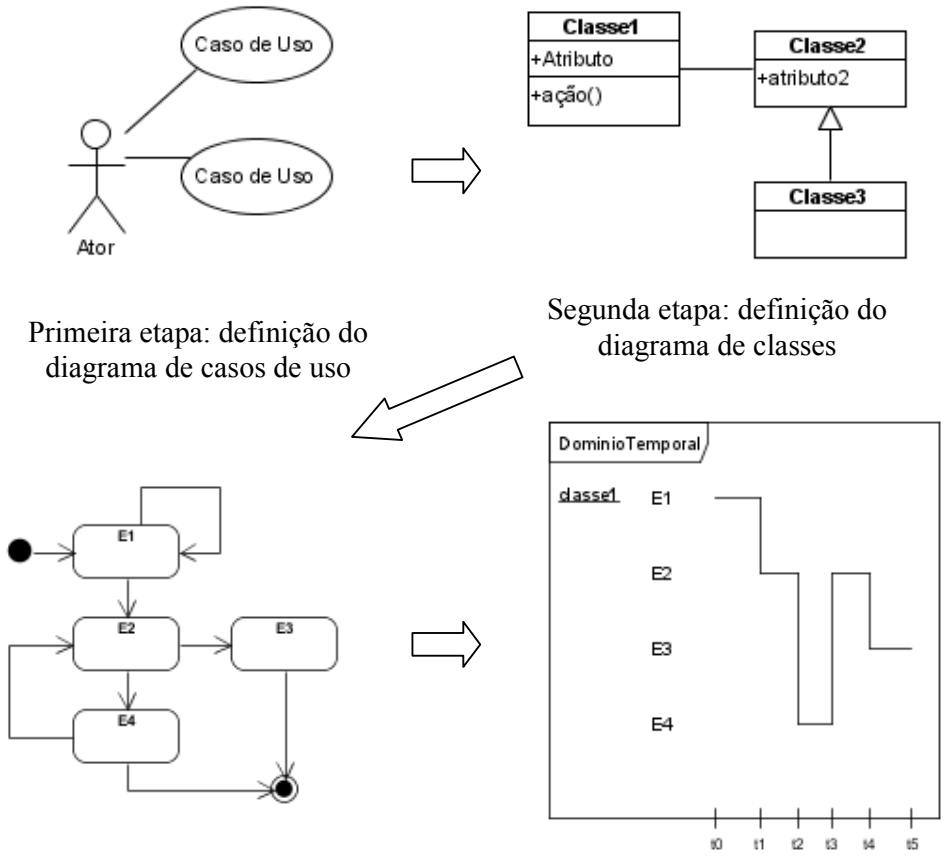


Figura 4.2 Seqüência de definição de um domínio em UML.P
Fonte: autor

A figura 4.2 ilustra a seqüência necessária para a definição de um domínio de planejamento em UML.P. Depois de apresentada a condição necessária para que o diagrama de tempo seja criado, é possível definir os elementos que compõem este diagrama:

Definição 4.1 - Elementos do diagrama de tempo

Um diagrama de tempo da UML.P é composto pelos elementos:

$D_t = \{O, E, T, A, D, Tc\}$ Onde:

- O = conjunto de objetos $o_1, o_2, ..., o_n \in O$ que participam do contexto temporal do domínio.
- E = {S,α } define um conjunto de estados
    - S é um conjunto de estados tal que $s_1, s_2, ..., s_n \in S$.

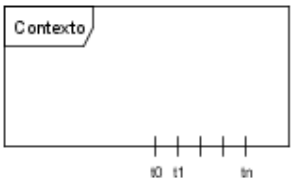
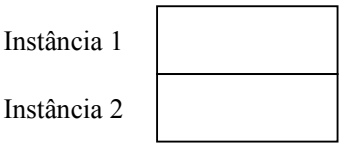
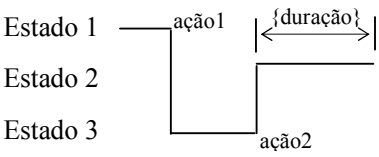
- α: S → O onde α é uma relação que mapeia cada estado ao seu respectivo objeto com ciclo de vida.

- T = $\{t_0, t_1, ..., t_n\}$ Conjunto de pontos no tempo que formam uma linha cronológica, tal que $t_0 < t_1 < ... < t_n$
- A = $\{A_t, \beta\}$ define um conjunto de ações
  - A_t é um conjunto de ações temporais
  - β: $(a_i \mid a_i \in A_t) \rightarrow (t_i \mid t_i \in T)$ onde β é uma relação que mapeia as ações aos pontos no tempo.
- D = $\{R, \gamma\}$ define um conjunto de restrições de duração
  - R é um conjunto de restrições
  - γ: R → i x β, tal que $i = \{(t_i, t_j) \mid t_i, t_j \in T \wedge t_i < t_j\}$ onde γ é uma relação que mapeia as restrições de duração a um par de pontos e esta relação é mapeada às ações pela relação β.
- Tc = $\{R_t, \kappa\}$ define um conjunto de restrições temporais
  - R_t é um conjunto de restrições que definem o início de uma ação
  - κ: R_t → E onde κ é uma relação que mapeia cada restrição temporal a uma ação.

Depois de definidos os elementos do diagrama, é possível analisar a representação gráfica a partir da UML e a adaptação para UML.P. Composto primeiramente por um quadro com o nome do domínio/ação no canto superior esquerdo, este quadro é dividido em partes conforme a quantidade de instâncias de classes modeladas previamente nos diagramas de máquina de estados, onde cada parte permite modelar o ciclo de vida da respectiva instância. A adaptação da UML para o contexto de planejamento sugere a estipulação da dependência entre os diagramas de classe e máquina de estados e o foco na modelagem do contexto do domínio ou de uma ação temporal.

O intervalo de duração das ações ou o tempo de permanência de uma instância em um estado é determinado por dois ou mais pontos no tempo. A permanência da instância em um estado é visualizada por um segmento de reta que passa pelos pontos definidos na linha de tempo. A partir do início de uma ação, é possível definir a duração com o auxílio da linguagem OCL (*Object Constraint Language*). Na tabela 4.1 são apresentados os elementos que compõem o diagrama e a participação de cada um na modelagem.

Tabela 4.1 Elementos do diagrama de tempo da UML.P

Tipo	Notação	Descrição
Quadro		O quadro define um contexto temporal. Pode ser aplicado na modelagem do domínio, contendo as instâncias de todos os objetos relevantes, ou na modelagem de uma ação específica, contendo apenas as instâncias dos objetos que participam da ação. O nome do contexto deve ser descrito no canto superior esquerdo e não deve haver mais que um diagrama com o mesmo nome. Cada quadro possui uma linha de tempo com instantes indexados. É possível dividir a linha de tempo conforme a quantidade de instantes necessários para o contexto. Na figura ao lado, modelado como t0, t1,..., tn.
Ciclo de vida		O ciclo de vida de uma instância define a participação de uma determinada classe no contexto temporal. As classes participantes devem estar presentes em um diagrama de máquina de estado. Uma instância nunca é repetida em um quadro e o número máximo de instâncias é a quantidade total de classes.
Linha de tempo para os estados de uma determinada instância		A linha de tempo permite visualizar o instante em que as ações são disparadas e seu tempo de duração. A linha na posição horizontal significa o tempo de permanência do objeto no estado. Os estados são obrigatoriamente oriundos dos diagramas de máquinas de estados. A duração da ação ou o tempo de permanência em um estado podem ser definidos e visualizados no diagrama e cada duração é definida na linguagem OCL (*Object Constraint Language*). As mudanças estão atreladas aos instantes definidos no quadro.

Fonte: Adaptado do diagrama de tempo da UML (OMG, 2003)

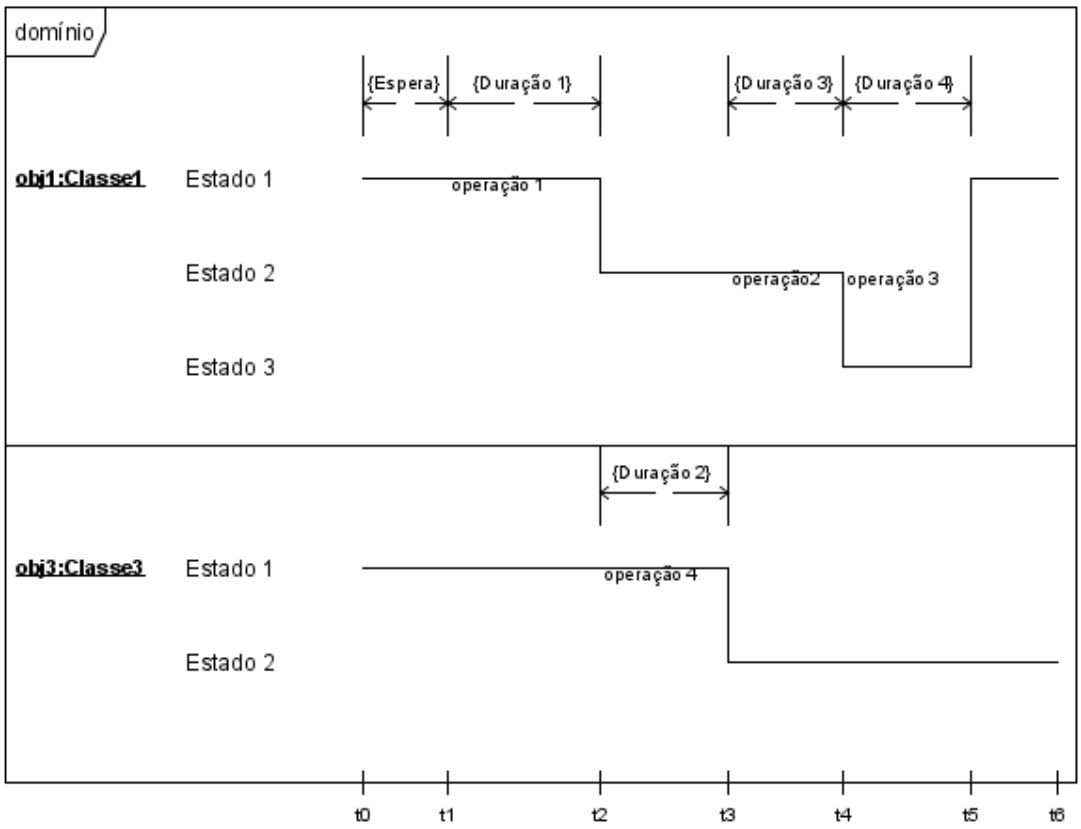


Figura 4.3 Diagrama de tempo da UML.P para os objetos do tipo Classe1 e Classe3
Fonte: autor

A coesão entre os elementos do diagrama define um contexto temporal. Uma linha de tempo é compartilhada entre todas as instâncias de classes. Desta forma, é possível analisar a situação de todas as entidades em um determinado instante ou intervalo de tempo. A figura 4.3 ilustra a modelagem do contexto temporal para os objetos criados a partir de duas classes (classe1 e classe3).

O processo é iniciado com a instância obj1 no 'Estado 1', em t_0, permanecendo sem nenhuma ação até t_1, quando a ação 'operação 1' é iniciada. A duração desta ação é indicada por '{Duração 1}' e está entre t1 e t2. A instância da classe 3 permanece no estado 1 (não é o mesmo estado da instância da classe 1) do início até a conclusão da operação 1, então, a operação 4 é iniciada e levará esta instância ao estado 2 após o tempo de duração '{Duração 2}'. Neste tempo, a instância da classe 1 permanece no 'Estado 2'. O ciclo de vida da instância da classe 3 se encerrará neste estado, enquanto a instância da classe 1 passará pelas operações 2 e 3 e pelo 'Estado 3', com duração '{Duração 3}' e '{Duração 4}'

respectivamente, retornando então para o ‘Estado 1’. Este contexto descreve um cenário geral, com a participação de todos os elementos que influenciam o ciclo de vida do domínio.

Embora a representação geral dos aspectos temporais seja fundamental para o entendimento do domínio, a precisão obtida por este nível de abstração não é suficiente para visualizar quando um determinado atributo deve ser válido ao longo de uma ação e como uma variação numérica deve evoluir. Por esta razão, torna-se necessário um diagrama que permita modelar o contexto de uma ação temporal. No próximo tópico será apresentada uma variação do diagrama de tempo com foco nestas ações.

4.6. Representação de ações temporais

A análise dos aspectos temporais em um contexto global é importante para identificar o dinamismo de um domínio. Entretanto, além desta análise, é necessário avaliar um contexto localizado, formado pelas ações temporais, que pode representar três tipos diferentes de abordagem: ações discretas, que permitem somente um número finito de atividades entre dois pontos quaisquer; ações de efeito contínuo, que possuem uma variável numérica que cresce ou decresce conforme um fator de tempo; ações com duração restrita por uma inequação, que possibilitam que fatores externos estejam envolvidos na determinação da extensão temporal (FOX; LONG, 2003).

Para modelar as ações através de um diagrama, é necessário representar os momentos em que os atributos assumem diferentes valores ao longo da execução da ação, bem como a própria duração da ação. Uma proposta de diagrama deve considerar todos os objetos que fazem parte da ação e seus atributos sob alguma forma de representação de tempo. Outro aspecto a ser considerado é a concepção principal das ações: elas levam um ou mais objetos de um estado original a um estado destino. Para que isto ocorra, os valores dos atributos (ou fluentes) devem atender aos critérios de precondição e, durante a evolução da ação, estes atributos têm seus valores alterados gerando os efeitos esperados, ou pós-condições.

A partir destas afirmações, para possibilitar a representação gráfica de cada elemento, é necessário definir exatamente quais são os elementos que devem compor um diagrama de tempo para ações. A definição 4.2 abaixo elenca todos estes elementos para a proposta da UML.P.

Definição 4.2 Elementos do Diagrama de Tempo para ações. Um diagrama de tempo para ações temporais é composto por:

EDT = {N, T, P, D} onde:

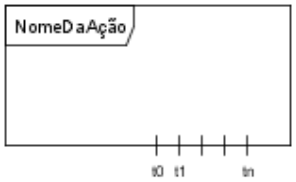
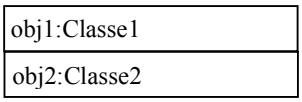
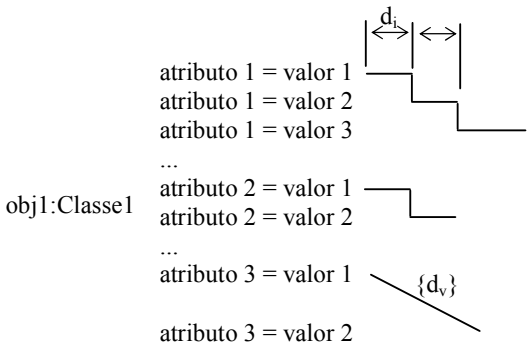
- N = Nome da ação;
- T = $\{t_0, t_1,..., t_n\}$ Conjunto de pontos no tempo que formam uma linha cronológica específica para a ação, tal que $t_0 < t_1 < ... < t_n$.
- P = conjunto de parâmetros da ação, tal que P = {I, A, R} onde:
  - I = Conjunto de instâncias de classes (objetos) que fazem parte dos parâmetros da ação;
  - A = Conjunto de atributos instanciados;
  - R = Função que etiqueta cada atributo $a_i \in A$ à uma instância de classe $i_n \in I$.
- D = conjunto de restrições temporais, tal que $D = \{D_a, D_i, D_v, M\}$, onde:
  - D_a = Restrição de duração da ação;
  - D_i = Conjunto de restrições de permanência dos atributos instanciados;
  - D_v = Conjunto de restrições para variação contínua dos atributos;
  - M = Função que etiqueta cada restrição temporal aos pontos que formam a linha cronológica T, onde $M = \{m_i \times t_i\}$

Com base nos elementos desta definição, é possível analisar uma proposta de interpretação gráfica que permita a modelagem visual para estes elementos. Ao longo da execução de uma ação, os resultados podem ocorrer a partir do instante em que a ação é iniciada e as condições podem ser requeridas em qualquer instante até o limite do encerramento da ação. Como vários atributos participam de uma ação, é possível definir o comportamento de cada um deles. A duração da ação pode ser determinada de duas formas:

1. Definição direta: Pela definição da duração entre o instante inicial e o instante final. Neste caso, uma expressão OCL define a duração e as expressões relacionadas aos atributos podem interferir como mecanismos de segurança para garantir uma execução correta.
2. Definição indireta: Pelo somatório de todas as definições realizadas em cada atributo. Neste caso, não é especificada diretamente a duração na ação e a complexidade é significativamente aumentada.

A tabela 4.2 apresenta os componentes do diagrama para ações temporais a partir da definição dos elementos do diagrama.

Tabela 4.2 Elementos para notação do diagrama para ações temporais na UML.P

Notação	Descrição
	O quadro para representar a ação temporal é idêntico ao quadro para representar o contexto do domínio. A linha de tempo refere-se ao ciclo de vida da ação e esta é única para todos os parâmetros da ação.
	Cada parâmetro, ou instância de classe, participa do ciclo de vida da ação, cada um com sua divisão própria. A forma de representação é idêntica ao diagrama anterior.
	Diferentemente do diagrama anterior, as linhas de tempo são definidas para cada atributo valorizado da instância. Um segmento de reta horizontal define o período em que um atributo é válido. Este período pode ser limitado por uma expressão OCL (*Object Constraint Language*) representada pelo elemento d_i na notação ao lado. Quando há variação contínua ao longo do tempo, um segmento de reta oblíquo pode designar a variação. Neste segmento, será relacionada uma expressão OCL (representado por d_v) com uma regra ou função que defina a variação contínua.

Fonte: autor

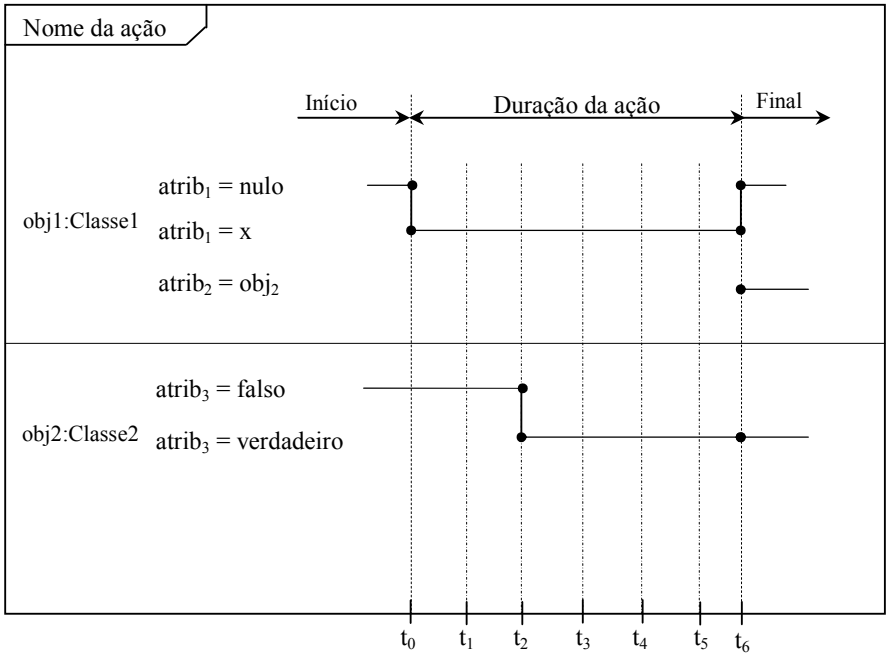


Figura 4.4 Diagrama para representação de ações temporais
Fonte: autor

A figura 4.4 ilustra todos estes componentes de forma coesa. Os instantes (t_0,..., t_5) que compõem a duração da ação podem ser definidos em uma linha de tempo válida para o intervalo de execução, sendo possível representar as definições de pré e pós-condições, mas sem referenciar qualquer instante, pois estes momentos não necessitam de um detalhamento. Os objetos que compõem a ação são tratados separadamente e cada atributo do objeto tem uma representação na linha de tempo, que é limitada à duração da ação. Desta forma, é possível determinar exatamente os instantes em que estes atributos são válidos ao longo da execução da ação. É possível observar neste exemplo que '$atrib_1$ = nulo' é precondição para a execução da ação e durante toda a execução da ação '$atrib_1$ = x'. Após o encerramento da ação, '$atrib_1$' retorna ao valor nulo.

Com esta abordagem, é possível aproximar a modelagem ao o planejamento com operadores temporais apresentado no capítulo 2, de forma que os instantes determinados correspondam aos instantes em uma base temporal, aumentando consideravelmente a precisão. Um exemplo deste diagrama será apresentado no capítulo 6 com a modelagem do estudo de caso do domínio de tratamento térmico de metais.

No próximo tópico será apresentado como o diagrama de tempo pode tratar a questão da variação numérica de um atributo com dependência direta da duração da ação.

