JORGE LUIS SALVI

RELACIONAMENTOS TEMPORAIS ENTRE REDES DE PETRI E PLANEJAMENTO AUTOMÁTICO

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

CURITIBA

2009

JORGE LUIS SALVI

RELACIONAMENTOS TEMPORAIS ENTRE REDES DE PETRI E PLANEJAMENTO AUTOMÁTICO

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

CURITIBA

2009

Dedico este trabalho aos meus pais, pelo exemplo, pela confiança, pelo estímulo e por, em todos os momentos, nunca terem medido esforços em transformar este meu sonho em realidade.

AGRADECIMENTOS

Foram muitos, os que contribuíram para tornar este trabalho possível.

Meus sinceros agradecimentos...

...à minha família, pela confiança e pelo apoio;

...aos amigos do mestrado e doutorado, pelas conversas e pela amizade;

...aos professores Luiz Allan, Castilho e Cesar Tacla pelas valiosas sugestões na banca de qualificação;

...ao professor Fabiano, pela oportunidade, orientação exemplar, paciência e confiança.

RESUMO

Este trabalho apresenta relacionamentos entre problemas de planejamento temporal em inteligência artificial e alcançabilidade em redes de Petri com tempo, estendendo assim os trabalhos de Silva [1], Hickmott et al. [2], Petry [3] e Edelkamp e Jabbar [4], que não tratam informações temporais. Estas relações são dadas pela tradução de problemas de um formalismo para outro. Tais traduções proporcionam o uso dos métodos formais de análise das redes de Petri e também os sofisticados algoritmos de busca da área de planejamento automático para solucionar problemas que fazem uso informações temporais. É comparado o poder de expressividade das transições temporais e temporizadas de uma rede de Petri com uma ação durativa em PDDL, demonstrando que as duas primeiras têm poder de expressividade reduzido em relação à última. Também é apresentada uma rede de Petri temporal capaz de simular a execução de um conjunto de ações durativas descritas pela linguagem PDDL.

ABSTRACT

This work presents relationships among temporal planning problems in artificial intelligence and reachability in Petri nets with time. It extends the works of Silva [1], Hickmott et al. [2], Petry [3], and Edelkamp and Jabbar [4] to support temporal information. These relationships are given by the translation of problems from one formalism to another. Such translations permit the use of the formal analysis methods of the Petri nets as well as the fast search algorithms of automatic planning area to solve problems with temporal information. The expressiveness power of the time and timed transitions of a Petri net is comparated with the duractive actions in PDDL language, showing that the first two have reduced expressiveness power. Also, it is presented a temporal Petri net capable to simulate the execution of a set of duractive actions described in PDDL.

LISTA DE FIGURAS

LISTAGENS

LISTA DE SIGLAS

ADL	do inglês *Action Description Language*
CN	do inglês *Constraint Network*
CSP	do inglês *Constraint Satisfaction Problem*
D-Graph	grafo de distância
IA	Inteligência Artificial
ICAPS	do inglês *International Conference on Automated Planning and Scheduling*
IPC	do inglês *International Planning Competition*
PDDL	do inglês *Planning Domain Definition Language*
PMDC	PDDL Model for DeadLock Checking
RdP	Rede de Petri
STN	do inglês *Simple Temporal Networks*
STP	do inglês *Simple Temporal Problems*
STRIPS	do inglês *Stanford Research Institute Problem Solver*
TCSP	do inglês *Temporal Constraint Satisfaction Problem*
TdPN	Timed Petri Net (Rede de Petri Temporizada)
TPN	Time Petri Net (Rede de Petri Temporal)

SIMBOLOGIA MATEMÁTICA

$t_1 \# t_2$	t_1 é independente de t_2.
M[t>M'	o disparo de t leva a rede da marcação M para a marcação M'.
M[t>	t é disparável a partir da marcação M.
$\widehat{p}$	é o lugar complementar de p.
$R(M_0)$	conjunto de todas as marcações alcançáveis da rede.
$RPM_1 \cong RPM_2$	duas redes de petri são equivalentes entre sí.
cfl(t,M)	é o conjunto de conflitos de t sobre uma marcação M.
$t_1 \neq t_2$	as transições $t_1 \neq t_2$ são independentes.
v(t)	refere-se aos pré e pós-conjuntos de t.
M(p)	é o número de marcas do lugar p em uma marcação M.
Pre(p,t)	é o peso do arco de lugar-transição.
Pos(p,t)	é o peso do arco de transição-lugar.
k(p)	é a capacidade máxima de marcas de um lugar $p \in P$.
Z(x)	é o tempo de consumo do elemento x da rede.
S(t)	lugares de $t\bullet$ que, eventualmente, poderão estar em situação de contato.

SUMÁRIO

CAPÍTULO 1

INTRODUÇÃO

Planejamento automático é uma área central da Inteligência Artificial (IA), que envolve o projeto de linguagens e modelos computacionais para raciocínio sobre ações, mudança e tempo [8]. De forma geral, a função de um planejador automático é encontrar uma sequência de ações que, partindo de um estado inicial, possa chegar ao objetivo proposto.

Para descrever um problema de planejamento clássico são utilizadas linguagens de representação do conhecimento. A que mais se destaca é a STRIPS [9], por conseguir definir os estados, ações e objetivos necessários para serem aplicados à planejadores clássicos. Apesar disso, esta linguagem não permite modelar problemas que levam em consideração informações temporais.

Tendo em vista que a representação e o raciocínio sobre o tempo têm crucial importância para muitos sistemas de inteligência artificial [10], assim como para muitos outros sistemas do mundo real, então torna-se atrativo fazer com que planejadores se adaptem à esta característica. Assim, surgiram as linguagens de representação temporal e também os planejadores temporais, a fim de suprir esta lacuna.

Atualmente, dentre os planejadores temporais, pode-se citar ZENO [11], VHPOP [12], SAPA [13] e CRIKEY [14]. Já dentre as linguagens temporais, a que mais se destaca é a PDDL [15], por ter um alto poder de expressividade e por ter se tornado um padrão nas competições do IPC (*International Planning Competition*), parte do ICAPS (*International Conference on Automated Planning and Scheduling*). Além disso, a PDDL permite descrever problemas tanto para o planejamento clássico quanto para o planejamento temporal.

Contudo, geralmente, encontrar um plano em planejamento temporal é mais difícil do que no planejamento clássico [16]. Sendo assim, uma das táticas, também usadas pelo planejamento clássico, é a tradução de um problema para diferentes tipos de formalismo,

com o intuito de tirar proveito de técnicas mais elaboradas que cada um deles pode proporcionar.

Uma ferramenta que pode ser promissora para resolver problemas de planejamento são as Redes de Petri (RdP). Este formalismo possibilita a modelagem gráfica e matemática de sistemas concorrentes, assíncronos, distribuídos, paralelos, não-determinísticos e/ou estocásticos, permitindo a modelagem, análise formal e a concepção de sistemas dinâmicos a eventos discretos [17]. Uma das grandes vantagens de se utilizar redes de Petri é que diferentes tipos de problemas, tais como representação de recursos e tempo, podem ser tratados com uso de um mesmo formalismo.

Sabendo dos potenciais benefícios que a tradução de um problema para outro pode trazer, Silva [1] e Hickmott et al. [2] propõem a tradução de problemas de planejamento para RdP, sem levar em consideração informações temporais. Por outro lado, Petry [3] e Edelkamp e Jabbar [4] propõem a tradução no sentido oposto. Como Peterson [18] e Allen [19] afirmam que as redes de Petri e planejadores automáticos têm sido aplicados especialmente em sistemas temporais onde os eventos podem ocorrer em paralelo, então o presente trabalho estende as abordagens apresentadas por Silva [1], Hickmott et al. [2], Petry [3] e Edelkamp e Jabbar [4] estabelecendo assim novas relações, entre ambos os formalismos, que permitem traduzir problemas que fazem uso de informações temporais. Na figura 1.1 são apresentadas as relações citadas, onde aquelas identificadas por linhas tracejadas são as novas relações descritas pelo presente trabalho.

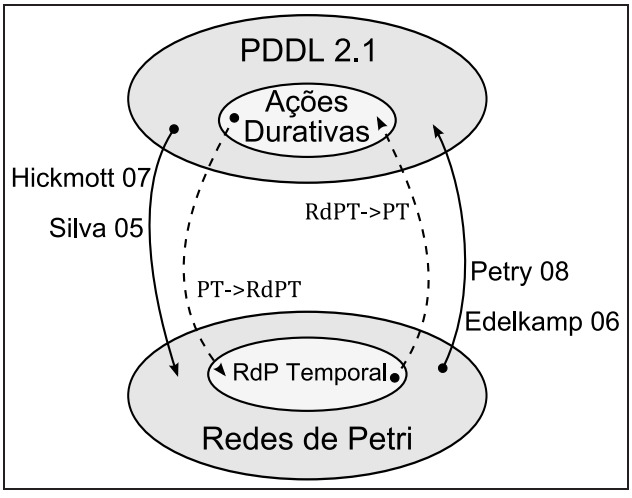

Figura 1.1: Traduções feitas entre diferentes formalismos de planejamento [5].

Outra questão importante abordada está relacionada ao poder de expressividade dos formalismos. São apresentadas fortes evidências de que as redes de Petri não possuem a mesma capacidade de representação da linguagem PDDL para problemas que estão condicionados à tratar informações temporais.

De forma geral, o presente trabalho inicia com algumas noções básicas sobre planejamento seguido pela descrição das formas de representação de conhecimento, em especial a linguagem PDDL. No capítulo 3 é feita uma abordagem sobre planejamento temporal apresentando as suas principais características. No capítulo 4, é apresentado o funcionamento e as propriedades das redes de Petri. Em seguida são apresentados os métodos de tradução entre redes de Petri com tempo e planejamento temporal e, por fim, as conclusões e trabalhos futuros.

CAPÍTULO 2

REPRESENTAÇÃO DO CONHECIMENTO EM PLANEJAMENTO

Este capítulo aborda algumas definições básicas sobre planejamento em inteligência artificial que serão utilizados ao longo deste trabalho, seguido por uma breve descrição sobre a evolução da representação do conhecimento aplicada em planejamento automático. Em seguida, também é apresentada a linguagem PDDL, bem como a sua utilização na modelagem de cenários clássicos.

2.1 Conceitos básicos em planejamento

O termo planejamento tem diferentes significados para diferentes grupos de pessoas [20]. Em IA esse mesmo termo não tem uma definição precisa, apesar de ser bastante difundido na área [21]. Neste trabalho será assumido que planejamento é o processo que procura por um plano dentro de um conjunto finito de ações que transformam o estado corrente do mundo, de maneira que saindo de um estado inicial possa se chegar a um estado objetivo. Um estado do mundo é caracterizado pela descrição do ambiente onde o agente se encontra em um dado instante de tempo. Esta descrição é feita através de um conjunto de variáveis de estado. Um novo estado do mundo pode ser obtido através da execução de uma ação, o que acarreta na mudança do conjunto de variáveis de estado que compõem o mundo.

Um problema de planejamento envolve um espaço de estados que captura todas as situações que podem ocorrer, sendo que em boa parte das aplicações o espaço de estados é muito grande para ser tratado [20].

Definição 1 (Literais)**.** *Define-se* $\mathtt{V}$ *como um conjunto de variáveis de estado positivas. O conjunto de literais sobre o conjunto* $\mathtt{V}$ *é* $\mathtt{L} = \mathtt{V} \cup \{\neg \mathtt{v} \mid \mathtt{v} \in \mathtt{V}\}$.

Definição 2 (Predicado)**.** *Um predicado* $\mathtt{R(v)}$ *é uma expressão que define uma propriedade* $\mathtt{R}$ *para um conjunto de literais* $\mathtt{v}$*, tal que* $\mathtt{R(v)}$ *é verdadeiro ou falso.*

Definição 3 (Estado)**.** *Um estado é a descrição do mundo em dado instante de tempo, isto é, um conjunto de variáveis de estado verdadeiras naquele instante.*

Definição 4 (Ação)**.** *Uma ação é um operador que transforma um estado do mundo em um outro. Essa transformação é dada pela condição* $\mathtt{o} = \langle \mathtt{Cond}, \mathtt{Eff} \rangle$*, onde* $\mathtt{Cond}$ *é o conjunto de predicados que compõem as condições e* $\mathtt{Eff}$ *é uma dupla* $\langle \mathtt{Add} \cup \mathtt{Del} \rangle$*, sendo* $\mathtt{Add}$ *o conjunto de literais positivos e* $\mathtt{Del}$ *o conjunto literais negativos, que compõe os efeitos da ação.*

Definição 5 (Plano)**.** *Um plano* $\mathtt{PLN}$ *é um conjunto parcialmente ordenado de ações que leva à solução de um dado problema, ou seja,* $\mathtt{PLN} = \{\mathtt{o_1}, \mathtt{o_2}, ..., \mathtt{o_n}\}$*, tal que* $\mathtt{o_i}$ *é uma ação.*

2.2 Representação do conhecimento

A representação do conhecimento pode ser definida como um conjunto de convenções sintáticas e semânticas que torna possível descrever coisas [22], formando uma linguagem. Em planejamento, as linguagens de representação são utilizadas para descrever estados, ações e objetivos. Segundo Russell e Norvig [22], uma linguagem deve ser suficientemente expressiva para descrever uma grande quantidade de problemas, mas ao mesmo tempo deve ser restrita ao ponto de permitir que os algoritmos possam operá-la. Desta forma, é possível desenvolver algoritmos poderosos e também flexíveis para a resolução de problemas.

A forma de representação definida como STRIPS (*Stanford Research Institute Problem Solver*) [9] surgiu após o lançamento do primeiro sistema de planejamento (de mesmo nome: STRIPS) em 1971, que foi projetado como o componente de planejamento para o robô Shakey. A partir de então, por ser uma linguagem de descrição cujo modelo de representação é simples e compacto, a linguagem STRIPS passou a ter muito mais influência do que as abordagens baseadas em lógica e prova de teoremas. Assim, a maioria

dos planejadores existentes passou a utilizar formas semelhantes àquelas presentes em STRIPS.

A linguagem STRIPS representa o estado do mundo utilizando conjunções de proposições que usam uma representação de primeira ordem (predicados). Com isso, os predicados especificados são dados como verdadeiros ou, caso contrário, são dados como falsos, ou seja, não são assumidos pelo estado do mundo em questão. Com este mecanismo, a linguagem é capaz de descrever conjuntos de ações, que são constituídas por conjunções de predicados que formam as pré-condições e os efeitos da mesma. De forma semelhante, também é descrito o estado inicial e objetivo do problema. As ações têm condições que devem ser verdadeiras para que sejam executadas e assim transformar o mundo com a aplicação de seus efeitos, que nada mais são do que a adição ou a remoção de predicados do estado corrente do mundo.

Apesar da linguagem STRIPS ter tido muito sucesso, com o passar do tempo ela se tornou insuficientemente expressiva para modelar alguns problemas de planejamento, especialmente problemas que necessitam computar informações temporais. Foram desenvolvidas muitas variantes dessa linguagem, entre elas a linguagem ADL (*Action Description Language*) [23]. Esta relaxou algumas das restrições da linguagem STRIPS permitindo a representação de disjunção nas pré-condições, condições nos efeitos, e quantificadores universais nas pré-condições e efeitos, tornando possível codificar problemas mais realistas.

Como já mencionado, houve muitas derivações da linguagem STRIPS, devido a um grande número de pesquisadores na área de planejamento automático. No entanto, por causa das diferentes nomenclaturas e formatos usados para explicar seus experimentos, tornou-se necessário estabelecer critérios de forma que todos tivessem uma estrutura padrão para descrever os problemas de planejamento. A seção a seguir apresenta a linguagem PDDL [15, 24] na qual o presente trabalho está focado e que, atualmente, é referência padrão para a comunidade de planejamento.

2.3 PDDL

A linguagem PDDL (*Planning Domain Definition Language*) é uma extensão da linguagem STRIPS clássica, que inclui diversas funcionalidades opcionais, tais como: efeitos condicionais, variáveis tipadas, quantificadores universais e existenciais, ações durativas, entre outras. Desenvolvida pelo comitê da competição IPC-98 (*International Planning Competition*), esta linguagem suporta a representação de planejamento clássico e também as extensões temporais. Com o passar do tempo foram incorporadas novas funcionalidades, e hoje pode ser dividida em 5 níveis, conforme seu poder de expressividade:

- Nível 1: STRIPS e ADL;
- Nível 2: extensões numéricas (*fluents*);
- Nível 3: ações durativas discretas;
- Nível 4: ações durativas contínuas;
- Nível 5: consiste em todos as extensões anteriores mais o suporte adicional à eventos instantâneos e processos físicos.

Segundo Fox [24], nas competições apenas são usados os três primeiros níveis, pois para os níveis 4 e 5 ainda não há tecnologia de planejamento suficientemente avançada para tratar as complexidades adicionais.

Em PDDL, a linguagem é centrada na descrição das ações, onde são usadas pré e pós-condições para descrever a aplicabilidade e os efeitos. O comportamento do domínio de planejamento é dado pela descrição dessas ações, separando-as da descrição do problema, que apresenta os objetos que podem fazer parte do mundo, as condições iniciais do ambiente e os objetivos que se deseja alcançar. Sendo assim, um problema de planejamento fica composto pela união das descrições do domínio e do problema.

Definição 6 (Domínio)**.** *Um domínio é dado por uma tripla* $\langle \mathtt{R}, \mathtt{F}, \mathtt{A} \rangle$*, onde* $\mathtt{R}$ *são predicados,* $\mathtt{F}$ *são funções numéricas aplicadas sobre os predicados e* $\mathtt{A}$ *são ações aplicáveis a um determinado mundo [24].*

Definição 7 (Problema)**.** *Um problema é dado por uma tripla* $\langle \mathtt{O}, \mathtt{I}, \mathtt{G} \rangle$*, tal que* `O` *são os objetos do domínio que irão compor o mundo,* `I` *é o estado inicial e* `G` *é o objetivo do problema [24].*

Definição 8 (Problema de Planejamento)**.** *Um problema de planejamento em PDDL é dado por um par* $\langle \mathtt{Dom}, \mathtt{Prob} \rangle$*, onde* `Dom` *é a descrição do domínio e* `Prob` *é a descrição do problema [24].*

2.3.1 Ações durativas

A partir da IPC-2002 foi adicionada a noção de ações durativas ao PDDL. A linguagem PDDL é descrita aplicando restrições sobre as condições e os efeitos. As restrições temporais aplicáveis às condições podem ser associadas tanto no início (*at **S**tart*) quanto no final (*at **E**nd*) das ações, além de possibilitar restrições invariantes que devem ser mantidas durante (***O**ver all*) todo o intervalo da ação. Para os efeitos apenas são consideradas as restrições de início e de fim. A figura 2.1 apresenta um esquema para representação de uma ação temporal (retângulo branco) que é composta pelo seu nome e a sua duração. Além disso, a ação contém as restrições temporais aplicáveis às condições (parte superior da ação) e aos efeitos (parte inferior).

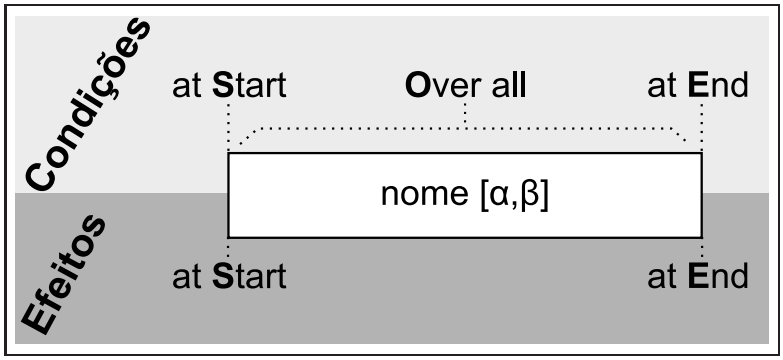

Figura 2.1: Notação das restrições temporais de uma ação.

Definição 9 (Ação Durativa)**.** *Uma ação durativa em PDDL é uma extensão da Definição 4 na qual os predicados das condições e dos efeitos passam a ser temporalmente detalhados com as restrições temporais* at start*,* over all *e* at end *além da adição de uma duração, dada pelo intervalo* $[\alpha, \beta]$*, à ação, tal que* $\alpha \in \mathbb{Q}^+$*,* $\beta \in (\mathbb{Q}^+ \cup \infty)$ *e* $\beta \geq \alpha$ *. Assim sendo,*

$\mathtt{o} = \langle \mathtt{Cond_s}, \mathtt{Cond_o}, \mathtt{Cond_e}, \mathtt{Eff_s}, \mathtt{Eff_e}, [\alpha, \beta] \rangle$ *onde* $\mathtt{Cond_s}, \mathtt{Cond_o}, \mathtt{Cond_e}$ *são os conjuntos de condições aplicados às restrições temporais* at start*,* over all *e* at end *e* $\mathtt{Eff_s}, \mathtt{Eff_o}, \mathtt{Eff_e}$ *são os conjuntos de efeitos aplicados às mesma restrições temporais, respectivamente.*

A duração de uma ação pode ser definida de três maneiras distintas:

- A primeira é definida de maneira *fixa*, onde a duração da ação é especificada de forma explícita e, portando, sempre conhecida, não podendo ser alterada. Ocorre sempre que o valor de $\alpha = \beta$. Ex.: `(= ?duration 3)`

- A segunda forma é usando uma *função* que depende do valor de outras variáveis para determinar a duração. Ex.: o tempo que uma pessoa leva para caminhar de um lugar para outro considera a velocidade média com que a pessoa anda e também qual é a distância entre origem e destino.
  `(= ?duration (/ (- ?destino ?origem) ?vel_media))`

- A última é usando *inequações*, onde é possível especificar o instante de tempo a partir do qual a ação deve ocorrer, isto é, pelo intervalo $[\alpha, \infty]$. Quando o intervalo de tempo é dado por $[\alpha, \beta]$, tal que $\beta \geq \alpha$, então a duração da ação é descrita por duas desigualdades Ex.: O tempo para descarregar um caminhão leva no mínimo 2 horas e no máximo 4 horas.
  `(and (<= ?duration 4) (>= ?duration 2))`

Considerando o símbolo grego λ (*lambda*) para denotar uma pequena fração de tempo que separa dois instantes de tempo, tal que $\lambda > 0$, então as restrições aplicáveis às condições são descritas por:

- *at start:* o predicado deve ser válido no início da ação (t_s) e, pelo menos, t_s - λ unidades de tempo antes que t_s.

- *at end:* o predicado deve ser válido no fim da ação (t_e) e, pelo menos, t_e - λ unidades de tempo antes de t_e.

- *over all:* o predicado deve ser válido durante o intervalo de t_s + λ até t_e - λ. Por exemplo, se uma ação tem duração igual a 3 unidades de tempo e é iniciada no tempo 0, considerando que $\lambda = 0,01$, então *over all* irá ocorrer no intervalo de tempo 0 + 0,01= 0,01 até 3 - 0,01= 2,99.

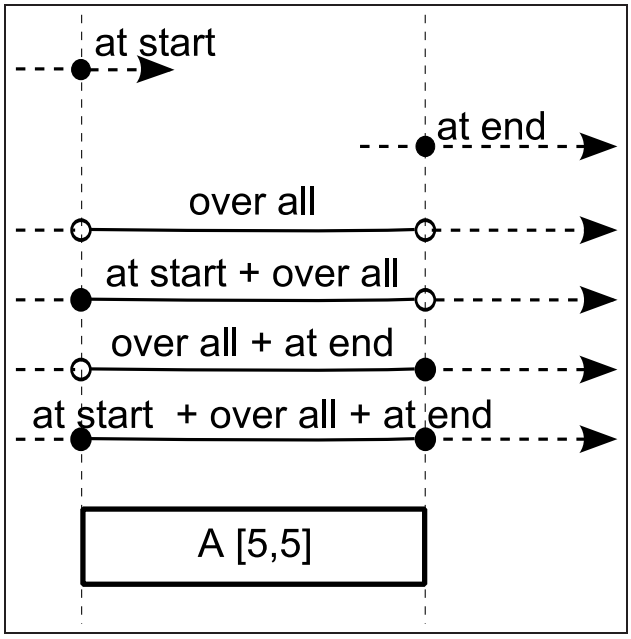

Figura 2.2: Restrições temporais aplicáveis às condições de uma ação em PDDL.

As restrições apresentadas anteriormente estão, respectivamente, representadas pela figura 2.2. As linhas tracejadas chegando a um ponto (aberto ou fechado) significam que o predicado tem que ser verdadeiro pelo menos λ unidades de tempo antes do intervalo. Quando as linhas tracejadas seguidas por uma seta se apresentam após o intervalo quer dizer que o predicado tem que ser verdadeiro pelo menos λ unidades de tempo depois do intervalo, podendo se propagar para o futuro. As linhas sólidas significam que o predicado deve se manter verdadeiro durante este período.

Quando é necessário que um fato se mantenha em um intervalo de tempo fechado, então é necessário que os três tipos de restrições sejam associadas para este mesmo fato, como por exemplo, *(at start p)*, *(over all p)* e *(at end p)*. Isto ocorre porque *over all* tem intervalo aberto no início e no fim, como apresentado na figura 2.2.

As restrições que podem ser aplicadas aos efeitos de uma ação são:

- *positivo at start:* o predicado será válido a partir do ponto de tempo t_s e se propagará para o futuro.

- *negativo at start:* o predicado deixa de ser verdadeiro a partir do ponto de tempo t_s. O predicado em questão não precisa ser necessariamente verdadeiro antes do tempo t_s.

- *positivo at end:* o predicado será válido a partir do ponto de tempo t_e e se propagará para o futuro.

- *negativo at end:* o predicado deixa de ser verdadeiro a partir do ponto de tempo t_e.

As restrições citadas são apresentadas pela figura 2.3. Os círculos com preenchimento sólido seguidos por uma linha tracejada com a seta indicam que o efeito se propagará para o futuro. Já as linhas sólidas seguidas por um círculo sem preenchimento significam que o predicado em questão pode ou não ser verdadeiro antes de ocorrer a negação do predicado.