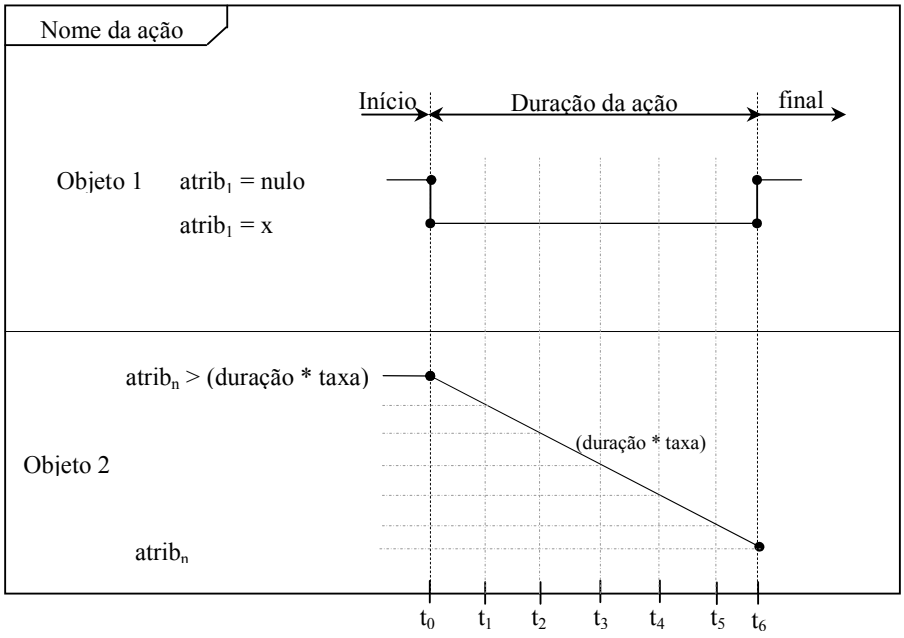


Figura 4.5 Diagrama para representação de ações temporais com variação numérica contínua
Fonte: autor

4.6.1. Variação numérica contínua ao longo do tempo

Neste cenário, a variação do valor do atributo está condicionada ao tempo de duração da ação e os valores dos atributos não são influenciados por fatores externos, não são observáveis ao longo da execução da ação e não estão sujeitos a alteração (FOX, LONG, 2003). A figura 4.5 ilustra a modelagem de uma ação com variação numérica ao longo de sua execução. A ação tem uma duração determinada (valor ou fórmula) e o valor do atributo $atrib_n$ antes da execução da ação deve ser maior que (duração * taxa). Desta forma, ao final da ação seu valor será positivo mesmo que seja reduzido durante o ciclo de vida da ação. No término da ação, o atributo terá o valor: $atrib_n = atrib_n - (duração * taxa)$.

Esta modelagem possibilita a visualização da variação de um atributo como um gráfico de uma função que define o valor de variável para cada ponto de tempo. No próximo tópico, será apresentada uma modelagem que permite que eventos externos à ação alterem os valores das variáveis e o tempo de execução da ação.

4.6.2. Ações durativas de efeito contínuo

Os atributos cujos valores sofrem variação contínua podem ser avaliados em qualquer instante durante a execução da ação. Diferentemente das ações com variação numérica contínua, as ações durativas de efeito contínuo podem ser afetadas por fatores externos, causando uma alteração nos valores dos atributos, embora estes continuem a sofrer os efeitos contínuos da ação. Este tipo de modelagem permite analisar o comportamento do domínio

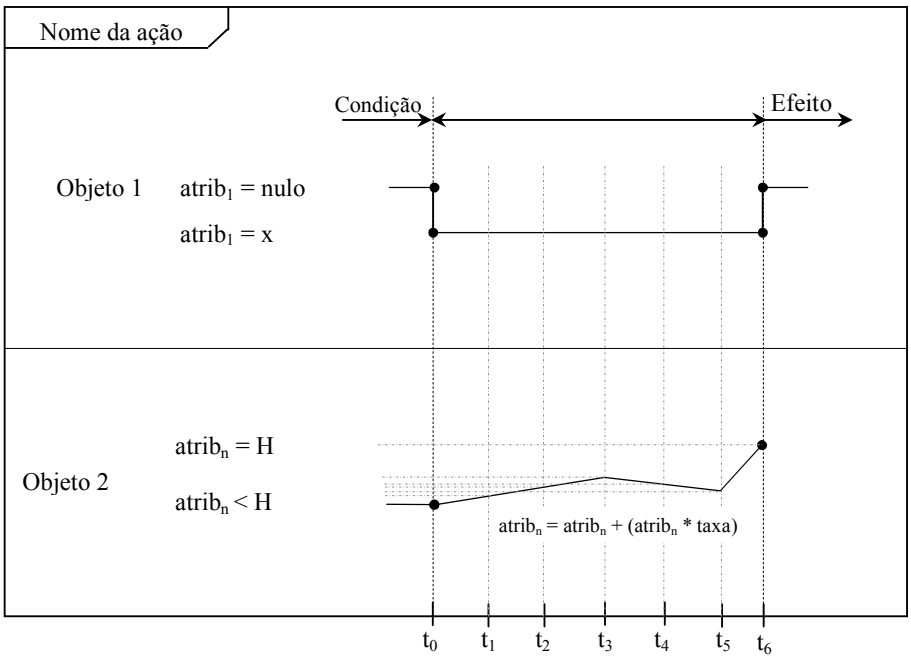


Figura 4.6 Diagrama para representação de ações durativas de efeito contínuo.
Fonte: autor

quando duas ações diferentes afetam a mesma variável. Por exemplo, no domínio de tratamento térmico, a ação de aquecer o forno eleva a temperatura interna, porém, se o forno tiver sua porta aberta enquanto estiver aquecendo, a temperatura irá cair, mas o forno continuará em processo de aquecimento e o tempo da ação será afetado.

A figura 4.6 ilustra uma ação com efeitos contínuos. Nesta figura, a duração da ação está condicionada ao tempo em que o atributo $atrib_n$ alcança a temperatura H. Além da taxa de acréscimo, fatores externos podem interferir nos valores deste atributo, neste exemplo, no momento t_3 o valor do atributo foi afetado por um fator externo, causando a redução no valor de $atrib_n$.

A proposta de modelagem de ações temporais dentro do escopo deste trabalho fora apresentada até este ponto. No próximo tópico, será abordada uma forma de modelagem necessária para um contexto específico, que se origina no diagrama de máquina de estados: tempo ramificado.

4.7. Tempo em estados ramificados

Em um sistema de transição de estado é possível que um estado seja origem para dois ou mais estados destinos. O diagrama de máquina de estados permite visualizar as seqüências

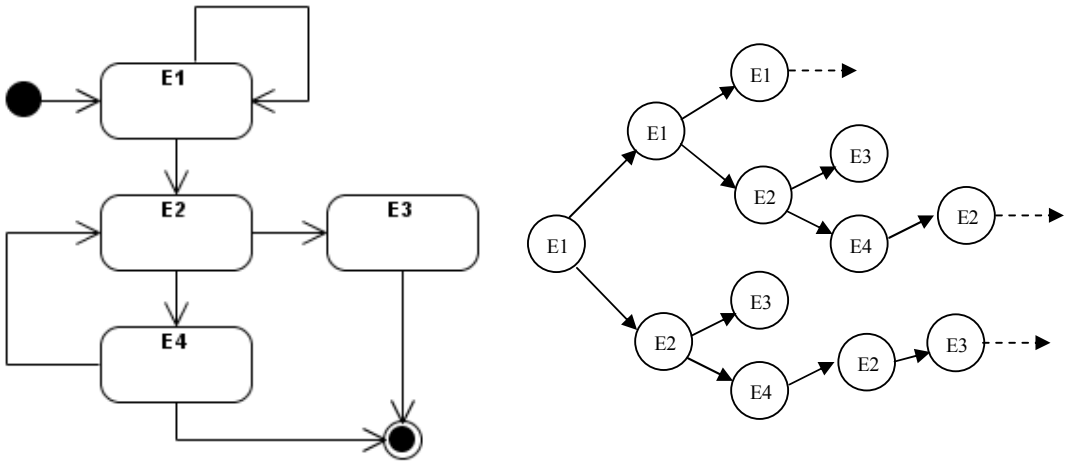


Diagrama de máquina de estados Árvore de computação

Figura 4.7 Exemplo de diagrama de máquina de estados e respectiva árvore de computação
Fonte: autor

possíveis a partir de um estado e todo dinamismo do domínio pode ser capturado por uma árvore de computação de profundidade ilimitada, cuja raiz é o estado inicial do diagrama (CUNHA, 2006). Cada uma destas seqüências participa de um contexto temporal específico e pode ser representado pelo diagrama de tempo proposto.

Na figura 4.7, os estados E1, E2 e E4 possuem mais que um estado destino. A respectiva árvore computacional está representada ao lado do diagrama e é possível observar as ramificações e as repetições geradas por estes três estados. Os elementos apresentados nesta figura compõem uma estrutura Kripke M = (S, S_I, R, V) (KRIPKE, 1963) onde S = conjunto não vazio de estados; S_I é o estado inicial; R $\subseteq$ SxS é uma relação de transição total que verifica $\forall s \in S . \exists s' \in S . (s, s') \in R$; V = S $\rightarrow$ 2^P função que etiqueta cada estado ao conjunto de proposições válidas nestes estados. Um caminho de M iniciado em $s_0 \in S$ é uma seqüência infinita de estados $\pi = s_0, s_1, s_2, ...$ onde $\forall i \geq 0 . (s_i, s_i+1) \in R$.

O conceito de tempo ramificado define várias combinações de estados para os mesmos objetos. Nos domínios de planejamento com enfoque temporal, um objeto não pode estar em dois ou mais estados no mesmo instante ou intervalo de tempo. Porém, vários objetos distintos podem ser avaliados no mesmo instante. As representações efetuadas no diagrama de tempo da UML.P são referentes aos estados dos objetos, onde cada estado é definido pelo agrupamento de predicados, que são avaliados durante o tempo de execução do domínio. Este diagrama não permite dois ou mais estados para um objeto no mesmo instante, assim, a representação temporal é linear.

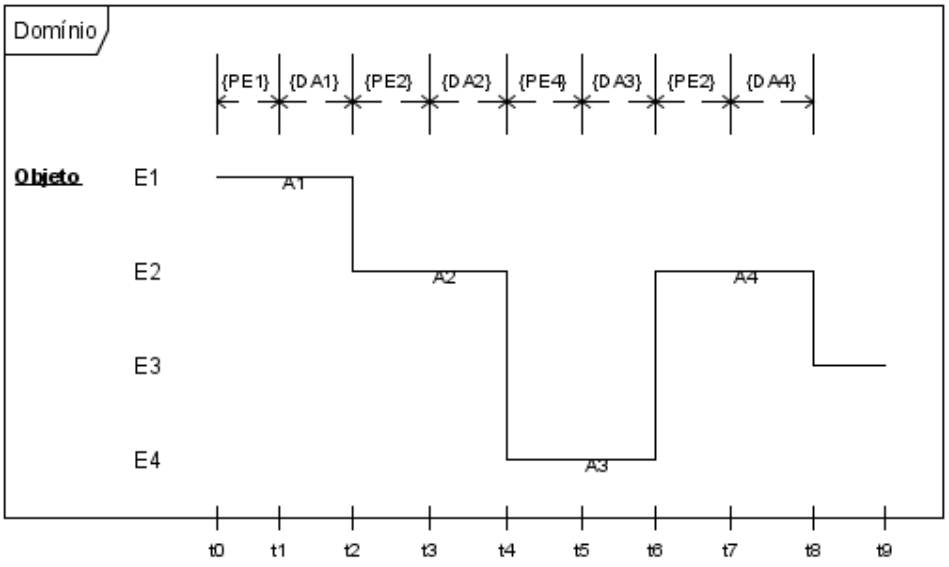


Figura 4.8 Diagrama de tempo correspondente a uma ramificação de estados
Fonte: autor

De um modo geral, o tempo contínuo presente neste tipo de contexto é definido pela repetição de uma seqüência de estados. O controle do tempo em cada seqüência possibilita a definição do ciclo de vida a partir das possíveis combinações de seqüências. Um estado pode atender a precondição para mais de uma ação e cada uma destas pode levar a uma linha de tempo diferente.

A figura 4.8 ilustra o diagrama de tempo para a ramificação E1, E2, E4, E2, E3 supracitada. Neste diagrama, PE1 refere-se ao tempo de permanência do objeto no estado E1, respectivamente PE2 para o estado E2 e PE4 para o estado E4. O estado E3 não possui uma especificação de permanência de tempo, pois é o último estado. Os tempos de duração das ações A1, A2, A3 e A4 estão referenciados por DA1, DA2, DA3 e DA4 respectivamente e o tempo total da ramificação é obtido pelo somatório de todas as durações (t_0,..., t_9).

O diagrama proposto também é capaz de representar um cenário em que diversas ramificações estão modeladas no mesmo diagrama. Para representar mais que uma ramificação no mesmo diagrama de tempo, todos os estados devem ter uma duração determinada e depois do último estado, o ciclo é reiniciado e o objeto retorna ao estado inicial. A partir deste ponto, cada ramificação subseqüente pode ter uma definição diferente do tempo de permanência do objeto no estado e as ações não sofrem alteração, pois são as mesmas em todos os cenários.

Desta forma, a partir do contexto apresentado, é possível afirmar que:

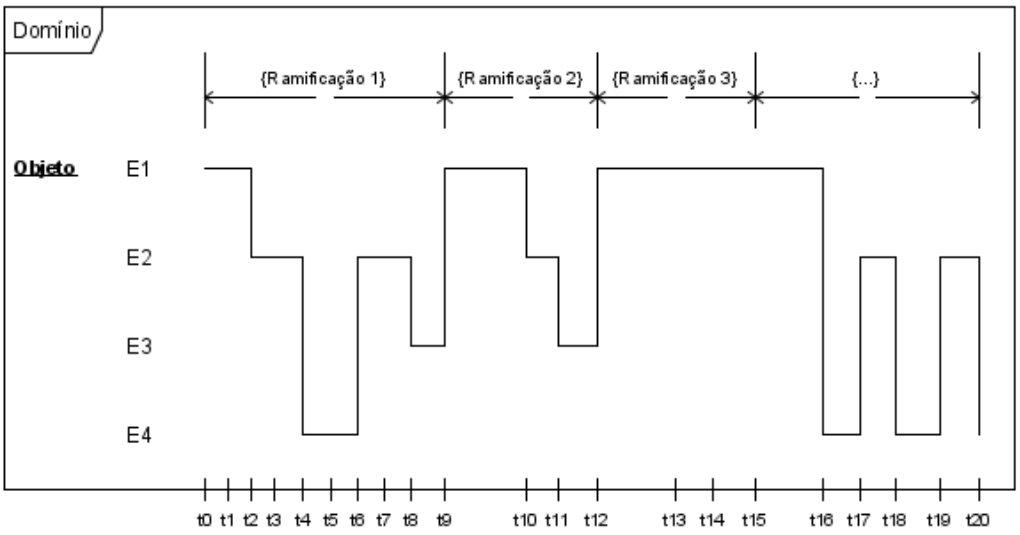


Figura 4.9 Diagrama de tempo correspondente a várias ramificações de estados
Fonte: autor

- Um sistema de transição de estados que possui uma seqüência de estados que configura uma ramificação pode ser representado por um diagrama de tempo capaz de representar os diversos ciclos de vida do sistema.
- Mesmo que em um sistema de transição de estados seja possível alcançar diversos estados a partir de um único estado, na representação temporal estes estados terão uma seqüência cronológica e todo objeto pertencente ao domínio está exatamente em um único estado s em um determinado instante t.
- As especificações temporais realizadas na modelagem definem os parâmetros necessários para que o planejador otimize o tempo dos planos a serem obtidos para resolução dos problemas.

A figura 4.9 ilustra um cenário em que diversas ramificações estão modeladas no mesmo diagrama. Todos os estados possuem uma duração determinada e é possível observar que o estado "E1" é o ponto inicial para cada início de ciclo. A partir deste ponto, cada ramificação subseqüente possui uma duração específica. Como a linha de tempo é contínua, será possível obter mais que um diagrama de tempo para um único diagrama de máquina de estados. O detalhamento de cada ação pode ser obtido com um diagrama específico.

Até este ponto foram apresentadas formas para representar tempo em planejamento e com uma proposta de modelagem baseada em UML, foram propostas duas variações do diagrama de tempo, uma capaz de modelar o contexto geral do domínio e outra capaz de representar uma ação temporal.

Embora o escopo proposto para o diagrama de tempo seja abrangente, é necessário comprovar sua capacidade de representar qualquer relação entre dois intervalos. A base para que isto seja possível é a verificação de sua equivalência com as relações de intervalos propostas por Allen (1983). Esta verificação será realizada no próximo tópico a partir da representação de cada relação através do diagrama de tempo proposto.

4.8. Abrangência do diagrama de tempo da UML.P

Para que o diagrama de tempo proposto seja aplicável, é necessário comprovar sua capacidade de modelar qualquer relação entre dois intervalos. A partir do trabalho de Allen (1983) apresentado no capítulo 2, que propõe que qualquer relação pode ser definida por sete relações mais suas relações inversas, neste tópico será apresentada uma proposta de equivalência entre o diagrama de tempo da UML.P e as relações temporais propostas por Allen. O resultado esperado é uma representação no diagrama da UML.P para cada relação temporal, consolidando a aplicabilidade do diagrama proposto.

A partir dos pontos extremos de dois intervalos (inicial e final), é possível determinar quais são as possíveis relações entre estes intervalos. Um domínio possui um conjunto de estados cujos elementos de uma relação de permanência podem ser dados por:

D = {O, E, T, R} onde:

- O = Conjunto de objetos do domínio {objeto$_1$, objeto$_2$,..., objeto$_n$}
- E = {S, α} define um conjunto onde:

  S = conjunto de estados onde {s_1, s_2,..., $s_n \in S$};

  α = S → O, onde α é uma relação que mapeia cada estado ao seu objeto.
- T = {t_0, t_1,..., t_n} Conjunto de pontos no tempo que formam uma linha cronológica, tal que $t_0 < t_1 < ... < t_n$.
- R = {λ, δ} onde:

  λ = S → T, onde λ é uma relação que mapeia cada estado $s \in S$ a um par de instantes t_i, $t_j \in T$ tal que $t_i < t_j$.

  δ = S → S, onde δ é uma relação de transição temporal que mapeia os intervalos de dois estados s_1 e s_2 conforme as restrições temporais definidas por Allen (1983) e apresentadas no capítulo 2, de forma que:

  - A^- é o instante inicial e A^+ é o instante final do intervalo A em que um objeto permanece no estado s_1;

- B^- é o instante inicial e B^+ é o instante final do intervalo B em que um objeto permanece no estado s_2.

As partir da relação de transição temporal δ tem-se:

((A *antes de* B) ∧ (B *depois de* A)) ⇒ $A^- < B^-$

(A é *igual* a B) ⇒ $(A^- = B^-) \wedge (A^+ = B^+)$

((A *encontra* B) ∧ (B *é encontrado por* A)) ⇒ $A^+ = B^-$

((A *durante* B) ∧ (B *contém* A)) ⇒ $(A^- > B^-) \wedge (A^+ < B^+)$

((A *inicia* B) ∧ (B *é iniciado por* A)) ⇒ $(A^- = B^-) \wedge (A^+ < B^+)$

((A *finaliza* B) ∧ (B *é finalizado por* A)) ⇒ $(A^- > B^-) \wedge (A^+ = B^+)$

((A *sobrepõe* B) ∧ (B *é sobreposto por* A)) ⇒ $(A^- < B^-) \wedge (A^+ < B^+) \wedge (A^+ > B^-)$

Os elementos do conjunto D supracitado estão presentes em um diagrama de tempo. A partir da modelagem realizada entre dois intervalos, é possível comparar esta modelagem com as definições das relações de Allen (1983) e certificar que o diagrama representa adequadamente um cenário do domínio.

Com a utilização de uma representação gráfica, a visualização das relações facilita identificar e definir os instantes que compõem um intervalo. A partir desta representação gráfica, a seguir, cada relação será analisada e interpretada conforme a disposição de dois intervalos no diagrama de tempo da UML.P. O resultado esperado é a obtenção de um diagrama para cada relação.

Desta forma, a verificação pode ser efetuada da seguinte forma:

Sejam: objeto1 e objeto2 dois objetos de um dado domínio;

x1, x2 e x3 são estados do objeto1

y1, y2 e y3 são estados do objeto2;

t0, t1, ..., tn os instantes do domínio que representam o início e fim dos intervalos.

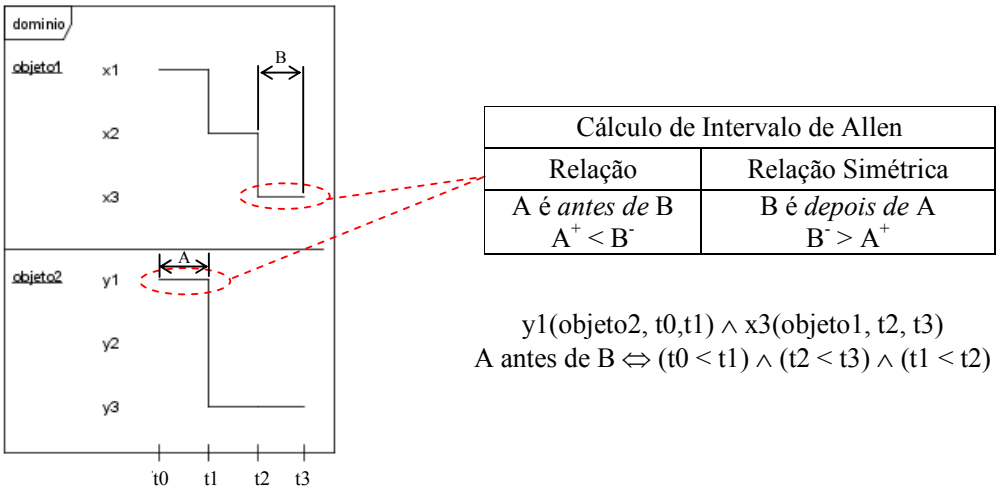


Figura 4.10 Diagrama de tempo para representação da primeira/segunda relação temporal de Allen
Fonte: autor

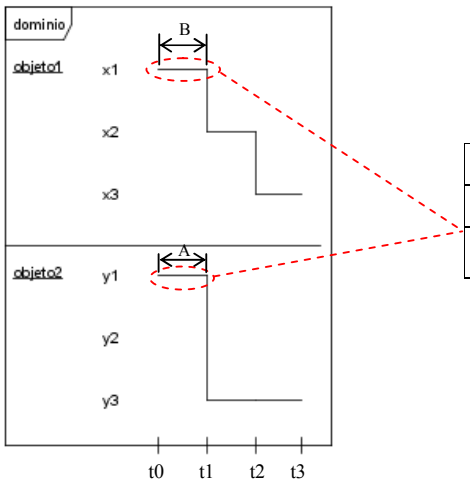
O diagrama de tempo apresentado na figura 4.10 ilustra a modelagem do comportamento de dois objetos com três estados: x1, x2 e x3 para o objeto 1 e y1, y2 e y3 para o objeto 2. A posição dos intervalos definidos por A e B referentes aos estados y1 e x3 corresponde à relação *antes de* proposta por Allen (1983). A partir desta representação temporal é possível definir o seguinte teorema:

Teorema 4.1: Dois intervalos A e B modelados em um diagrama de tempo D da UML.P com linha cronológica T = {t0, t1, t2, t3,..., tn} tal que t0 < t1 < t2 < t3 < ... < tn, representam a relação A *antes de* B se, e somente se, existir um intervalo não vazio entre A e B, cuja representação é dada por $A^+ < B^-$.

Prova:

- Seja A o intervalo definido pelos instantes t0, t1, sendo que $A^- = t0$ é o instante inicial de A e $A^+ = t1$ é o instante final de A e t0 < t1;
- Seja B o intervalo definido pelos instantes t2, t3, sendo que $B^- = t2$ é o instante inicial de B e $B^+ = t3$ é o instante final de B e t2 < t3;
- Supondo t0 < t1 < t2 < t3, então, existe um intervalo não vazio entre os pontos t2 e t3 que torna a prova decorrente da relação *antes de* demonstrada na tabela 2.2, no capítulo 2, onde $A^+ < B^-$. ou $B^- - A^+ > 0$.

Portanto, a relação *antes de* e sua relação inversa *depois de* podem ser modeladas pelo diagrama da UML.P através de dois intervalos disjuntos com um intervalo não vazio entre estes. □



Cálculo de Intervalo de Allen	
Relação	Relação Simétrica
A é *igual* a B	∅

$$y1(objeto2, t0,t1) \wedge x1(objeto1, t0, t1)$$
$$A = B \Leftrightarrow (t0 < t1)$$

Figura 4.11 Terceira relação temporal de Allen representada em um diagrama de tempo da UML.P
Fonte: autor

O diagrama de tempo apresentado na figura 4.11 ilustra a modelagem do comportamento de dois objetos que possuem três estados: x1, x2 e x3 para o objeto 1 e y1, y2 e y3 para o objeto 2. A posição dos intervalos A e B referentes ao tempo de permanência dos objetos nos estados x1 e y1 corresponde à relação *igual* proposta por Allen (1983). A partir desta representação temporal é possível definir o seguinte teorema:

Teorema 4.2: Dois intervalos A e B modelados em um diagrama de tempo D da UML.P com linha cronológica T = {t0, t1, t2, t3,..., tn} tal que t0 < t1 < t2 < t3 < ... < tn representam a relação A *igual a* B se, e somente se, estes intervalos forem estritamente idênticos, cuja representação é $(A^- = B^-) \wedge (A^+ = B^+)$.

Prova:

- Seja A o intervalo definido pelos instantes t0, t1, sendo que $A^- = t0$ é o instante inicial de A e $A^+ = t1$ é o instante final de A e t0 < t1;
- Seja B o intervalo definido também pelos instantes t0, t1, sendo que $B^- = t0$ é o instante inicial de B e $B^+ = t1$ é o instante final de B e t0 < t1;
- Supondo que o ponto inicial t0 é comum aos dois intervalos, ou seja, $A^- = t0 = B^-$ e o ponto final t1 também é comum aos dois intervalos, ou seja, $A^+ = t1 = B^+$, t2 < t3, então a prova torna-se decorrente da relação *igual* demonstrada na tabela 2.2, no capítulo 2, onde $(A^- = B^-) \wedge (A^+ = B^+)$.

Portanto, a relação *igual* pode ser modelada pelo diagrama de tempo da UML.P através de dois intervalos cujos instantes inicias e finais compartilham os mesmos pontos iniciais e finais. □

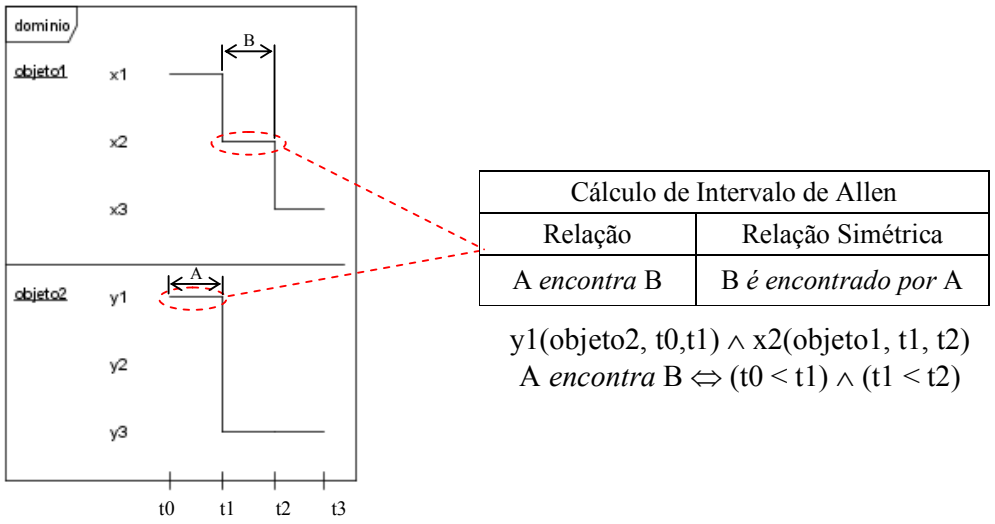


Figura 4.12 Quarta/quinta relação temporal de Allen representada no diagrama de tempo da UML.P
Fonte: autor

O diagrama de tempo apresentado na figura 4.12 ilustra a modelagem do comportamento de dois objetos que possuem três estados: x1, x2 e x3 para o objeto 1 e y1, y2 e y3 para o objeto 2. A posição dos intervalos A e B referentes ao tempo de permanência dos objetos nos estados x2 e y1 corresponde à relação *encontra* proposta por Allen (1983). A partir desta representação temporal é possível definir o seguinte teorema:

Teorema 4.3: Dois intervalos A e B modelados em um diagrama de tempo D da UML.P com linha cronológica T = {t0, t1, t2, t3,..., tn}, tal que t0 < t1 < t2 < t3 < ... < tn, representam a relação A *encontra* B se, e somente se, não existir um intervalo não vazio entre A e B, cuja representação é dada por $A^+ = B^-$.

Prova:

- Seja A o intervalo definido pelos instantes t0, t1, sendo que $A^- = t0$ é o instante inicial de A e $A^+ = t1$ é o instante final de A e t0 < t1;
- Seja B o intervalo definido também pelos instantes t1, t2, sendo que $B^- = t1$ é o instante inicial de B e $B^+ = t2$ é o instante final de B e t1 < t2;
- Supondo que o ponto t1 é comum aos dois intervalos, ou seja, $A^+ = t1 = B^-$, então não existe nenhum intervalo entre A^+ e B^-, assim, a prova torna-se decorrente da relação *encontra* demonstrada na tabela 2.2, no capítulo 2, onde ($A^+ = B^-$).

Portanto, a relação *encontra* se, pode ser modelada pelo diagrama de tempo da UML.P através de dois intervalos cujo instante final de um é idêntico ao instante inicial do outro. □

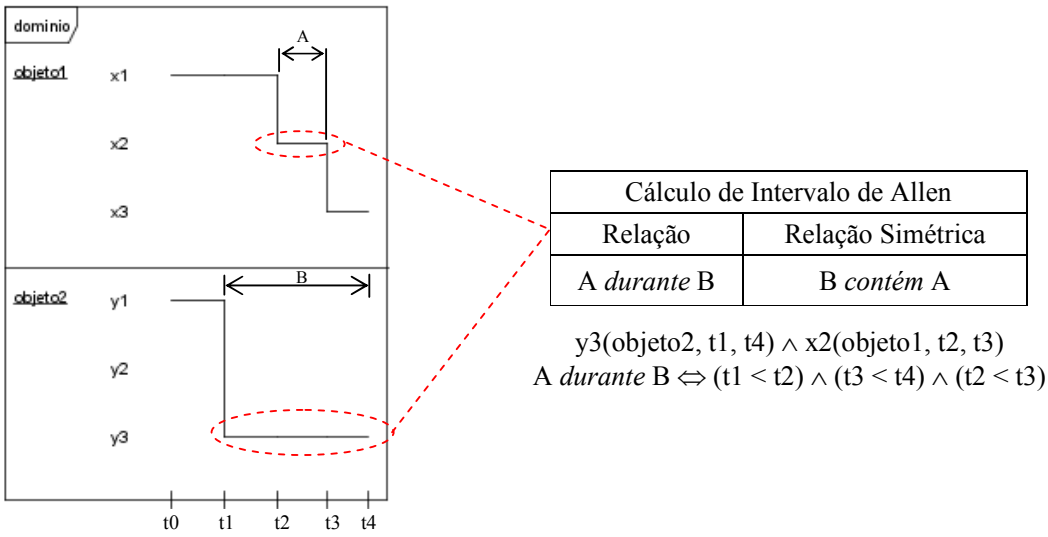


Figura 4.13 Sexta/sétima relação temporal de Allen representada no diagrama de tempo da UML.P
Fonte: autor

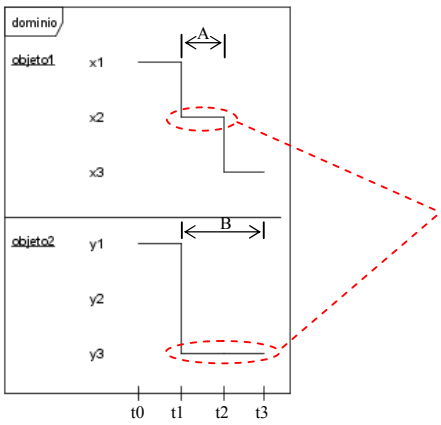
O diagrama de tempo apresentado na figura 4.13 ilustra a modelagem do comportamento de dois objetos que possuem três estados: x1, x2 e x3 para o objeto 1 e y1, y2 e y3 para o objeto 2. A posição dos intervalos A e B referentes ao tempo de permanência dos objetos nos estados x2 e y3 corresponde à relação *durante* proposta por Allen (1983). A partir desta representação temporal é possível definir o seguinte teorema:

Teorema 4.4: Dois intervalos A e B modelados em um diagrama de tempo D da UML.P com linha cronológica T = {t0, t1, t2, t3, t4..., tn}, tal que t0 < t1 < t2 < t3 < ... < tn, representam a relação A *durante* B se, e somente se, o ponto inicial de A ocorre depois do ponto inicial de B e seu ponto final ocorre antes do ponto final de B, cuja representação é dada por $(A^- > B^-) \wedge (A^+ < B^+)$ para t0 < t1 < .. < tn.

Prova:

- Seja A o intervalo definido pelos instantes t2, t3, sendo que $A^- = t2$ é o instante inicial de A e $A^+ = t3$ é o instante final de A e t2 < t3;
- Seja B o intervalo definido também pelos instantes t1, t4, sendo que $B^- = t1$ é o instante inicial de B e $B^+ = t4$ é o instante final de B e t1 < t4;
- Supondo um intervalo não vazio entre cada ponto $t \in T$, o ponto t1 ocorre antes do ponto t2, ou seja, A é iniciado depois B. O ponto t4 é maior que o ponto t3, ou seja, B é finalizado depois de A. Assim, a prova torna-se decorrente da relação *durante* demonstrada na tabela 2.2, no capítulo 2, onde $(A^- > B^-) \wedge (A^+ < B^+)$.

Portanto, a relação *durante* se, pode ser modelada pelo diagrama de tempo da UML.P através de dois intervalos de forma que um está estritamente contido no outro. □



Cálculo de Intervalo de Allen	
Relação	Relação Simétrica
A *inicia* B	B *é iniciado por* A

$y3(objeto2, t1,t3) \wedge x2(objeto1, t1, t2)$
A *inicia* B $\Leftrightarrow (t1 < t2) \wedge (t2 < t3)$

Figura 4.14 Oitava/Nona relação temporal de Allen representada pelo diagrama de tempo da UML.P
Fonte: autor

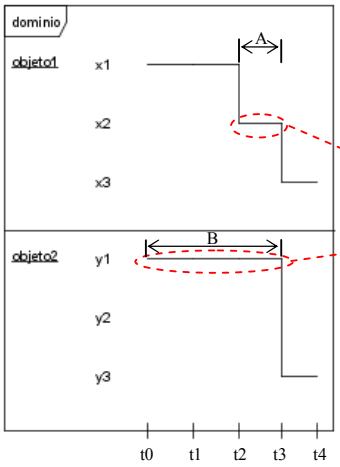
O diagrama de tempo apresentado na figura 4.14 ilustra a modelagem do comportamento de dois objetos que possuem três estados: x1, x2 e x3 para o objeto 1 e y1, y2 e y3 para o objeto 2. A posição dos intervalos A e B referentes ao tempo de permanência dos objetos nos estados x2 e y3 corresponde à relação *inicia* proposta por Allen (1983). A partir desta representação temporal é possível definir o seguinte teorema:

Teorema 4.5: Dois intervalos A e B modelados em um diagrama de tempo D da UML.P com linha cronológica T = {t0, t1, t2, t3,..., tn}, tal que t0 < t1 < t2 < t3 < ... < tn, representam a relação A *inicia* B se, e somente se, A possui o mesmo ponto inicial de B, mas está estritamente contido em B, cuja representação é dada por $(A^- = B^-) \wedge (A^+ < B^+)$.

Prova:

- Seja A o intervalo definido pelos instantes t1, t2, sendo que $A^- = t1$ é o instante inicial de A e $A^+ = t2$ é o instante final de A e t1 < t2;
- Seja B o intervalo definido também pelos instantes t1, t3, sendo que $B^- = t1$ é o instante inicial de B e $B^+ = t3$ é o instante final de B e t1 < t3;
- Supondo um instante t1 comum ao início dos intervalos A e B, um instante t2 maior que t1 em que o intervalo A se encerra e t3 o instante em que B se encerra e t1 < t2 < t3, então, tem-se um intervalo A que inicia um intervalo B, mas se encerra antes de B. Assim, a prova torna-se decorrente da relação *inicia* demonstrada na tabela 2.2, no capítulo 2, onde $(A^- = B^-) \wedge (A^+ < B^+)$.

Portanto, a relação inicia pode ser modelada pelo diagrama de tempo da UML.P através de dois intervalos com pontos iniciais idênticos e diferentes pontos finais. □



Cálculo de Intervalo de Allen	
Relação	Relação Simétrica
A *finaliza* B	B *é finalizado por* A

x2(objeto1, t2, t3) ∧ y1(objeto2, t0, t3)
A *finaliza* B ⇔ (t0 < t3) ∧ (t2 < t3)

Figura 4.15 Décima/décima primeira relação temporal de Allen representada em UML.P
Fonte: autor

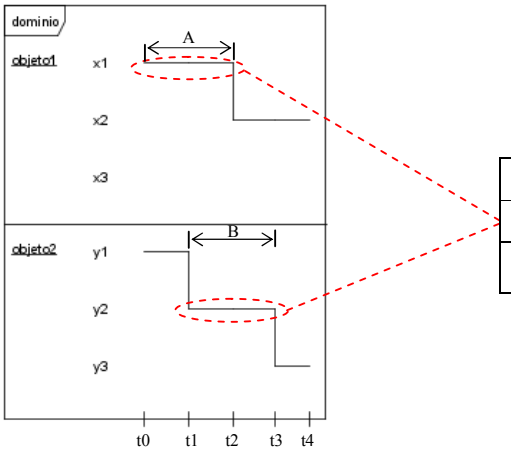
O diagrama de tempo apresentado na figura 4.15 ilustra a modelagem do comportamento de dois objetos que possuem três estados: x1, x2 e x3 para o objeto 1 e y1, y2 e y3 para o objeto 2. A posição dos intervalos A e B referentes ao tempo de permanência dos objetos nos estados x2 e y1 corresponde à relação *finaliza* proposta por Allen (1983). A partir desta representação temporal é possível definir o seguinte teorema:

Teorema 4.6: Dois intervalos A e B modelados em um diagrama de tempo D da UML.P com linha cronológica T = {t0, t1, t2, t3,..., tn}, tal que t0 < t1 < t2 < t3 < ... < tn, representam a relação A *finaliza* B se, e somente se, A possui o mesmo ponto final de B e está estritamente contido em B, cuja representação é dada por $(A^- > B^-) \wedge (A^+ = B^+)$.

Prova:

- Seja A o intervalo definido pelos instantes t2, t3, sendo que $A^- = t2$ é o instante inicial de A e $A^+ = t3$ é o instante final de A e t2 < t3;
- Seja B o intervalo definido pelos instantes t0, t3, sendo que $B^- = t0$ é o instante inicial de B e $B^+ = t3$ é o instante final de B e t0 < t3;
- Supondo um instante t3 comum ao fim dos intervalos A e B, um instante t2 em que o intervalo A se inicia e t0 o instante em que B se inicia e t0 < t1 < t2 < t3, então, tem-se um intervalo A que encerra um intervalo B e está estritamente contido em B. Assim, a prova torna-se decorrente da relação *finaliza* demonstrada na tabela 2.2, no capítulo 2, onde $(A^- > B^-) \wedge (A^+ = B^+)$.

Portanto, a relação finaliza pode ser modelada pelo diagrama de tempo da UML.P através de dois intervalos com pontos finais idênticos e diferentes pontos iniciais. □



Cálculo de Intervalo de Allen	
Relação	Relação Simétrica
A *sobrepõe* B	B *é sobreposto por* A

x1(objeto1, t0, t2) ∧ y2(objeto2, t1,t3)
A *sobrepõe* B ⇔ (t0 < t1 < t2 < t3)

Figura 4.16 Décima segunda/décima terceira relação temporal de Allen representada em UML.P
Fonte: autor

O diagrama de tempo apresentado na figura 4.16 ilustra a modelagem do comportamento de dois objetos que possuem três estados: x1, x2 e x3 para o objeto 1 e y1, y2 e y3 para o objeto 2. A posição dos intervalos A e B referentes ao tempo de permanência dos objetos nos estados x1 e y2 corresponde à relação *sobrepõe* proposta por Allen (1983). A partir desta representação temporal é possível definir o seguinte teorema:

Teorema 4.7: Dois intervalos A e B modelados em um diagrama de tempo D da UML.P com linha cronológica T = {t0, t1, t2, t3, t4,..., tn}, tal que t0 < t1 < t2 < t3 < ... < tn, representam a relação A *sobrepõe* B se, e somente se, A é iniciado antes de B e B é finalizado após A, cuja representação é dada por $(A^- < B^-) \wedge (A^+ < B^+) \wedge (A^+ > B^-)$.

Prova:

- Seja A o intervalo definido pelos instantes t0, t2, sendo que $A^- = t0$ é o instante inicial de A e $A^+ = t2$ é o instante final de A e t0 < t2;
- Seja B o intervalo definido pelos instantes t1, t3, sendo que $B^- = t1$ é o instante inicial de B e $B^+ = t3$ é o instante final de B e t1 < t3;
- Supondo t0 < t1 < t2 < t3, o início do intervalo A em t0 e o fim em t2 e o início de B em t1 e o fim em t3, então, o intervalo A inicia antes de B e também finaliza antes de B, determinando um deslocamento de B em relação à A. Assim, a prova torna-se decorrente da relação *sobrepõe* demonstrada na tabela 2.2, no capítulo 2, onde $(A^- < B^-) \wedge (A^+ < B^+) \wedge (A^+ > B^-)$.

Portanto, a relação *sobrepõe* pode ser modelada pelo diagrama de tempo da UML.P através de dois intervalos onde o primeiro é iniciado antes do segundo e finalizado durante este. □

Com base na modelagem apresentada, é possível afirmar que:

Teorema 4.8 Se as relações definidas por Allen (1983) são capazes de representar qualquer relação entre dois intervalos e o diagrama de tempo da UML.P é capaz de interpretar todas as relações definidas por Allen, logo, o diagrama de tempo também pode representar qualquer relação entre dois intervalos.

Prova:

A prova decorre naturalmente dos teoremas 4.1 a 4.7 que provam a correspondência entre cada relação de Allen com o diagrama de tempo da UML.P. Portanto, é possível concluir que o diagrama de tempo proposto é capaz de representar qualquer relação temporal entre dois intervalos. □

Com os teoremas apresentados até este ponto, se pode observar que a modelagem da relação entre dois intervalos pode ser realizada com o diagrama de tempo da UML.P. A partir desta proposta, se torna necessária uma interpretação deste diagrama em PDDL. No próximo tópico, será apresentada uma proposta para esta interpretação com base nos elementos apresentados até este ponto.