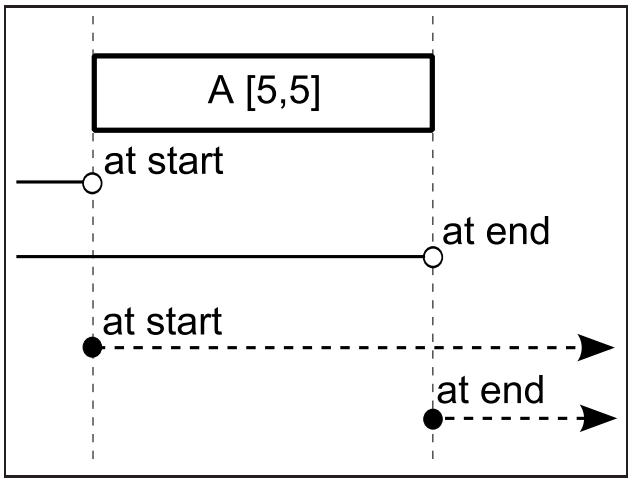

Figura 2.3: Restrições temporais aplicáveis aos efeitos de uma ação em PDDL.

É importante salientar que os efeitos apenas são efetivamente aplicados no mundo se todas as condições da ação forem atendidas, caso contrário a ação é dada como não aplicável naquele momento. Um exemplo típico ocorre quando as condições estão no final e efeitos no início da ação. Isto acontece porque quando uma ação gera, por exemplo, P como efeito no início da ação dando impressão que este efeito já foi aplicado ao mundo,

mas ele apenas se propagará para as ações que estão sendo executadas em paralelo naquele instante. No entanto, para P ser realmente um fato do mundo é necessário que ao final da ação todas as condições tenham sido satisfeitas.

A seguir é apresentado, de foma básica, como a PDDL é utilizada para descrever problemas de planejamento clássico. Para maiores detalhes sobre a linguagem PDDL, consulte o trabalho de Fox e Long [24].

2.4 Planejamento clássico em PDDL

Para exemplificar a utilização do PDDL em planejamento tomaremos como exemplo o domínio *logistics* [25], que se caracteriza pelo transporte de objetos entre diferentes localidades, utilizando caminhões para transporte dentro de uma mesma cidade e aviões para transporte entre cidades. Porém, para simplificar serão utilizados apenas os seguintes elementos para a representação do mundo:

- Há apenas 2 pacotes (P1, P2) e um caminhão (C1) para efetuar a movimentação dos pacotes;
- Os pacotes podem ser movimentados apenas entre as localidades L1 e L2;
- A capacidade de carga de um caminhão é um pacote, portanto, uma vez carregado o caminhão somente pode ser descarregado ou movimentado;
- Um pacote somente pode ser carregado em um caminhão se ambos estão na mesma localização.

Tendo conhecimento sobre o sistema a ser modelado então é possível descrever o arquivo do domínio. O exemplo do domínio citado acima é apresentado na listagem 2.1. Note que o primeiro passo é atribuir um nome ao domínio (linha 1). Em seguida são definidos os requisitos que serão necessários para representar a descrição do modelo e que neste caso são: *strips* e *typing* (que permite atribuir tipos na declaração das variáveis). Além destes há outros, que podem estender ainda mais a expressividade da descrição do domínio. Entre os que são mais comumente utilizados estão:

- *negative-preconditions*: permite a descrição de predicados negados nas condições da ação;
- *conditional-effects*: permite o uso de *when* no efeito das ações (quando X é verdade então Y);
- *fluents*: permite a definição de predicados numéricos e o uso de operações aritméticas;
- *universal-preconditions*: permite o uso do quantificador universal (*forall*) na descrição dos efeitos (para todo X faça Y).

Como foi definido o uso de tipos, então também é necessário efetuar a definição dos mesmos. Portanto, foram criados os tipos "local", "caminhao" e "pacote", sendo que os dois últimos são do tipo "objeto", ou seja, um objeto pode ser um caminhão ou um pacote. Em seguida são criados os predicados que irão compor as regras para a criação das ações. Aqui está a razão de se criar o tipo "objeto", pois assim é possível criar um único predicado (linha 7) para dizer onde um caminhão ou um pacote estão. Como o caminhão só pode carregar um pacote, então o outro predicado usado serve apenas para indicar se o caminhão está carregado ou não.

Por último, basta descrever as ações que irão compor o cenário. Desta forma, foram definidas três ações: carregar, descarregar e mover. Os parâmetros são definidos com o uso de variáveis (sempre iniciadas com "?"), que quando instanciadas permitem seguir os testes com as condições das ações. Tomando como exemplo a ação carregar, as pré-condições da ação serão satisfeitas se, e somente se, os parâmetros instanciados se enquadrarem aos predicados da pré-condição, ou seja, o caminhão deverá estar vazio e estar no mesmo lugar do pacote. O efeito da ação "carregar" é tirar o pacote do local onde o mesmo se encontra e o colocá-lo no caminhão. Para o restante das ações, o comportamento de sua execução se dará da mesma forma.

```
(define (domain logistics)
  (:requirements :strips :typing)
  (:types objeto
          caminhao pacote - objeto
          local )
  (:constants L1 L2 - local)
  (:predicates (em ?o - objeto ?l - local)
               (vazio ?c - caminhao) )

  (:action carregar
     :parameters (?c - caminhao ?p - pacote ?l - local)
     :precondition (and (em ?c ?l)
                        (em ?p ?l)
                        (vazio ?c) )
     :effect (and (not (em ?p ?l))
                  (not (vazio ?c)) )
  )
  (:action descarregar
     :parameters (?c - caminhao ?p - pacote ?l - local)
     :precondition (and (not (vazio ?c))
                        (em ?c ?l))
     :effect (and (em ?p ?l)
                  (vazio ?c))
  )
  (:action mover
     :parameters (?c - caminhao ?origem ?dest - local)
     :precondition (and (em ?c ?origem) )
     :effect (and (not (em ?c ?origem))
                  (em ?c ?dest) )
  )
)
```

Listagem 2.1: Uma versão do domínio de *logistics*.

Com a descrição do domínio completa, o próximo passo consiste em especificar a descrição do arquivo de problema. Para isso é necessário saber qual é o estado inicial e o estado objetivo, como por exemplo os estados apresentados pela figura 2.4, onde dois pacotes e também o caminhão estão inicialmente localizados no lugar L1 e o objetivo é ter o pacote P1 no lugar L2 e o caminhão no lugar L1. Sabendo disso, o primeiro passo é especificar o nome do problema e em seguida o nome do domínio a que ele pertence. Posteriormente são especificados os objetos que compõe o mundo. Note que os lugares L1 e L2 não foram declarados aqui, pois estes são constantes e já foram declaradas no arquivo de domínio.

A listagem 2.2 apresenta, nas linhas 5 a 9, o estado inicial do domínio *logistics* formulados em PDDL e também a formulação dos objetivos, nas linhas 10 a 12.

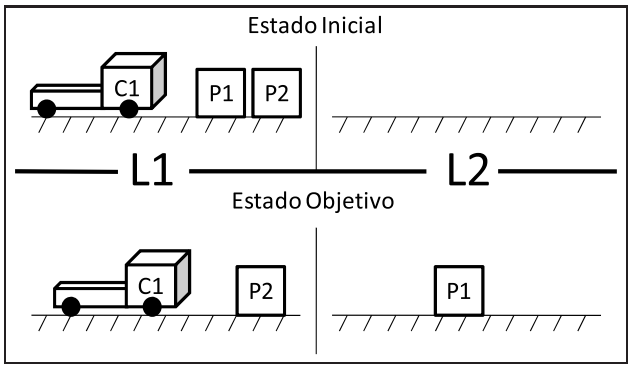

Figura 2.4: Representação do estado inicial e objetivo do mundo.

```
(define (problem p_logistics)
  (:domain logistics)
  (:objects P1 P2 - pacote
            C1 - caminhao)
  (:init
        (em P1 L1)
        (em P2 L1)
        (em C1 L1)
        (vazio C1) )
  (:goal (and (em P1 L2)
              (em C1 L1) )
  )
)
```

Listagem 2.2: Descrição de um problema para *logistics*.

A resposta para o problema pode ser descrita como: carregar P1 no caminhão, mover o caminhão para L2, descarregar o caminhão e voltar com o caminhão para L1, conforme listagem 2.3. Sendo assim, é possível perceber que, mesmo não tendo o tempo explicitamente descrito nas ações, no planejamento clássico há uma noção temporal aplicada à execução do resultado encontrado, na qual os passos do plano são executadas de forma parcialmente ordenada, o que não quer dizer que a execução de uma ação se dá imediatamente após o término de outra. Para este caso, a solução apresentada é ótima, ou seja, é o menor número de ações que leva do estado inicial ao objetivo.

```
1:(carregar C1 P1 L1)
2:(mover C1 L1 L2)
3:(descarregar C1 P1 L2)
4:(mover C1 L2 L1)
```

Listagem 2.3: Solução do problema apresentado.

2.5 Considerações

Este capítulo tratou de apresentar como os problemas de planejamento são modelados de maneira que os planejadores possam absorver as informações que representam o conhecimento sobre um determinado problema. Foi visto também que a capacidade de representação da STRIPS é limitada, assim como os planejadores clássicos, que, apesar de resolver uma grande quantidade de problemas, são incapazes de lidar com informações temporais.

Por outro lado, a representação PDDL tem um poder de expressividade bastante elevado, podendo, inclusive, representar ações durativas. Com isso, a PDDL tornou-se um padrão para a representação de conhecimento na área de planejamento, tanto clássico quanto não-clássico.

CAPÍTULO 3

PLANEJAMENTO TEMPORAL

Apesar de no planejamento clássico ser possível resolver uma grande variedade de problemas, adicionar noções temporais aos modelos possibilita o tratamento de outros problemas onde o tempo é um fator fundamental.

As definições apresentadas neste capítulo são fundamentadas nos trabalhos de Cushing et al. [6, 26], Wullinger [27] e Ghallab et al. [28], os quais explicam o que é e como funciona o planejamento temporal. Além disso, a linguagem PDDL é utilizada como base para a descrição de exemplos e das definições formais deste capítulo. Ao final é exposto um exemplo de modelagem temporal usando a linguagem PDDL.

3.1 Representação do tempo

Os problemas de representação e raciocínio temporal abrangem áreas multidisciplinares, inclusive a de Inteligência Artificial [29]. Segundo Villa [10], o tempo é modelado como um conjunto infinito de instantes, denso e não-circular sobre o qual há uma relação de ordem definida. No entanto, esta grandeza possibilita sequencializar eventos, comparar a duração de eventos assim como o intervalo existente entre eles.

Conforme apresentado por Villa [10], Lever e Richards [30] e Allen [19], o tempo pode ser dividido em dois períodos: não decomponível, chamado de instante (ponto do tempo), e período decomponível, chamado de intervalo (intervalo de tempo). O primeiro pode ser considerado como um intervalo com duração igual a zero e se refere, de forma precisa, a um momento no qual uma ação (evento) pode ocorrer. Já um intervalo de tempo corresponde ao período de tempo entre dois pontos de tempo (x, y) tal que $x \leq y$, ou seja, um pedaço do espaço de tempo no qual os eventos ocorrem [29].

Definição 10 (Duração)**.** *Duração é o período de tempo ocupado por um intervalo, ou seja, é a diferença de tempo entre os instantes inicial e final da execução de uma sequência*

de ações.

Em relação à representação do tempo, é possível estabelecer duas classes distintas: qualitativa (relação relativa) e quantitativa (relação métrica). A primeira apresenta as restrições temporais entre duas variáveis (O_i e O_j) usando uma disjunção

$$(O_j\ r_1\ O_j) \vee ... \vee (O_i\ r_k\ O_j)$$

onde O_i e O_j devem ser pontos ou intervalos de tempo e r_i é a relação existente entre as duas variáveis. Existem três categorias de relação entre ações:

1. *Relação ponto-ponto:* conjunto de relações que podem ocorrer entre um par de pontos. Introduzido por Vilain e Kautz [31] é denotada usando três tipos de restrições, conforme apresentado pela figura 3.1 a).

2. *Relação ponto-intervalo:* conjunto de relações que podem ocorrer entre um ponto e um intervalo ou entre um intervalo e um ponto. É uma extensão da lógica de Allen que foi criada por Zaidi [32]. Possui cinco possíveis relações conforme apresentado pela figura 3.1 b).

3. *Relação intervalo-intervalo:* conjunto de relações que podem ocorrer entre um par de intervalos. Proposta por Allen [29] como base para uma álgebra de cálculos sobre intervalos, leva em consideração 7 possíveis relações (e suas inversas), apresentadas na figura 3.1 c. Estas relações definem a exclusão mútua de relacionamentos que podem existir entre dois intervalos, além disso, usando estas relações é possível descrever como as ações afetam o mundo [21]. Dentre as relações apresentadas na figura 3.1 c, apenas aquelas apresentadas mais ao topo não são consideradas complexas (<, >, m e mi), pois implicam uma execução de forma sequencial, enquanto as restantes (o, oi, d, di, =, s, si, f, e fi) podem ser executadas concorrentemente.

Contudo, segundo Dechter et al. [33] e Meiri [34], a álgebra qualitativa não oferece mecanismos convenientes para representar informações métricas. Já na classe quantitativa, a informação temporal deve estar explicitamente disponível de forma numérica e portanto,

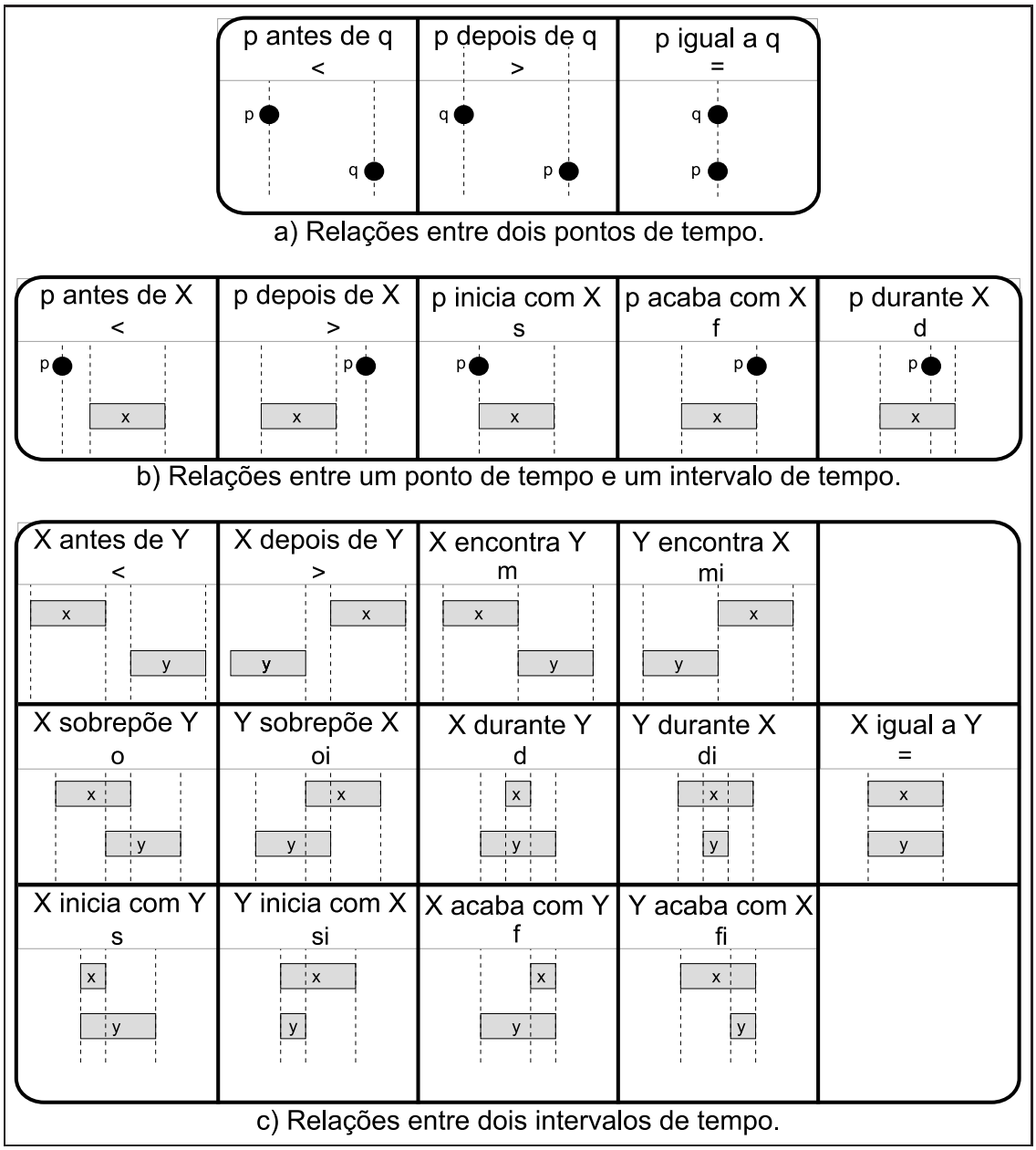

Figura 3.1: Possíveis relações ponto-ponto (a), ponto-intervalo (b) e intervalo-intervalo (c) com seus respectivos símbolos.

segundo Vila [10], pode ser facilmente computada. Como o raciocínio sobre as restrições temporais pode ser visto como um problema de satisfação de restrições [33, 35] (CSP, do inglês *Constraint Satisfaction Problem*) então Dechter et al. [33] desenvolveram um trabalho baseado nesta idéia, chamado de problema de satisfação de restrições temporais (TCSP, do inglês *Temporal Constraint Satisfaction Problems*). Dois tipos de restrições podem ser representadas:

- *Restrição Unária:* a restrição limita a posição de um ponto de tempo (P_i) para

um dado conjunto de intervalos $([a_1, b_1], ..., [a_n, b_n])$. Ex.: $P_1 \mid P_1 \in \{(3, 6)\}$ ou $3 \leq P_1 \leq 6$.

- *Restrição Binária:* restringe os valores admissíveis para a distância $P_j - P_i$ entre pontos para um dado conjunto de intervalos $([a_1, b_1], ..., [a_n, b_n])$. Ex.: $P_2 - P_1 \in \{(3, 6)\}$ ou $3 \leq P_2 - P_1 \leq 6$.

Além disso, as duas formas, qualitativa e quantitativa, são representadas por um conjunto de disjunções $((R_i) \vee ... \vee (R_n))$, onde R_i é uma restrição. Estas restrições podem ser representadas por uma rede de restrições temporais (CN, do inglês *Constraint Network*), usando um grafo direcionado disjuntivo, onde os nodos são representados por variáveis (instante de tempo ou intervalo) e um arco $i \rightarrow j$ indica uma restrição entre variáveis, sendo rotulado por um conjunto de intervalos (quando a restrição é quantitativa) ou por um conjunto de relações qualitativas. Porém, sem a disjunção, o problema se torna temporalmente simples (STP, do inglês *Simple Temporal Problems*) e com isso pode ser representado por uma rede temporal simples (STN, do inglês *Simple Temporal Networks*), representado usando um grafo de distância (*d-Graph*). O uso de STPs permite respostas rápidas para perguntas temporais [12], pois pode ser resolvido em tempo polinomial.

A figura 3.2 apresenta uma rede CN quantitativa que é transformada em STN e, posteriormente, transformada em *d-graph*, conforme apresentado no parágrafo anterior.

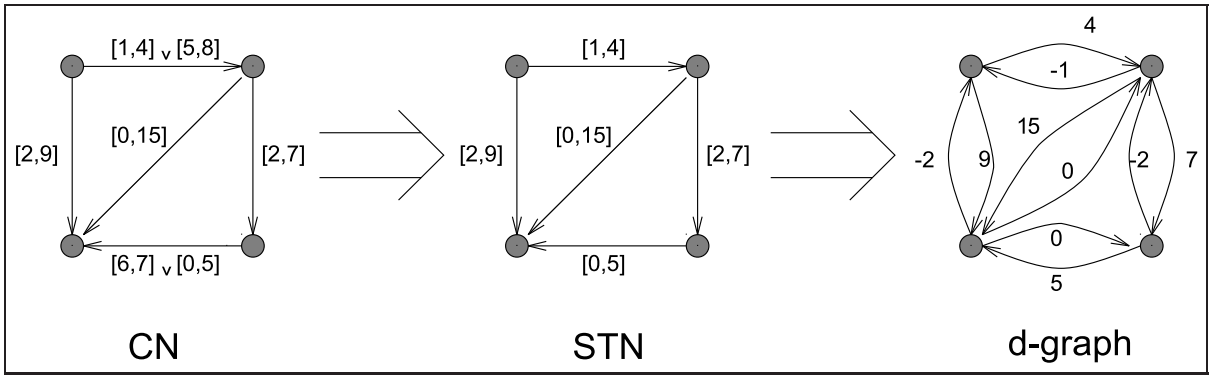

Figura 3.2: Tranformando um CN em um d-graph usado representação quantitativa.

Já a figura 3.3 apresenta um exemplo de rede para a representação qualitativa descrita por Ghallab et al. [28], onde o conjunto de restrições da figura 3.3(c) é representado gra-

ficamente por uma CN qualitativa para instantes (figura 3.3(a)) e para intervalos (figura 3.3(b))

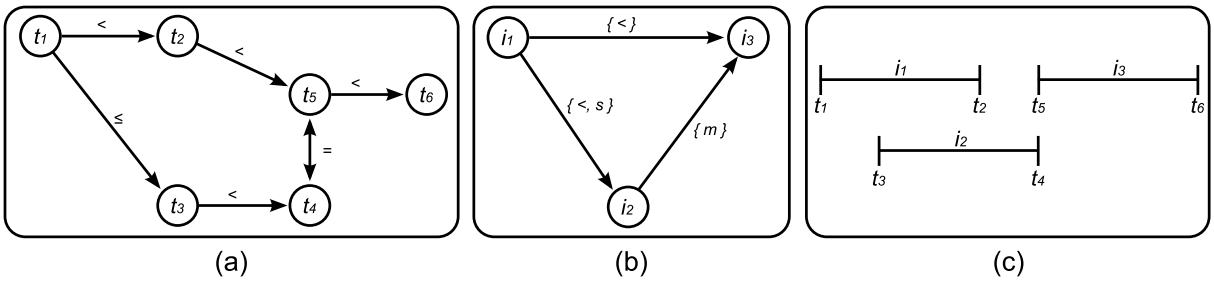

Figura 3.3: Exemplo de CN qualitativa para instantes (a) e para intervalos (b) para um dado conjunto de restrições (c).

3.2 Noções de planejamento temporal

Enquanto o planejamento clássico busca um plano com o menor número de passos, o planejamento temporal busca por um plano que tenha a menor duração, não importando a quantidade de ações executadas, mas sim o menor tempo para a execução da tarefa desejada.

O planejamento temporal possibilita descrever as ações durativas de duas formas: contínua e discreta. A primeira forma é mais complexa [24] e não será discutida neste trabalho. Já na segunda forma, as ações tem uma duração $[\alpha, \beta]$ definida, como já discutido no capítulo anterior (definição 9). Sendo assim, a ação inicia em um instante de tempo t_s e seu final ocorre em t_e, tal que $t_s + \alpha \leq t_e \leq t_s + \alpha + \beta$, o que corresponde a uma relação ponto-ponto (veja a seção anterior).

A representação temporal de uma ação é feita através da descrição temporal de cada um dos predicados que compõem as pré-condições e os efeitos da ação. Entretanto, essas informações são associadas a instantes de tempo onde a validade de um predicado varia de acordo com o tempo, ou seja, os predicados verdadeiros em um dado momento podem ser falsos no momento seguinte.

3.2.1 Concorrência requerida

Quando é necessário que um plano tenha a menor duração possível é necessário que, em muitos casos, as ações sejam executadas concorrentemente (figura 3.4 b). Por exemplo, dois caminhões podem ser descarregados ao mesmo tempo, desde que haja uma máquina de descarregamento independente para cada um. Apesar disso, o descarregamento sequencial (figura 3.4 a) dos mesmos também é um plano válido, porém, a duração consequentemente não é o melhor.

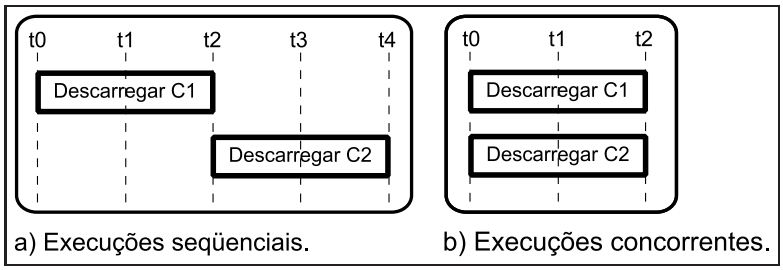

Figura 3.4: Comparação entre a duração de execução de ações sequenciais e concorrentes.

Definição 11 (Ações Concorrentes)**.** *Dado um plano* $\mathtt{PLN} = \{\mathtt{o}_1, \mathtt{o}_2, \mathtt{o}_3, ..., \mathtt{o}_n\}$*, duas ações são concorrentes se, e somente se*

$$\exists \mathtt{o}_i, \mathtt{o}_j \in \mathtt{PLN} \mid \mathtt{t}_s(\mathtt{o}_i) \leq \mathtt{t}_s(\mathtt{o}_j) \leq \mathtt{t}_e(\mathtt{o}_i) \vee \mathtt{t}_s(\mathtt{o}_j) \leq \mathtt{t}_s(\mathtt{o}_i) \leq \mathtt{t}_e(\mathtt{o}_j)$$

Contudo, muitas vezes a execução concorrente de algumas ações é o único caminho para se chegar à uma solução válida de um problema. Sendo assim, quando este tipo de problema ocorre então é dito que o problema de planejamento tem Concorrência Requerida.