4.9. Interpretação do diagrama de tempo da UML.P em PDDL 2.1

Depois que o conhecimento sobre o domínio fora registrado, inicia-se uma nova fase no processo de engenharia do conhecimento: a correspondência de todos os registros realizados para uma linguagem que seja interpretada pelos planejadores. De fato, o processo não se completa se isto não for possível. Desta forma, este tópico abordará a interpretação da UML.P para a PDDL com foco na versão da PDDL 2.1 (FOX, LONG, 2003).

Uma tradução da UML para a PDDL proposta em (VAQUERO, 2007) compreende um processo que extrai cada elemento principal do modelo, converte em um formato intermediário de armazenamento denominado XPDDL (*Extensible Planning Domain Definition Language*) (GOUCH, 2004) e posteriormente traduz para PDDL. Esta abordagem fora aplicada na implementação da ferramenta itSIMPLE (VAQUERO, TONIDANDEL, SILVA, 2005) e pode também ser aplicada em um trabalho futuro de implementação do diagrama de tempo da UML.P nesta mesma ferramenta. Por hora, o foco será em como a modelagem em UML.P pode ser diretamente interpretada na PDDL.

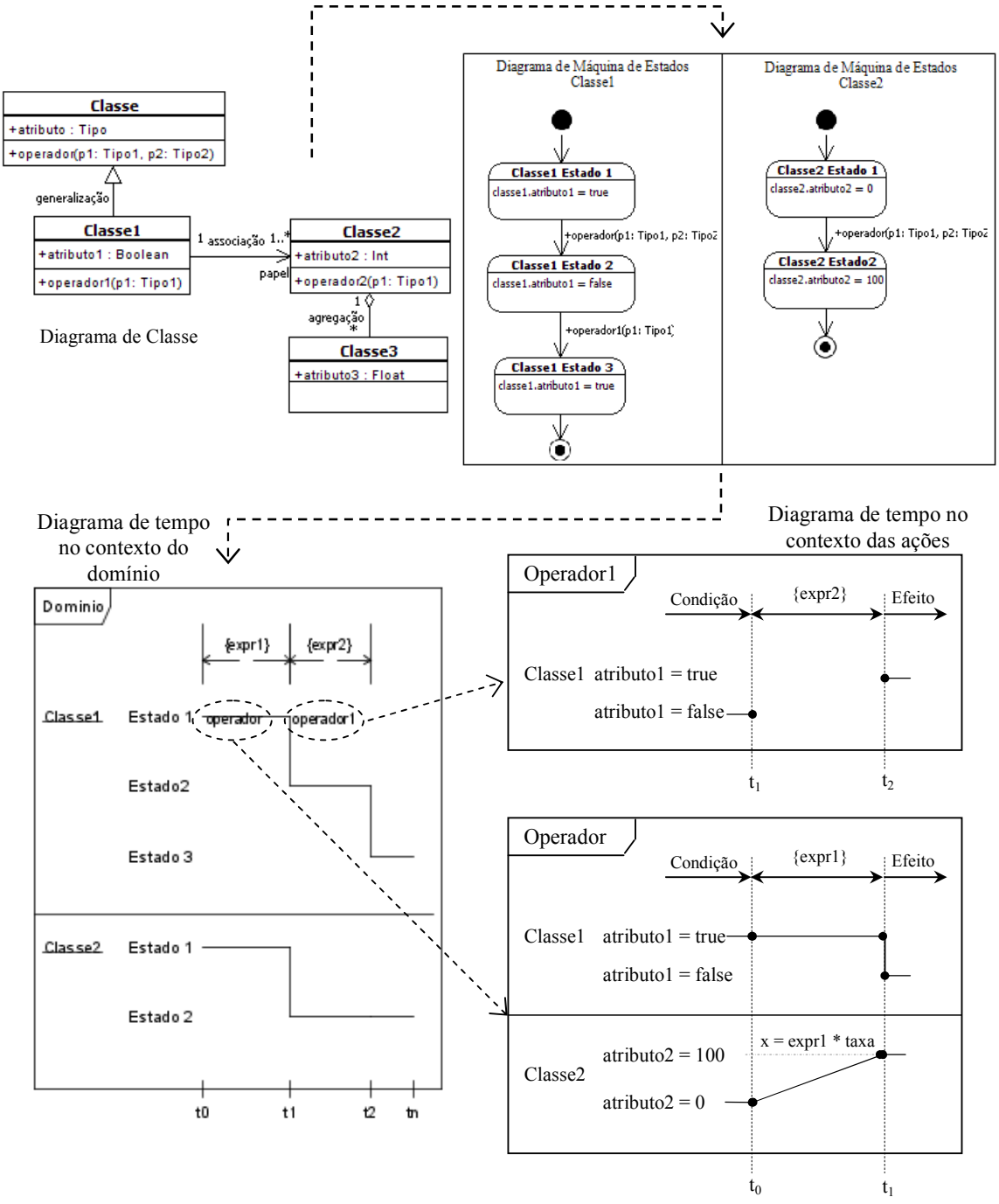


Figura 4.17 Seqüência de modelagem em UML.P
Fonte: Autor

A figura 4.17 ilustra a seqüência de modelagem em UML.P. Primeiramente são identificados os elementos do domínio e seus relacionamentos através do diagrama de classes. Posteriormente, são definidos os aspectos dinâmicos atemporais com o diagrama de máquina de estados, sendo um para cada classe relevante. Os aspectos temporais referentes ao contexto

do domínio são modelados pela junção dos diagramas de máquina de estados em um diagrama de tempo. O tempo de execução para 'operador' está definido pela expressão 'expr1'. Pelo diagrama de tempo, é possível observar que este é o tempo em que 'Classe1' permanece no 'Estado 1'. Cada ação relevante pode utilizar um diagrama de tempo para detalhamento. O próximo passo deste processo é a interpretação para PDDL.

O diagrama de tempo apresentado torna o dinamismo do diagrama de máquina de estados dependente do tempo, com a indexação dos acontecimentos a uma linha crescente em relação ao ciclo de vida do domínio. A interpretação em PDDL é realizada com base no conjunto de diagramas utilizados na modelagem do domínio, considerando inclusive o detalhamento das ações, cada uma com seu respectivo diagrama.

Conforme dito anteriormente, um trabalho de tradução da UML.P para PDDL já fora realizado em (VAQUERO, 2007) para aspectos estáticos e dinâmicos atemporais. No presente trabalho será apresentada uma proposta de interpretação do diagrama de tempo apresentado, o que possibilita agregar a modelagem dos aspectos temporais ao trabalho inicial.

4.9.1. Mapeamento para requerimento (:requirements)

Conforme visto no capítulo 3, os requerimentos necessários para que o planejador interprete adequadamente o domínio devem estar explícitos. O mapeamento considera a definição 4.1 do conjunto $D_t = \{O, E, T, A, D, Tc\}$ e a especificação da PDDL da seguinte forma:

- :durative-actions

  Adicionado automaticamente quando no domínio existir pelo menos um diagrama de tempo, ou seja, se o conjunto de elementos do diagrama de tempo não for vazio ($D_t \neq \{\varnothing\}$).

- :duration-inequalities

  Adicionado automaticamente quando no conjunto D existir uma restrição de duração de um intervalo cujo critério de comparação seja <, >, <= ou >=.

- :continuous-effects

  Adicionado automaticamente quando os critérios de definição de efeitos contínuos forem satisfeitos, ou seja, deve existir pelo menos uma instrução de incremento/decremento que esteja condicionada à duração da ação. Um exemplo deste caso é o diagrama apresentado anteriormente na figura 4.6.

4.9.2. Mapeamento para definição de ações com duração (:durative-action)

A definição de ação durativa está condicionada à inclusão desta ação no diagrama de tempo (contexto do domínio ou da ação). Durante o processo de modelagem, os elementos que formam o diagrama de tempo são adicionados ao conjunto, e, com base nos elementos da definição 4.1, se a ação estiver contida no conjunto de ações temporais (A_t) e estiver mapeada a um instante de tempo, então a ação será interpretada como ação durativa:

A partir do conjunto $D_t = \{O, E, T, A, D, Tc\}$ da definição 4.1:

$A = \{A_t, \beta\}$ define um conjunto de ações

- A_t é um conjunto de ações temporais
- β: $(a_i \mid a_i \in A_t) \rightarrow (t_i \mid t_i \in T)$ onde β é uma relação que mapeia as ações aos pontos no tempo.

$(\exists(a_i) \mid A(a_i) \wedge a_i \in A_t \wedge a_i \subset \beta) \rightarrow \text{durativa}(a_i)$

Se existe um elemento a_i tal que a_i seja uma ação e pertença ao conjunto de ações temporais e esteja contido na relação que mapeia uma ação a um ponto no tempo, então este elemento será uma ação durativa.

4.9.3. Mapeamento para definição da duração da ação (:duration)

Na UML.P uma restrição de duração é indexada a uma ação e a um par de instantes que correspondem ao início e fim desta ação. As expressões que determinam estas restrições são construídas em OCL (*Object Constraint Language*), que possui uma notação infixa. Estas expressões são traduzidas para a PDDL, que possui uma especificação em notação prefixada.

Uma expressão em OCL que determine uma duração envolvendo um objeto x e uma função de tempo pode ser descrita da seguinte forma: x.atributo * x.funcãoTempo. Esta mesma expressão em PDDL pode ser escrita como: (= ?duration (* (atributo ?x) (funcãoTempo ?x))). Desta forma, a tradução da OCL para PDDL é obtida pela implementação de um algoritmo que transforme uma notação infixa em prefixa e altere a notação dos objetos e atributos para o padrão LISP utilizado pela PDDL.

A semântica de cada linguagem restringe o escopo de interpretação dos elementos. Isto pode ser verificado pela especificação do instante em que a ação deve ser executada. Na UML.P são definidos intervalos através dos instantes de início e fim, contudo a duração destes intervalos é definida pela duração da ação. Na PDDL não existe uma especificação direta do instante em que uma ação deve ser executada, então, a tradução deve considerar apenas a duração das ações.

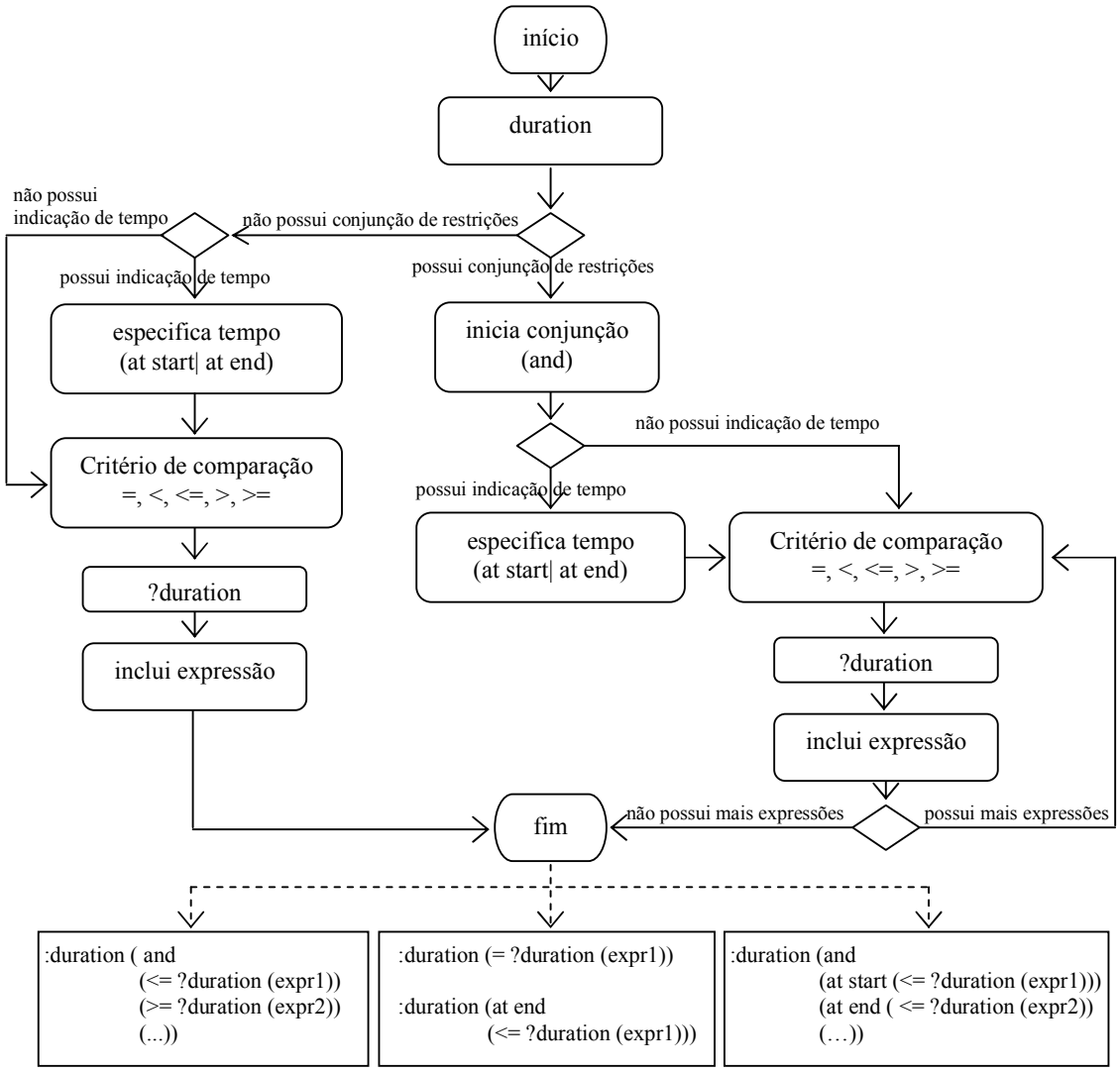


Figura 4.18 Possibilidades para determinação da restrição de duração em PDDL.
Fonte: criado pelo autor a partir da sintaxe da PDDL em (FOX; LONG, 2003)

As ações identificadas no conjunto D_t possuem uma duração determinada ou condicionada aos elementos que participam de seu ciclo de vida. Para visualizar como estas e outras características podem ser interpretadas, a figura 4.18 ilustra as possibilidades para determinar a duração de uma ação temporal em PDDL: a conjunção de expressões possibilita estabelecer limites inferiores e superiores para a duração. Também é possível determinar uma restrição no início ou no final da ação (at start | at end) como forma de garantir que a ação seja válida em um ambiente concorrente de acordo com uma expressão ou uma conjunção de expressões (FOX; LONG, 2003).

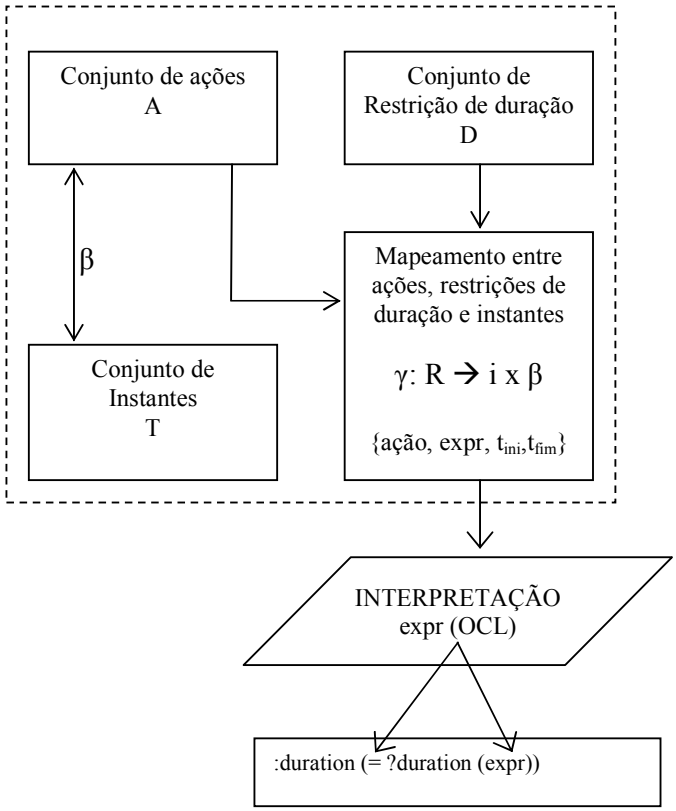


Figura 4.19 Ilustração do mecanismo de interpretação para definição da duração de uma ação
Fonte: autor

A figura 4.19 ilustra a interpretação da definição da duração da ação dada pelo conjunto da definição 4.1 (D_t = {O, E, T, A, D, Tc}) onde a partir da expressão OCL de restrição mapeada pela relação γ: R → i x β do conjunto D, são obtidos os critério de comparação e a expressão que define a duração da ação conforme definição 4.1:

- D = {R,γ} define um conjunto de restrições de duração
  - R é um conjunto de restrições

γ: R → i x β, tal que i = {(t_i, t_j) | t_i, t_j ∈ T ∧ t_i < t_j } onde γ é uma relação que mapeia as restrições de duração a um par de pontos e esta relação é mapeada às ações pela relação β, que mapeia as ações aos pontos no tempo.

4.9.4. Mapeamento para definição das condições (:condition)

O diagrama de tempo das ações fornece um panorama geral da duração da ação, permitindo observar o efeito do tempo sobre os atributos que participam da ação. A definição do tempo na condição de uma ação possibilita visualizar os momentos em que as variáveis

terão certos valores, por exemplo: para que uma ação seja disparada, é necessário que uma determinada variável tenha um valor no início da ação e mantenha este valor durante toda duração. No domínio de exemplo de tratamento térmico de metais, para que uma peça seja temperada, é necessário que o forno esteja na temperatura necessária e permaneça nesta temperatura durante todo o período em que a peça estiver sendo aquecida. No final, é necessário que o recipiente esteja disponível (somente no final, podendo ficar indisponível durante a ação.). Sem estas condições, a ação não é iniciada.

De fato, estas características temporais não são definidas no diagrama de máquina de estados, porém, este diagrama auxilia na definição das condições através dos critérios estabelecidos na precondição da ação. A indexação destes critérios ao tempo é definida a partir do diagrama de tempo para ações temporais com a definição 4.2 (Elementos do Diagrama de Tempo para ações). Desta forma, torna-se necessário definir a interpretação dos pontos inicias de uma ação em PDDL:

Pode-se afirmar que, se t_0 é o instante definido no diagrama de tempo como primeiro ponto em que os atributos são definidos e a execução da ação inicia-se neste ponto, então, pela definição de (FOX; LONG, 2003) a condição *at start* refere-se ao instante em que a ação é iniciada. Por esta definição, se i é o instante em que a ação é iniciada, então $t_0 = i$ e *at start* = i, logo t_0 = *at start*. Como a PDDL não especifica nenhum instante durante a execução da ação,o intervalo de uma ação em PDDL é equivalente à seqüência ordenada de pontos t_i, t_{i+1}, … t_{n-1} definida na duração da ação em UML.P.

Na PDDL, o intervalo de uma ação é aberto nas extremidades, ou seja, uma variável que permanece com um valor durante a ação e é especificada com *over all* não terá este valor controlado no instante inicial e final da ação, somente durante a execução (FOX; LONG, 2003). Desta forma, considerando um conjunto de instantes de uma ação sendo t_0 o instante inicial da ação e t_n o instante final, com $t_0 < t_n$, a equivalência para a UML.P será dada pelos intervalos entre t_1 e t_{n-1}. O fato da PDDL não utilizar instantes específicos na definição da ação resulta na ausência de uma interpretação precisa quando uma variável no diagrama de tempo sofre oscilação em seu valor durante a execução da ação.

Assim como o instante inicial, o instante final de um a ação é especificado em PDDL, o que torna a tradução entre a UML.P e a PDDL mais direta. O ponto final da ação em PDDL *at end* é equivalente ao instante t_n do diagrama de tempo da UML.P.

Na UML.P, t_n é o instante em que a ação é finalizada. Em PDDL, pela definição de (FOX; LONG, 2003) a condição *at end* refere-se ao instante em que a ação é finalizada, então, t_n = *at end*. Com esta definição, a definição das condições fica concluída. No próximo tópico será abordado mapeamento para os efeitos da ação.

4.9.5. Mapeamento para definição dos efeitos (:effects)

Nas ações temporais, os efeitos são referenciados a um determinado momento no ciclo de vida da ação. Uma única ação pode ter diversos efeitos, conforme sua lista de adição e remoção, e estes podem ocorrer de forma dispersa no tempo, alguns imediatamente após o início da ação, outros somente no final e outros podem variar constantemente ao longo da ação. Para cada uma destas situações, existe uma forma de representação em UML.P e em PDDL. Conforme visto neste capítulo, na UML.P a representação temporal faz parte de um processo composto por diagramas interdependentes, onde cada um trata uma abordagem específica e, conseqüentemente, a interpretação de um cenário em PDDL depende de todos estes diagramas.

Em um contexto muito simples, o diagrama de tempo apenas indexa tempo às condições e efeitos determinados no diagrama de máquina de estados, porém, em cenários com variação contínua, por exemplo, o diagrama de tempo tem um papel muito mais abrangente, pois permite uma visão detalhada da evolução dos valores das variáveis ao longo da ação, além da definição de restrições. A maior diferença entre o diagrama de tempo da UML.P e a PDDL 2.1 é a precisão na determinação de pontos no tempo: na UML.P é possível determinar quantos pontos forem necessários para detalhar o ciclo de vida da ação e na PDDL 2.1, conforme visto anteriormente, existem as definições de início, durante e final.

Tabela 4.3 interpretação de aspectos temporais da PDDL e UML.P para efeitos de uma ação (continua)

PDDL	Interpretação em UML.P
Especificador de tempo: at start over all at end	A interpretação destes especificadores é idêntica à das condições. Desta forma, os mesmos teoremas do tópico anterior podem ser utilizados para determinar
increase decrease	Quando uma expressão numérica na forma p.atributo = p.atributo + num Onde: p é um objeto com atributo numérico do tipo inteiro ou ponto flutuante.

Tabela 4.3 interpretação de aspectos temporais da PDDL e UML.P para efeitos de uma ação (última)

PDDL	Interpretação em UML.P
(* #t <f-exp>)	A partir da especificação do conjunto D_{ta} = {N, Dur, Tr_c, O, Exp} as expressões que definem a variação contínua pertencem ao subconjunto Ai do conjunto O. Um atributo cujo valor varia continuamente ao longo da duração da ação terá sua expressão OCL da seguinte forma: o.atributo = o.atributo <-+> (tn * taxa) Considerando que "o" é um objeto, "tn" refere-se a qualquer instante $t_i \in Tr_c$, "taxa" refere-se ao fator de incremento/decremento. Exemplo: p.nível = p.nível – (tn * r.taxa) Será interpretado como: (decrease (nivel ?p) (* #t (taxa ?r)))

Fonte: autor

A tabela 4.3 aborda uma interpretação do diagrama de tempo considerando os efeitos da ação para PDDL. Esta interpretação corresponde ao campo *effects* da ação.

Com esta interpretação, o escopo da tradução é atingido. A partir das definições apresentadas, é possível desenvolver um algoritmo de tradução que possa, a partir dos elementos definidos para um diagrama de tempo, gerar um domínio equivalente em PDDL 2.1.

4.10. Resumo e conclusão do capítulo

Neste capítulo foi abordada uma extensão da UML.P com base no diagrama de tempo da UML 2.0. Com este diagrama, é possível associar tempo aos aspectos dinâmicos de um domínio de planejamento em um contexto geral com a participação de todos os elementos relevantes e também em um contexto local, restrito às ações temporais. As relações temporais de Allen foram utilizadas para garantir a aplicabilidade do diagrama e adicionar um formalismo em sua definição.

A interpretação do diagrama proposto em PDDL é importante para garantir que o processo de engenharia do conhecimento fique completo. Também é importante salientar que não há uma interpretação para todos os aspectos modelados no diagrama, por exemplo, não há

como especificar em PDDL um momento exato em que os efeitos de uma ação durativa acontecem. Esta limitação já havia sido apontada em (SMITH, 2003), que além deste aspecto, cita também o fato dos efeitos da ação acontecerem no início, durante todo intervalo e no final, de forma arbitrária, limitando detalhes importantes nos domínios reais. Desta forma, existe a possibilidade de uma interpretação em UML.P para PDDL não ser precisa.

Apesar dos comentários e restrições impostas às ações durativas na PDDL 2.1 (SMITH, 2003) e (McCLUSKEY, 2003) sobre como a PDDL trata os aspectos temporais, a evolução da linguagem para tratar aspectos temporais certamente trouxe avanços para a área de planejamento. Para obter uma visão prática do diagrama proposto, no próximo capítulo o domínio de exemplo *tratamento térmico de metais* será modelado parcialmente com a participação de todos os diagramas da UML.P.

CAPÍTULO V

5. Estudo de Caso

5.1. Exemplo: Tratamento térmico de metais

Com o objetivo de analisar a proposta de modelagem de domínios temporais em UML.P, o domínio de exemplo tratamento térmico de metais será modelado com um pouco mais de profundidade neste capítulo. A partir da especificação do problema no capítulo 2, será realizada toda a modelagem em UML.P até a interpretação em PDDL. A ferramenta utilizada na modelagem do domínio será o itSIMPLE.

Conforme descrito no capítulo 4, a primeira fase do processo é o entendimento do problema pela definição dos casos de uso. A partir de uma descrição do domínio, é possível entender os aspectos dinâmicos e obter uma visão geral para entendimento do comportamento do domínio com o auxílio do diagrama de casos de uso. A figura 5.1 ilustra o diagrama de casos de uso identificado. O ator do sistema é um operador de forno que realiza o tratamento térmico. Os casos de uso identificados são *Aquecer forno para têmpera*, *Preparar para têmpera*, *Temperar peça*, *Resfriar forno para revenimento*, *Preparar para revenimento*, *Revenir peça* e *Entregar peça*. A seta tracejada indica que existe uma dependência entre os casos de uso. Neste exemplo, o caso de uso *Preparar para têmpera* é dependente do caso de uso *Aquecer forno para têmpera*.

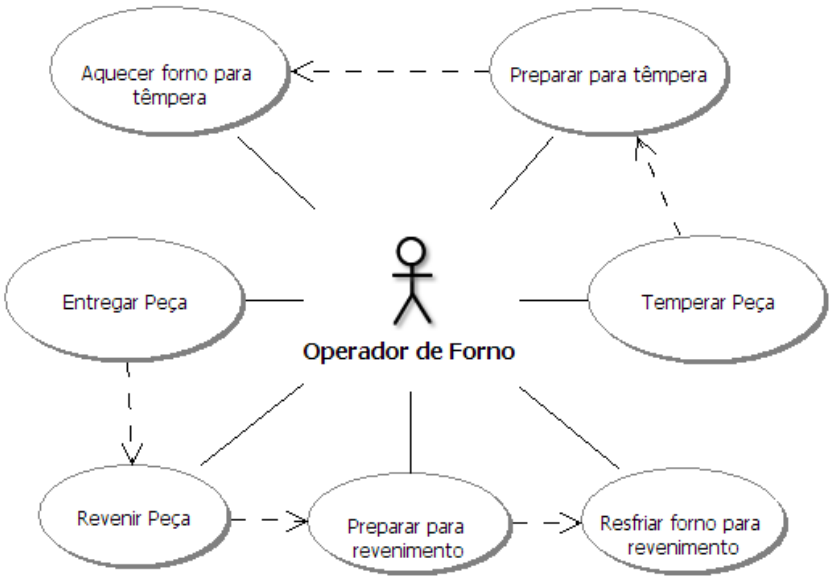


Figura 5.1 Diagrama de caso de uso para domínio de tratamento térmico
Fonte: modelado pelo autor na ferramenta itSIMPLE (VAQUERO, 2007).

Com a especificação de cada caso de uso, é possível identificar os elementos que farão parte do domínio do sistema, qual será sua dinâmica e quais serão as restrições. A tabela 5.1 a seguir, apresenta os detalhes de cada caso de uso identificado no diagrama da figura 5.1. Para não tornar o domínio muito extenso, algumas restrições não foram consideradas, como por exemplo, se o tamanho do forno comporta o tamanho da peça.

Tabela 5.1 Tabela de casos de uso do domínio de tratamento térmico

Caso de Uso: *Aquecer forno para Têmpera*
Ator: Operador de Forno **Descrição:** O agente Operador de Forno aquece o forno à temperatura necessária para temperar a peça. ***Restrições:*** **Pré-condições** – O forno deve estar vazio; – O forno deve estar ligado; – A temperatura do forno deve ser menor ou igual à temperatura para temperar a peça; – A peça deve estar sem tratamento. **Pós-condições** – Forno na temperatura necessária para têmpera; **Restrições temporais** – O forno permanece aquecendo conforme a equação: $$duração = \frac{\left(\frac{temperaturaMáxima + temperaturaMínima}{2}\right) - temperaturaDoForno}{taxaDeAquecimento}$$ Onde: temperaturaMáxima e temperaturaMínima referem-se ao intervalo de temperatura necessário para temperara a peça;
Caso de Uso: *Preparar para temperar*
Ator: Operador de Forno **Descrição:** O agente Operador de Forno prepara a peça para ser temperada. ***Restrições:*** **Pré-condições**

– O forno deve estar vazio;
– O forno deve estar ligado e na temperatura média para temperar a peça em questão;
– A peça deve estar sem tratamento;
– A peça deve estar sobre a bancada de entrada;

Pós-condições:

– A peça não está mais sobre a bancada;
– A peça deve estar dentro do forno;
– A peça fica pronta para ser temperada, com temperatura média para têmpera;
– O forno não deve estar vazio.

Restrições temporais:

– A peça deve permanecer no forno até atingir a temperatura e por mais um período de estabilização, conforme a expressão:

$$duração = \left(\frac{\frac{(mp * cep * tp) + (mf * cef * tf)}{(mp * cep) + (mf * cef)}}{taf} \right) * iep$$

Onde:

mp = massa da peça;
cep = calor específico da peça;
tp = temperatura da peça;
mf = massa do forno;
cef = calor específico do forno;
tf = temperatura do forno;
taf = taxa de aquecimento do forno;
iep = índice de estabilização da peça.

Caso de Uso: *Temperar Peça*

Ator: Operador de Forno

Descrição: O agente Operador de Forno retira a peça do forno quando esta estiver pronta para têmpera e a resfria bruscamente no reservatório.

Restrições:

Pré-condições

– A peça deve estar dentro do forno;
– A peça deve estar na temperatura de têmpera e estabilizada;
– A peça deve estar pronta para têmpera;
– O recipiente deve estar livre;

Pós-condições:

– O forno deve estar vazio;

– A peça deve estar imersa no recipiente de resfriamento;
– O recipiente fica ocupado;
– A peça fica temperada e não está mais pronta para têmpera;

Restrições temporais:

– O recipiente para resfriamento deve estar livre no momento em que a peça tiver atingido o período de estabilização;
– Depois de submersa no recipiente de resfriamento, a peça deve permanecer até atingir a temperatura ambiente. Este tempo varia conforme a capacidade de resfriamento do recipiente, a massa do líquido e a massa da peça;

$$duração = \left(\frac{(mp * cep * tp) + (ml * cel * tl)}{(mp * cep) + (ml * cel)} \over trl \right)$$

Onde:

mp = massa da peça;
cep = calor específico da peça;
tp = temperatura da peça;
ml = massa do líquido;
cel = calor específico do líquido;
tl = temperatura do líquido;
trl = taxa de resfriamento do líquido.

– Depois de resfriada, a peça fica pronta para revenimento.

Caso de Uso: *Resfriar Forno para Revenimento*

Ator: Operador de Forno

Descrição: O agente Operador de Forno resfria o forno para poder revenir a peça.

Restrições:

Pré-condições

– O forno deve com a temperatura maior que a temperatura para revenimento;
– A peça deve estar temperada;
– A peça não deve estar revenida;
– O forno deve estar ligado;

Pós-condições:

– O forno deve estar na temperatura de revenimento.

Restrições temporais:

– O tempo em que o forno permanece resfriando segue a expressão:

$$duração = \left(\frac{(tmf) - \left(\frac{tmar + tmir}{2} \right)}{txr} \right)$$

Onde:

tmf = temperatura do forno;
tmar = temperatura máxima para revenimento da peça;
tmir = temperatura mínima para revenimento da peça;
txr = taxa de resfriamento do forno.

Caso de Uso: *Preparar para Revenimento*

Ator: Operador de Forno

Descrição: O agente Operador de Forno prepara a peça para ser revenida.

Restrições:

Pré-condições

– O forno deve estar vazio e ligado e na temperatura para revenimento;
– A peça deve estar temperada, imersa no recipiente e em temperatura ambiente;
– A peça não deve estar revenida;
– A peça deve esta pronta para revenimento.

Pós-condições:

– A peça não está mais imersa no recipiente;
– A peça deve estar dentro do forno;
– A peça deve estar pronta para ser revenida, com temperatura média para revenimento;
– O forno não deve estar vazio.

Restrições temporais:

– A peça deve permanecer no forno até atingir a temperatura de revenimento, conforme a expressão:

$$duração = \left(\frac{(mp * cep * tp) + (mf * cef * tf)}{(mp * cep) + (mf * cef)} \right)$$

Onde:

mp = massa da peça;
cep = calor específico da peça;
tp = temperatura da peça;
mf = massa do forno;
cef = calor específico do forno;
tf = temperatura do forno;
taf = taxa de aquecimento do forno.

Caso de Uso: *Revenir Peça*

Ator: Operador de Forno

Descrição: O agente Operador de Forno retira a peça pronta para revenimento do forno e a deposita no recipiente.

Restrições:

Pré-condições

– A peça deve estar dentro do forno;
– O forno deve estar na temperatura de revenimento;
– A peça deve estar pronta para revenimento;
– A peça não deve estar revenida;

Pós-condições:

– A peça não deve estar dentro do forno;
– A peça deve estar imersa no recipiente de resfriamento;
– O recipiente fica ocupado;
– A peça fica revenida;
– O forno fica vazio.

Restrições temporais:

– O recipiente para resfriamento deve estar livre no momento em que a peça tiver atingido a temperatura para revenimento;
– Depois de submersa no recipiente de resfriamento, a peça deve permanecer até atingir a temperatura ambiente. Este tempo varia conforme a capacidade de resfriamento do recipiente, a massa do líquido e a massa da peça, de acordo com a expressão:

$$duração = \left(\frac{(mp * cep * tp) + (ml * cel * tl)}{(mp * cep) + (ml * cel)} \over trl \right)$$

Onde:

mp = massa da peça;
cep = calor específico da peça;
tp = temperatura da peça;
ml = massa do líquido;
cel = calor específico do líquido;
tl = temperatura do líquido;
trl = taxa de resfriamento do líquido.

– Depois de resfriada, a peça fica pronta para ser entregue.

Caso de Uso: *Entregar Peça*

Ator: Operador de Forno

Descrição: O agente Operador de Forno retira a peça revenida do recipiente e a disponibiliza na bancada de saída.

Restrições:

Pré-condições

– A peça deve estar no recipiente e à temperatura ambiente;
– A bancada de saída deve comportar a peça;
– A peça deve estar temperada e revenida.

Pós-condições:

– A peça deve estar sobre a bancada de saída;
– A peça não deve estar imersa no recipiente de resfriamento;
– O recipiente deve estar livre.

Restrições temporais:

– O tempo de entrega da peça deve ser determinado conforme o tipo de problema a ser solucionado;

Fonte: autor

A partir do detalhamento dos casos uso é realizada identificação das entidades do domínio e a representação no diagrama de classes. As entidades suficientes para modelar este domínio são: Bancada, Peça, Recipiente, Forno e Operador. Pela descrição dos casos de uso, uma ou mais peças poderão permanecer sobre uma bancada. Uma peça ou nenhuma pode estar submersa em um recipiente. Uma peça ou nenhuma pode estar dentro de um forno. Não há um relacionamento explícito entre o operador e os demais elementos.

Os relacionamentos *sobre*, *dentro* e *imersa* definem atributos da classe Peça que posteriormente serão interpretados como predicados na PDDL. Os atributos definidos para a classe Peça são necessários para definir as condições em que o tratamento térmico é realizado. Uma peça pode ser fabricada com diferentes tipos de metais, cada um com sua própria temperatura para têmpera e revenimento e calor específico. Não é relevante ao domínio especificar o tipo de material da peça. Os atributos das classes Forno e Recipiente além de auxiliar na designação dos estados das classes, auxiliam no dinamismo do domínio.

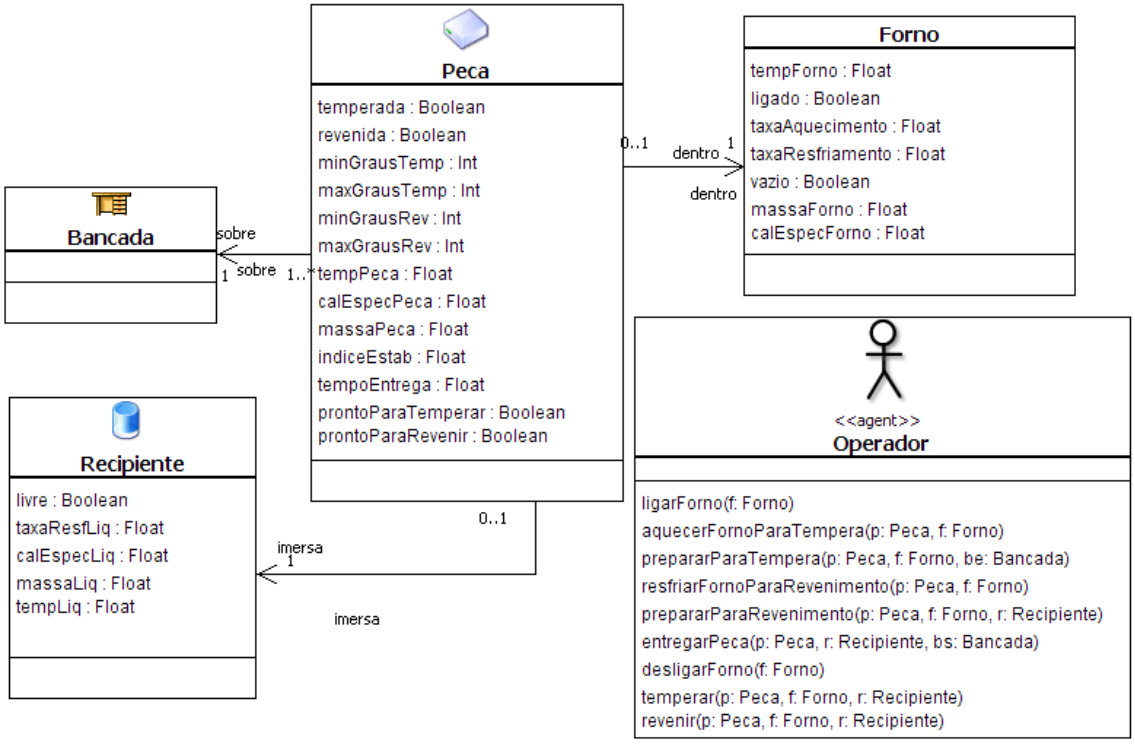


Figura 5.2 Diagrama de classes do domínio de tratamento térmico de metais
Fonte: modelado pelo autor na ferramenta itSIMPLE

O diagrama de classes, ilustrado na figura 5.2, exibe o relacionamento, os atributos e operações das classes identificadas a partir dos casos de uso. A tabela 5.2 abaixo detalha cada atributo das classes e fornece uma descrição dos métodos da classe Operador.

Tabela 5.2 Descrição dos atributos e métodos das classes modeladas no diagrama de classes

Classe:	Peça	
Atributo	**Tipo**	**Descrição**
temperada	Booleano	Indica se a peça está temperada (verdadeiro) ou não (falso).
revenida	Booleano	Indica se a peça está temperada (verdadeiro) ou não (falso).
minGrausTemp	Inteiro	Define a temperatura mínima para têmpera em graus Celsius.
maxGrausTemp	Inteiro	Define a temperatura máxima para têmpera em graus Celsius.
minGrausRev	Inteiro	Define a temperatura mínima para revenimento em graus Celsius.
maxGrausRev	Inteiro	Define a temperatura máxima para revenimento em graus Celsius.
tempPeca	Ponto Flutuante	Indica a temperatura da peça em graus Celsius no instante em que está sendo verificada.

calEspecPeca	Ponto Flutuante	Calor específico da peca. Utilizado para o cálculo de estabilização da temperatura. Uma peça de aço a 20graus Celsius tem calor específico de 448.
massaPeca	Ponto Flutuante	Determina a massa da peça em quilogramas.
indiceEstab	Ponto Flutuante	Define um índice necessário para determinar o tempo de estabilização da temperatura na têmpera.
tempoEntrega	Ponto Flutuante	Define um valor constante para o tempo de entrega da peça.
prontoParaTemperar	Booleano	Indica se a peça está pronta para temperar (verdadeiro) ou não (falso).
prontoParaRevenir	Booleano	Indica se a peça está pronta para revenir (verdadeiro) ou não (falso).
Classe	Forno	
Atributo	**Tipo**	**Descrição**
tempForno	Ponto Flutuante	Indica a temperatura do forno no momento.
ligado	Booleano	Indica se o forno está ligado (verdadeiro) ou desligado (falso).
taxaAquecimento	Ponto Flutuante	Define uma taxa de aquecimento específica para o forno. Será utilizado na definição do tempo em que o forno atinge a temperatura desejada.
taxaResfriamento	Ponto Flutuante	Define uma taxa de resfriamento específica para o forno. Será utilizada na definição do tempo em que o forno permanece perdendo calor até atingir a temperatura desejada.
vazio	Booleano	Indica se o forno está vazio (verdadeiro) ou não (falso).
massaForno	Ponto Flutuante	Define a massa do forno. Será necessária para calcular o tempo de aquecimento.
calEspecForno	Ponto Flutuante	Define o calor específico do forno. Será utilizado no cálculo de equilíbrio de temperatura.
Classe	Recipiente	
Atributo	**Tipo**	**Descrição**
livre	Booleano	Indica se o recipiente está livre (verdadeiro) ou se está ocupado (falso)
taxaResfLiq	Ponto Flutuante	Define uma taxa de resfriamento para o líquido contido no recipiente.
calEspecLiq	Ponto Flutuante	Define o calor específico do líquido de resfriamento. Será utilizado no cálculo de equilíbrio de temperatura quando a peça for temperada ou revenida.
massaLiq	Ponto Flutuante	Massa do líquido (peso) em quilogramas. Será utilizado no cálculo de estabilização da temperatura.
tempLiq	Ponto Flutuante	Indica a temperatura do líquido em graus Celsius.
Classe	Operador	

Método	Descrição
ligarForno	Realiza a ligação do forno.
aquecerFornoParaTempera	Define uma temperatura para o forno e aguarda.
prepararParaTempera	Coloca a peça no forno para que seja temperada.
resfriarFornoParaRevenimento	Depois de temperar a peça, reduz a temperatura do forno para o revenimento.
prepararParaRevenimento	Coloca a peça no forno para que esta seja revenida.
entregarPeca	Entrega a peça, retirando do recipiente de resfriamento e colocando na bancada de entrega.
desligarForno	Desliga o forno depois de todas as operações.
temperar	Retira a peça em temperatura de têmpera já estabilizada do forno e resfria bruscamente no recipiente.
revenir	Retira a peça em temperatura de revenimento e resfria bruscamente no recipiente.