Definição 12 (Concorrência Requerida)**.** *Um problema tem concorrência requerida se para todo plano, que é sua solução, existem ações executando concorrentemente, ou seja, não existe nenhum plano sequencial que soluciona o problema [26].*

Um exemplo onde há concorrência requerida é apresentado na figura 3.5. Neste exemplo, tendo como estado inicial G e R falsos, é impossível se chegar ao objetivo (G) sem a execução concorrente das ações. Isto ocorre porque a solução somente pode ser encontrada

através da execução da ação B que, a principio, não tem suas pré-condições satisfeitas. No entanto, a única forma de se obter a solução deste problema é iniciar pela ação A e em seguida iniciar a ação B antes do término de A. Ainda é necessário que o final de execução de B ocorra após o final da execução de A, pois caso contrário a solução G da ação B seria negada novamente no final da execução de A.

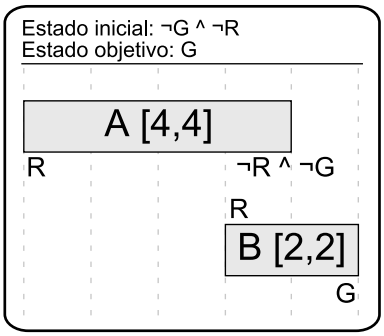

Figura 3.5: Exemplo de problema com concorrência requerida [6].

Um caso de concorrência requerida mais complicado é apresentado na figura 3.6. É possível notar que com apenas a execução de A e B já se chega à solução do problema (G1 e G2), mas com a execução de B é gerado ¬Q no início, o que impede o final da execução de A, que tem que assegurar a condição final (Q). Sendo assim, é necessário uma segunda execução da ação A para se obter a solução.

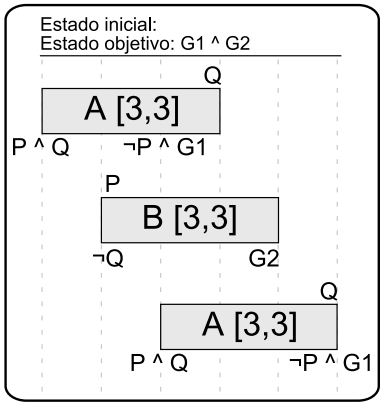

Figura 3.6: Exemplo de problema com concorrência requerida [7].

3.3 Linguagens de ações temporais

Muitas linguagens que modelam ações durativas já foram propostas, dentre elas a linguagem TGP [36], porém a linguagem PDDL é mais amplamente utilizada na área de

planejamento por ser mais flexível.

As linguagens de representação são definidas através das diferentes relações temporais e restrições que suas ações podem representar. No caso da PDDL, devido ao seu grande poder de expressividade, a linguagem pode capturar uma vasta quantidade de relações e restrições temporais [37], tais como àquelas apresentadas pela figura 3.1. Sendo assim, a linguagem PDDL pode ser representada por $L_{s,e}^{s,o,e}$, onde as restrições sobrescritas são associadas às condições e as restrições subscritas são associadas aos efeitos das ações, assim como apresentado na seção 2.3.1. No entanto, PDDL pode derivar várias outras linguagens, conforme a figura 3.7.

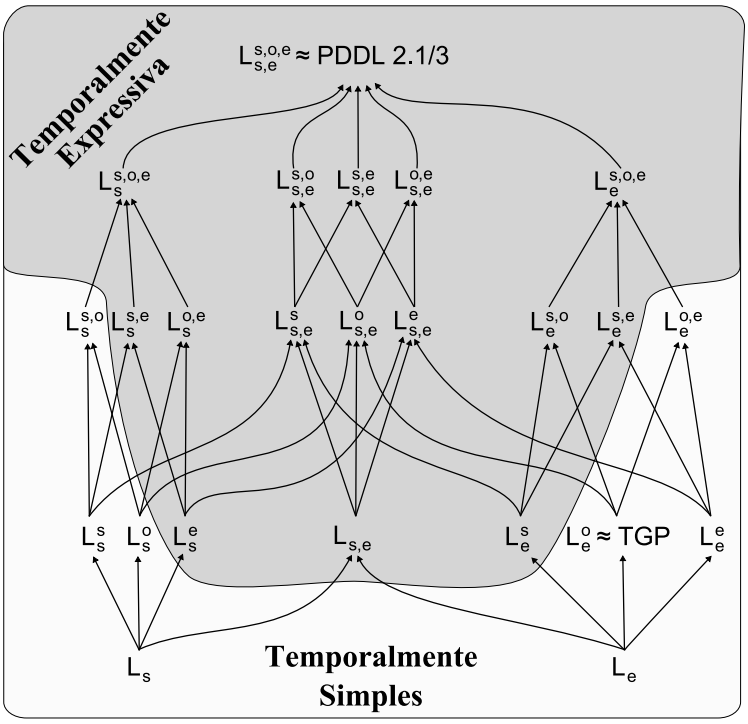

Figura 3.7: Taxonomia das linguagens temporais e sua expressividade [6].

Através do conceito de concorrência requerida é possível dividir as linguagens em dois grupos distintos: linguagens temporalmente simples e linguagens temporalmente expressivas. Esta divisão tem grande importância pois, conforme apresentado por Cushing et al. [6], todos os problemas temporalmente simples podem ser resolvido por planejadores clássicos,ou seja, as informações temporais não tem influência sobre o plano que resolve o problema.

Definição 13 (Linguagens Temporalmente Simples/Expressivas)**.** *Uma linguagens L é*

temporalmente expressiva se ela pode descrever problemas com concorrência requerida, caso contrário, L é temporalmente simples.

Um outro critério mais simples, chamado de lacuna temporal, pode ser usado para dividir as linguagens nos dois grupos citados. Quando uma linguagem possui lacuna temporal, então ela é classificada como temporalmente expressiva e, caso contrário, temporalmente simples.

Definição 14 (Lacuna Temporal)**.** *Uma linguagem ou um domínio possui lacuna temporal se, potencialmente, suas ações têm lacuna temporal, ou seja, existe pelo menos uma ação onde [26]:*

1. *Há uma condição ou efeito sobre um predicado x em* AT START *e;*

2. *Há uma condição ou efeito sobre um predicado y , possivelmente diferente, em* AT END.

Por exemplo, uma ação que não tem lacuna temporal e tem *at start* como condição ou efeito não pode ter efeito ou condição no final (*at end*). A quantidade de possibilidades de lacunas temporais de uma linguagem depende de sua expressividade. Sendo assim, pode haver linguagens onde não há lacuna temporal, com é o caso da L_s^s, pois todas as restrições são aplicadas em um mesmo instante de tempo. Considerando todas as restrições temporais apresentadas pela PDDL, então qualquer linguagem que deriva das linguagens L_e^s, L_s^e ou L_{se} possui lacuna temporal, ou seja, uma ação tem lacuna temporal sempre que houver uma lacuna entre: uma pré-condição e um efeito no fim (figura 3.8(a)), entre uma pós-condição e um efeito no início (figura 3.8(b)) ou entre dois efeitos (figura 3.8(c)).

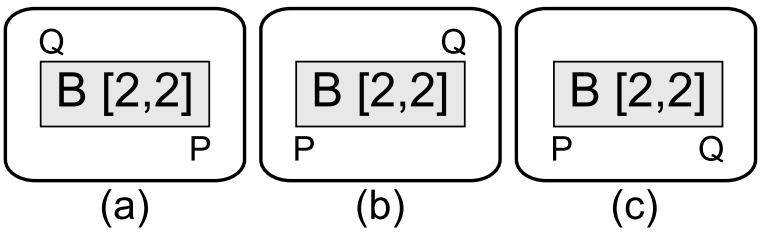

Figura 3.8: Lacunas temporais que podem ocorrer em uma ação.

Apesar de tudo, mesmo que um domínio contenha ações com lacuna temporal o mesmo pode não ser temporalmente expressivo, ou seja, não requer concorrência. Este fato pode ocorrer por dois motivos:

1. porque o domínio pode conter dependências sobre as ações, o que faz com que o plano gerado sempre seja sequencial.

2. porque o problema não possui concorrência requerida. Ex.: encontrar planos que possuem uma única ação.

3.4 Planejamento temporal em PDDL

Para o mesmo problema de planejamento apresentado na seção 2.4 (página 12), agora são adicionadas restrições temporais que devem ser obedecidas para que haja a execução de uma ação, conforme apresentado pela listagem 3.1. Para isto, é necessário que haja três alterações básicas em relação à descrição clássica. A primeira mudança é a adição de *:durative-actions* nos requisitos da linguagem. Na segunda são trocados ":action" por ":durative-action", "precondition" por "condition" e adicionado o rótulo ":duration", onde será especificado a duração. Por último, a adição das restrições temporais a cada predicado que compõem as condições e efeitos das ações.

```
(define (domain logistics_temp)
  (:requirements :strips :typing :durative-actions)
  (:types objeto
          caminhao pacote - objeto
          local )
  (:constants L1 L2 - local)
  (:predicates (em ?o - objeto ?l - local)
               (vazio ?c - caminhao) )

  (:durative-action carregar
     :parameters (?c - caminhao ?p - pacote ?l - local)
     :duration (= ?duration 5)
     :condition (and (over all (em ?c ?l))
                     (at start (em ?c ?l))
                     (at start (em ?p ?l))
                     (at start (vazio ?c)) )
     :effect (and (at start (not (em ?p ?l)))
                  (at end (not (vazio ?c))) )
  )
  (:durative-action descarregar
     :parameters (?c - caminhao ?p - pacote ?l - local)
```

```
        :duration (= ?duration 3)
        :condition (and (at start (not (vazio ?c)))
                        (at start (em ?c ?l))
                        (over all (em ?c ?l)) )
        :effect (and (at end (em ?p ?l))
                     (at start (vazio ?c)) )
    )
    (:durative-action mover
        :parameters (?c - caminhao ?origem ?dest - local)
        :duration (= ?duration 30)
        :condition (and (at start (em ?c ?origem)) )
        :effect (and (at start (not (em ?c ?origem)))
                     (at end (em ?c ?dest)) )
    )
)
```

Listagem 3.1: Uma versão do domínio temporal para *logistics*.

Para exemplificar uma ação durativa, novamente será usada a ação "carregar". Com duração igual a 5, esta ação somente poderá ser executada se no início o caminhão estiver vazio e o pacote e o caminhão estiverem no mesmo lugar no instante inicial e durante todo o processo de carregamento. Como não foi especificado que o caminhão deverá permanecer no local até o final do carregamento, então isto quer dizer que o caminhão pode iniciar a movimentação imediatamente com o fim da ação carregar e, caso contrário, haveria uma lacuna de tempo entre as duas ações.

No resultado para o problema da listagem 2.2, as ações foram executadas uma após a outra, como no planejamento clássico, porém cada uma iniciando em um determinado tempo t e tendo como duração o tempo total da execução da tarefa, conforme listagem 3.2. O resultado é dado pela seguinte sintaxe:

<tempo de início> : (<ação executada>) [<tempo de execução da ação>]

```
0.00: (carregar C1 P1 L1) [5]
5.00: (mover C1 L1 L2) [30]
35.01:(descarregar C1 P1 L2) [3]
38.01:(mover C1 L2 L1) [30]

Duração: 68.01
```

Listagem 3.2: Solução de um problema temporal.

No caso citado, o resultado não teve ações executando concorrentemente, mas isto é possível com a simples adição de mais um caminhão (C2) e adicionar ao objetivo a

condição de que o pacote P2 deva estar em L2. O resultado para este caso é apresentado na listagem 3.3. No entanto é possível notar que a duração não se alterou com relação ao do problema anterior, isto porque os dois caminhões executam as mesmas tarefas ao mesmo tempo.

```
0.00: (carregar C2 P2 L1) [5]
0.00: (carregar C1 P1 L1) [5]
5.00: (mover C2 L1 L2) [30]
5.00: (mover C1 L1 L2) [30]
35.01:(descarregar C2 P2 L2) [3]
35.01:(descarregar C1 P1 L2) [3]
38.01:(mover C1 L2 L1) [30]

Duração: 68.01
```

Listagem 3.3: Solução de um problema temporal com concorrência.

3.5 Considerações

O objetivo deste capítulo foi apresentar como as restrições temporais são aplicadas em planejamento temporal, assim como apresentar os possíveis problemas que a inclusão de parâmetros temporais acarretam na busca por um plano. Além disso, dependendo do poder de expressividade das ações temporais, encontrar um plano para um problema em planejamento temporal pode ser computacionalmente mais complexo do que em planejamento clássico [16].

CAPÍTULO 4

REDES DE PETRI

A teoria sobre Redes de Petri surgiu em 1962 a partir da tese de doutorado desenvolvida por Carl Adam Petri [38]. Entre 1968 e 1976 um grupo de pesquisadores do Instituto de Tecnologia de Massachussetts (MIT), nos Estados Unidos, lançou as bases do que hoje é conhecido como Redes de Petri.

As Redes de Petri, no entanto, se constituem em uma ferramenta gráfica e matemática muito útil para a especificação, modelagem, análise formal, simulação e controle de sistemas a eventos discretos [17].

Neste capítulo são apresentados os principais conceitos sobre redes de Petri (RdP), tanto na notação gráfica quanto na notação matemática. Além disso, será abordado as redes de Petri com o uso de tempo, que possibilita representar e analisar problemas ligados à diversas atividades onde a avaliação do desempenho temporal é um critério de fundamental importância.

4.1 A rede

Na sua representação gráfica, a RdP é composta por um grafo direcionado bipartido constituído pelos seguintes elementos: lugares, transições e arcos.

Definição 15 (Rede de Petri)**.** *Uma Rede de Petri é uma tripla,*

$$\mathtt{RP} = (\mathtt{P}, \mathtt{T}, \mathtt{F})$$

onde $\mathtt{P}$ *é um conjunto finito não-vazio de lugares,* $\mathtt{T}$ *é um conjunto finito não-vazio de transições e* $\mathtt{F}$ *é uma dupla* $\langle \mathtt{Pre}, \mathtt{Pos} \rangle$ *que representa a relação de fluxo dos arcos, onde* $\mathtt{Pre}$ *é uma função de entrada tal que* $\mathtt{Pre} \mid (\mathtt{P} \times \mathtt{T}) \rightarrow \mathbb{N}$ *e* $\mathtt{Pos}$ *é uma função de saída tal que* $\mathtt{Pos} \mid (\mathtt{P} \times \mathtt{T}) \rightarrow \mathbb{N}$*.*

Na rede, os lugares ($\mathtt{P}$) são representados por círculos, que denotam algo passivo tal como uma condição ou um estado local de um componente, enquanto as transições ($\mathtt{T}$) são representadas por retângulos, que denotam algo ativo tal como uma ação, um evento ou a execução de uma instrução. Já os arcos ($\mathtt{F}$) são representados por setas denotando causas ou efeitos. Além disso, existem arcos que podem ligar um lugar $\mathtt{p_i}$ à uma transição $\mathtt{t_j}$ se, e somente se, $\mathtt{Pre}(\mathtt{p_i}, \mathtt{t_j}) > 0$, caracterizando uma pré-condição, e/ou ligar uma transição $\mathtt{t_j}$ à um lugar $\mathtt{p_i}$ se, e somente se, $\mathtt{Pos}(\mathtt{p_i}, \mathtt{t_j}) > 0$, caracterizando uma pós-condições. Os arcos também podem conter um peso $\mathtt{w}$ associado, ou seja, $\mathtt{Pre}(\mathtt{p_i}, \mathtt{t_j}) = \mathtt{w}$ e $\mathtt{Pos}(\mathtt{p_i}, \mathtt{t_j}) = \mathtt{w}$, tal que $\mathtt{w} \in \mathbb{N}$.

Definição 16 (Conjuntos Predecessores e Sucessores)**.** *Seja uma rede* $\mathtt{RP} = (\mathtt{P}, \mathtt{T}, \mathtt{F})$*, então para um conjunto predecessor e sucessor de um lugar* $\mathtt{p} \in \mathtt{P}$ *e de uma transição* $\mathtt{t} \in \mathtt{T}$ *define-se:*

i) $\bullet\mathtt{p} = \{\mathtt{t} \mid \mathtt{Pos}(\mathtt{p}, \mathtt{t}) > 0\}$ = *conjunto das transições de entrada de* $\mathtt{p}$.

ii) $\mathtt{p}\bullet = \{\mathtt{t} \mid \mathtt{Pre}(\mathtt{p}, \mathtt{t}) > 0\}$ = *conjunto das transições de saída de* $\mathtt{p}$.

iii) $\bullet\mathtt{t} = \{\mathtt{p} \mid \mathtt{Pre}(\mathtt{p}, \mathtt{t}) > 0\}$ = *conjunto dos lugares de entrada de* $\mathtt{t}$.

iv) $\mathtt{t}\bullet = \{\mathtt{p} \mid \mathtt{Pos}(\mathtt{p}, \mathtt{t}) > 0\}$ = *conjunto dos lugares de saída de* $\mathtt{t}$.

A figura 4.1 (a) exemplifica uma rede de Petri. O quadro (b) da mesma figura apresenta a estrutura da rede, conforme a definição 15. Já no quando (c) é apresentado um exemplo do uso da definição 16 para os conjuntos de predecessores e sucessores de lugar e de transição.

Definição 17 (Rede Pura)**.** *Uma rede de Petri* $\mathtt{RP} = (\mathtt{P}, \mathtt{T}, \mathtt{F})$ *é pura se, e somente se,*

$$\forall \mathtt{t} \in \mathtt{T},\ \bullet\mathtt{t} \cap \mathtt{t}\bullet = \emptyset$$

Definição 18 (Vizinhança de uma Transição)**.** *Dada uma rede de Petri* $\mathtt{RP} = (\mathtt{P}, \mathtt{T}, \mathtt{F})$*, a vizinhança de uma transição* $\mathtt{t} \in \mathtt{T}$ *é definida como*

$$\mathtt{v}(\mathtt{t}) = \bullet\mathtt{t} \cup \mathtt{t}\bullet$$

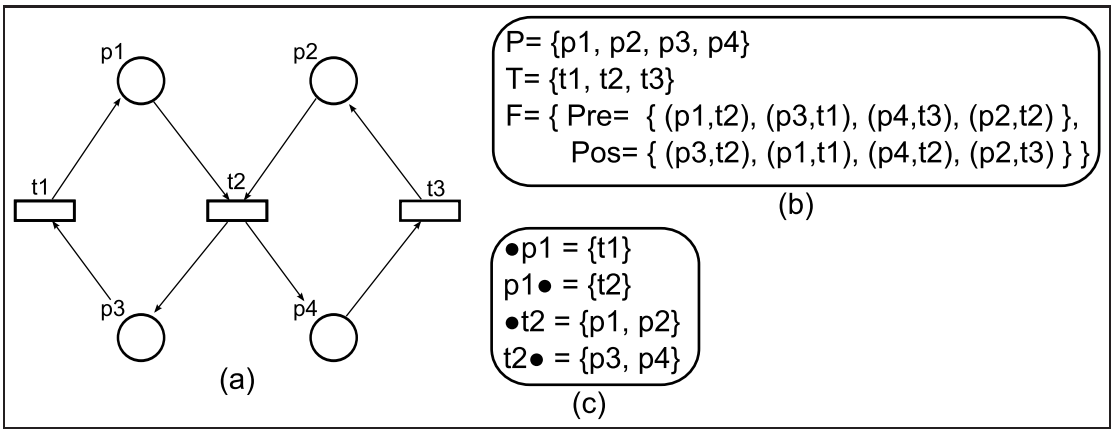

Figura 4.1: Exemplo de uma rede de Petri.

Definição 19 (Transições Independentes)**.** *Duas transições* $\mathtt{t}_1 \neq \mathtt{t}_2 \in \mathtt{T}$ *são independentes (*$\mathtt{t}_1 \# \mathtt{t}_2$*) se, e somente se,*

$$\mathtt{v}(\mathtt{t}_1) \cap \mathtt{v}(\mathtt{t}_2) = \emptyset$$

onde $\mathtt{v}(\mathtt{t}_\mathtt{i})$ *são os lugares vizinhos de* $\mathtt{t}_\mathtt{i}$*.*

4.2 Marcação de uma rede de Petri

Uma Rede de Petri é caracterizada por possuir um componente dinâmico associado a sua estrutura e por permitir uma análise matemática sobre seus modelos estrutural e comportamental. O comportamento dinâmico é introduzido ao modelo através do conceito de marcação e da evolução do sistema entre marcações. Uma marcação consiste na introdução de marcas (em inglês, *tokens*) nos lugares. Estas marcas são representadas graficamente por pequenos pontos dentro de um lugar.

Definição 20 (Marcação)**.** *Uma marcação* $\mathtt{M}$ *de uma rede de Petri é uma função* $\mathtt{M} \,|\mathtt{P} \rightarrow \mathbb{N}$*, que representa o número de marcas contidas em cada lugar* $\mathtt{p} \in \mathtt{P}$*. A marcação* $\mathtt{M}$ *é representada por um vetor coluna cuja dimensão é o número de lugares da rede.*

$$\mathtt{M} = [\mathtt{M}(\mathtt{p}_1), \mathtt{M}(\mathtt{p}_2), \ldots, \mathtt{M}(\mathtt{p}_\mathtt{n})]^\mathtt{T}$$

Definição 21 (Rede de Petri Marcada)**.** *Uma rede de Petri marcada é uma dupla*

$$\mathtt{RPM} = (\mathtt{RP}, \mathtt{M_0})$$

onde $\mathtt{RP}$ *é uma rede de Petri e* $\mathtt{M_0} \mid \mathtt{P} \to \mathbb{N}$ *é a marcação inicial da rede.*

Desta forma é possível dizer que um lugar contém n marcas, tal que $n \in \mathbb{N}$. Uma marcação, portanto, permite representar o estado da rede em um determinado momento.

As seções subsequentes descrevem o funcionamento das redes de Petri de maneira mais detalhada e formal, assumindo que todas as redes são puras.

4.3 Comportamento dinâmico

A dinâmica de uma rede de Petri se dá através da movimentação das marcas pelos lugares da rede, gerando um novo estado da rede. Toda e qualquer movimentação das marcas apenas pode ocorrer se uma transição estiver habilitada, ou seja, se, e somente se, o número de marcas contidas em cada lugar pertencente ao conjunto de entrada da transição for maior ou igual ao peso dos arcos respectivos.

Definição 22 (Transição Habilitada)**.** *Uma transição* $\mathtt{t} \in \mathtt{T}$ *está habilitada para a marcação* $\mathtt{M}$*, denotado por* $\mathtt{M}[\mathtt{t}>$*, de uma rede de Petri* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M_0})$*, isto é, ela está pronta para disparar, se, e somente se,*

$$\forall \mathtt{p} \in {\bullet}\mathtt{t}, \mathtt{M}(\mathtt{p}) \geq \mathtt{Pre}(\mathtt{p}, \mathtt{t})$$

onde, $\mathtt{M}(\mathtt{p})$ *é o número de marcas do lugar* $\mathtt{p} \in \mathtt{P}$ *em uma marcação* $\mathtt{M}$ *e* $\mathtt{Pre}(\mathtt{p}, \mathtt{t})$ *é o peso do arco de lugar-transição.*

O disparo de uma transição habilitada fará com que n marcas sejam retiradas de um lugar, tal que n é o peso do arco que incide sobre a transição. Este procedimento ocorre para todas os lugares que são pré-condição da transição. Contudo, a transição gera marcas nos lugares que são sua pós-condição, onde a quantidade de marcas de cada lugar também é especificada pelo peso de cada arco.

Definição 23 (Disparo de uma Transição)**.** *Se* $\mathtt{t} \in \mathtt{T}$ *é uma transição habilitada por uma marcação* $\mathtt{M}$*, então o disparo de* $\mathtt{t}$ *retira* $\mathtt{Pre}(\mathtt{p},\mathtt{t})$ *marcas de* $\mathtt{p} \in {}^{\bullet}\mathtt{t}$ *e adiciona* $\mathtt{Pos}(\mathtt{p},\mathtt{t})$ *marcas em* $\mathtt{p} \in \mathtt{t}^{\bullet}$*, levando a rede a uma nova marcação* $\mathtt{M}'$*, denotada por* $\mathtt{M}[\mathtt{t} > \mathtt{M}'$*. A evolução da rede é descrita pela equação:*

$$\mathtt{M}' = \mathtt{M} - \mathtt{Pre}(\mathtt{p},\mathtt{t}) + \mathtt{Pos}(\mathtt{p},\mathtt{t})$$

Para exemplificar uma rede de Petri foram adicionadas marcas aos lugares e pesos aos arcos da figura 4.1, como apresentado pela figura 4.2. Esta figura exemplifica uma rede de Petri, onde é enfatizado o disparo da transição `t2`. Tomando com base a figura 4.2 (a) é possível notar que a transição `t2` está habilitada, pois o número de marcas contidas nos lugares `p1` e `p2` é maior ou igual ao peso dos seus arcos. Sendo assim, a transição `t2`, quando disparada, consome duas marcas do lugar `p1` e uma marca de `p2`, e então coloca uma marca em `p3` e outra em `p4`. A marcação da rede após o disparo da transição `t2` é apresentada pela figura 4.2 (b).