Depois de definida a estrutura do domínio através do diagrama de classes, é necessário realizar a modelagem dos aspectos dinâmicos. Conforme abordado no capítulo 4, cada classe que participa ativamente de um contexto dinâmico do domínio possui pelo menos um diagrama de máquina de estados. Neste domínio, as classes peça e recipiente foram modeladas com um diagrama de máquina de estados e a classe forno com dois diagramas.

Para realizar o tratamento térmico é necessário que a peça passe pelo estado inicial ‘Sem tratamento’ até chegar ao estado ‘Revenida’ e ser entregue. Cada mudança de estado é causada por um método (ação) que pode participar de mais de um diagrama de máquina de estados. Abaixo a representação da ação prepararParaTempera em OCL. É observável que as pré-condições e pós-condições são oriundas dos estados dos objetos recebidos como parâmetros.

```
context Operador::prepararParaTempera(p: Peca, f: Forno, be: Bancada)
pre:
  -- conditions
     p.temperada = false and p.revenida = false and p.prontoParaTemperar = false
     and p.prontoParaRevenir = false and p.sobre->includes(be)
  -- conditions
      f.tempForno >= p.minGrausTemp and f.tempForno <= p.maxGrausTemp
      and f.ligado = true and    f.vazio = true

post:
  -- conditions
     p.temperada = false and p.revenida = false and p.prontoParaTemperar = true
     and p.prontoParaRevenir = false and p.sobre->excludes(be)
     and p.tempPeca = f.tempForno and p.dentro->includes(f)
  -- conditions
     f.ligado = true and f.vazio = false
```

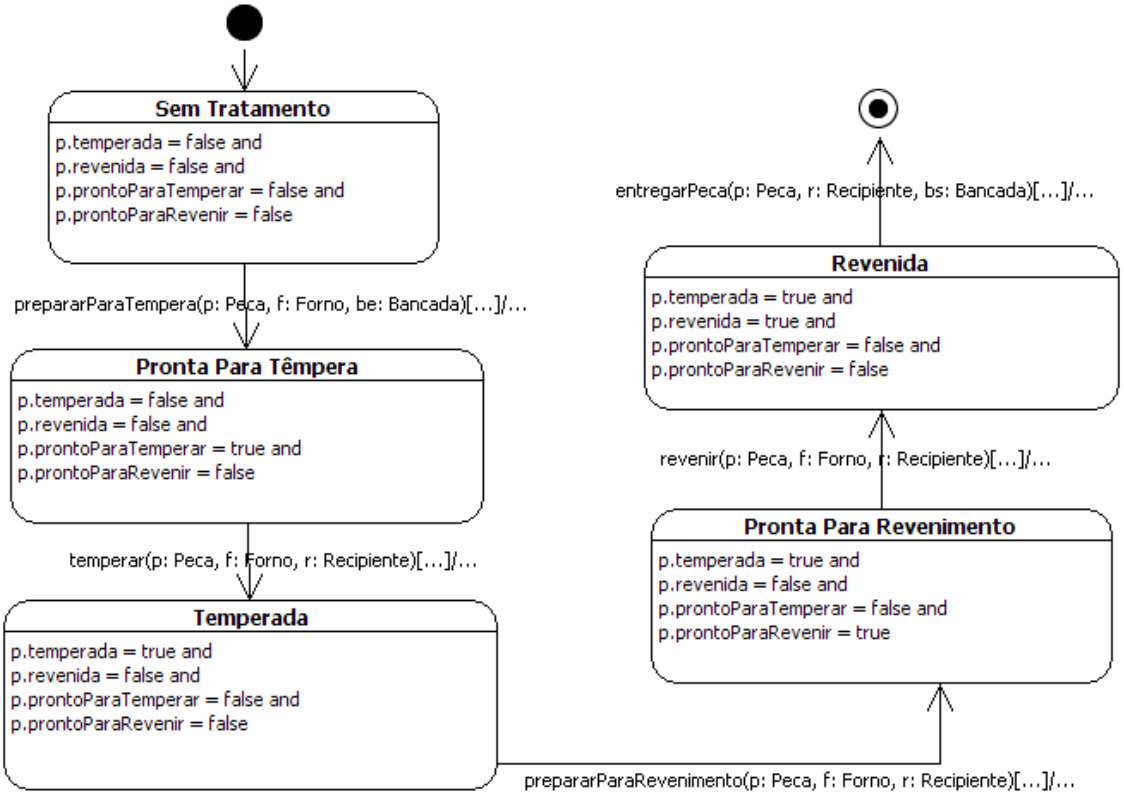


Figura 5.3 Diagrama de máquina de estados do objeto peça
Fonte: modelado pelo autor na ferramenta itSIMPLE

A figura 5.3 ilustra o diagrama de máquina de estado para um objeto p da classe Peça. Cada estado é definido por um conjunto de atributos com valores da classe e todos os estados farão parte de um contexto temporal no diagrama de tempo. Os tempos que serão analisados futuramente serão referentes ao tempo em que o objeto permanece no estado e o tempo necessário para troca de estado. Neste diagrama, o objeto 'peça' passa do estado inicial 'Sem Tratamento' para um novo estado 'Pronta Para Têmpera' através da ação 'prepararParaTempera'. Os demais estados têm uma evolução semelhante. A definição da temperatura da peça em cada estado está oculta.

Cada diagrama de máquina de estados possibilita a visualização dos estados de um único objeto por vez. Cada ação pode receber mais de um objeto, mas a UML direciona para a modelagem individual dos objetos. No diagrama de maquina de estados é possível visualiza o que muda e, no diagrama de tempo, quando estas mudanças ocorrem. Depois que todas as possíveis ações do objeto forem devidamente modelados, será possível iniciar o diagrama de tempo.

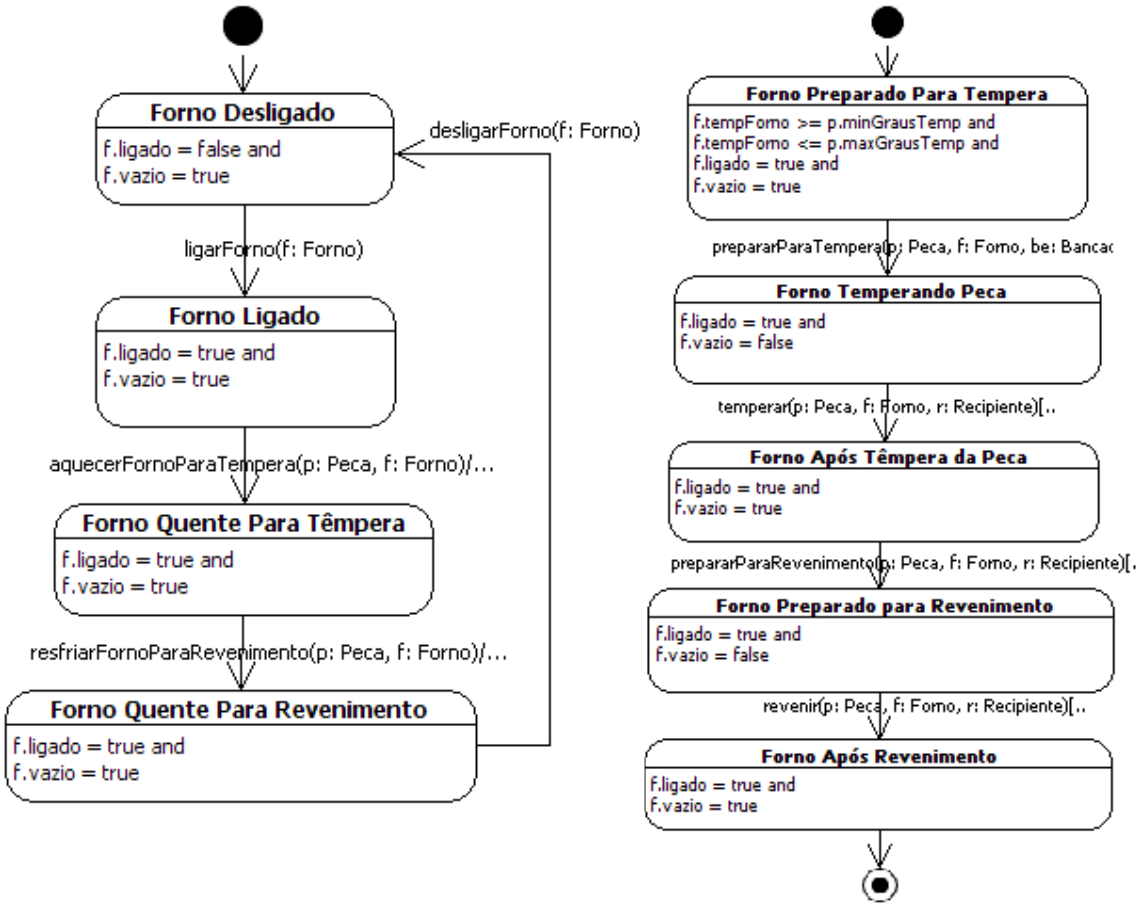


Figura 5.4 Diagramas de máquinas de estados para o objeto forno
Fonte: modelado pelo autor na ferramenta itSIMPLE

A figura 5.4 ilustra todos os estados possíveis para o objeto forno. Para este objeto, existem dois contextos específicos: o primeiro, representado pelo diagrama à esquerda onde seu estado inicial é 'Desligado', passando pelos estados necessários para realizar o tratamento térmico até retornar ao estado inicial; o segundo refere-se ao uso do forno para prepará-lo para realizar a têmpera e revenimento da peça. Mesmo participando de diagramas diferentes, o objeto peça pode participar do mesmo diagrama de tempo com estados dos dois diagramas.

Abaixo a ação ligarForno definida em OCL:

```
context Operador::ligarForno(f: Forno)
pre:
  -- conditions
    f.ligado = false and f.vazio = true

post:
  -- conditions
    f.ligado = true and f.vazio = true
```

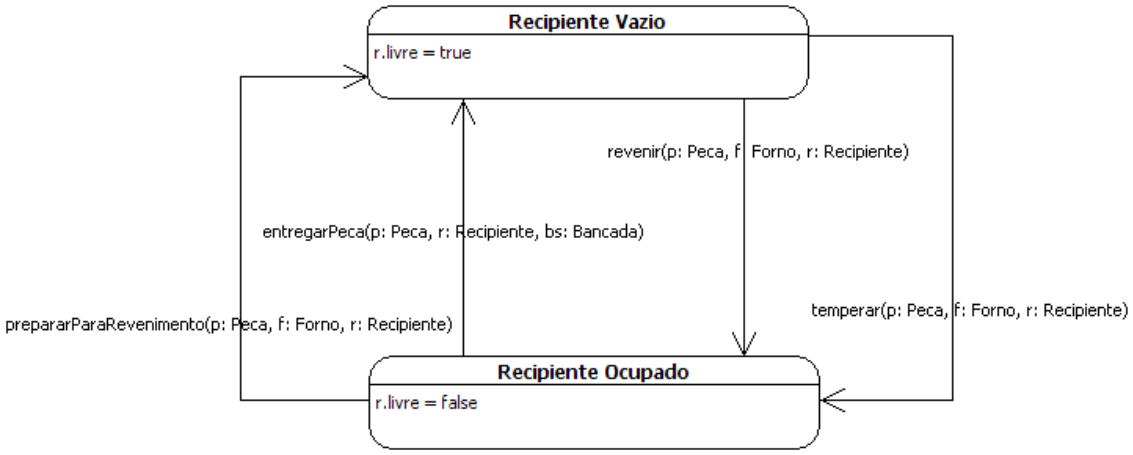


Figura 5.5 Diagrama de máquina de estados do objeto recipiente
Fonte: modelado pelo autor na ferramenta itSIMPLE

A figura 5.5 ilustra o diagrama de máquina de estados do objeto recipiente. O estado 'Recipiente Vazio' pode ser alcançado pelas ações 'prepararParaRevenimento' e 'entregarPeca' e o estado 'Recipiente Ocupado' é alcançado pelas ações 'temperar' e 'revenir'. Este diagrama contém aspectos de tempo ramificado, conforme observado no capítulo 4.

Até este ponto da modelagem foram tratados os aspectos atemporais com o diagrama de classes e o diagrama de máquina de estados. A partir deste ponto, os aspectos temporais serão abordados em dois contextos: o primeiro é geral, com o objetivo de visualizar a dinâmica do domínio, onde todos os objetos que participaram dos diagramas de máquina de estados participam também do diagrama de tempo; o segundo contexto é específico de cada ação, onde o foco é visualizar as mudanças dos valores dos atributos dos objetos que participam da ação ao longo do ciclo de vida da ação.

Para não estender demasiadamente este estudo de caso, serão consideradas apenas as ações 'prepararParaTempera', 'temperar', 'prepararParaRevenimento' e 'revenir' na modelagem. As demais ações terão um modo semelhante e as quatro ações propostas são suficientes para ilustrar como o diagrama proposto pode ser aplicado no contexto dos domínios de planejamento.

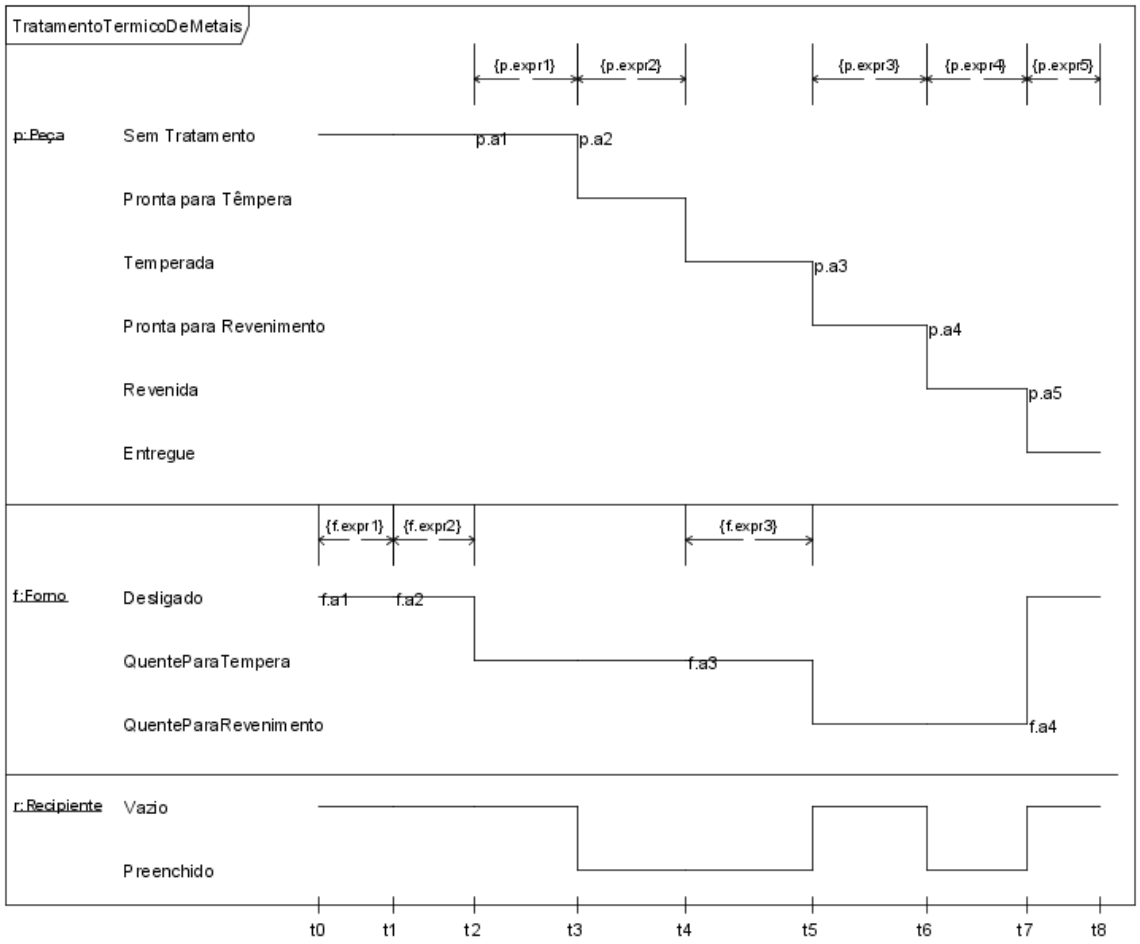


Figura 5.6 Diagrama de tempo para contexto global do domínio
Fonte: autor

A figura 5.6 ilustra o diagrama de tempo no contexto geral do domínio. Este diagrama é criado a partir da junção dos diagramas de máquinas de estados modelados. O diagrama possui o mesmo nome do domínio 'TratamentoTermicoDeMetais' e um ciclo de vida divididos em instantes, iniciado em t_0 até o instante t_8. Cada objeto modelado em um diagrama de máquina de estados participa do ciclo de vida, pois são definidos os períodos em que permanecem em um estado e o período necessário para transição, no caso das ações temporais.

Uma linha define o tempo em que o objeto permanece no estado. Pelo exemplo, o objeto 'p', do tipo 'Peca', permanece no estado 'Sem Tratamento' do instante t_0 até o instante t_3. No instante t_2, a ação 'prepararParaTempera' rotulada como 'p.a1' é iniciada e, conforme definido no caso de uso, possui um tempo de execução, rotulado neste exemplo como {f.expr1}. É possível visualizar que o objeto permanece neste estado pelo tempo definido na

soma dos intervalos $\{(t_0, t_1), (t_1, t_2), (t_2, t_3)\}$, que correspondem aos tempos das expressões {f.expr1} + {f.expr2} + {p.expr1}. Fica clara a dependência do tempo do objeto f, do tipo forno, sair do estado 'Desligado' até o estado 'QuenteParaTempera'. Enquanto isto não ocorrer, o objeto 'p' permanece no estado 'Sem Tratamento'.

Neste diagrama, as ações foram rotuladas para facilitar a visualização, da mesma forma como as expressões, que correspondem ao tempo de duração das ações. Desta forma, têm-se: p.a2 = temperar; p.a3 = prepararParaRevenimento; p.a4 = revenir; p.a5 = entregarPeca; f.a1 = ligarForno; f.a2 = aquecerFornoParaTempera; f.a3 = resfriarFornoParaRevenimento; f.a4 = desligarForno.

O objeto 'r' do tipo recipiente participa do ciclo de vida do domínio com a contribuição no tempo de resfriamento da peça nas ações de têmpera e revenimento. É possível observar no diagrama os momentos em que este permanece vazio ou preenchido.

O segundo contexto da modelagem trata os aspectos temporais sob a ótica das ações temporais. Como apresentado anteriormente, existem diferentes abordagens para as ações temporais e o domínio de exemplo possui tanto ações discretas quanto ações cujas variáveis tem efeito contínuo ao longo do ciclo de vida da ação.

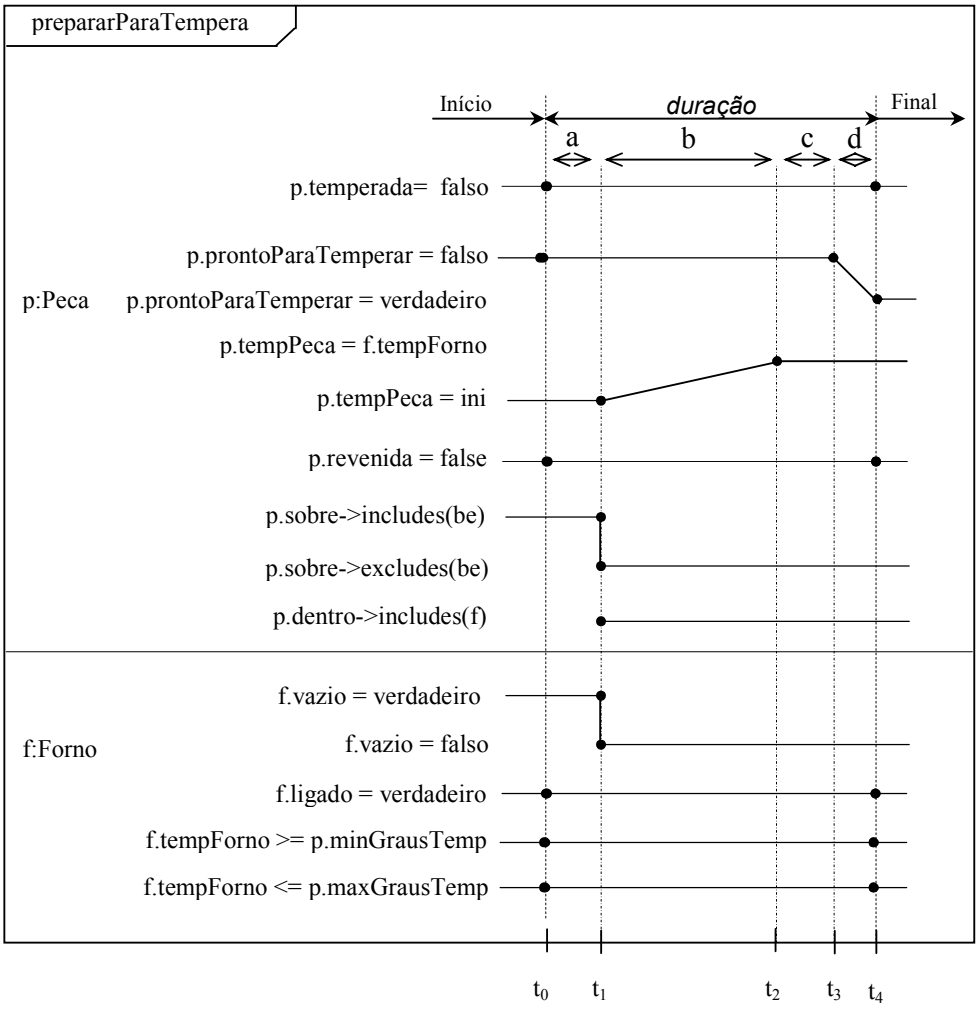


Figura 5.7 Diagrama de tempo para ação temporal prepararParaTempera
Fonte: autor

A figura 5.7 ilustra a ação temporal prepararParaTempera. A ação recebe os parâmetros p:peca, f:forno e be:bancada, somente p:peca e f:forno e possui uma linha cronológica para a mudança das variáveis de cada um destes objetos. O tempo total de execução da ação refere-se ao intervalo definido pelos instantes t_0 a t_4. A duração deste intervalo é definida pela expressão oriunda do caso de uso e os atributos que participam da expressão não desempenham outro papel na ação além de definir variáveis. A duração também pode ser definida pela soma dos intervalos entre t_0 e t_1, na figura, cotado como ‘a’, t_1 e t_2 cotado como ‘b’, t_2 e t_3 cotado como ‘c’ e t_3 e t_4 cotado como d.

$$duração = \left(\frac{\left(mp * cep * tp\right) + \left(mf * cef * tf\right)}{\left(mp * cep\right) + \left(mf * cef\right)} \right) * iep$$

Onde:

mp = p.massaPeca;
cep = p.calEspecPeca;
tp = p.tempPeca;
mf = f.massaForno;
cef = f.calEspecForno;
tf = f.tempForno;
taf = f.taxaAquecimento;
iep = p.indiceEstab.
	Expressão OCL da duração da ação:
	
	(((p.massaPeca * p.calEspecPeca * p.tempPeca)+
	(f.massaForno * f.calEspecForno * f.tempForno))/
	((p.massaPeca * p.calEspecPeca) +
	(f.massaForno * f.calEspecForno)))/
	(f.taxaAquecimento) * p.indiceEstab

Figura 5.8 Duração da ação preparar ParaTempera **definida em expressão e em OCL**
Fonte: autor

Ao iniciar e durante todo intervalo da ação, a peça não fica temperada (p.temperada = falso) e da mesma forma não fica revenida (p.revenida = falso). Do ponto t_0 até o ponto t_3, depois de certo tempo que a peça atingiu a temperatura de têmpera, a peça começa a se tornar pronta para têmpera até atingir t_4, quando está completamente pronta. Normalmente, a peça está na temperatura ambiente (ini) no início da ação e atinge a temperatura do forno em t2, quando inicia o processo de estabilização. Inicialmente a peça está sobre a bancada (p.sobre -> includes(be)) e no momento em que vai para o forno (p.dentro -> includes(f)) livra a bancada (p.sobre -> excludes(be)).

O forno inicia a operação vazio, mas mantém-se não vazio (f.vazio = false) a partir do instante t1 até o final da ação. Durante todo ciclo de vida da ação o forno permanece ligado (f.ligado = verdadeiro) e sua temperatura deve, no máximo, oscilar entre a temperatura mínima e a máxima para têmpera.

Diferentemente do diagrama de máquina de estados, o diagrama de tempo permite a participação de todos os objetos envolvidos no contexto, com a possibilidade de especificar o instante em que cada atributo sofrerá alteração de valor. Com isto, é possível visualizar o tempo em que um objeto permanece em um estado aguardando outro objeto concluir um ciclo.

A figura 5.8 ilustra a fórmula da duração da ação, conforme definido anteriormente no caso de uso. Cada variável desta fórmula é um dos atributos dos objetos recebidos como parâmetro da ação. A duração da ação é obtida a partir da expressão OCL da ação.

```
(:durative-action prepararParaTempera
 :parameters (?p - Peca ?f - Forno ?be - Bancada)
 :duration (= ?duration (* (/ (/ (+ (* (* (massaPeca ?p) (calEspecPeca ?p)) (tempPeca ?p))
                        (* (* (massaForno ?f) (calEspecForno ?f)) (tempForno ?f)))
                        (+ (* (massaPeca ?p) (calEspecPeca ?p)) (* (massaForno ?f)
                        (calEspecForno ?f)))) (taxaAquecimento ?f)) (indiceEstab ?p)))
 :condition
     (and
        (at start (not (temperada ?p)))
        (over all (not (temperada ?p)))
        (at end   (not (temperada ?p)))
        (at start (not (revenida ?p)))
        (over all (not (revenida ?p)))
        (at end   (not (revenida ?p)))
        (at start (not (prontoParaTemperar ?p)))
        (at start (sobre ?p ?be)
        (at start (>= (tempForno ?f) (minGrausTemp ?p)))
        (at start (<= (tempForno ?f) (maxGrausTemp ?p)))
        (over all (>= (tempForno ?f) (minGrausTemp ?p)))
        (over all (<= (tempForno ?f) (maxGrausTemp ?p)))
        (at end   (>= (tempForno ?f) (minGrausTemp ?p)))
        (at end   (<= (tempForno ?f) (maxGrausTemp ?p)))
        (at start (= (tempPeca ?p) (ini ?p)))
        (at start (ligado ?f))
        (over all (ligado ?f))
        (at end   (ligado ?f))
        (at start (vazio ?f)
     )
 :effect
     (and
        (at end   (prontoParaTemperar ?p)
        (at start (not (sobre ?p ?be)))
        (over all (assign (tempPeca ?p) (tempForno ?f)))
        (at start (dentro ?p ?f)))
        (at start (not (vazio ?f)))
     )
)
```

Figura 5.9 Ação prepararParaTempera interpretada em PDDL 2.1
Fonte: autor

A figura 5.9 ilustra a ação ‘prepararParaTempera’ interpretada em PDDL 2.1 a partir do diagrama de tempo. Conforme visto no capítulo 4, na PDDL não há correspondência direta entre os pontos definidos no diagrama de tempo, o que reduz a precisão na especificação do momento exato em que uma mudança de valor deve ocorrer. Entretanto, com a proposta de interpretação, também do capítulo 4, o início, o decorrer e o final da ação podem ser determinados conforme os instantes em que as mudanças ocorrem. ‘At start’ é definido de acordo com os instantes t_0 e t_1. ‘Over all’ é definido entre t_1 e t_{n-1} e ‘At end’ é definido de acordo com t_n.

De fato, há um comprometimento da precisão com esta interpretação, porém, há uma oportunidade de aprimoramento da PDDL de forma que seja possível especificar instantes exatos em que as proposições sofram alteração de valor. Esta proposta seria válida em contextos com necessidade de controles apurados do tempo e compartilhamento de recursos.

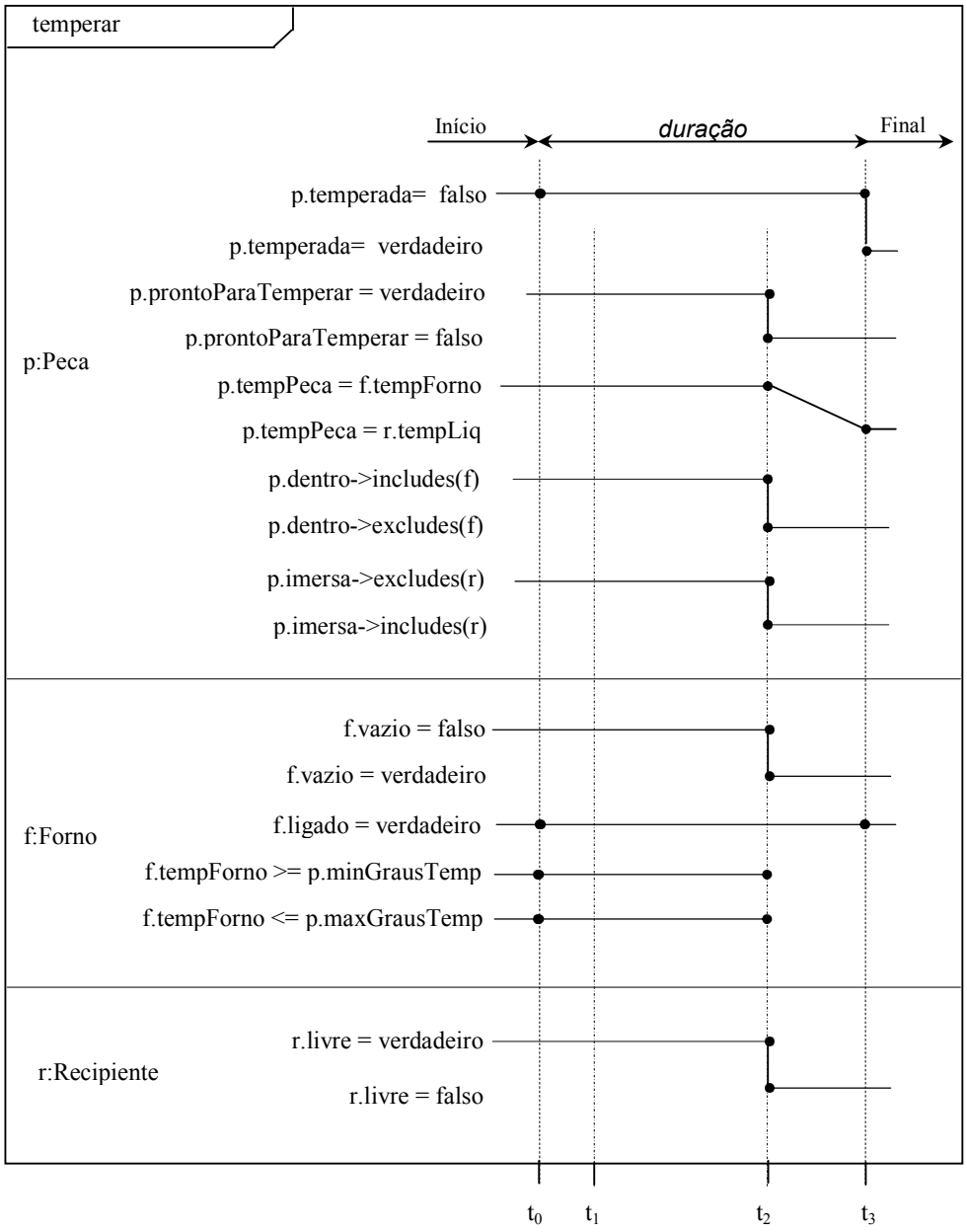


Figura 5.10 Diagrama de tempo para a ação temperar
Fonte: autor

A figura 5.10 ilustra a ação temporal ‘temperar’ modelada em um diagrama de tempo. A ação recebe os parâmetros p:peca, f:forno e recipiente. O tempo total de execução da ação é definido pelos intervalos definidos pelos instantes t_0 a t_3. A duração da ação é obtida conforme expressão definida no caso de uso. Neste diagrama, a peça inicia não temperada e permanece nesta condição entre o instante inicial t_0 e o instante final t_3. A partir deste instante a peça está temperada. Entre t0 e t2, a fica pronta para têmpera.

$$duração = \left(\frac{(mp * cep * tp) + (ml * cel * tl)}{(mp * cep) + (ml * cel)} \over trl \right)$$

Onde:

mp = massa da peça; *cep* = calor específico da peça; *tp* = temperatura da peça; *ml* = massa do líquido; *cel* = calor específico do líquido; *tl* = temperatura do líquido; *trl* = taxa de resfriamento do líquido.	*Expressão OCL da duração da ação:* (((p.massaPeca * p.calEspecPeca * p.tempPeca)+ (r.massaLiq * r.calEspecLiq * r.tempLiq))/ ((p.massaPeca * p.calEspecPeca) + (f.massaForno * f.calEspecLiq)))/ (f.taxaResfLiq)

Figura 5.11 Duração da ação temperar definida em expressão e em OCL
Fonte: autor

No instante t_2 a peça é retirada do forno (p.dentro -> excludes(f)) e introduzida no recipiente (p.imersa -> includes(r)). A temperatura da peça gradativamente atinge a temperatura do recipiente a partir do instante t_2. O intervalo entre t_2 e t_3 define o tempo em que este resfriamento ocorre.

No momento t_2 em que a peça sai do forno, há uma mudança no atributo 'vazio' que pode ser considerada como instantânea. A partir deste instante, o forno está disponível para ser utilizado por outra ação, ou pela mesma ação com outros objetos. Até o instante t_2, a temperatura do forno permanece no intervalo da temperatura de têmpera definida pela peça.

O recipiente deve estar livre até o instante em que a peça precisa sair do forno. Não é estritamente necessário que este esteja livre no instante inicial da ação, apesar de estar modelado no diagrama como necessário estar livre no instante t_0.

A figura 5.11 ilustra a fórmula da duração da ação 'temperar', conforme definido anteriormente no caso de uso. Cada variável desta fórmula é um dos atributos dos objetos recebidos como parâmetro da ação. A duração da ação é obtida a partir da expressão OCL da ação com a alteração da notação infixa para prefixa.

```
(:durative-action temperar
 :parameters (?p - Peca ?f - Forno ?r - Recipiente)
 :duration (= ?duration (/ (/ (+ (* (* (massaPeca ?p) (calEspecPeca ?p)) (tempPeca ?p))
           (* (* (massaLiq ?r) (calEspecLiq ?r)) (tempLiq ?r)))
           (+ (* (massaPeca ?p) (calEspecPeca ?p))
           (* (massaLiq ?r) (calEspecLiq ?r)))) (taxaResfLiq ?r)))
 :condition
   (and
       (at start (not (temperada ?p)))
       (over all (not (temperada ?p)))
       (at start (prontoParaTemperar ?p))
       (at start (= (tempPeca ?p) (tempForno ?f)))
       (over all (= (tempPeca ?p) (tempForno ?f)))
       (at start (dentro ?p ?f))
       (over all (dentro ?p ?f))
       (at start (livre ?r))
       (over all (livre ?r))
       (at start (ligado ?f))
       (over all (ligado ?f))
       (at start (not (vazio ?f)))
       (over all (not (vazio ?f)))
       (over all (>= (tempForno ?f) (minGrausTemp ?p)))
       (over all (<= (tempForno ?f) (maxGrausTemp ?p)))
     )
   )

 :effect
   (
     (and
       (at end (temperada ?p))
       (at end (not (prontoParaTemperar ?p)))
       (at end (imersa ?p ?r))
       (at end (assign (tempPeca ?p) (tempLiq ?r)))
       (at end (not (dentro ?p ?f)))
       (at end (not (prontoParaTemperar ?p)))
       (at end (not (livre ?r))
       (at end (vazio ?f))
     )
   )
)
```

Figura 5.12 Ação temperar interpretada em PDDL 2.1
Fonte: autor

A figura 5.12 ilustra a ação durativa discreta ‘temperar’ interpretada em PDDL 2.1. A duração da ação é determinada conforme a expressão definida em OCL. A PDDL não possibilita visualizar o instante exato em que a peça sai do forno e entra no recipiente. A interpretação possível, conforme a proposta de interpretação do capítulo 4, é que a peça esteja no forno no início (at start) e durante (over all) a ação. Somente no final (at end) é possível determinar, como um efeito, que a peça não está mais no forno e está imersa no recipiente. Novamente, a precisão em que os atributos sofrem alteração de valor não corresponde exatamente ao especificado no diagrama de tempo.

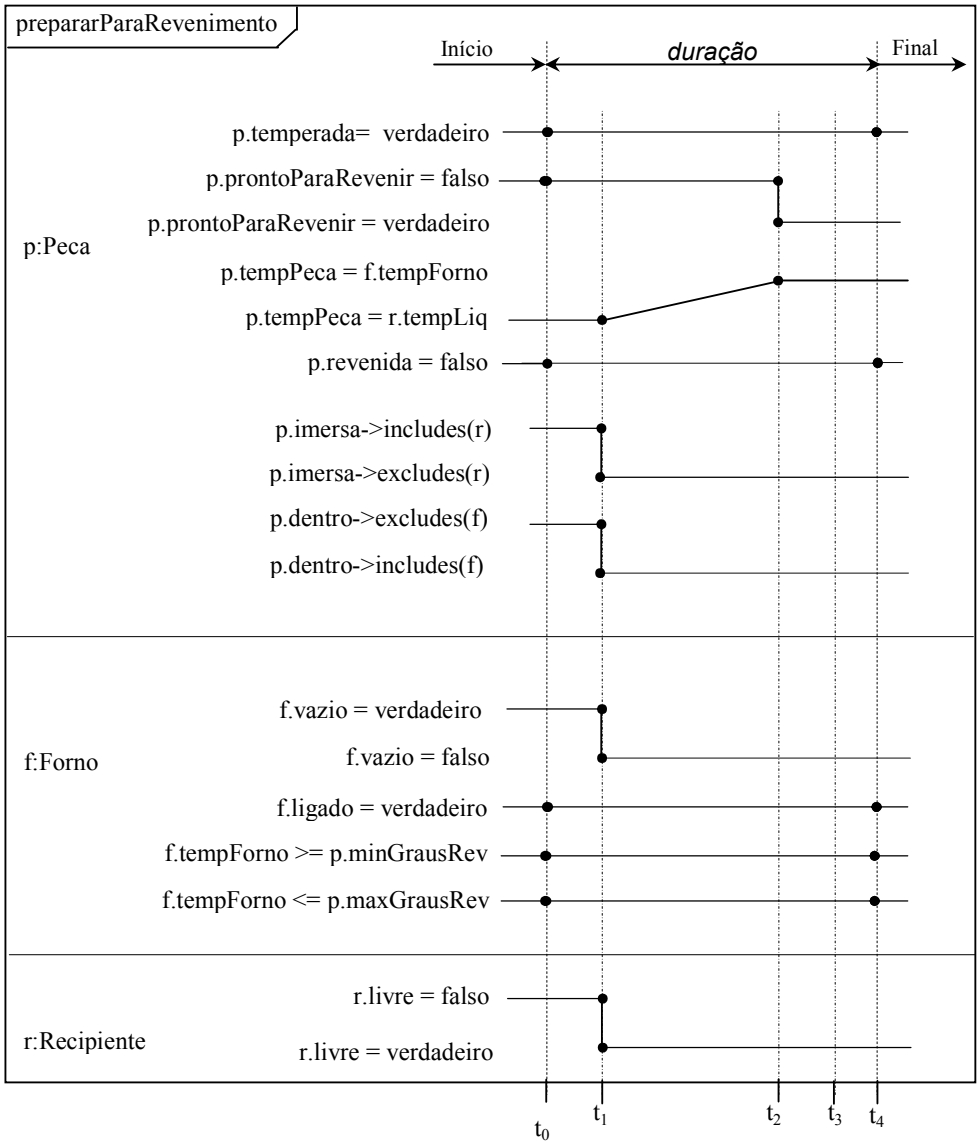


Figura 5.13 Diagrama de tempo para a ação prepararParaRevenimento
Fonte: autor

A figura 5.13 ilustra a ação 'prepararParaRevenimento' modelada em um diagrama de tempo. A ação recebe os parâmetros p:peça, f:forno e r:recipiente. O tempo de duração da ação é definido pelos instantes t_0 e t_4. A preparação para revenimento (p.imersa -> includes(r)) se inicia com a peça dentro do recipiente do instante t_0 até o instante t_1. A partir do instante t_1, quando a peça é colocada no forno (p.dentro -> includes(f)) a temperatura da peça começa a receber calor do forno e atinge sua temperatura no instante t_2. A partir deste ponto, a peça fica pronta para revenimento. O forno mantém a temperatura média de revenimento durante todo o tempo de execução e deve permanecer ligado durante todos os intervalos da ação.