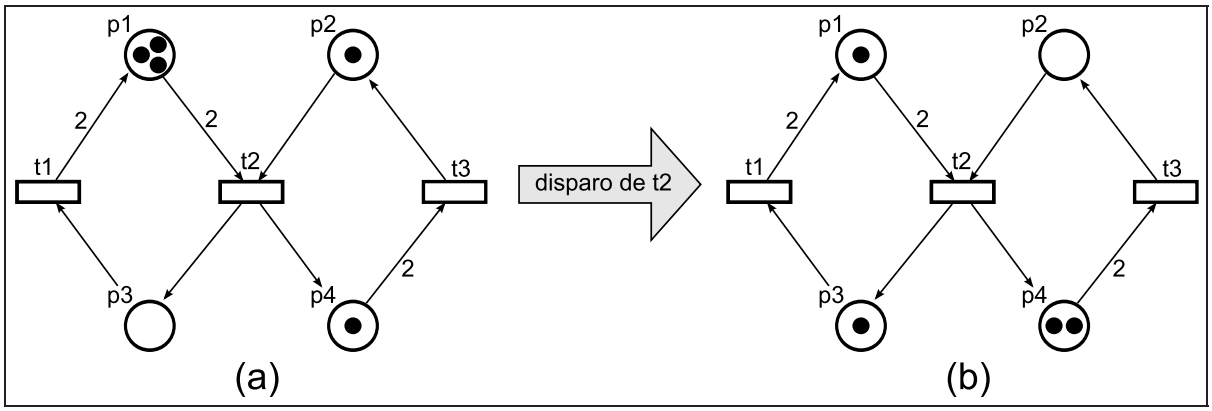

Figura 4.2: Exemplo de uma rede de Petri marcada.

4.4 Algumas situações fundamentais

Nesta seção são apresentadas algumas situações que podem ocorrer em uma rede, que envolvem noções de dependência e/ou independência entre lugares e transições. O conteúdo aqui apresentado tem como base os trabalhos de Balbo et al. [39], Buchholz [40] e Murata [17].

A primeira situação é chamada de sequência de disparos. É a mais básica de todas e serve para descrever quais disparos de transições são necessários para que uma outra transição específica possa disparar. Esta situação é ilustrada na figura 4.3, onde o disparo da transição $\mathtt{t_i}$ é necessária para o disparo de $\mathtt{t_j}$, ou seja, $\mathtt{t_j}$ só pode ocorrer após $\mathtt{t_i}$.

Definição 24 (Sequência de Disparos)**.** *Dada uma rede de Petri* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M_0})$*, duas transições* $\mathtt{t_i}, \mathtt{t_j} \in \mathtt{T}$ *estão em sequência em uma marcação* $\mathtt{M}$ *se, e somente se,*

$$\mathtt{M}[\mathtt{t_i} > \wedge \neg(\mathtt{M}[\mathtt{t_j} >) \wedge \mathtt{M'}[\mathtt{t_j} >$$

onde $\mathtt{M}[\mathtt{t_i} > \mathtt{M'}$*. O disparo sequencial de transições* $\mathtt{s} = \{\mathtt{t_1}, ..., \mathtt{t_n}\}$*, onde* $\mathtt{t_i} \in \mathtt{T}$*, que levam a rede de uma marcação* $\mathtt{M}$ *à uma marcação* $\mathtt{M'}$ *é dado por:* $\mathtt{M}[\mathtt{s} > \mathtt{M'}$*.*

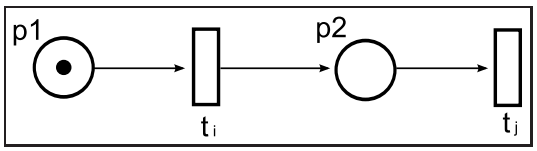

Figura 4.3: Disparo sequencial de $\mathtt{t_i}$ e $\mathtt{t_j}$.

Outra situação comumente utilizada é a concorrência, que possibilita que as transições sejam disparadas concorrentemente sem que o disparo de uma inviabilize o disparo de outra. A concorrência pode se apresentar de duas formas: estrutural e efetiva, conforme as definições 25 e 26 respectivamente. Na primeira forma, duas transições são concorrentes se não possuem nenhum lugar de entrada em comum, enquanto na segunda forma, as transições devem possuir concorrência estrutural entre si e também estar habilitadas para uma dada marcação $\mathtt{M}$. A figura 4.4 ilustra estas duas situações.

Definição 25 (Concorrência Estrutural)**.** *Dada uma rede de Petri* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M_0})$*, um conjunto de transições* $\mathtt{A} \subseteq \mathtt{T}$ *é estruturalmente concorrentemente se, e somente se,*

$$\forall \mathtt{t_i}, \mathtt{t_j} \in \mathtt{A}, \bullet\mathtt{t_i} \cap \bullet\mathtt{t_j} = \emptyset$$

ou seja, não existe lugar da rede que pertença simultaneamente aos pré-conjuntos de $\mathtt{t_i}$ *e de* $\mathtt{t_j}$*.*

Definição 26 (Concorrência Efetiva)**.** *Dada uma rede de Petri* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M_0})$*, um conjunto de transições* $\mathtt{A} \subseteq \mathtt{T}$ *está habilitada concorrentemente numa marcação* $\mathtt{M}$ *se, e somente se, cada uma das transições está habilitada individualmente e cada uma é independente de todas as outras.*

$$\forall \mathtt{t} \in \mathtt{A}, \mathtt{M}[\mathtt{t} > \wedge \forall \mathtt{t_i}, \mathtt{t_j} \in \mathtt{A} \mid \mathtt{t_i} \# \mathtt{t_j}$$

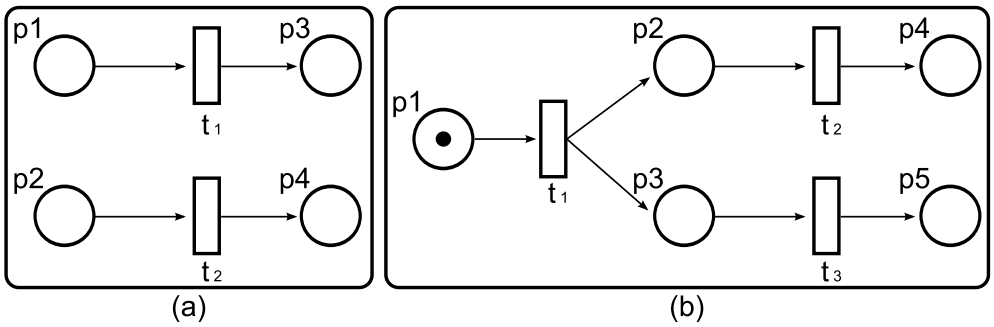

Figura 4.4: Concorrência estrutural (a) e concorrência efetiva (b).

Note que na figura 4.4 (a) as duas transições, `t1` e `t2`, são independentes, o que caracteriza uma concorrência estrutural. Já na figura 4.4 (b) há concorrência efetiva porque as transições `t2` e `t3`, além de terem concorrência estrutural, também estão habilitadas para a marcação $\mathtt{M}'$, que se dá através do disparo de `t1` (denotado por $\mathtt{M}[\mathtt{t1} > \mathtt{M}'$).

Os conflitos, assim como na concorrência, também se apresentam na forma estrutural e efetiva. No conflito estrutural duas transições estão em conflito se possuem pelo menos um lugar de entrada em comum, como é o caso do lugar `p2` da figura 4.5 (a), o que torna `t1` e `t2` em conflito. Já no conflito efetivo, duas transições `t1` e `t2` podem disparar individualmente em uma marcação `M` mas elas não podem ambas disparar nesta mesma marcação. Este tipo de conflito ocorre porque as transições não são independentes ($\neg(\mathtt{t1}\#\mathtt{t2})$) pois compartilham suas pré ou pós-condições, conforme ilustrado pela figura 4.5 (b) e (c). É importante salientar que o conflito efetivo compartilhando pós-condições somente ocorre em redes k-limitadas, ou seja, redes onde a capacidade de marcas suportadas por cada lugar é limitada (ver definição 37), que na figura 4.5 (c) ocorre porque k(p3=1).

Definição 27 (Conflito Estrutural)**.** *Dada uma rede de Petri* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M_0})$*, um conjunto de transições* $\mathtt{A} \subseteq \mathtt{T}$ *está em conflito estrutural se, e somente se,*

$$\forall \mathtt{t_i}, \mathtt{t_j} \in \mathtt{A}, \bullet\mathtt{t_i} \cap \bullet\mathtt{t_j} \neq \emptyset$$

ou seja, existe ao menos um lugar da rede que esteja simultaneamente no pré-conjunto de $\mathtt{t_i}$ *e de* $\mathtt{t_j}$*.*

Definição 28 (Conflito Efetivo)**.** *Dada uma rede de Petri* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M_0})$*, um conjunto de transições* $\mathtt{A} \subseteq \mathtt{T}$ *está em conflito efetivo em uma marcação* $\mathtt{M}$ *se, e somente se,*

$$\forall \mathtt{t_i}, \mathtt{t_j} \in \mathtt{A}, \mathtt{M}[\mathtt{t_i} > \wedge\, \mathtt{M}[\mathtt{t_j} > \wedge\, \neg(\mathtt{M}[\mathtt{s}>) \mid \mathtt{s} = \{\mathtt{t_i}, \mathtt{t_j}\}$$

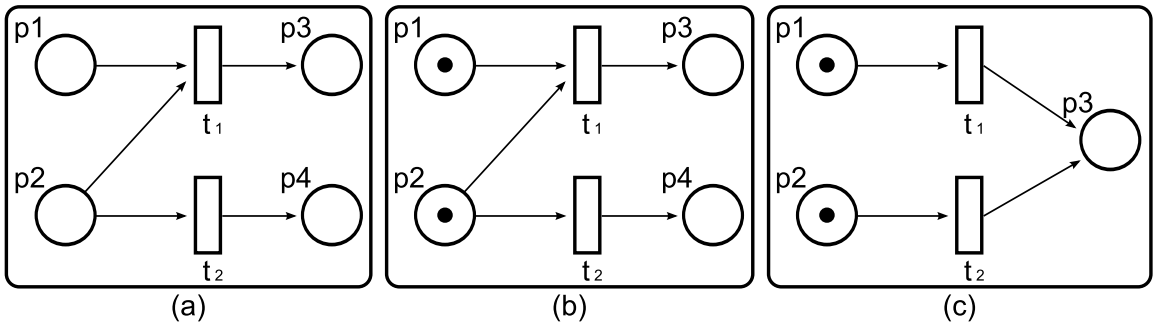

Figura 4.5: Conflito estrutural (a), conflito efetivo compartilhando as pré-condições(b) e conflito efetivo compartilhando as pós-condições (c), no caso de uma RdP 1-limitada.

Quando são misturados os conceitos de concorrência e de conflito então se tem uma nova situação que é chamada de confusão (Definição 29). Em princípio as ações concorrentes são independentes, mas em alguns casos estas ações podem interferir umas nas outras.

Definição 29 (Confusão)**.** *Seja o conjunto de conflitos de* $\mathtt{t}$ *sobre uma marcação* $\mathtt{M}$*, denotado por* $\mathtt{cfl}(\mathtt{t}, \mathtt{M})$*. Então duas transições concorrentes* $\mathtt{t_1}$ *e* $\mathtt{t_2} \in \mathtt{T}$ *estão em confusão para uma marcação* $\mathtt{M}$ *se:*

$$\mathtt{cfl}(\mathtt{t_1}, \mathtt{M}) \neq \mathtt{cfl}(\mathtt{t_1}, \mathtt{M'})$$

onde $\mathtt{M}[\mathtt{t_2} > \mathtt{M'}$*.*

Tomando a figura 4.6 (a) como exemplo para o uso da Definição 29 tem-se:

$$\mathtt{M} = \{\mathtt{p1}, \mathtt{p2}\}$$

$$\mathtt{M}[\mathtt{t_3} > \mathtt{M}' \Rightarrow \mathtt{M}' = \{\mathtt{p1}, \mathtt{p5}\}$$

$$\mathtt{cfl}(\mathtt{t_1}, \mathtt{M}) = \{\mathtt{t_2}\} \neq \emptyset = \mathtt{cfl}(\mathtt{t_1}, \mathtt{M}')$$

o que significa que $\mathtt{t_1}$ e $\mathtt{t_3}$ estão em confusão para a marcação $\mathtt{M}$, pois $\{\mathtt{t_2}\} \neq \emptyset$. Na figura 4.6 (b), onde $\mathtt{t_1}$ e $\mathtt{t_2}$ são concorrentes, se disparada a transição $\mathtt{t_1}$ por primeiro, então não haverá conflito e $\mathtt{t_2}$ poderá ser disparada em seguida, mas se disparada a transição $\mathtt{t_2}$ por primeiro então haverá conflito entre $\mathtt{t_1}$ e $\mathtt{t_3}$.

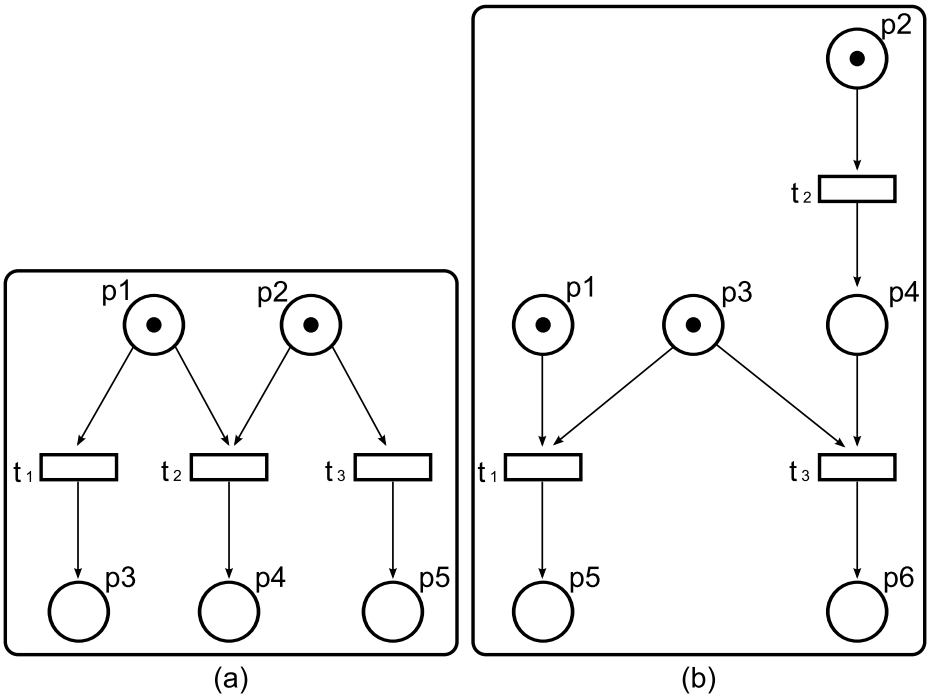

Figura 4.6: Confusão simétrica (a) e confusão assimétrica (b).

Outra situação importante em uma redes de Petri é o contato. Ele ocorre em redes k-limitadas (ver definição 37) sempre que uma transição está habilitada e, com o seu suposto disparo, a capacidade de marcas do lugar seja excedida. Sempre que uma transição possuir contato, então ela será desabilitada. A figura 4.7 exemplifica um caso onde há contato. É possível notar que a transição $\mathtt{t_1}$ está habilitada, porém o lugar $\mathtt{p4}$, por estar marcado, ocasiona contato, impedindo assim que $\mathtt{t_1}$ seja disparada.

Definição 30 (Contato)**.** *Uma rede de Petri* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M_0})$ *está em contato se, e somente se,*

$$\mathtt{M}[\mathtt{t} > \wedge\, \exists\, \mathtt{p} \in \mathtt{t}\bullet \mid \mathtt{M}(\mathtt{p}) + \mathtt{Pos}(\mathtt{p}, \mathtt{t}) > \mathtt{k}(\mathtt{p})$$

onde $\mathtt{k}(\mathtt{p})$ *é a capacidade máxima de marcas de um lugar* $\mathtt{p} \in \mathtt{P}$*.*

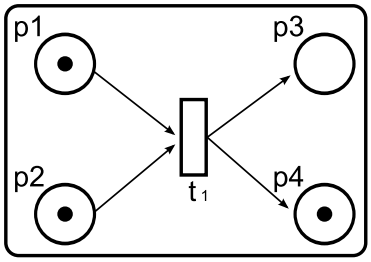

Figura 4.7: Situação de contato para $\mathtt{k}(\mathtt{p4}) = \mathtt{1}$.

Sabendo que os contatos tornam os cálculos da dinâmica da rede mais complexos, então foi introduzido o conceito de lugar complementar que visa eliminar a situação de contato. No entanto, qualquer rede de Petri pode ser transformada em uma rede livre de contato com o mesmo comportamento, desde que todos os seus lugares tenham sua capacidade de marcas limitada. A figura 4.8(a) ilustra uma rede onde há presença de contato. O contato ocorre porque t1 pode disparar e p2 já está marcado, sabendo que $\mathtt{k}(\mathtt{p2}) = \mathtt{1}$. Já a figura 4.8(b) apresenta a mesma rede citada anteriormente, porém, totalmente livre de contatos. Na próxima seção será visto o porquê que as duas redes tem o mesmo comportamento, ou seja, são redes equivalentes (ver Definição 35).

Definição 31 (Lugar Complementar)**.** *Dada uma rede de Petri* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M_0})$*, dois lugares* $\mathtt{p}, \widehat{\mathtt{p}} \in \mathtt{P}$ *são complementares entre si, se, e somente se, para toda marcação M seu comportamento dinâmico obedecer a*

$$\forall \mathtt{p} \in \mathtt{P}, \mathtt{M}(\mathtt{p}) + \mathtt{M}(\widehat{\mathtt{p}}) = \mathtt{k}(\mathtt{p}) = \mathtt{k}(\widehat{\mathtt{p}})$$

e se sua estrutura também obedecer a

$$\forall \mathtt{t} \in \mathtt{T}, (\mathtt{p} \subseteq \bullet\mathtt{t} \wedge \mathtt{p} \cap \mathtt{t}\bullet = \emptyset) \wedge (\widehat{\mathtt{p}} \subseteq \mathtt{t}\bullet \wedge \widehat{\mathtt{p}} \cap \bullet\mathtt{t} = \emptyset)$$

Definição 32 (Eliminação de Contatos)**.** *Dada rede de Petri* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M_0})$*, um conjunto de novos lugares* $\widehat{\mathtt{P}}$ *disjunto de* $\mathtt{P}$*, e uma bijecção* $\varphi \mid \mathtt{P} \rightarrow \widehat{\mathtt{P}}$*, é possível obter uma rede equivalente livre de contatos (*$\mathtt{RPM'} = (\mathtt{P'}, \mathtt{T'}, \mathtt{F'}, \mathtt{M'_0})$*) tal que*

$$\mathtt{P'} = \mathtt{P} \cup \widehat{\mathtt{P}},$$

$$\mathtt{T'} = \mathtt{T},$$

$$\mathtt{F'} = \mathtt{F} \cup \{\mathtt{Pos}(\varphi(\mathtt{p}), \mathtt{t}) \mid \mathtt{t} \in \mathtt{T} \wedge \mathtt{Pre}(\mathtt{p}, \mathtt{t}) \in \mathtt{F}\}$$
$$\cup \{\mathtt{Pre}(\varphi(\mathtt{p}), \mathtt{t}) \mid \mathtt{t} \in \mathtt{T} \wedge \mathtt{Pos}(\mathtt{p}, \mathtt{t}) \in \mathtt{F}\},$$

$$\mathtt{M'_0} = \mathtt{M_0} \cup \mathtt{M}(\varphi(\mathtt{p}))) \mid \forall \mathtt{p} \in \mathtt{P}, \mathtt{M}(\varphi(\mathtt{p}))) = \mathtt{k}(\mathtt{p}) - \mathtt{M_0}(\mathtt{p}).$$

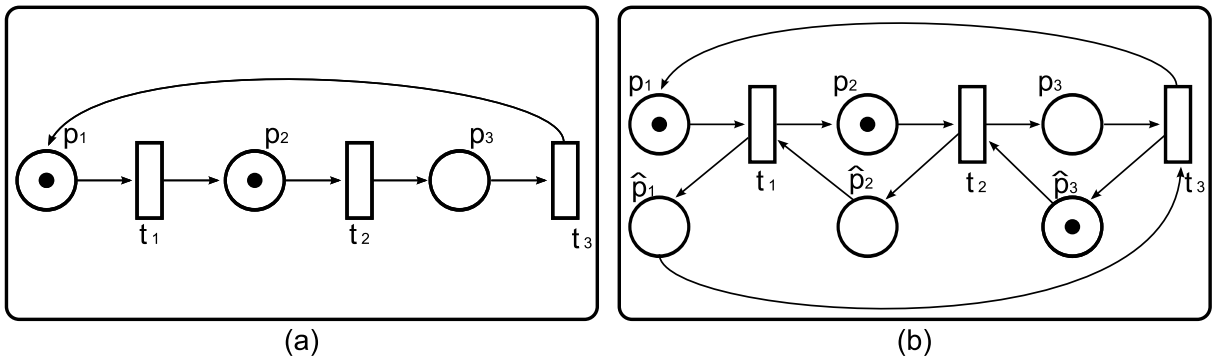

Figura 4.8: Rede de Petri com contato (a) e a mesma rede livre de contato (b).

4.5 Propriedades

Redes de Petri é uma ferramenta que não apenas se restringe à descrição de sistemas, com ela também é possível identificar propriedades da rede que permitem verificar suas características. Segundo Murata [17], dois tipos de propriedades podem ser estudadas: as comportamentais (dependentes da marcação da rede) e as estruturais (dependentes da topologia da rede). No entanto, para o presente trabalho, serão discutidas apenas as propriedades comportamentais das redes de Petri, as quais estão baseadas nos trabalhos de Murata [17] e Buchholz [40], levando em consideração apenas as propriedades básicas necessárias para o entendimento deste trabalho.

A propriedade de *alcançabilidade* é a base fundamental para o estudo das propriedades dinâmicas de uma rede. O conjunto de marcações alcançáveis é o conjunto de todos

os estados que uma dada rede de Petri pode alcançar a partir do disparo de alguma sequência de transições. Esta propriedade pode ser representada por um grafo, o grafo de alcançabilidade, que é composto por nodos, que correspondem a marcações da rede, e por arcos direcionados, que descrevem o disparo de uma transição habilitada. O processo de ramificação da rede gera uma árvore que contém o conjunto de todas as marcações alcançáveis da rede ($\mathtt{R}(\mathtt{M}_0)$) a partir de $\mathtt{M}_0$. Os grafos sempre serão limitados (finitos), se as redes também forem limitadas (ver Definição 37).

Definição 33 (Marcação Alcançável)**.** *Dada uma rede de Petri* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M}_0)$ *e dada uma marcação* $\mathtt{M}_\mathtt{n} \subseteq \mathtt{P}$ *qualquer de* RPM*, então uma marcação* $\mathtt{M}_\mathtt{n}$ *é dita alcançável a partir de uma marcação* $\mathtt{M}_0$ *(denotado por* $\mathtt{M}_0[> \mathtt{M}_\mathtt{n}$*) se, e somente se,*

$$\exists \mathtt{s} \mid \mathtt{M}_0[\mathtt{s} > \mathtt{M}_\mathtt{n} \wedge \mathtt{M}_\mathtt{n} \in \mathtt{R}(\mathtt{M}_0)$$

onde s *é uma sequência de disparos e* $\mathtt{R}(\mathtt{M}_0)$ *é o conjunto de todas as marcações alcançáveis da rede.*

Definição 34 (Grafo de Alcançabilidade)**.** *O grafo de alcançabilidade* $\mathtt{EG}(\mathtt{M}_0)$ *de uma rede de Petri* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M}_0)$ *é uma tripla*

$$\mathtt{G} = (\mathtt{V}, \mathtt{E}, \mathtt{r}_0)$$

onde $\mathtt{V} \in \mathtt{R}(\mathtt{M}_0)$ *são vértices (nodos),* $\mathtt{E} = (\mathtt{M}, \mathtt{t}, \mathtt{M}') \mid \mathtt{t} \in \mathtt{T}, \mathtt{M} \in \mathtt{V} \wedge (\mathtt{M}[\mathtt{t} > \mathtt{M}')$ *são as arestas (arcos) rotuladas com* t *e* $\mathtt{r}_0 = \mathtt{M}_0$ *é o nodo raiz da rede.*

Definição 35 (Redes de Petri Equivalentes)**.** *Duas redes de de Petri* $\mathtt{RPM}_1$ *e* $\mathtt{RPM}_2$ *são equivalentes (*$\mathtt{RPM}_1 \cong \mathtt{RPM}_2$*) se, e somente se, as duas redes tem grafos de alcançabilidade isomorfos (ver Definição 36).*

Definição 36 (Grafos Isomorfos)**.** *Dois grafos direcionados e com arestas rotuladas* $\mathtt{G}_1 = (\mathtt{V}_1, \mathtt{E}_1, \mathtt{r}_1)$ *e* $\mathtt{G}_2 = (\mathtt{V}_2, \mathtt{E}_2, \mathtt{r}_2)$ *são isomorfos (*$\mathtt{G}_1 \cong \mathtt{G}_2$*) se, e somente se, existirem*

duas funções bijectivas $\alpha \mid \mathtt{V}_1 \rightarrow \mathtt{V}_2$ *e* $\beta \mid \mathtt{E}_1 \rightarrow \mathtt{E}_2$ *tal que*

$$\alpha(\mathtt{V}_1) = \mathtt{V}_2 \wedge \forall \mathtt{t} \in \mathtt{T}_1, (\mathtt{M}, \mathtt{t}, \mathtt{M}') \in \mathtt{E}_1 \leftrightarrow (\alpha(\mathtt{M}), \beta(\mathtt{t}), \alpha(\mathtt{M}')) \in \mathtt{E}_2$$

onde $(\mathtt{M}, \mathtt{t}, \mathtt{M}')$ *é uma aresta que liga o vértice* $\mathtt{M}$ *ao vértice* $\mathtt{M}'$.

A figura 4.9 ilustra a ocorrência de isomorfismo entre os grafos G e H, pois existe uma correspondência um-para-um entre os vértices dos dois grafos, tal que o número de arestas entre dois vértices de G é idêntico ao número de arestas entre os vértices correspondentes de H, como pode ser observado nos valores dados pelas funções α e β.

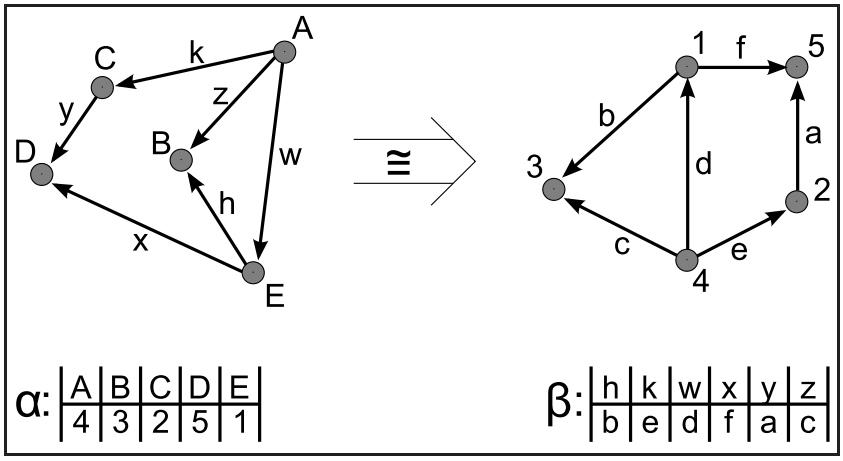

Figura 4.9: Exemplo de isomorfismo entre grafos direcionados e com arestas rotuladas.

Tomando a figura 4.8(a) como exemplo, o grafo de alcançabilidade à ela relacionado é ilustrado pela figura 4.10(a). Como já comentado na seção 4.4, transformar uma rede com contatos em uma rede livre de contato não altera o seu comportamento. Esta situação pode ser visualizada através da construção do grafo de alcançabilidade da duas redes da figura 4.8, onde os dois grafos resultantes são isomorfos. É importante salientar que duas redes equivalentes sempre contêm o mesmo número de transições, porém, o número de lugares pode ser diferente, como é o caso deste exemplo.

As redes de Petri elementares têm uma propriedade que está introduzida no próprio conceito deste tipo de rede: é a *limitabilidade*. Em redes elementares todos os lugares que compõe a rede tem o limite máximo de uma marca. Em outros tipos de redes de Petri esta propriedade permite garantir que, nos casos dos lugares modelarem *buffers* ou registros,

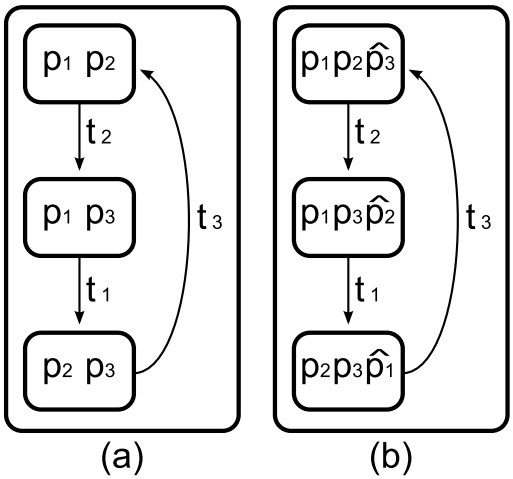

Figura 4.10: Exemplo de grafos de alcançabilidade e de redes de petri equivalentes.

nunca será verificada a situação de *overflow*[1].

Definição 37 (Rede de Petri k-limitada)**.** *Uma rede de Petri* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M_0})$ *é dita k-limitada se, e somente se,*

$$\forall \mathtt{p} \in \mathtt{P}, \forall \mathtt{M} \in \mathtt{R}(\mathtt{M_0}), \mathtt{M}(\mathtt{p}) \leq \mathtt{k}(\mathtt{p})$$

onde $\mathtt{k} \in \mathbb{N}$ *é o número máximo de marcações de um lugar . As redes 1-limitadas também são chamadas de redes seguras, como é o caso das redes de Petri elementares*[2]*.*

Outra propriedade está relacionada a *vivacidade* da rede. Neste caso quanto se tem uma rede de Petri viva então se tem a garantia de que a estrutura da rede não provoca bloqueio, também conhecido como *deadlock*[3]. Além disso, uma rede viva também garante que, para qualquer uma de suas transições, sempre existe um caminho que as torna disparáveis.

Definição 38 (Rede de Petri Viva)**.** *Dada uma rede de Petri* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M_0})$*, ela é viva se, e somente se, todas as suas transições são vivas, ou seja, para todas as suas transições*

[1] A situação de *overflow* ocorre quando o número de marcas de um lugar ultrapassa sua capacidade máxima de armazenamento.

[2] Rede de Petri Elementar é um tipo de rede ordinária, ou seja, uma rede onde todos os seus arcos têm peso 1 e, além disso, todos os seus lugares são 1-limitados.

[3] Deadlock é um estado da rede no qual nenhuma transição da rede está habilitada no sistema e portanto não há estado sucessor.

existe uma sequência de disparo de transições $\mathtt{s}$ *que a habilita.*

$$\forall \mathtt{t} \in \mathtt{T}, \mathtt{M} \in \mathtt{R}(\mathtt{M_0}), \exists \mathtt{s} \mid \mathtt{M_0}[\mathtt{s} > \mathtt{M} \ \wedge \ \mathtt{M}[\mathtt{t} >$$

Por último, a propriedade de *reiniciabilidade*. Esta ocorre sempre que existir uma sequência de disparos que leva a rede de uma marcação qualquer à marcação inicial. Por tanto, o seu grafo de alcançabilidade é sempre fortemente conexo. Esta propriedade é comumente utilizada em sistemas de manufatura onde o sistema de produção é, geralmente, cíclico [41].

Definição 39 (Rede de Petri Reiniciável)**.** *Uma rede de Petri* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M_0})$ *é reiniciável (cíclica) se, e somente se*

$$\forall \mathtt{M}' \in \mathtt{R}(\mathtt{M_0}), \exists \mathtt{s} \mid \mathtt{M}'[\mathtt{s} > \mathtt{M_0}$$

4.6 Representação matricial

Além da representação gráfica (veja a seção 4.1), as redes de Petri possibilitam a sua representação sob o ponto de vista matemático. Para isso, as redes são definidas matricialmente, onde é armazenado o fluxo $\mathtt{F}$ dos arcos entre os lugares e transições. Esta matriz é chamada de matriz de incidência ($\mathtt{C}$). No entanto, para construí-la é necessário que haja a subtração das matrizes de entrada e de saída, onde são especificados os pesos dos arcos ($\mathtt{Pre}(\mathtt{p_i}, \mathtt{t_j})$) e $\mathtt{Pos}(\mathtt{p_i}, \mathtt{t_j})$.

Definição 40 (Matriz de Incidência)**.** *Dada uma rede* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M_0})$*. Seja* $\mathtt{m}$ *o número de lugares de* $\mathtt{P}$ *e* $\mathtt{n}$ *o número de transições de* $\mathtt{T}$*, com* $\mathtt{i}$ *e* $\mathtt{j}$ *inteiros variando dentro de, respectivamente,* $\{1, 2, ..., \mathtt{m}\}$ *e* $\{1, 2, ..., \mathtt{n}\}$*. As matrizes características de* $\mathtt{RPM}$ *são:*

i) Matriz incidência de entrada, $\mathtt{Pre} = [\mathtt{Pre}(\mathtt{p_i}, \mathtt{t_j})]_{\mathtt{m}\times\mathtt{n}}$

ii) Matriz incidência de saída, $\mathtt{Pos} = [\mathtt{Pos}(\mathtt{p_i}, \mathtt{t_j})]_{\mathtt{m}\times\mathtt{n}}$

iii) Matriz incidência, $\mathtt{C} = \mathtt{Pos} - \mathtt{Pre}$

Tomando a figura 4.8 (b) como exemplo, a construção das matrizes de incidência para esta rede é:

$$\mathtt{Pre} = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 \end{matrix} \\ \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \end{array} \begin{matrix} p_1 \\ p_2 \\ p_3 \\ \widehat{p}_1 \\ \widehat{p}_2 \\ \widehat{p}_3 \end{matrix} \quad \mathtt{Pos} = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 \end{matrix} \\ \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \end{array} \begin{matrix} p_1 \\ p_2 \\ p_3 \\ \widehat{p}_1 \\ \widehat{p}_2 \\ \widehat{p}_3 \end{matrix} \quad \mathtt{C} = \begin{array}{c} \begin{matrix} t_1 & t_2 & t_3 \end{matrix} \\ \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \end{array} \begin{matrix} p_1 \\ p_2 \\ p_3 \\ \widehat{p}_1 \\ \widehat{p}_2 \\ \widehat{p}_3 \end{matrix}$$

Muitas RdP são modeladas com transições cujo disparo repõe marcas em um ou mais lugares do seu pré-conjunto, caracterizando assim, uma transição em laço (*self-loop*). Neste caso, como a matriz de incidência não representa toda a estrutura da rede[4], então é necessário garantir que rede sempre seja pura (ver definição 17).

A equação para obter a nova marcação pode ser escrita como: $\mathtt{M}' = (\mathtt{M} - {\bullet}\mathtt{t} + \mathtt{t}{\bullet})$ ou ainda $\mathtt{M}' = \mathtt{M} + \mathtt{C}(\mathtt{t})$, onde $\mathtt{C}(\mathtt{t})$ é o vetor coluna da transição $\mathtt{t} \in \mathtt{T}$ e $\mathtt{M}$ é o vetor coluna da marcação atual da rede. Então para o disparo da transição $\mathtt{t_1}$ se tem

$$\mathtt{M}' = [0, 0, 1]^{\mathtt{T}} + [1, 0, -1]^{\mathtt{T}} = [1, 0, 0]^{\mathtt{T}}$$

No entanto, esta equação funciona apenas para o disparo de um única transição por vez. Sendo assim, quando se deseja saber a marcação da rede após uma sequência de disparos utiliza-se a equação fundamental.

Definição 41 (Equação Fundamental)**.** *A evolução de uma rede* $\mathtt{RPM} = (\mathtt{P}, \mathtt{T}, \mathtt{F}, \mathtt{M_0})$ *para uma sequência de disparos de transições que leva a rede de uma marcação* $\mathtt{M}$ *para* $\mathtt{M}'$ *(*$\mathtt{M}[\mathtt{s} > \mathtt{M}'$*) é dado pela equação*

$$\mathtt{M}' = \mathtt{M} + \mathtt{C}.\bar{\mathtt{s}}$$

onde $\bar{s}$ *é o vetor característico que contém o número de vezes que cada transição de s foi disparada.*

[4]A estrutura da rede não é totalmente representada porque a transição gera uma marca para o lugar e retira outra marca do mesmo lugar e, portanto, o lugar será representado por 0 (zero) na matriz C, o que assinala a inércia da transição sobre aquele lugar.

Ainda utilizando a figura 4.8 (b), a nova marcação $\mathtt{M}'$ obtida utilizando a equação fundamental a partir do disparo das transições $\mathtt{t_2}$ e $\mathtt{t_1}$ ($\mathtt{M}[\{\mathtt{t_2},\mathtt{t_1}\} > \mathtt{M}'$) é dado por $\mathtt{M}' = [\mathtt{0},\mathtt{1},\mathtt{1},\mathtt{1},\mathtt{0},\mathtt{0}]$.

$$\mathtt{M}' = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 1 \\ 1 & -1 & 0 \\ 0 & 1 & -1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} \begin{matrix} p_1 \\ p_2 \\ p_3 \\ \widehat{p}_1 \\ \widehat{p}_2 \\ \widehat{p}_3 \end{matrix}$$

O vetor $\bar{s}$ apresentado no caso acima é disparável para a sequência $\mathtt{s} = \{\mathtt{t_2},\mathtt{t_1}\}$. Entretanto, pode haver sequências que não são disparáveis, como é o caso de $\mathtt{s} = \{\mathtt{t_1},\mathtt{t_2}\}$, que são caracterizadas pelo mesmo vetor $\bar{\mathtt{s}} = [\mathtt{1},\mathtt{1},\mathtt{0}]^{\mathtt{T}}$. No entanto, não se tem garantia da existência de $\mathtt{s}$.

Além disso, através da representação matricial também é possível efetuar uma análise estrutural da rede, que também permite obter informações adicionais sobre a dinâmica da rede, através das invariantes de lugar e de transição [42, 17, 41].

4.7 Redes de Petri e a representação do tempo

Como comentado na seção 3.1, a informação temporal possibilita sequencializar eventos, comparar suas durações, assim como determinar o intervalo existente entre eles. Desta forma, é possível representar e analisar problemas ligados a diversas atividades onde a avaliação do tempo é um critério de fundamental importância. Como a noção de tempo não é explicitamente dada na definição original de redes de Petri, então, este conceito foi incorporado posteriormente. Com a adição de tempo, o indeterminismo com relação ao disparo de uma transição é, de certa forma, reduzido já que a informação temporal acrescenta uma nova relação de ordem entre os disparos das transições [43].

As redes de Petri com tempo podem ser divididas em duas classes [17]: estocásticas e determinísticas. Na primeira, o tempo é modelado com variáveis aleatórias com distribuição exponencial, permitindo executar as transições através de uma função probabilística

onde é possível considerar incertezas nos tempos. Nas determinísticas, existem quatro abordagens para associar tempo aos elementos constituintes das redes de Petri:

1. tempo associado às transições;
2. tempo associado aos lugares;
3. tempo associado às marcas;
4. tempo associado aos arcos.

O funcionamento de uma rede de Petri com tempo tem o comportamento dinâmico semelhante àquela apresentado nas seção 4.3, com a diferença de que aqui cada elemento da rede, dependendo da abordagem escolhida, consome `Z` unidades de tempo. Por exemplo, se for usada a abordagem 1, então o tempo é associado às transições.

Definição 42 (Rede de Petri com Tempo)**.** *De forma genérica, uma rede de Petri com tempo pode ser definida como uma dupla*

$$\texttt{RPT} = (\texttt{RPM}, \texttt{Z}(\texttt{x}))$$

onde `RPM` *é uma rede de Petri (definição 21) e* `Z` *é a informação temporal aplicada a cada elemento* `x` *da rede, onde* `x` *é uma transição, um lugar, uma marca ou um arco.*

Atualmente há três modelos de RdP que mais se destacam. O primeiro modelo, conhecido como RdP Temporizadas (TdPNs, do inglês *Timed Petri Nets*), foi introduzido por Ramchandani [44] onde o tempo é descrito através da duração do disparo, expresso por um número racional positivo. Os disparos das transições neste tipo de rede são executados em três passos:

1. uma transição `t` é disparada no momento em que ela é habilitada (figura 4.11 (a)), consumindo as marcas do seu pré-conjunto;
2. a marcação fica retida por `Z(t)` unidades de tempo pela transição (figura 4.11 (b));

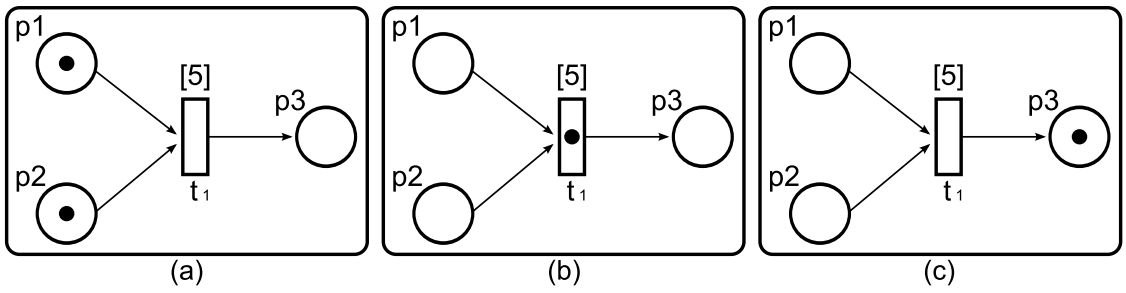

Figura 4.11: Exemplo do disparo de uma rede de Petri temporizada.

3. após decorridas Z(t) unidades de tempo, as marcas são depositadas no pós-conjunto de t (figura 4.11 (c)).

No segundo modelo de RdP o tempo é associado aos lugares. Este modelo foi introduzido por Sifakis [45], e é conhecido como RdP P-temporizadas. Neste caso, Z(p) associa um atraso a cada lugar p ∈ P fazendo com que as marcas permaneçam Z(p) unidades de tempo no lugar antes de serem utilizadas para habilitar as transições do seu pós-conjunto.

Desta forma é possível definir dois tipos de marcas: disponíveis e indisponíveis. Quando uma marca chega a um determinado lugar p ∈ P, ela sempre está no estado indisponível. Ela se tornará disponível após decorrido o tempo associado ao lugar em questão. O estado atual da rede é dado pela soma das marcas disponíveis e indisponíveis de cada lugar. Já o disparo de uma transição ocorre de forma semelhante às TdPNs.

As redes de Petri P-temporizadas e as TdPNs podem representar um mesmo sistema, pois elas são expressivamente equivalentes [46]. Sendo assim, existe uma forma de transformar um modelo em outro. As figuras 4.12 e 4.13 mostram como são feitas estas transformações.

Por fim, o terceiro modelo: as RdP Temporais. Este modelo de RdP é mais abrangente, pois as TdPNs e as P-temporizadas estão contidas neste modelo. Uma vez que este modelo é o que possui melhores ferramentais de análise, ele será o modelo adotado neste trabalho.

4.7.1 Redes de Petri temporais

O modelo das Redes de Petri Temporais (TPNs, do inglês *Time Petri Nets*) foi proposto por Merlin [47]. O tempo é descrito através de um intervalo de disparo, onde, após

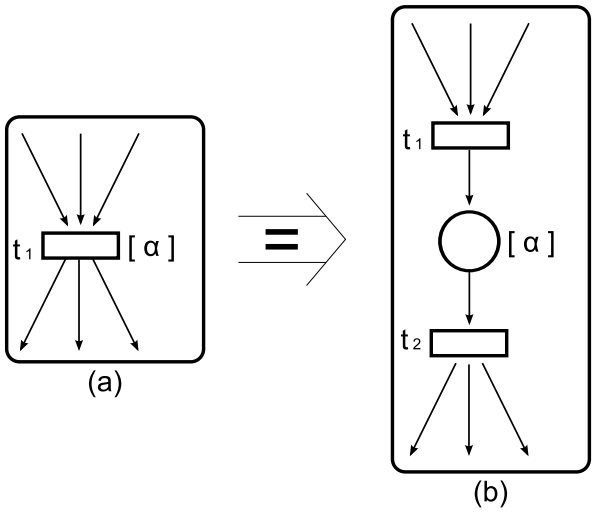

Figura 4.12: Transformação de uma rede de Petri temporizada (a) em uma rede de Petri temporizada de lugar (b).

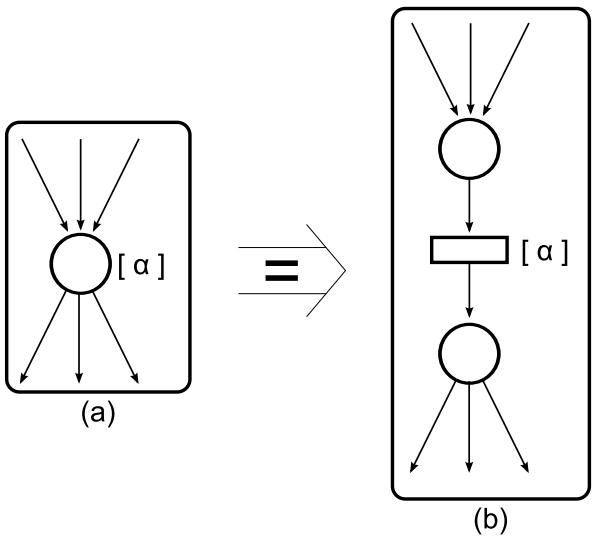

Figura 4.13: Transformação de uma rede de Petri temporizada de lugar(a) em uma rede de Petri temporizada (b).

a habilitação de uma transição, tem-se um tempo mínimo durante o qual a transição deve esperar até que possa disparar e um tempo máximo admissível de espera para o disparo.

Definição 43 (Rede de Petri Temporal)**.** *Uma rede de Petri com tempo* $\mathtt{RPT} = (\mathtt{RPM}, \mathtt{Z}(\mathtt{t}))$ *é dita uma Rede de Petri Temporal se, e somente se,*

$$\forall \mathtt{t} \in \mathtt{T}, \mathtt{Z}(\mathtt{t}) = [\alpha, \beta] \mid \alpha \in \mathbb{Q}^+ \wedge \beta \in (\mathbb{Q}^+ \cup \infty) \wedge \beta \geq \alpha$$

onde $\mathtt{Z}(\mathtt{t})$ *é o intervalo de sensibilização de cada transição* $\mathtt{t} \in \mathtt{T}$*, tal que* α *e* β *são os*

limites inferiores e superiores, respectivamente, de $\mathtt{t}$.

Nas TPNs cada transição deve possuir um relógio independente para o controle de seu tempo. Quando uma transição é habilitada o relógio da transição começa a decrementar o tempo a ela atribuído, tanto para o valor de α quanto para o de β. Sendo assim, haverá um ponto que o valor de α chegará ao valor 0 (zero), o que significa que a transição já pode disparar. Porém, o valor de β pode ainda não ter zerado, o que significa que este é o tempo máximo possível para retardar o disparo.

O disparo de uma transição ocorre de forma semelhante às redes de Petri convencionais [39]: ele é instantâneo e ocorre em um instante após decorridos α unidades de tempo e antes de β unidades de tempo após a sua habilitação. É importante salientar que para ocorrer o disparo de uma transição é necessário que a mesma permaneça continuamente habilitada, caso contrário, ela é desabilitada e seu relógio removido. Este tipo de disparo é chamado de disparo atômico. Em outras palavras, se uma transição $\mathtt{t} \in \mathtt{T}$ for habilitada no tempo τ, então $\mathtt{t}$ deve permanecer continuamente habilitada até o seu disparo. Entretanto, $\mathtt{t}$ não pode disparar antes $\tau + \alpha$, mas obrigatoriamente deve disparar até o tempo $\tau + \beta$, a não ser que $\mathtt{t}$ seja desabilitada pelo disparo de outra transição através do consumo de marcas de suas pré-condições.

Definição 44 (Condição de Disparo)**.** *Se* $\mathtt{t} \in \mathtt{T}$ *é uma transição habilitada no momento* τ *então* $\mathtt{t}$ *pode disparar se, e somente se,*

$$\mathtt{M}[\mathtt{t} > \wedge(\tau + \alpha) \leq \tau \leq (\tau + \beta)$$

Além disso, durante o intervalo de tempo $[\tau, \tau + \alpha]$ *a transição deve estar continuamente habilitada e também deve disparar até* $\tau + \beta$ *caso* $\mathtt{t}$ *não seja desabilitado por outra transição.*

Um exemplo da semântica das transições temporais é apresentado pela Figura 4.14. O quadro (a) representa o estado inicial da rede que possui a transição $\mathtt{t_1}$ habilitada para o instante 0 (zero). Porém esta transição ainda não pode disparar, pois ela precisa esperar exatamente duas unidades de tempo pois α é igual a β. Após decorrido este tempo, a

transição t_1 dispara no instante 2 do relógio da rede, como observado no quadro (b) e (c), onde a transição pronta para disparar está na cor cinza. Avançando mais uma unidade de tempo, então t_3 já pode disparar (quadro (d)), mas ela ainda pode retardar este disparo em até 4 unidades de tempo, como acontece neste exemplo. No instante 4, representado pelo quadro (e), passa-se a ter duas transições prontas para disparar e se as duas transições retardam seus disparos por mais uma unidade de tempo (quadro (f) e (g)), então t_2 é obrigado a disparar no instante 5, porque o valor de β já está zerado. Por fim, a transição t_3, a partir deste momento, ainda pode ser disparada em qualquer um dos instantes 5, 6 ou 7.

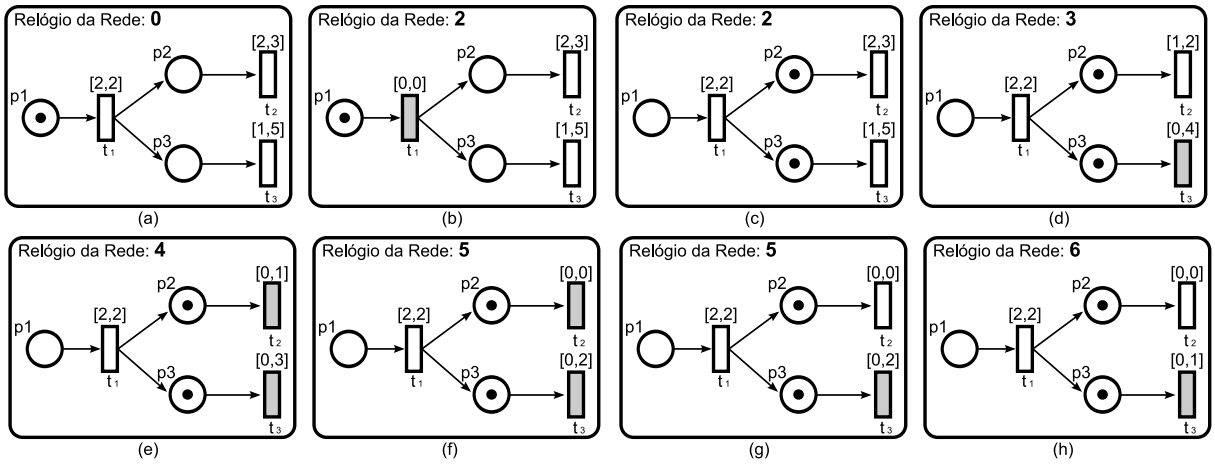

Figura 4.14: Exemplo de uma sequência de disparos de uma rede de Petri temporal com disparo atômico.

Quando uma rede tem transições em conflito, há casos que é necessário escolher uma transição para o disparo. Se considerarmos o tempo $\tau = 2$, então t_1 e t_2 podem disparar (figura 4.15). Se escolhido t_1, então desabilitará t_2. Mas o que fazer com o tempo restante da transição que foi desabilitada e com o tempo do restante das transições da rede? Há três abordagens que podem ser adotadas [39]:

1. *Resampling*: Após cada disparo todos os tempos, de todas as transições da rede, são reiniciadas. Neste caso não há memória, e depois de descartados todos os tempos, novos valores de tempo somente serão aplicados ao novo conjunto de transições que são habilitadas a partir da nova marcação da rede.

2. *Enabling memory*: Após cada disparo, os tempos das transições que foram desa-

bilitadas são reiniciados. As demais transições mantêm seus valores. É esta a abordagem que será adotada por este trabalho.

3. *Age memory*: Após cada disparo, todos os tempos de todas as transições mantêm seus valores.

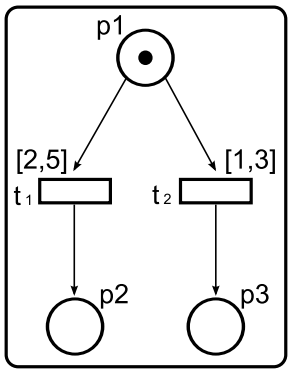

Figura 4.15: Conflito em uma transição temporal.

A diferença entre TdPN e TPN, quando há conflito, é que na primeira, por ter seu disparo em três fases, se faz a escolha de uma transição para disparar dentre um conjunto de transições habilitadas, obrigando esta transição a disparar e não dando mais chance às demais. Já na TPN o disparo atômico faz com que as marcas não sejam removidas na habilitação da transição. Qualquer transição pode ser disparada, desde que obedeça a definição 44. Como uma TdPN está contida em uma TPN temporal, então existe um mapeamento que traduz a primeira para última, assim como apresentado pela figura 4.16.

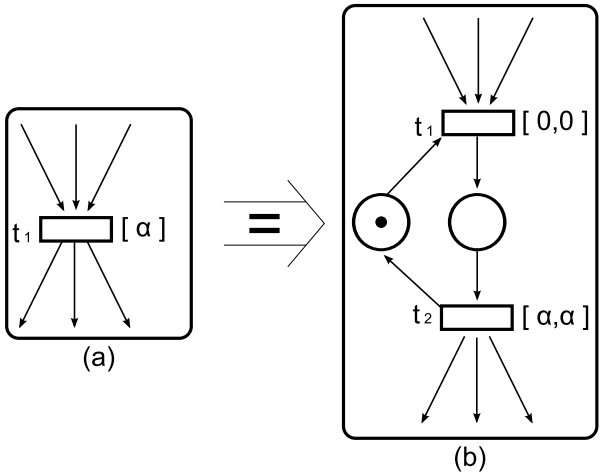

Figura 4.16: RdP temporizada (a) traduzida para RdP temporal (b).

4.8 Considerações

O presente capítulo teve como objetivo apresentar as Redes de Petri com tempo, que serão utilizadas no próximo capítulo. Desta forma, com os parâmetros temporais, as RdP expandiram sua capacidade de representação, assim como aconteceu na área de planejamento. Os modelos temporais propostos por Ramchandani [44], Merlin [47] e Sifakis [45] são os mais comumente utilizados, porém, segundo Balbo [39], uma rede de Petri temporal com disparo atômico pode preservar o comportamento básico do modelo das redes de Petri sem tempo.

CAPÍTULO 5

RELAÇÕES ENTRE PLANEJAMENTO TEMPORAL E REDES DE PETRI COM TEMPO

O presente capítulo apresenta relacionamentos entre redes de Petri com tempo e planejamento temporal. Estas relações são dadas pela tradução de problemas de um formalismo para outro, a fim de possibilitar o uso dos algoritmos de ambas as áreas para a resolução de problemas de alcançabilidade. A seção 5.1 apresenta métodos de tradução de RdP temporais e temporizadas para problemas de planejamento temporal em PDDL enquanto a seção 5.2 apresenta a tradução de problemas de planejamento temporal em PDDL para RdP temporais.

5.1 Tradução de RdP com tempo para planejamento temporal

Tendo em vista que um problema de alcançabilidade em RdP pode ser visto como um problema de planejamento, então a tradução de uma RdP com tempo para Planejamento Temporal é trivial, pois na área de planejamento existem algoritmos eficientes para resolver este tipo de problema. Esta abordagem também já foi adotada pelo trabalho de Edelkamp e Jabbar [4] para a detecção de bloqueios em RdP, porém aplicada somente em RdP que não levam em consideração as restrições temporais.

Foram desenvolvidas 3 traduções distintas, conforme apresentado nas próximas seções, dentre as quais as duas primeiras são de propósito geral e a última é de caráter exclusivo para a verificação de bloqueios em RdP. As traduções apresentadas possuem algumas restrições, dentre as quais é exigido que todas as RdP sejam seguras (definição 37), ou seja, os lugares devem possuir no máximo uma marca e todos os arcos da RdP possuir peso 1. Outra restrição importante é que as RdP devem ser livres de contato (definição 30), pois caso contrário os planejadores podem obter planos que não correspondem a uma sequência válida de disparos da RdP porque a execução de uma ação pode tentar adicionar

marcas em um lugar que já está marcado, o que não é possível já que a rede é segura. Por outro lado, as restrições citadas anteriormente podem ser facilmente eliminadas conforme apresentado na seção 5.1.5, o que proporciona traduzir RdP com outras propriedades. A base das traduções apresentadas tem como referência o trabalho de Edelkamp e Jabbar [4].

De maneira geral, todas as traduções possuem um mesmo padrão para a descrição do problema de planejamento, onde a única diferença está relacionada a forma como cada ação deve ser descrita para representar o comportamento de disparo das transições para um determinado tipo de RdP. Utilizando a PDDL, apresentada no capítulo 2, como linguagem de representação das traduções das RdP para os planejadores automáticos o cabeçalho do arquivo de domínio deve ser descrito pelos requisitos `:strips` e `:durative-actions` e também por um predicado nomeado de `(marked ?p)` que determinará se um lugar possui ou não marcas. Além disso, cada lugar da RdP é traduzido como uma constante fazendo com que não haja a necessidade de criar um predicado específico para a descrição dos arcos da rede, pois já se conhece quais são as pré-condições e efeitos de cada transição. A listagem 5.1 apresenta um exemplo do cabeçalho do arquivo de domínio para a RdP da figura 5.1.

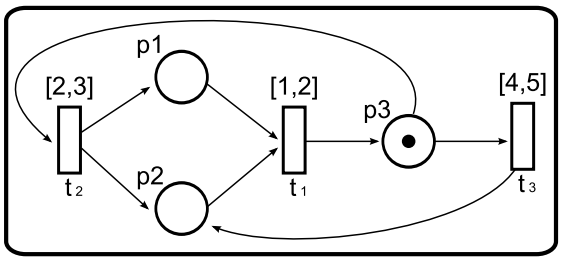

Figura 5.1: Exemplo de rede de Petri temporal.

```
1 (define (domain Test)
    (:requirements :strips :durative-actions)
    (:constants p1 p2 p3 )
    (:predicates (marked ?p)) )
```

Listagem 5.1: Cabeçalho de um arquivo de domínio.

Para descrever uma ação não é necessário utilizar parâmetros, pois já se conhece todas as informações necessárias sobre a transição a ser traduzida. A utilização de parâmetros

faz com que o planejador tenha que instanciar todas as combinações possíveis para os parâmetros recebidos, o que é fatorial. Não utilizar parâmetros implica em um arquivo de domínio maior, pois a descrição das ações se torna enumerativa mas que, ao mesmo tempo, despreza o processo de instanciação das variáveis pelo planejador.

As peculiaridades da tradução das ações para redes de Petri temporais, temporizadas e para verificação de bloqueios são apresentadas nas próximas seções, assim como a descrição do arquivo de problema de planejamento.

5.1.1 Tradução de transições temporais

A linguagem PDDL pode derivar outras linguagens, tal como apresentado pela figura 3.7 da seção 3.3. Entretanto, uma transição temporal possui limitações que, devido a sua semântica, impedem a representação em todas as linguagens derivadas da PDDL.

Teorema 1. *Uma transição temporal em RdP pode ser representada pelas sub-linguagens* L_e^{soe}*,* L_s^s *ou* L_e *da PDDL.*

Demonstração. A equivalência entre uma transição temporal e uma sub-linguagem da PDDL pode ser verificada através da análise comportamental que ambas podem expressar. Dada uma ação durativa em PDDL contendo todas as restrições temporais possíveis, ou seja, uma ação descrita pela linguagem L_{se}^{soe}, e uma transição temporal tal que $Pre(p,t) = Pos(p,t) \neq \emptyset$, ambas possuindo a mesma duração, então aplicando as condições de disparo da transição temporal sobre a ação durativa é possível verificar em quais restrições temporais da PDDL a marcação da RdP sofre mudanças.

A figura 5.2 apresenta, através de um traço contínuo, a regra de disparo de uma transição temporal, onde o conjunto $\bullet t$ deve permanecer marcado durante toda a execução da ação, ou seja, nas restrições *at start, over all*, e *at end*, e através da linha tracejada o instante no qual as marcas de $\bullet t$ são removidas, que somente acontece ao final da ação (*at end*). Já o conjunto $t\bullet$ apenas recebe as marcações do disparo da transição no instante *at end*. Sobre esta análise é possível concluir que toda transição temporal com tempo $\alpha > 0$ pode ser representa pela linguagem L_e^{soe}. Por outro lado, quando uma transição

não possui pré-condições, então esta transição pode ser representada por L_e, que é uma sub-linguagem de L_e^{soe}. Também há casos onde uma transição possui disparo instantâneo, isto é, $\alpha = \beta = 0$. Quando esta situação acontece, a transição pode ser representada pela linguagem L_s^s.

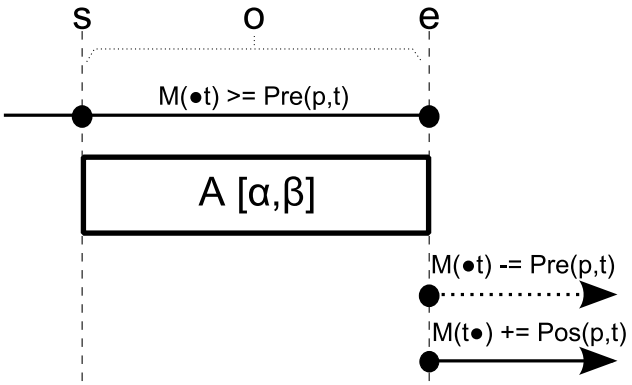

Figura 5.2: Modelo comportamental de uma transição temporal sobre uma ação durativa.

□

Sendo assim, a tradução de uma transição temporal para uma ação deve usar a linguagem L_e^{soe}, que é temporalmente expressiva (definição 13), para as transições que possuem duração maior que zero. O tempo de uma transição deve ser descrito na forma de uma inequação sempre que a sua duração for expressa por um intervalo de tempo $[\alpha, \beta]$, tal que $\alpha < \beta$, e de forma fixa quando o valor de $\alpha = \beta$, assim como apresentado na seção 2.3.1.

A tradução do pré-conjunto da transição gera as condições de execução da ação assim como o seu pós-conjunto gera os efeitos. Como a semântica de uma transição temporal diz que todos os lugares do seu pré-conjunto devem permanecer marcados durante toda a duração da ação então cada um destes lugares é descrito através das 3 restrições temporais, ou seja, `(at start p)`, `(over all p)` e `(at end p)`. Na tradução dos efeitos, os lugares que pertencem ao pós-conjunto da transição devem receber marcas ao final da execução da ação e, além disso, todos os lugares presentes nas condições perdem marcas, exceto os lugares ligados às transições por arcos bidirecionais. A listagem 5.2 apresenta a tradução da transição temporal t_1 da figura 5.1, onde o tempo é descrito na forma de uma inequação, os lugares p1 e p2 tratados como condições que devem estar presentes durante toda a ação

e, nos efeitos, o consumo e a geração de marcas ao final de sua execução.

Uma transição temporal $\mathtt{t} \in \mathtt{T}$, tal que $\alpha \in Z(t) > 0$, pode ser formalmente traduzida para uma ação durativa $\mathtt{o} = \langle \mathtt{Cond_s}, \mathtt{Cond_o}, \mathtt{Cond_e}, \mathtt{Eff_s}, \mathtt{Eff_e}, [\alpha, \beta] \rangle$ por

$$\mathtt{o} = \cup \begin{cases} \mathtt{Cond_s} = \mathtt{Cond_o} = \mathtt{Cond_e} = \bullet\mathtt{t} \\ \mathtt{Eff_s} = \emptyset \\ \mathtt{Eff_e} = \mathtt{t}\bullet \cup \{\neg \mathtt{p} \mid \mathtt{p} \in \bullet\mathtt{t}\} \\ [\alpha, \beta] = \mathtt{Z(t)} \end{cases}$$

onde $\mathtt{o}$ é uma ação composta pelas condições $\mathtt{Cond_s}, \mathtt{Cond_o}$ e $\mathtt{Cond_e}$ aplicáveis, respectivamente, ao instante inicial, durante e final da ação, enquanto $\mathtt{Eff_s}$ e $\mathtt{Eff_e}$ são os efeitos de início e fim da ação. O intervalo $[\alpha, \beta]$ representa a duração da ação.

```
1   (:durative-action t1
      :parameters ()
      :duration (= ?duration 4)
      :condition (and
5         (at start (marked p1))
          (over all (marked p1))
          (at end (marked p1))
          (at start (marked p2))
          (over all (marked p2))
10        (at end (marked p2))
          )
      :effect ( and
          (at end (not (marked p1)))
          (at end (not (marked p2)))
15        (at end (marked p3))
        )
    )
```

Listagem 5.2: Tradução de uma transição temporal.

Quando uma transição é instantânea, isto é, $\alpha = \beta = 0$ para o intervalo de tempo $[\alpha, \beta]$, então todas as restrições temporais da linguagem PDDL ocorrem no mesmo instante de tempo, o que caracteriza uma linguagem temporalmente simples (definição 13). Sendo assim, a descrição da listagem anterior pode ser simplificada através da eliminação das linhas que contêm as restrições *over all* e *at end* das condições e substituindo todas *at end* dos efeitos por *at start*, ou seja, utilizando somente a linguagem L_s^s ou L_e^e, já que as duas podem representar ações instantâneas. Supondo que a transição t_1 da figura 5.1 possua

intervalo de tempo igual a [0,0], então a sua tradução é descrita pela ação apresentada pela listagem 5.3.

$$\mathtt{o} = \cup \begin{cases} \mathtt{Cond_s} = \bullet\mathtt{t} \\ \mathtt{Cond_o} = \mathtt{Cond_e} = \mathtt{Eff_e} = \emptyset \\ \mathtt{Eff_s} = \mathtt{t} \bullet \cup \{\neg \mathtt{p} \mid \mathtt{p} \in \bullet\mathtt{t}\} \\ [\alpha, \beta] = \mathtt{Z(t)} \end{cases}$$

```
(:durative-action t1
  :parameters ()
  :duration (= ?duration 0)
  :condition (and
      (at start (marked p1))
      (at start (marked p2))
      )
  :effect ( and
      (at start (not (marked p1)))
      (at start (not (marked p2)))
      (at start (marked p3))
    )
)
```

Listagem 5.3: Tradução de uma transição temporal com disparo instantâneo.

5.1.2 Tradução de transições temporizadas

A tradução das transições temporizadas é dada de forma semelhante a das transições temporais. A única diferença está relacionada a semântica deste tipo de transição, que pode ser representada em outra linguagem da PDDL.

Teorema 2. *Uma transição temporizada em RdP pode ser representada pelas sub-linguagens L^s_{se}, L^s_s ou L_e da PDDL.*

Demonstração. Assim como na prova do teorema 1, a linguagem L_e pode representar uma transição que não possui pré-condições para seu disparo enquanto a linguagem L^s_s pode representar uma transição com disparo instantâneo. Sabendo que uma RdP temporizada está contida em uma RdP temporal, assim como comentado na seção 4.7, então a linguagem que representa o disparo de uma transição temporizada é dada pela combinação de

um sub-conjunto das linguagens que representam uma transição temporal. No entanto, tal combinação é dada por L_s^s e L_e gerando a linguagem L_{se}^s, que representa um comportamento similar a de uma transição temporizada. A figura 5.3 ilustra que os lugares de $\bullet t$ devem permanecer marcados somente o instante inicial da ação (*at start*) e tendo como efeito o consumo imediado de suas marcas, representado pela linha tracejada. Por outro lado, $t\bullet$ receberá suas marcas somente no instante *at end* da ação. Portanto, sobre a análise da figura 5.3 é possível concluir que uma transição temporizada pode ser representada pela sub-linguagem L_{se}^s.

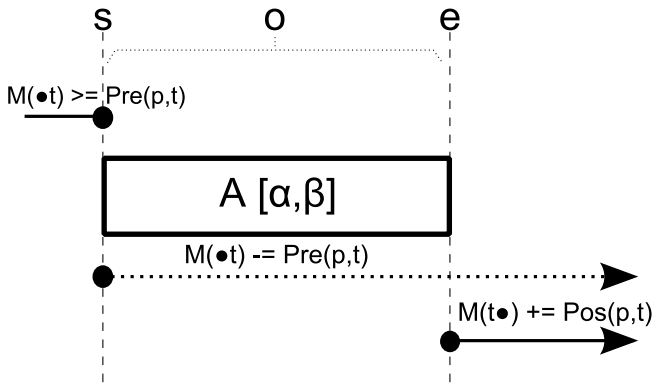

Figura 5.3: Modelo comportamental de uma transição temporizada sobre uma ação durativa.

□

No entanto, como as marcas do pré-conjunto da transição devem ser consumidas logo que esta seja habilitada e o seu pós-conjunto apenas receberá as marcas ao final de seu disparo, então a ação resultante da tradução da transição t_1, supondo que sua duração seja [1,1], é apresentado pela listagem 5.4. É possível notar que existe semelhança com a tradução de uma transição temporal com disparo instantâneo, com a diferença de existir uma duração maior que zero e pela geração das marcas no instante *at end* da ação.

$$\mathtt{o} = \cup \begin{cases} \mathtt{Cond_s} = \bullet\mathtt{t} \\ \mathtt{Cond_o} = \mathtt{Cond_e} = \emptyset \\ \mathtt{Eff_s} = \{\neg \mathtt{p} \mid \mathtt{p} \in \bullet\mathtt{t}\} \cup \{\mathtt{p} \in \mathtt{t}\bullet \mid \widehat{\mathtt{p}} \in \bullet\mathtt{t}\} \\ \mathtt{Eff_e} = \mathtt{t}\bullet \setminus \{\mathtt{p} \in \mathtt{t}\bullet \mid \widehat{\mathtt{p}} \in \bullet\mathtt{t}\} \\ [\alpha, \beta] = \mathtt{Z(t)} \end{cases}$$

```
(:durative-action t1
  :parameters ()
  :duration (= ?duration 1)
  :condition (and
      (at start (marked p1))
      (at start (marked p2))
      )
  :effect ( and
      (at start (not (marked p1)))
      (at start (not (marked p2)))
      (at end (marked p3))
    )
)
```

Listagem 5.4: Tradução de uma transição temporizada.

A tradução das transições que englobam o conceito de lugar complementar requer atenção, pois a habilitação deste tipo de transição causa inconsistência nas variáveis de estado do sistema de planejamento. Um exemplo desta situação pode ser observada em uma transição onde $\bullet t = \{p\}$ e $t\bullet = \{\widehat{p}\}$ que após traduzida para planejamento a execução da ação que representa esta transição gera como efeito de início $\neg p$, ou seja, ambos os lugares não possuem marcas, o que vai contra o conceito de lugar complementar. No entanto, quando a transição se enquadra na situação citada acima então a tradução do lugar complementar que pertence a $t\bullet$ não deve ser atribuído à restrição *at end* dos efeitos da ação, mas sim a restrição *at start*. Com esta pequena mudança este problema é corrigido.

A tradução das transições temporizadas com disparo instantâneo é feita do mesmo modo como apresentado para as transições temporais com disparo instantâneo, isto porque ambas utilizam a mesma linguagem, a L_s^s ou a L_e^e, para as suas traduções.

5.1.3 Tradução de transições temporais para verificação de bloqueios

A tradução de transições temporais para verificação de bloqueios em RdP com ou sem tempo pode ser feita da mesma maneira, pois a única diferença é que as redes temporais podem conter um número menor de estados com bloqueios. Este fato ocorre porque nas redes temporais as transições habilitadas em situação de conflito podem nunca ser disparadas caso exista alguma transição onde seu limite inferior seja maior que o limite superior de outras transições que estão em conflito. Na Figura 5.1 a prevenção do bloqueio ocorre porque a transição t_2 sempre irá disparar antes da transição t_3. Caso o tempo de t_3 seja [0,1] então a rede passa a contar com um bloqueio. No entanto, transições que estão em conflito e que nunca poderão disparar são descartadas da tradução, no caso do exemplo citado, descarta-se a transição t_3 pois o valor α de t_3 é maior que o valor de β de t_3.

No entanto, a tradução de transições temporais não deve representar informações temporais, já que a verificação de bloqueios é feita somente sobre a estrutura da rede. Isto implica na utilização de planejadores clássicos, que são mais simples e mais eficientes. Sendo assim, a listagem 5.5 apresenta a tradução da transição t_1 da figura 5.1 na qual as informações temporais são ignoradas.

```
(:action t1
  :parameters ()
  :precondition (and
      (marked p1)
      (marked p2)
      )
  :effect ( and
      (not (marked p1))
      (not (marked p2))
      (marked p3)
    )
)
```

Listagem 5.5: Exemplo de tradução para a transição t_1 da Figura 5.1.

5.1.4 Descrição do problema de alcançabilidade

Como um problema de alcançabilidade em RdP é teoricamente equivalente a um problema de planejamento [1] então sua tradução torna-se bastante simples. Todos os lugares marcados, que representam o estado inicial da RdP, são descritos através do predicado `(marked ?p)` como o estado inicial do problema de planejamento. O mesmo ocorre com a descrição do estado objetivo. Esta tradução somente pode ser aplicada às modelagens referentes às RdP temporais e temporizadas, pois o arquivo de problema para verificação de bloqueio possui uma descrição diferenciada, assim como apresentado nos próximos parágrafos. É importante salientar que as informações temporais não devem ser mencionadas na descrição do arquivo de problema. A listagem 5.6 apresenta um exemplo de tradução do problema de alcançabilidade para planejamento, onde a marcação inicial da rede é p3 e o estado objetivo contém p1 e p2.

```
1 (define (problem TestProb)
    (:domain Test)
    (:init (marked p3) )
    (:goal (and (marked p1) (marked p2)) )
5 )
```

Listagem 5.6: Exemplo do arquivo de problema para RdP temporais e temporizadas 1-limitadas.

A verificação de bloqueios em RdP também pode ser encarada como um problema de alcançabilidade através de alguma condição que torne todas as transições desabilitadas. Uma destas condições pode ser modelada através da verificação da ausência de marca para algum lugar do pré-conjunto da transição. Se a condição for satisfeita para todas as transições da rede, então a rede está em bloqueio.

A tradução do problema de alcançabilidade para verificação de bloqueios em RdP é feita através da descrição do estado objetivo, que é composto por uma conjunção de disjunções. Cada disjunção representa a condição de desabilitação de uma transição. A listagem 5.7 apresenta o arquivo de problema para a rede da figura 5.1, onde as linhas 5 e 6 são a condição de bloqueio da transição t_1 e a linha 7 é a condição de bloqueio para t_2. Note que a condição de bloqueio da transição t_3 é a mesma da t_2 mas, caso fosse diferente, então t_3 não deveria ter sua condição de bloqueio no estado objetivo, pois é uma

transição que foi rejeitada na descrição do arquivo de domínio devido ao fato de nunca poder disparar, tal como apresentado na seção 5.1.3.

```
(define (problem TestProb)
  (:domain Test)
  (:init (marked p3) )
  (:goal (and
           ( or (not(marked p2))
                (not(marked p1))  )
           (not(marked p3))
         )
  )
)
```

Listagem 5.7: Tradução do estado objetivo para verificação de bloqueios.

5.1.5 Estendendo a representação das traduções

Para as traduções citadas nas seções anteriores existem limitações quanto às propriedades da RdP a serem utilizadas. É possível observar que o predicado `(marked ?p)` apenas pode informar se o lugar possui ou não marca, o que caracteriza uma RdP 1-limitada. No entanto, esta limitação pode ser descartada se o predicado `(marked ?p)` for substituído por um predicado numérico que armazena o número de marcas de cada lugar, tal como `(number-of-tokens ?p)`, que é declarado dentro do escopo `:functions`[1], assim como ocorre na declaração de predicados.

No entanto, como um predicado numérico somente pode atuar sobre operadores aritméticos, então as condições da ação devem ser descritas por uma desigualdade para cada lugar do pré-conjunto da transição. Esta inequação tem como função comparar a quantidade de marca dos lugares com o peso de seus respectivos arcos. Já na descrição dos efeitos, a geração e o consumo de marcas devem ser feitas através dos operadores `increase` e `decrease` conforme o peso dos arcos de entrada e saída da transição. A listagem 5.8 apresenta um exemplo de utilização dos predicados numéricos na tradução da transição t_1 da figura 5.1. Ainda, é importe lembrar que o uso de predicados numéricos exige a adição da declaração `:fluents` nos requerimentos do arquivo de domínio.

[1]Para obter maiores detalhes sobre a especificação gramatical da linguagem PDDL consulte Fox e Long [24].

```
(:durative-action t1
  :parameters ()
  :duration (= ?duration 1)
  :condition (and (at start (>= (number-of-tokens p2) 1) )
                  (over all (>= (number-of-tokens p2) 1) )
                  (at end (>= (number-of-tokens p2) 1) )
                  (at start (>= (number-of-tokens p1) 1) )
                  (over all (>= (number-of-tokens p1) 1) )
                  (at end (>= (number-of-tokens p1) 1) )
  )
  :effect ( and
      (at end (decrease (number-of-tokens p2) 1 ) )
      (at end (decrease (number-of-tokens p1) 1 ) )
      (at end (increase (number-of-tokens p3) 1 ) )
    )
)
```

Listagem 5.8: Exemplo de uso de predicados numéricos para RdP temporal k-limitada.

Com o uso de predicados numéricos a descrição do estado inicial e objetivo do problema de planejamento devem ser modificados, de forma a tornar-se compatível com o uso dos recursos numéricos. Para isso, no estado inicial, todos os lugares devem ser declarados para informar a quantidade de marcas que eles contêm, independente de possuírem ou não marcas em sua marcação inicial. Para o estado objetivo são declarados somente os lugares e a quantidade de marcas que cada um deverá possuir. A listagem 5.9 apresenta o mesmo arquivo de problema da listagem 5.7, porém, de forma numérica.

```
(define (problem TestProb)
  (:domain Test)
  (:init
    (= (number-of-tokens p1) 0)
    (= (number-of-tokens p2) 0)
    (= (number-of-tokens p3) 1)
  )
  (:goal (and
    (= (number-of-tokens p1) 1)
    (= (number-of-tokens p2) 1)
  )
)
```

Listagem 5.9: Exemplo do estado inicial e objetivo para modelagem de RdP k-limitadas.

A situação de contato também pode ser controlada através dos predicados numéricos. Para isto basta adicionar uma nova condição para cada lugar do pós-conjunto da transição. Estas novas condições verificarão, no instante inicial da ação, se cada lugar pertencente ao pós-conjunto da transição não excederá sua capacidade após o disparo da transição.

A condição pode ser descrita em PDDL por: `(at start (>= K (+ (number-of-tokens ?p) W ) ) )`, onde `K` e `W` são, respectivamente, o valor que representa a capacidade do lugar pertencente ao pós-conjunto da transição e o peso do arco que liga a transição a este lugar. Para exemplificar o seu uso, a rede da figura 5.4 é traduzida. Observe que na listagem 5.10 a descrição da condição de execução da ação possui, além das condições de disparo da transição (linhas 4 e 5), uma condição adicional (linha 6) que verifica a situação de contato, isto é, impede a execução da ação caso a soma da marcação atual de p3 com o peso do arco que adiciona marcas neste lugar (W=2) seja maior que a sua capacidade (k=3). Para a condição da ação não ser satisfeita basta a que a marcação inicial do lugar p3 seja maior que dois.

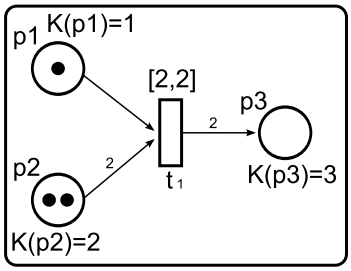

Figura 5.4: Exemplo de rede de Petri temporal k-limitada.

```
1   (:durative-action t1
      :parameters ()
      :duration (= ?duration 2)
      :condition (and (at start (>= (number-of-tokens p2) 2) )
5                     (over all (>= (number-of-tokens p2) 2) )
                      (at end (>= (number-of-tokens p2) 2) )
                      (at start (>= (number-of-tokens p1) 1) )
                      (over all (>= (number-of-tokens p1) 1) )
                      (at end (>= (number-of-tokens p1) 1) )
10                    (at start (>= 3 (+ (number-of-tokens p3) 2) ) )
      )
      :effect ( and
          (at end (decrease (number-of-tokens p2) 2 ) )
          (at end (decrease (number-of-tokens p1) 1 ) )
15        (at end (increase (number-of-tokens p3) 2 ) )
        )
    )
```

Listagem 5.10: Exemplo de uso de predicados numéricos para uma RdP k-limitada, com peso nos arcos maior que 1 e com controle da situação de contato.

Apesar dos exemplos de extensões apresentados anteriormente terem sido aplicados somente às RdP temporais, estas extensões também podem ser utilizadas por RdP tem-

porizadas. Para isso, durante a tradução, basta utilizar as restrições temporais adequadas para este tipo de RdP, conforme já apresentado na seção 5.1.2.

5.2 Tradução de planejamento temporal para RdP temporais

Sabendo dos potenciais benefícios que a tradução de um problema para outro pode trazer, Silva [1] e Hickmott et al. [2] propuseram a tradução de problemas de planejamento para RdP. Como a RdP é um modelo com uma vasta fundamentação teórica que pode contribuir na análise de sistemas temporais, então o presente trabalho se propõe a estender o trabalho de Silva e estabelecer mais um relacionamento para a tradução de um problema de planejamento temporal em PDDL para um problema de alcançabilidade em redes de Petri temporal.

O processo de transformar um problema de planejamento temporal em uma rede de Petri temporal é mais complexo, pois uma transição temporal não possui a capacidade de modelar as diversas linguagens que a PDDL pode derivar, assim como apresentado pelo teorema 1. No entanto, esta tradução pode ser dividida em três partes distintas: a definição da estrutura base da RdP que representa uma ação, a eliminação de literais negativos e a remoção das situações de contato, conforme apresentado nas seções 5.2.1, 5.2.2 e 5.2.3. Nas seções subsequentes são apresentados os procedimentos para descrever um problema de alcançabilidade sobre a RdP traduzida. Posteriormente, é apresentado como a RdP obtida pelo processo de tradução pode ser simplificada e, por fim, são apontadas as possíveis restrições da RdP gerada. Cabe frisar que a tradução apresentada nesta seção é aplicada somente a ações determinísticas, isto é, ações representadas por um intervalo de tempo $[\alpha, \beta]$ tal que $\alpha = \beta$.

5.2.1 Representando uma ação durativa em RdP temporais

A representação de uma ação durativa através de uma única transição temporal não é possível, pois uma transição temporal não possui expressividade suficiente para representar todas as restrições temporais dos problemas de planejamento, conforme comprovado

pelo teorema 1. Sendo assim, a tradução de uma ação deve ser feita por um conjunto de transições capazes de verificar todas as restrições temporais aplicáveis a uma ação, de modo a simular a execução de uma ação. A definição do conjunto de transições que representam uma ação é parcialmente inspirada no trabalho de Halsey et al. [14], porém em RdP. Segundo este autor uma ação durativa pode ser fragmentada em 3 ações instantâneas, uma para cada tipo de restrição temporal (*at start, over all, at end*), o que permite utilizar planejadores clássicos para encontrar um plano parcialmente ordenado e posteriormente utilizar um escalonador das informações temporais para produzir a solução final do problema.

Utilizando o mesmo princípio abordado por Halsey et al. [14], de tratar cada restrição temporal individualmente, foi desenvolvido uma RdP capaz de representar o comportamento de uma ação durativa. No entanto, a estrutura básica da RdP que traduz uma ação durativa é composta por uma sequência cíclica de 4 transições para o controle das restrições temporais juntamente com mais 3 transições para o controle do tempo sobre cada uma destas restrições, conforme ilustrado na figura 5.5.

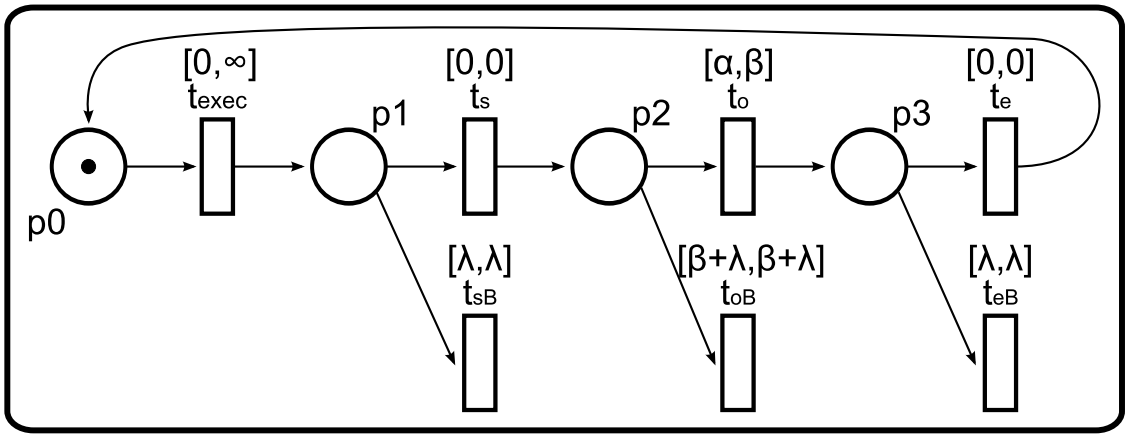

Figura 5.5: RdP temporal em alto nível representando a tradução de uma ação do problema de planejamento.

A função de cada transição sobre o modelo proposto é apresentada nos itens a seguir, onde é utilizado o símbolo grego λ (*lambda*) para denotar uma pequena fração de tempo:

- A transição t_{exec} tem por finalidade retardar o disparo das transições subsequentes, ou seja, não obriga uma ação disparar quando tiver suas pré-condições satisfeitas.

Sendo assim esta transição tem associado o intervalo de tempo [0,∞]. O lugar que antecede esta transição determina se a ação está em processo de execução, ou seja, se estiver marcada então a ação não está em execução;

- Sobre a transição t_s são associados todos predicados pertencentes às condições e efeitos que devem ocorrer no instante *at start* da ação. Sua duração sempre deve ser instantânea, ou seja, com intervalo de tempo [0,0];

- Na transição t_o são associados todos os predicados que não deverão sofrer mudança de estado durante a execução da ação, ou seja, todas as invariantes, que em PDDL são descritas pela restrição temporal *over all*. Esta transição deve possuir o intervalo de tempo $[\alpha, \beta]$ que corresponde ao tempo de execução da ação;

- A transição t_e, também instantânea, possui características semelhantes à transição $\mathtt{t_s}$, a não ser pelo fato de manter as condições e efeitos associados à restrição *at end*.

- As transições t_{sB}, t_{oB} e t_{eB} controlam o tempo sobre as transições $\mathtt{t_s}, \mathtt{t_o}$ e $\mathtt{t_e}$, respectivamente. Estas transições impedem que se encontre uma sequência inválida de disparos para o problema de alcançabilidade, pois elas bloqueiam a execução da ação caso ocorra um comportamento impróprio na simulação da execução da ação. Maiores detalhes sobre a função das transições $\mathtt{t_{sB}}, \mathtt{t_{oB}}$ e $\mathtt{t_{eB}}$ serão discutidos na seção 5.2.4, que trata de apresentar como um problema de alcançabilidade deve ser definido sobre o problema de planejamento temporal traduzido.

O processo de transformar uma ação durativa em uma RdP temporal é iniciado através da tradução de cada literal das condições e dos efeitos desta ação por um lugar. Estes lugares são interligados por arcos até a transição referente à restrição temporal de cada literal traduzido. O literal representado pelo lugar é verdadeiro se ele possui marca, caso contrário ele é falso. Utilizando esta argumentação para traduzir uma ação durativa para uma RdP temporal e tomando como exemplo a ação A1 da listagem 5.11, então a RdP resultante é a da figura 5.6. Note que todos os literais da restrição *at start* foram associados a transição $\mathtt{t_s}$ enquanto o literal Q foi associado a transição $\mathtt{t_e}$ pois sua restrição temporal

é *at end*. Apesar disso, a rede apresentada não possui o mesmo comportamento da ação A1, pois esta tradução possui dois tipos de inconsistências que são ocasionadas pelo uso de literais negativos e por possíveis situações de contato que podem ocorrer na rede. Os métodos para eliminar tais inconsistências são apresentadas, respectivamente, pelas seções 5.2.2 e 5.2.3.

```
(define (domain tradDom)
  (:requirements :strips :durative-actions)
  (:constants  A B C D E Q)
  (:predicates  (marked ?p))

  (:durative-action A1
    :duration (= ?duration 4)
    :condition (and (at start (A))
                    (at start (not (B)))
                    (at start (C)) )
    :effect (and (at start (not(A)))
                 (at start (B))
                 (at start (D))
                 (at start (not(E)))
                 (at end (Q)) )
  )

  (:durative-action A2
    :duration (= ?duration 0)
    :condition (and (at start (A))
                    (at start (B)) )
    :effect (and (at start (not(B))))
  )
)
```

Listagem 5.11: Exemplo de ações durativas em PDDL.

```
(define (problem TradProb)
  (:domain tradDom)
  (:init (marked A)
         (marked B)
         (marked C)
         (not(marked E)) )
  (:goal (and
          (marked D)
        )
  )
)
```

Listagem 5.12: Exemplo de um problema de planejamento para a listagem 5.11.

Os exemplos das próximas seções também utilizarão o domínio e o problema de planejamento apresentados pelas listagens 5.11 e 5.12.

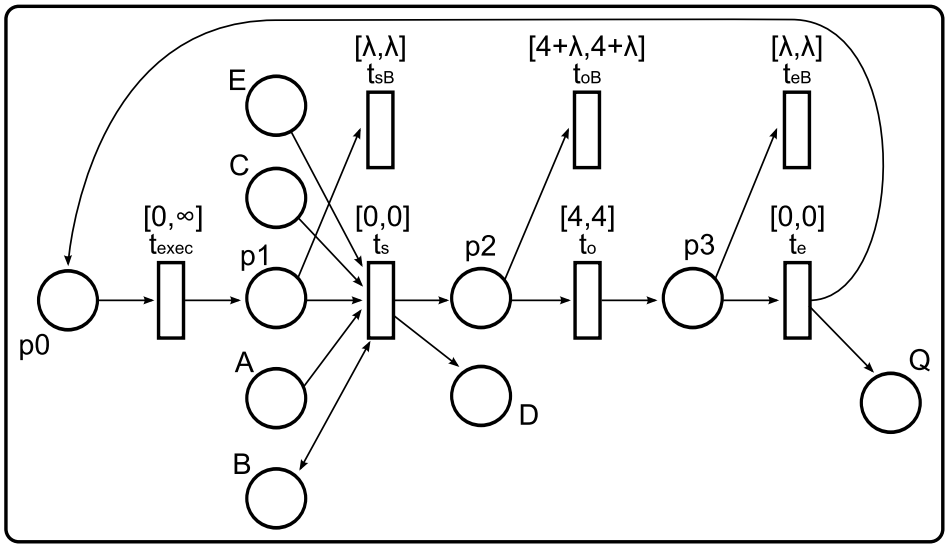

Figura 5.6: Problemas da tradução ocasionados por literais negativos.

5.2.2 Eliminando literais negativos

O problema ocasionado pelo uso de literais negativos é dado por dois motivos. O primeiro acontece quanto os literais aparecem como condições da ação e que, quando traduzida para RdP, impedem que a transição seja habilitada. Isto ocorre porque para representar o comportamento da ação é exigido que o lugar, que representa tal literal, não possua marcas para o disparo da transição, o que é contrário à própria condição de disparo de uma transição temporal. A figura 5.6 ilustra esta situação, onde a marcação $\mathtt{M(C)} = \mathtt{M(p1)} = \mathtt{M(A)} = 1, \mathtt{M(B)} = 0$ representaria a condição de disparo da transição $\mathtt{t_s}$ conforme o comportamento da ação traduzida, mas que em RdP não possui o mesmo comportamento pois se o lugar B não possuir marca então a transição não pode disparar.

O segundo problema ocorre quando é feita a tradução dos literais negativos presentes nos efeitos, o que acarreta a criação de pré-condições indesejadas caso este literal não pertença às condições da ação. Este fato ocorre porque o consumo de marcas de um lugar apenas pode ser feito por transições e este ato acaba estabelecendo uma nova condição para a transição. A figura 5.6 exemplifica esta situação, onde o lugar E deveria perder a marcação com o disparo da transição, mas que acaba representando um comportamento diferente da ação pois o lugar que representa o literal E deixa de pertencer aos efeitos da ação e passa a pertencer às pré-condições.

Ambos os problemas citados podem ser facilmente resolvidos. Para isso, além de associar os lugares, que representam os literais positivos das condições e dos efeitos da ação, às transições referentes as suas restrições temporais, os literais negativos também devem ser representados por lugares que representam a sua negação, ou seja, quando este lugar estiver marcado então quer dizer que é verdade que o literal está negado, tal que $\neg\mathtt{p} = \widehat{\mathtt{p}}$ e $\widehat{\neg\mathtt{p}} = \mathtt{p}$. Sendo assim, para uma ação durativa $\mathtt{o} = \langle \mathtt{Cond_s}, \mathtt{Cond_o}, \mathtt{Cond_e}, \mathtt{Eff_s}, \mathtt{Eff_e}, [\alpha, \beta] \rangle$ a sua tradução é dada por:

$$\mathtt{P} = \{\mathtt{l}, \widehat{\mathtt{l}} \mid \mathtt{l} \in \mathtt{L}\} \cup \{\mathtt{l}, \widehat{\mathtt{l}} \mid \neg\mathtt{l} \in \mathtt{L}\};$$

$$\mathtt{T} = \{\mathtt{t_{exec}}, \mathtt{t_s}, \mathtt{t_{sB}}, \mathtt{t_o}, \mathtt{t_{oB}}, \mathtt{t_e}, \mathtt{t_{eB}}\};$$

$$\mathtt{Z}(\mathtt{t_{exec}}) = [0, \infty];$$

$$\mathtt{Z}(\mathtt{t_s}) = \mathtt{Z}(\mathtt{t_e}) = [0, 0];$$

$$\mathtt{Z}(\mathtt{t_o}) = [\alpha, \beta];$$

$$\mathtt{Z}(\mathtt{t_{sB}}) = \mathtt{Z}(\mathtt{t_{eB}}) = [\lambda, \lambda];$$

$$\mathtt{Z}(\mathtt{t_{oB}}) = [\beta + \lambda, \beta + \lambda];$$

$$\mathtt{F} = \cup \begin{cases} \bullet\mathtt{t_{exec}} = \{\mathtt{p0}\} \\ \mathtt{t_{exec}}\bullet = \bullet\mathtt{t_{sB}} = \{\mathtt{p1}\} \\ \bullet\mathtt{t_s} = (\mathtt{Cond_s} \cap \mathtt{V}) \cup \{\widehat{\mathtt{l}} \mid \neg\mathtt{l} \in \mathtt{Cond_s}\} \cup \{\mathtt{p1}\} \\ \mathtt{t_s}\bullet = (\mathtt{Eff_s} \cap \mathtt{V}) \cup \{\widehat{\mathtt{l}} \mid \neg\mathtt{l} \in \mathtt{Eff_s}\} \cup \{\mathtt{p2}\} \\ \bullet\mathtt{t_o} = (\mathtt{Cond_o} \cap \mathtt{V}) \cup \{\widehat{\mathtt{l}} \mid \neg\mathtt{l} \in \mathtt{Cond_o}\} \cup \{\mathtt{p2}\} \\ \mathtt{t_o}\bullet = \bullet\mathtt{t_{eB}} = \{\mathtt{p3}\} \\ \bullet\mathtt{t_{oB}} = \{\mathtt{p2}\} \\ \bullet\mathtt{t_e} = (\mathtt{Cond_e} \cap \mathtt{V}) \cup \{\widehat{\mathtt{l}} \mid \neg\mathtt{l} \in \mathtt{Cond_e}\} \cup \{\mathtt{p3}\} \\ \mathtt{t_e}\bullet = (\mathtt{Eff_e} \cap \mathtt{V}) \cup \{\widehat{\mathtt{l}} \mid \neg\mathtt{l} \in \mathtt{Eff_e}\} \cup \{\mathtt{p1}\} \end{cases}$$

onde $\mathtt{V}$ é conjunto de literais positivos e $\mathtt{L}$ é o conjunto de literais positivos e negativos de um problema de planejamento, assim como apresentado pela definição de literais (definição 1, da seção 2.1).

As soluções para a eliminação de literais negativos da figura 5.6 é apresentado na figura 5.7, onde cada literal das condições e efeitos é representado por dois lugares, um complementar do outro.

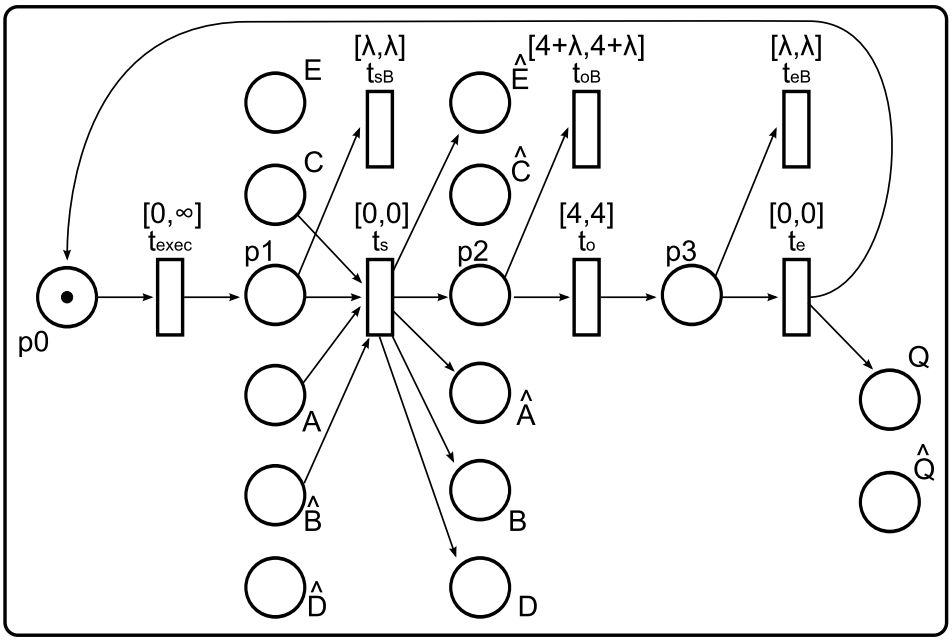

Figura 5.7: Tradução dos literais negativos de uma ação para uma RdP temporal.

5.2.3 Evitando situações de contato

O próximo passo da tradução é eliminar as eventuais situações de contato que podem ocorrer na RdP. Para isso é utilizada a abordagem apresentada por Hickmott et al. [2], onde a situação de contato pode ser verificada sempre que existir uma transição que não é segura, ou seja, existem lugares no pós-conjunto da transição que receberão marcas com o disparo da transição mas que não consumirão marcas de seus lugares complementares. Um exemplo de transição não segura é apresentado na figura 5.7, onde a situação de contato pode ocorrer sobre os lugares D e $\widehat{E}$ caso algum destes lugares já estejam marcados antes do disparo da transição t_s.

Definição 45 (Transição Segura)**.** *Dado o conjunto*

$$\mathtt{S(t)} = \mathtt{t}\bullet \setminus \{\mathtt{p} \in \mathtt{t}\bullet \mid \widehat{\mathtt{p}} \in \bullet\mathtt{t}\} \setminus \{\mathtt{p0}, \mathtt{p1}, \mathtt{p2}, \mathtt{p3}\} \quad [2]$$

[2]O operador $\setminus$ é a diferença entre conjuntos, tal que, para dados dois conjuntos A e B, a operação $A \setminus B$ e definida por {x ∈ A | x ∉ B}. Ex.: {a,b,c} $\setminus$ {b,c,d} = {a}.

então uma transição $t \in T$ é dita segura se, e somente se, $\mathtt{S(t)} = \emptyset$*, onde o conjunto* $\{\mathtt{p0}, \mathtt{p1}, \mathtt{p2}, \mathtt{p3}\}$ *são os lugares da estrutura básica da RdP que representam uma ação.*

Para solucionar o problema da situação de contato, cada transição não segura deve ser transformada em um conjunto de transições seguras capaz de manter o comportamento original da transição. Para isso, o número de transições que irão substituir a transição não segura é dado por $2^{|\mathtt{S(t)}|}$. Definindo $t\diamond$ como sendo o conjunto de todos os subconjuntos de $\mathtt{S(t)}$, que na teoria dos conjuntos também é chamado de *conjunto das partes*, então as novas transições são dadas pelo algoritmo abaixo:

```
1 Função EliminaContatos(t,t◇)
2   i=1;
3   para ∀x ∈ t◇ faça
4     •t_i = •t ∪ {p̂ | p ∈ x} ∪ (S(t)\ x);
5     t_i• = t• ∪ ((•t\ {p̂ | p ∈ t•}) ∩ {p0,p1,p2,p3});
6     Z(t_i) = [0,0];
7     i++;
8   fim para
9 fim EliminaContatos(t,t◇);
```

Listagem 5.13: Algoritmo para eliminação de contatos de uma RdP.

A tabela abaixo apresenta um exemplo do uso do algoritmo da listagem 5.13 aplicado sobre a transição $\mathtt{t_s}$ da figura 5.7. Sabendo que $\mathtt{S(t_s)} = \{\mathtt{D}, \widehat{\mathtt{E}}\}$ então as novas transições que a substituem $\mathtt{t_s}$ são:

$\bullet\mathtt{t_s} = \{\mathtt{A}, \widehat{\mathtt{B}}, \mathtt{C}, \mathtt{p1}\}$	$\mathtt{t_s}\bullet = \{\widehat{\mathtt{A}}, \mathtt{B}, \mathtt{D}, \widehat{\mathtt{E}}, \mathtt{p2}\}$	$\mathtt{t_s}\diamond = \{\emptyset, \{\mathtt{D}\}, \{\widehat{\mathtt{E}}\}, \{\mathtt{D}, \widehat{\mathtt{E}}\}\}$
$\bullet\mathtt{t_{s1}} = \{\mathtt{A}, \widehat{\mathtt{B}}, \mathtt{C}, \mathtt{D}, \widehat{\mathtt{E}}, \mathtt{p1}\}$	$\mathtt{t_{s1}}\bullet = \{\widehat{\mathtt{A}}, \mathtt{B}, \mathtt{D}, \widehat{\mathtt{E}}, \mathtt{C}, \mathtt{p2}\}$	$\mathtt{t_s}\diamond_1 = \emptyset$
$\bullet\mathtt{t_{s2}} = \{\mathtt{A}, \widehat{\mathtt{B}}, \mathtt{C}, \widehat{\mathtt{D}}, \widehat{\mathtt{E}}, \mathtt{p1}\}$	$\mathtt{t_{s2}}\bullet = \{\widehat{\mathtt{A}}, \mathtt{B}, \mathtt{D}, \widehat{\mathtt{E}}, \mathtt{C}, \mathtt{p2}\}$	$\mathtt{t_s}\diamond_2 = \{\mathtt{D}\}$
$\bullet\mathtt{t_{s3}} = \{\mathtt{A}, \widehat{\mathtt{B}}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{p1}\}$	$\mathtt{t_{s3}}\bullet = \{\widehat{\mathtt{A}}, \mathtt{B}, \mathtt{D}, \widehat{\mathtt{E}}, \mathtt{C}, \mathtt{p2}\}$	$\mathtt{t_s}\diamond_3 = \{\widehat{\mathtt{E}}\}$
$\bullet\mathtt{t_{s4}} = \{\mathtt{A}, \widehat{\mathtt{B}}, \mathtt{C}, \widehat{\mathtt{D}}, \mathtt{E}, \mathtt{p1}\}$	$\mathtt{t_{s4}}\bullet = \{\widehat{\mathtt{A}}, \mathtt{B}, \mathtt{D}, \widehat{\mathtt{E}}, \mathtt{C}, \mathtt{p2}\}$	$\mathtt{t_s}\diamond_4 = \{\mathtt{D}, \widehat{\mathtt{E}}\}$

Observado a estrutura da RdP que representa uma ação durativa é possível notar que as transições $\mathtt{t_{exec}}$ e $\mathtt{t_o}$ sempre serão seguras, pois os efeitos de uma ação nunca serão aplicados sobre estas transições. Também cabe observar que os lugares, que antes pertenciam ao pré-conjunto de uma transição mas não ao seu pós-conjunto, agora também pertencer seu pós-conjunto. Assim, é possível evitar o consumo indevido de marcas destes lugares após o disparo da transição, já que na ação os literais que representam estes

lugares nunca sofrem mudança de estado com a execução da ação pois eles não são efeito da mesma. Este fato ocorre na tabela acima, onde o lugar C, que antes pertencia apenas ao pré-conjunto de $\mathtt{t_s}$, agora também pertence pós-conjunto de $\mathtt{t_s}$.

A transição $\mathtt{t_e}$ também não é segura, pois $\mathtt{S}(\mathtt{t_s}) = \{\mathtt{Q}\}$. No entanto o mesmo algoritmo também deve ser aplicado a esta transição, gerando assim as transições da tabela abaixo.

$\bullet\mathtt{t_e} = \{\mathtt{p3}\}$	$\mathtt{t_e}\bullet = \{\mathtt{Q}, \mathtt{p0}\}$	$\mathtt{t_e}\diamond = \{\emptyset, \{\mathtt{Q}\}\}$
$\bullet\mathtt{t_{e1}} = \{\mathtt{Q}, \mathtt{p3}\}$	$\mathtt{t_{e1}}\bullet = \{\mathtt{Q}, \mathtt{p0}\}$	$\mathtt{t_e}\diamond_1 = \emptyset$
$\bullet\mathtt{t_{e2}} = \{\widehat{\mathtt{Q}}, \mathtt{p3}\}$	$\mathtt{t_{e2}}\bullet = \{\mathtt{Q}, \mathtt{p0}\}$	$\mathtt{t_e}\diamond_2 = \{\mathtt{Q}\}$

O fim do processo de tradução de uma ação durativa para uma RdP temporal é dado quando todas as transições da rede forem seguras. Portanto, a RdP temporal que representa a ação A1 da listagem 5.11 é ilustrada pela figura 5.8.

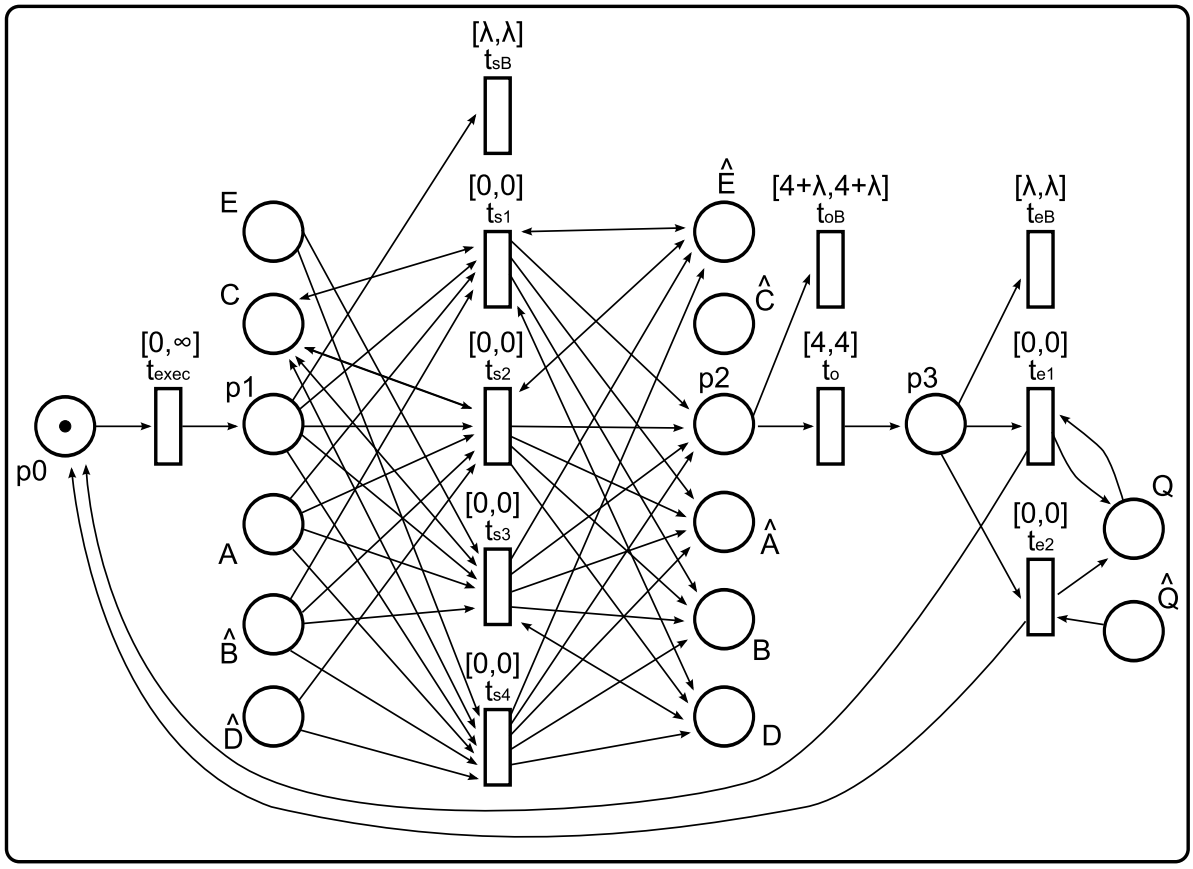

Figura 5.8: RdP com transições seguras.

5.2.4 Definindo o problema de alcançabilidade

O último passo da tradução de um problema de planejamento para RdP temporal é definir a marcação inicial e os estado objetivo da rede que permitem verificar a sua propriedade de alcançabilidade. Dado que um problema de planejamento é uma tripla $\langle \mathtt{O}, \mathtt{I}, \mathtt{G} \rangle$, então a marcação inicial da rede é comporta pelos lugares que representam os literais do conjunto $\mathtt{I}$, sabendo que $\neg l = \widehat{l}$. Literais que não estão declarados no estado inicial do problema de planejamento são considerados falsos pelos planejadores. Sabendo disso, então os lugares que representam a negação dos literais não declarados no conjunto $\mathtt{I}$ também devem fazer parte da marcação inicial da RdP juntamente com todos os lugares que representam p0 na tradução de cada ação. Desta maneira, a marcação inicial da rede pode ser formalmente definida por:

$$\mathtt{M_0} = (\mathtt{I} \cap \mathtt{V}) \cup \{\widehat{\mathtt{l}} \mid \mathtt{l} \in (\mathtt{V} \setminus (\mathtt{I} \cap \mathtt{V})\} \cup \mathtt{P0}$$

tal que P0 é o conjuntos de todos os lugares p0 que representam a condição inicial de execução de cada ação.

Contudo, a tradução completa do domínio e do problema de planejamento das listagens 5.11 e 5.12 é apresentado na figura 5.9.

A tradução do estado objetivo do problema de planejamento é feita de maneira idêntica ao do estado inicial. No entanto, é importante observar que o conjunto $\mathtt{P0}$ também faz parte de $\mathtt{M}'$, pois é justamente este conjunto que impede que a marcação objetivo seja gerada pela execução parcial das ações, isto é, a execução de uma ação sempre deve ser dada pelo disparo das 4 transições que a representam $(\mathtt{t_{exec}}, \mathtt{t_s}, \mathtt{t_o}, \mathtt{t_e})$ para poder encontrar uma sequência de disparos equivalente ao plano de um problema de planejamento.

$$\mathtt{M}' = (\mathtt{G} \cap \mathtt{V}) \cup \{\widehat{\mathtt{l}} \mid \mathtt{l} \in (\mathtt{V} \setminus (\mathtt{G} \cap \mathtt{V})\} \cup \mathtt{P0}$$

O processo de busca que encontra uma sequência de disparos para resolver um problema de alcançabilidade deve ser interrompido sempre que a soma do tempo entre o instante da primeira habilitação e do disparo de uma transição, dentro do ciclo que re-

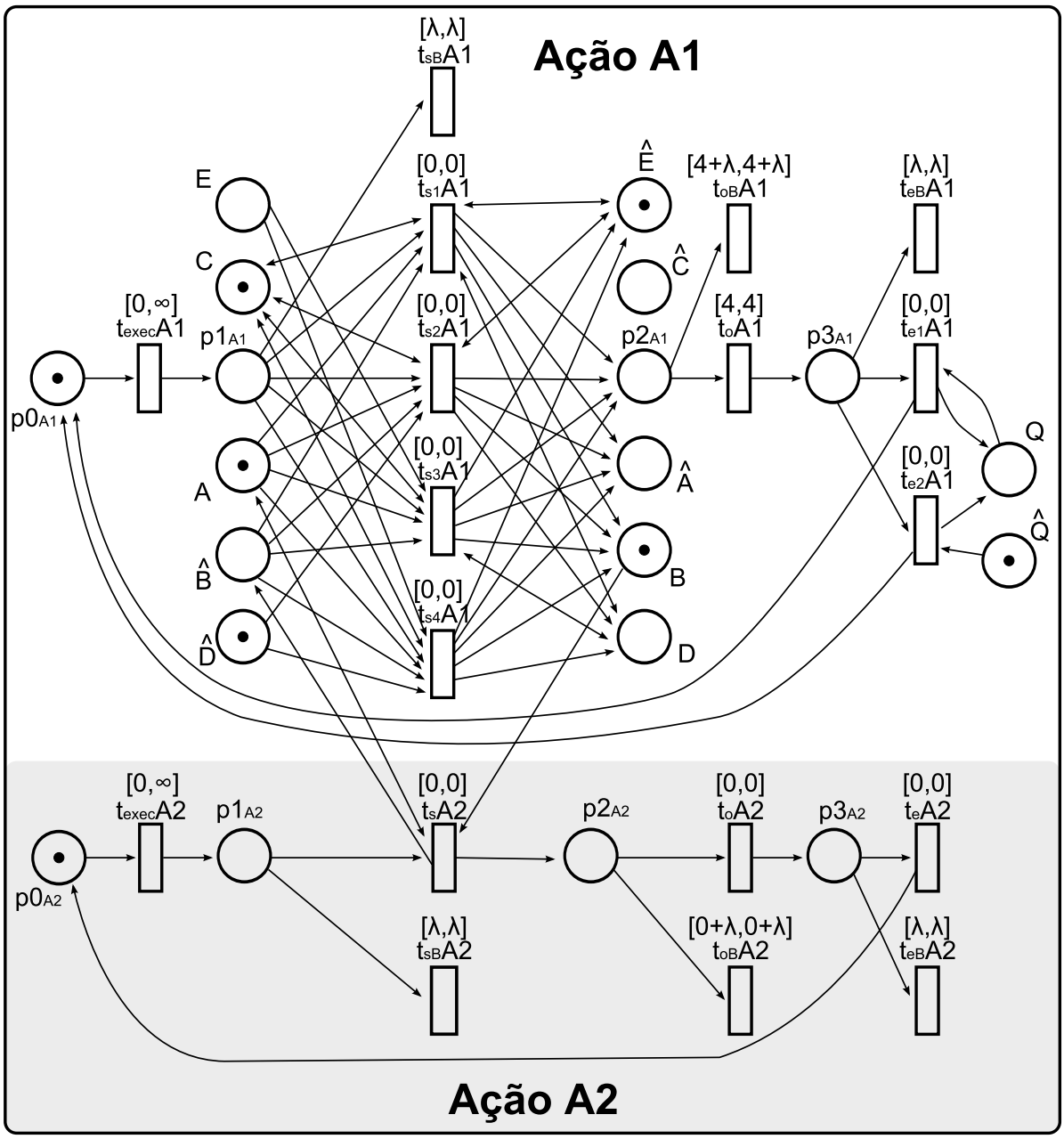

Figura 5.9: Tradução completa do domínio e problema de planejamento das listagens 5.11 e 5.12.

presenta a execução da ação, for maior que o próprio tempo indicado por esta transição. Um exemplo que ilustra esta situação pode ser observado na figura 5.10, onde duas ações concorrentes disputam a marca do lugar R para validar seus disparos. Neste exemplo, o tempo de execução da ação A1 pode ultrapassar o tempo real de sua execução, que é 4 unidades de tempo, se a transição $\texttt{t}_o\texttt{A1}$ permanecer habilitada por um tempo superior a 1 e, em seguida, a ação A2 começa a executar concorrentemente a A1. Sendo assim, o disparo da transição $\texttt{t}_s\texttt{A2}$ reinicia a contagem do tempo da transição $\texttt{t}_o\texttt{A1}$, ou seja, se

t_oA1 já estava habilitada a 1 unidade de tempo e seu relógio foi reiniciado pelo disparo de t_sA2 então o tempo para obter a execução completa desta ação será 5 unidades de tempo, o que não condiz com a ação descrita em planejamento.

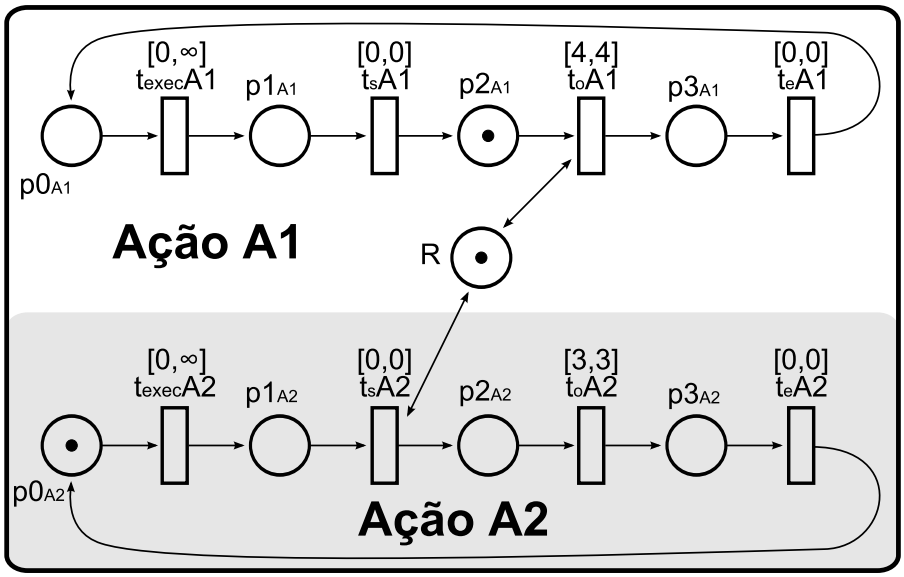

Figura 5.10: Exemplo de uma sequência de disparos inválida para a execução de uma ação.

Como uma transição não consegue controlar o tempo entre o instante da sua primeira habilitação e de seu disparo dentro do ciclo que representa a execução da ação, então é necessário adicionar uma transição extra para cada restrição temporal para efetuar este controle de tempo, ou seja, as transições t_{sB}, t_{oB} e t_{eB}. A ideia é fazer com que estas transições interrompam a execução da ação sempre que o tempo de permanência da marcação dos lugares $p1, p2$ ou $p3$ for superior ao tempo atribuído a transição que possui um destes lugares em seu pré-conjunto. O bloqueio da execução da ação sempre invalida a sequência de disparos para encontrar a solução do problema de alcançabilidade, já que o disparo de qualquer transição que controla o tempo sobre as restrições impede que o lugar $p0$ volte a possuir marca. Com o bloqueio da rede, o resolvedor do problema de alcançabilidade pode antecipar o processo de *backtracking* sobre uma sequência de disparos inconsistente, o que diminui a busca no espaço de estados.

No entanto, todos os estados da RdP que levam a solução de um problema de alcançabilidade sempre devem possuir o seguinte comportamento:

- A marcação dos lugares $\mathtt{p0_i}, \mathtt{p1_i}, \mathtt{p2_i}$ e $\mathtt{p3_i}$ deve ser mutualmente exclusiva, isto é, $\mathtt{M(p0_i)} + \mathtt{M(p1_i)} + \mathtt{M(p2_i)} + \mathtt{M(p3_i)} = \mathtt{1}$, onde $\mathtt{p0_i}$ corresponde ao lugar p0 da estrutura que representa a ação $\mathtt{i}$ em RdP.

- A marcação dos demais lugares da rede obedecem a definição 31 (lugar complementar, da seção 4.4), onde para todo $\mathtt{p} \in \mathtt{P}, \mathtt{M(p)} + \mathtt{M}(\widehat{\mathtt{p}}) = \mathtt{1}$, já que a rede é 1-limitada.

5.2.5 Restrições e observações

A tradução de um problema de planejamento para uma RdP temporal possui algumas restrições quanto a sua representação:

a) não é possível obter a solução de um problema de alcançabilidade que exija o conceito de concorrência requerida sobre uma mesma ação, pois a marcação dos lugares p0, p1, p2 e p3, que representam a estrutura cíclica da rede, é mutualmente exclusiva e portanto não pode simular a execução de uma mesma ação no mesmo instante de tempo;

b) muitas vezes, apesar da simulação da execução das ações em RdP prover uma sequência válida do disparo das transições, a ordem cronológica de execução pode ser incorreta. Este fato sempre ocorre quando é exigida que a execução de uma ação inicie ou termine após o início ou fim de outra ação. Dois exemplos deste problema podem ser verificados na figura 5.11, onde mais à esquerda é apresentado o plano defeituoso e à direita o plano correto. A sequência de execução das ações está ordenada de cima para baixo. Em todas as figuras a ordem de execução está correta, porém, na figura 5.11 a), o escalonamento do tempo não está correto, pois B deve iniciar sua execução após o fim de A e não ao mesmo tempo, assim como apresentado pelo quadro b). No entanto, para se obter um plano correto é necessário que haja uma pequena fatia de tempo entre o fim da ação A e o início de B, ilustrado pela figura 5.11 b) pela área hachurada. Semelhante ao anterior, o exemplo da figura 5.11 c) também possui problema na ordem cronológica de execução, porém a pequena fatia de tempo adicional deve estar entre o fim da ação A e o fim de B. Este problema pode ser facilmente resolvido com uso de

uma fase adicional de pós-processamento para inserir instantes de tempo entre início e fim tais ações;

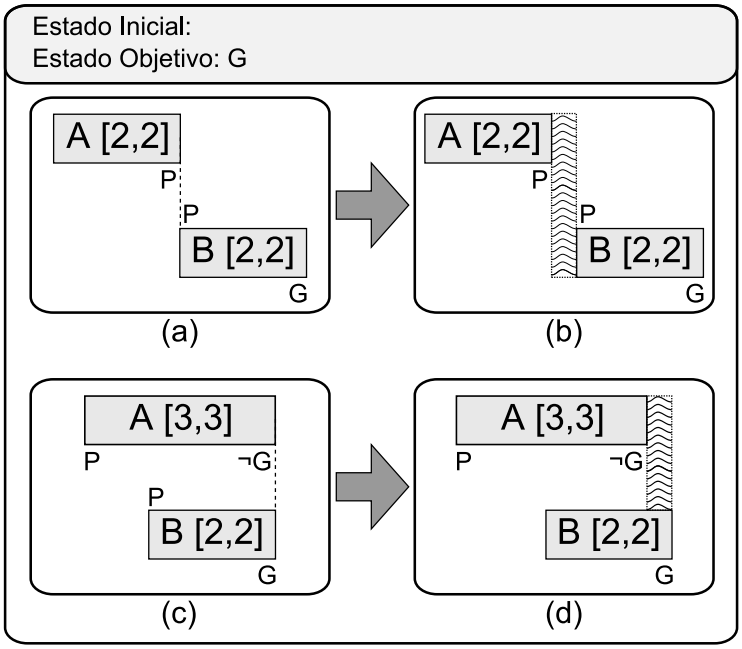

Figura 5.11: Possíveis erros provenientes da simulação da execução paralela entre ações em RdP temporais.

c) todas as transições t_s e t_e sempre possuem o intervalo de tempo igual a [0,0], pois caso contrário a verificação das condições e aplicação dos efeitos de uma ação não são efetivados no instante correto. Caso adicionado tempo na transição t_s então os efeito aplicáveis a esta restrição ocorrem após o início da execução ação. Caso adicionado tempo na transição t_e então as pós-condições da ação são verificadas antes do término da execução da mesma;

d) não é possível traduzir uma ação que faz uso de uma *função* (ver seção 2.3.1) para determinar a sua duração, pois em RdP a atribuição de tempos às transições não é dinâmica;

e) a tradução de uma ação durativa para uma RdP temporal apenas contempla problemas de planejamento que tenham como requerimento a linguagem STRIPS e pré-condições negativas. Demais condições da linguagem PDDL, tais como efeitos condicionais (*when*), quantificadores universais e existenciais, disjunções, predicados tipados e predicados numéricos, não são abordadas, pois o escopo principal deste trabalho é a

tradução das restrições temporais. Maiores detalhes sobre os requisitos da linguagem PDDL podem ser obtidos em Fox e Long [24];

f) quando uma ação durativa é traduzida para RdP temporais, então o tamanho do conjunto de transições que representa esta ação é igual à

$$\left(\sum_{\forall t \in \{t_{exec}, t_s, t_o, t_e\}} 2^{|S(t)|} + 1 \right) - 1 \tag{5.1}$$

onde $S(t)$ é o número de transições não seguras.

5.3 PMDC: Um modelo alternativo para verificação de bloqueios em redes de Petri

Edelkamp e Jabbar [4] apresentam um modelo em planejamento para o problema de verificação de bloqueio em RdP seguras que se sustenta em uma extensão da linguagem PDDL proposta por eles. Esta extensão exige modificações nos planejadores para o tratamento adequado dos arquivos de entrada e no mecanismo de busca para que tal extensão represente o ganho em desempenho esperado. Neste sentido nosso trabalho não depende de tais modificações e usa exclusivamente as estruturas nativas do PDDL.

O modelo proposto por Edelkamp e Jabbar é composto por 3 ações. Quando uma transição se encontra habilitada ela é tratada pela ação de disparo, que remove as marcas de $\bullet t$ e põe marcas em $t\bullet$. Com o disparo, as transições concorrentes são desabilitadas enquanto as transições que dependem de $t\bullet$ podem ser habilitadas. Entretanto, como a ação de disparo não passa ao planejador qualquer informação sobre a condição de habilitação das transições, torna-se necessária uma ação secundária, que desabilita qualquer transição que não satisfaz a condição de disparo. Ainda é necessária uma terceira ação que tem a função de verificar se todas as transições estão desabilitadas e, caso constatado, gerar a informação de que a rede está em bloqueio e que o objetivo foi alcançado.

A nova tradução proposta por esta seção, PMDC (*PDDL Model for DeadLock Checking*), é um modelo alternativo às traduções propostas por Edelkamp e Jabbar e por

aquelas apresentadas nas seções 5.1.3 e 5.1.4, já que ambas não obtiveram resultados satisfatórios quando aplicados ao planejador FF-Metric. O modelo PMDC visa simplificar a representação da estrutura da rede e, ao mesmo tempo, possibilita que a mesma seja estendida a partir redes seguras para também suportar redes k-limitadas e com pesos nos arcos maiores que 1. Como nas redes seguras cada lugar pode conter apenas uma marca, a condição de habilitação de uma transição pode ser descrita pela igualdade entre o número total de marcas de $\bullet t$ e o total de lugares do conjunto $\bullet t$. Aplicar esta idéia aos problemas de planejamento facilita a forma de computar os próximos estados da rede e também faz com que o único requisito seja trabalhar com operadores numéricos da linguagem PDDL.

A descrição do arquivo de domínio é iniciada com a declaração de cada lugar como uma constante. Também há a necessidade de adicionar uma variável numérica, nomeada `(number-of-tokens ?p)`, que tem como objetivo armazenar a quantidade de marcas de cada lugar. Para as redes seguras, os valores destas variáveis são 0 ou 1. Cada transição da rede é convertida em uma ação que descreve o comportamento de disparo da transição. As pré-condições da ação são descritas pela condição de habilitação da transição, ou seja, pela equação $\sum_{p \in \bullet t} M(p) = |\bullet t|$. A descrição dos efeitos é feita através dos operadores de incremento e decremento sobre o número de marcas de cada lugar, ou seja, decrementa-se todas as marcas pertencentes a $\bullet t$ e incrementa-se uma marca para cada lugar de $t\bullet$. O Código 5.14 apresenta a tradução da transição t_1 da Figura 5.1 para uma ação em PDDL que descreve o seu comportamento de disparo.

```
(:action Fire_T1
  :parameters ()
  :precondition (and ( = 2 (+(number-of-tokens P2)
                             (number-of-tokens P1) ) ) )
  )
  :effect (and   (decrease (number-of-tokens P2) 1 )
                 (decrease (number-of-