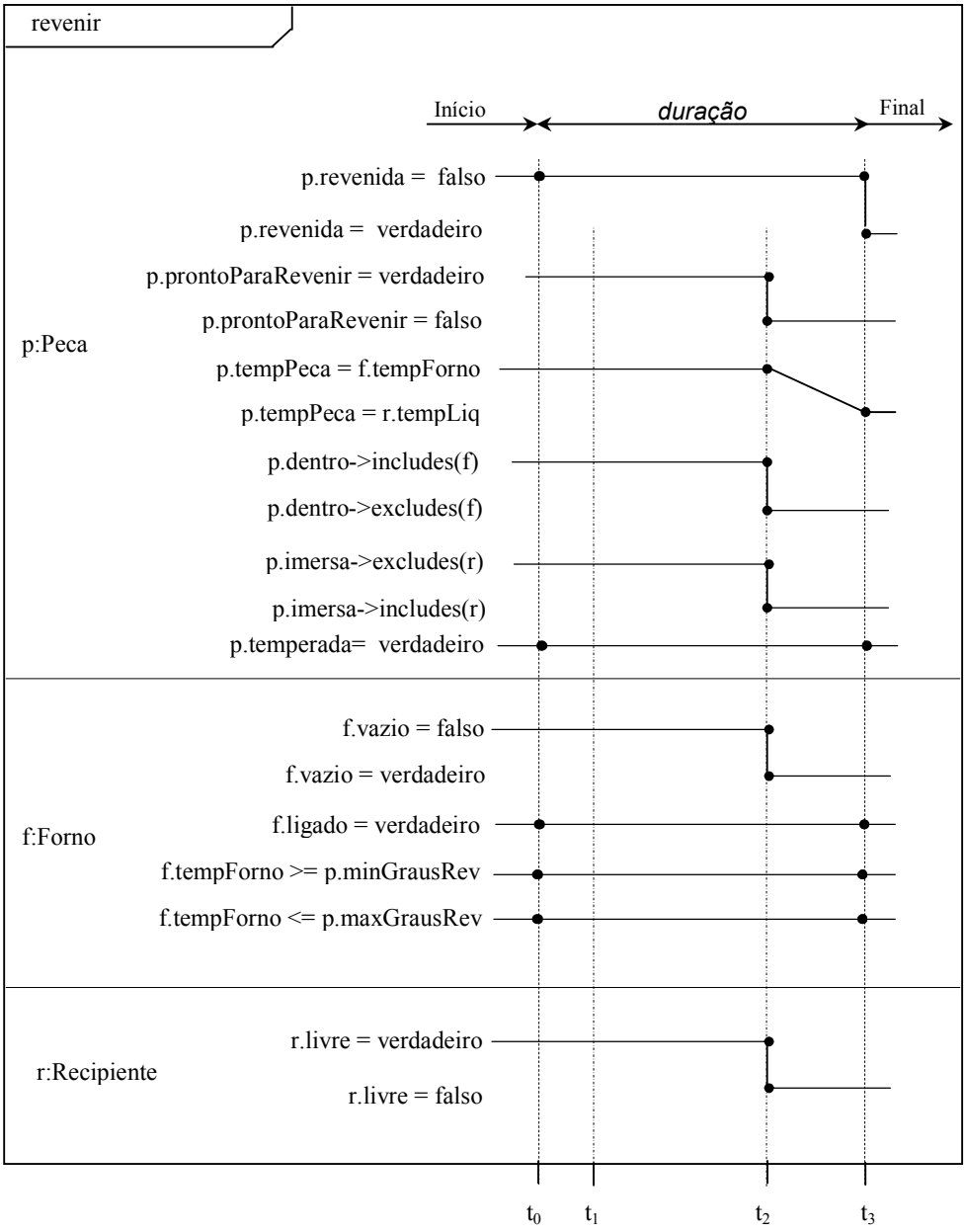


Figura 5.14 Diagrama de tempo para a ação prepararParaRevenimento
Fonte: autor

A figura 5.14 ilustra a ação durativa discreta 'revenir' modelada em um diagrama de tempo. A ação recebe os parâmetros p:peça, f:forno e r:recipiente. O tempo de duração da ação é definido pelos instantes t_0 e t_3. A peça inicialmente está dentro do forno (p.dentro -> includes(f)) e permanece não revenida até o último instante da ação, t_3, com esta dentro do recipiente (p.imersa -> includes (r)). O forno permanece na temperatura de revenimento durante todo o intervalo da ação. A peça gradativamente atinge a temperatura do recipiente a partir do instante t_2. Este tempo define o tempo principal na execução da ação.

5.2. Resumo e conclusão do capítulo

Neste capítulo fora apresentado um estudo de caso com a aplicação do diagrama de tempo proposto em um domínio de exemplo. Foi possível observar o uso em um contexto geral do domínio e nos contextos das ações, com a indexação do dinamismo das variáveis na linha cronológica da ação. Com a interpretação, foi possível observar os pontos de inconsistência entre o diagrama de tempo e a PDDL, principalmente na definição dos instantes em que as proposições têm seus valores alterados.

Desta forma, completa-se a apresentação do diagrama, com a definição e validação no capítulo 4 e a apresentação de um exemplo no capítulo 5. No próximo capítulo, uma discussão mais elaborada será apresentada como conclusão deste trabalho.

CAPÍTULO VI

6. Conclusão e trabalhos futuros

6.1. Conclusão

A área de planejamento automático evoluiu significativamente depois da primeira competição internacional de planejamento e escalonamento (AIPS 98) com diversas propostas de arquiteturas para planejamento temporal, com sistemas baseados em espaço de planos (YOUNES; SIMMONS, 2003), sistemas baseados em grafos, como o TGP (GEREVINE; SERINA, 2002) e busca através de um espaço de estados estendido com o planejador SAPA (DO; KAMBHAMPATI, 2001). Contudo, para avaliar as capacidades reais destes sistemas, é necessário fornecer-lhes elementos que compõem um mundo real. Para isto, é necessário, também, evoluir as técnicas de aquisição, documentação e manutenção do conhecimento adquirido acerca dos domínios de planejamento. Estas técnicas agregadas determinam um processo de Engenharia do Conhecimento (McCLUSKEY, 2001).

Apesar da grande contribuição dos estudos acerca do raciocínio sobre tempo em planejamento (REICHGELT, 1989) e sistemas temporais (BACCHUS; TENENBERG; KOOMEN, 1991) e (SHANAHAN, 1990), que basearam a extensão da PDDL para tratar domínios temporais, a representação temporal em planejamento ainda apresenta pontos sem uma proposta clara de representação (CUSHING *et al,* 2007) e (McCLUSKEY, 2003). Com isto, torna-se visível uma oportunidade de contribuição na área de engenharia de conhecimento para planejamento temporal, assim, o presente trabalho objetiva agregar a uma metodologia existente a capacidade de modelar aspectos temporais em domínios de planejamento.

Com base na proposta de utilizar uma linguagem de modelagem amplamente difundida e consolidada como a UML (OMG, 2003) e sua extensão para modelagem de domínios de planejamento, a UML.P (VAQUERO *et al,* 2006), foi apresentado neste trabalho um diagrama de tempo capaz de agregar aspectos temporais ao contexto do domínio, com a participação dos objetos no ciclo de vida do domínio e uma variação deste diagrama, capaz de agregar aspectos temporais às ações. Com o uso da OCL (OMG, 2001), estes diagramas podem representar um amplo espectro de problemas temporais, com a especificação de fórmulas na duração das ações e na evolução dos valores das variáveis de uma ação ao longo de sua execução.

Ao longo do trabalho, houve uma preocupação em garantir a aplicabilidade do diagrama e atingir a maior gama de problemas possíveis. Então, com o auxílio do trabalho de Allen (1983) sobre intervalos, foi demonstrado nesta dissertação que qualquer relação entre dois intervalos pode ser representada pelo diagrama de tempo proposto.

No capítulo 4 foi apresentada uma interpretação do diagrama em PDDL, porém, o diagrama proposto não é dependente de nenhuma linguagem de planejamento. Estas são necessárias para submeter um domínio aos planejadores. Conforme discussão deste capítulo, existem lacunas entre a PDDL e a UML.P a serem tratadas em trabalhos futuros, como o detalhamento dos instantes em que os atributos são atualizados no ciclo de vida da ação.

Os resultados obtidos com o diagrama proposto foram apresentados com a modelagem do domínio de exemplo no capítulo 5, que também mostra o processo completo para modelagem de um domínio em UML.P, com a visão geral através do diagrama de tempo para o contexto do domínio e uma visão específica para uma ação temporal. Portanto, sem a pretensão de induzir a uma conclusão favorável, o diagrama de tempo proposto mostrou-se como uma alternativa viável na modelagem de domínios temporais de planejamento.

6.2. Trabalhos futuros

Os pontos abaixo apontam as diretrizes para os trabalhos futuros:

- Implementação do diagrama de tempo na ferramenta itSIMPLE com a integração entre os diagramas de máquinas de estados e diagrama de classes. Esta implementação possibilitará maior precisão nas especificações realizadas no diagrama de máquina de estados;
- Desenvolver uma proposta de extensão da PDDL para proporcionar maior precisão na especificação de instantes nas ações temporais. Mesmo com a PDDL+ nível 5 (FOX; LONG, 2001) que propõe o uso de processos e eventos, ainda fica a lacuna de especificar momentos específicos de mudança de valores das variáveis ao longo da execução de ações ou processos.
- Desenvolver uma tradução formal entre o diagrama de tempo e a PDDL.
- Desenvolver uma correspondência semântica entre o diagrama de tempo da UML.P e a lógica temporal, que, como uma extensão da lógica de primeira ordem (VENEMA, 2000), possui o propósito de verificar e validar sistemas reativos de estados finitos.

BIBLIOGRAFIA

ALLEN, J. F.: ***Maintain Knowledge About Temporal Intervals.*** CACM 26, 1983 p. 832-843.

ALLEN, J. F.: ***Towards a General Theory of Action and Time.*** Department of Computer Science, University of Rochester, NY. Elsevier, 1984.

ALLEN, J. F.: ***Planning as Temporal Reasoning.*** Department of Computer Science University of Rochester, NY, 1991.

ALLEN, J. F.: ***Temporal Reasoning and Planning.*** In Readings in Planning. USA, Morgan Kaufmann, 1990.

ALLEN, J. F.; KOOMEN, J. A. ***Planning Using a Temporal World Model.*** Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 1983.

ARRINGTON, C.T. ***Enterprise Java With UML.*** OMG Press, New York, USA: Willey, 2001.

BACCHUS , F., TENENBERG, J., KOOMEN, J. ***A non-reified temporal logic for AI***. Artificial Intelligence, 52, 87–108., 1991

BLUM, A.; FURST M.: ***Fast Planning through Planning Graphs Analysis***, Artificial Intelligence, v.90, p. 281-300, 1997.

BITTNER, T.: ***Approximated Qualitative Temporal Reasoning***. Qualitative Reasoning Group, Northwestern University, 2001.

BIUNDO, S. (Ed.): ***Technological Roadmap On AI Planning and Scheduling***. Ulm, Germany, 2003.

CHAPMAN, D. ***Planning for Conjuntive Goals***. Artificial Intelligence, v.32, n.3, p.333-377, 1987.

CUSHING, W., KAMBHAMPATI, K., TLAMADUPLA, M, WELD, D. ***"Evaluating Temporal Planning Domains,"*** International Conference on Automated Planning and Scheduling (ICAPS-07). Providence, RI. September 2007.

CUSHING, W.; KAMBHAMPATI, R., WELD. ***"When is Temporal Planning Really Temporal?***," International Joint Conference on Artificial Intelligence (IJCAI-07). Hyderabad, INDIA, January 2007b.

CUNHA, M. A.: ***Lógica Temporal.*** Workshop. Minho, Portugal, 2006.

DEAN, T., McDERMOTT, D. ***Temporal Data Base Management.*** Artificial Intelligence, USA, 1987.

DEAN, T. WELLMAN, M. ***Planning and Control.*** Morgan Kaufmann, USA, 1991.

EL-KHOLY, A., RICHARD, B. ***Temporal and Resource Reasoning in Planning: The ParcPlan approach***. In Proceedings of the European Conference on Artificial Intelligence (ECAI). Wiley, USA 1996.

FIKES, R.; NILSSON, N. ***STRIPS: A new approach to the application of theorem proving to problem solving.*** Califórnia: Stanford Research Institute, 1970 (Technical Note).

FOX, M.; LONG, D. ***PDDL 2.1 An Extension to PDDL for Expressing Temporal Planning Domains***. Journal of Artificial Intelligence Research, 2003.

FOX, M.; LONG, D. PDDL+ ***level 5: An Extension to PDDL2.1 for Modelling Planning Domains with Continuous Time-dependent.*** Technical report, University of Durham, UK, 2001.

GEREVINI A.; SERINA, I. ***LPG: A Planner Based on Local Search for Planning Graphs with Actions Costs***. In: International Conference on Artificial Intelligence Planning and Scheduling - AIPS´02, 6., Toulouse, 2002. ***Preprints***. AAAI Press, 2002, p.281-290.

GHALLAB, M.; NAU, D.; TRAVERSO, P. **Automated Planning: Theory and Practice.** San Francisco, USA: Morgan Kaufmann, 2004.

GHALLAB, M. LARUELLE, H. ***Representation and Control in IxTeT, a Temporal Planner.*** In Proceedings of the International Conference on AI Planning Systems (AIPS), 1994.

GOUCH, J. ***XPDDL 0.1b: A XML version of PDDL.*** Disponível em http://www.cis.strath.ac.uk/~jg/XPDDL/. 2004. Acesso em dezembro/2008.

GREEN, C. ***Application of Theorem Proving to Problem Solving***. In: INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE IJCAI-69, s.l., 1969. ***Proceedings***, s.n., 1969, p. 219-240.

HENZINGER, T.; MANNA, Z.; PNUELI, A.: ***Timed Transition System.*** *Proceedings of the REX Workshop ``Real-Time: Theory in Practice''*, volume 600 of *Lecture Notes in Computer Science*, pages 226-251. Springer-Verlag, 1992.

HETZBERG, J.: ***On Building a Planning Toolbox***. IOS Press, Amsterda, 1996.

HOFFMANN, J.; NEBEL, B.. ***The FF Planning System: Fast Plan Generation Through Heuristic Search.*** Journal of Artificial Intelligence Research. v.14, p.253 – 302. 2001.

KAMBHAMPATI, S; DO, ***Planning Graph-based for Cost-sensitive Temporal Planning***. *Technical Report, ASU* 2001

KRIPKE, S. : ***Semantical Considerations on Modal Logic***. *Acta Philosophica Fennica.* New York, 1963. 16:83–94

KROKHIN, A.; JEAVONS, P. ***Reasoning About Temporal Relations: The Tractable Subalgebras of Allen's Interval Algebra.*** Journal of the ACM (Association of Computing Machinery), Oxford, UK, 2003.

KRUCHTEN, P. ***The Rational Unified Process: An Introduction.*** USA: Addison-Wesley, 2000.

KULPA, Z.: ***Diagrammatic Representation And Reasoning***. Machine Graphics & Vision, 3: 77-103, Poland, 1994.

LARKING, J.; SIMON, H. ***Why a diagram is worth ten thousand words***. Cognitive Science, 11: 65-99, 1987.

LADKIN, P. B.: ***The Logic of Time Representation.*** PHD Thesis. University of California. 1987.

McCARTHY, J. ***Recursive functions of symbolic expressions and their computation by machine***. Communications of the ACM 3(4):184-195, 1960.

McCARTHY, J.; HAYES, P. **Some Philosophical Problems from Standpoint of Artificial Intelligence, in Machine Intelligence 4.** California: B. Meltzer and D. Michie, 1969 pgs 463-502.

McCARTHY, J.: ***Epistemological Problems of Artificial Intelligence***. In preceding of the Fifth Intenational Joint Conference on Artificial Intelligence, p. 1038-1044, Cambridge, 1977.

McCLUSKEY, T.L.: ***PDDL: A Language with a Porpouse?*** Proceedings of the ICAPS-03 workshop on PDDL, International Conference on Automated Planning and Scheduling, UK, June, 2003 (ICAPS'03).

McCLUSKEY, T.L (Ed.) ***Knowledge Engineering for Planning***. Proceedings of the ICAPS-03 workshop on PDDL, International Conference on Automated Planning and Scheduling, UK, June, 2003b.

McCLUSKEY, T. L.; SIMPSON, R.: ***Knowledge Formulation for AI Planning*** in: Engineering Knowledge in the Age of the Semantic Web. Lecture Notes in Artificial Intelligence, 2004. No. 3257, p 449 – 465.

McCLUSKEY, T.L; ***The Nature of Knowledge Engineering***. 2001. Disponível em http://scom.hud.ac.uk/planet/roaddir/roadmap-tax.html

McDERMOTT, D.; HENDLER, J. ***Planning: What it is, What it could be, An Introduction to the Special Issue on Planning and Scheduling.*** Artificial Intelligence, Yale, USA: Elsevier, v.76, p1-16, 1995.

McDERMOTT, D. *et al*.: ***PDDL – The Planning Domain Definition Language.*** Tech Report: Yale Center for Computational Vision and Control, 1998.

McDERMOTT, D. ***The 1998 AI planning systems competition***. AI Magazine, UK, 2000.

McDERMOTT, D.: ***The Formal Semantics of Processes in PDDL***. Proc. ICAPS Workshop on PDDL, 2003.

McDERMOTT, D.: ***A Temporal Logic for Reasoning about Process and Plans***. Cognitive Science, p 101-155, 1982.

MOTTA, E.: ***The Knowledge Modeling Paradigm In Knowledge Engineering***. Handbook of Software Engineering and Knowledge Engineering. World Scientific Publishing, 2000, UK.

NEWELL, A.; SIMON, H.A. ***GPS: A program that simulates a human thought, in Edward A. Feigenbaum and Julian Feldman, Computers and Thought.*** New York: McGraw-Hill, 1963 pgs 279-293.

NEWELL, A.; SHAW, J. C.; SIMON, H.A. ***Report on a General Problem-Solving Program.*** New York: Proceedings of International Conference on Information Processing, 1959 pgs 256-264.

OMG - Object Management Group. ***Unified Modeling Language: Superstructure***, 2007. Disponível em <http://www.omg.org/docs/formal/07-11-04.pdf> Acesso em 05/01/2009.

OMG - Object Management Group. ***UML Resource Page***. 2008. Disponível em < http://www.uml.org/#Links-Tutorials>. Acesso em 05/01/2009.

OMG - Object Management Group. ***OCL 2.O – Object Constraint Language.*** Disponível em http://www.omg.org/cgi-bin/doc?formal/2003-10-14. Acesso em 05/01/2009.

PEDNAULT, E.P.D. ***ADL: exploring the Middle Ground between STRIPS and the situation calculus.*** In: International Conference On Principles of Knowledge Representation and Reasoning, Toronto, 1989. Proceedings. s.n. 1989, p.324-332.

PILONE, D.; PITMAN, N. ***UML 2.0 in a Nutshell***. Califórnia, USA: O'Reilly, 2005.

PNUELLI, A. ***The Temporal Logic of Programs***. In *Proceedings of the 18th IEEE Symposium Foundations of Computer Science (FOCS 1977)*, pages 46-57, 1977

POLLOCK, J. L.: ***Perceiving and Reasoning about Changing World***. Computational Intelligence, 14: 498-562, 1998.

REICHGELT, H. ***A comparison of first order and modal theories of time.*** In Jackson, P., Reichgelt, H., & van Harmelen, F. (Eds.), Logic-based knowledge representation, pp. 143–176. MIT Press, USA, 1989.

REITER, R. ***On Closed-World Databases.*** In H. Gallaire and J. Minker, editors, Logic and Data Bases. New York: Plenum Press, 1978 pgs 55-76.

RUSSEL, S.; NORVIG, P. ***Inteligência Artificial: tradução da segunda edição.*** Rio de Janeiro: Elsevier, 2004 – 4ª reimpressão.

RUMBAUGH, J.; JACOBSON, I.; BOOCH, G.: ***The Unified Modeling Language Reference Manual***. Reading, Mass.: Addison-Wesley, 1999.

SACERDOTI, E. D. ***The Nonlinear Nature of Plans.*** 1975. In: ALLEN, J.; HENDLER, J.; TATE, A. Readings in Planning, San Mateo: Morgan Kaufmann, 1990. p.162-170.

SHANAHAM, M. ***Solving the Frame Problem: A Mathematical Investigation of the Common Sense Law of Inertia***. MIT Press, Cambridge, 1997.

SIMPSON, R.M. ***Gipo: Graphical Interface for Planning With Objects.*** Technical Report. Department of Computing and Mathematical Sciences, University of Huddersfield, 2005.

SIMPSON, R.M.; McCLUSKEY, T. L.; ZHAO, W.; AYLETT, R.S.; DONIAT, C.: ***An Integrated Graphical Tool to support Knowledge Engineering in AI Planning.*** Proceedings of the European Conference on Planning, Toledo, Spain, 2001.

SUSSMAN, G.J. ***A Computer Model of Skill Acquisition.*** Elsevier Science Inc. New York, NY, USA. 1975.

SMITH, D., FRANK, J., JÓNSSON: ***Bridging the Gap Between Planning ad Scheduling.*** California, 2000.

SMITH, D. E., WELD, D.: ***Temporal Planning with Mutual Exclusion Reasoning***. IJCAI 1999.

SMITH, D. E.: ***The Case for Durative Actions: A Commentary on PDDL2.1***. Journal of Artificial Intelligence Research 20. California, 2003 pg. 149-154

TATE, A. ***Generating Projects Networks.*** 1977. In: Allen, J.; Hendler, J.; Tate, A. Readings in Planning, San Mateo: Morgan Kaufmann, 1990. p. 291-296.

TONIDANDEL, F.; ***Desenvolvimento e Implementação de um Sistema de Planejamento Baseado em Casos***. São Paulo: USP, Tese de Doutorado, 2003.

VAQUERO, T. E. ***itSIMPLE: ambiente integrado de modelagem e análise de domínios de planejamento automático.*** 2007. 284p. Dissertação (Mestrado em Engenharia Elétrica) – Escola Politécnica da Universidade de São Paulo, Departamento de Engenharia Mecatrônica e de Sistemas Mecânicos, São Paulo.

VAQUERO, T.S.; TONIDANDEL, F.; SILVA, J. R. ***The itSIMPLE tool for Modeling Planning Domains.*** ICAPS 2005 Competition on Knowledge Engineering for Planning and Scheduling, Monterey, California, USA. 2005.

VAQUERO, T.S.; TONIDANDEL, F.; BARROS, L. N.; SILVA, J. R. ***On the Use of UML.P for Modeling a Real Application as a Planning Problem.*** Short paper in Preceedings of ICAPS 2006 International Conference on Automated Planning and Scheduling, Cumbria, UK, 2006.

VENEMA, Y.: ***Temporal Logic.*** Notre Dame Journal of Formal Logic, 2000

WILKINGS, D. ***Practical Planning: Extending the Classical AI planning Paradigm***. San Mathew, Morgan Kaufmann, 1988.

YOUNES, L. S. SIMMONS, R. ***Probabilistic verification of discrete event systems using acceptance sampling***. In Proc. 14th International Conference on Computer Aided Verification, volume 2404 of LNCS. disponível em http://www-2.cs.cmu.edu/~lorens/papers/cav2002.pdf. Último acesso em 020/01/2009.

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo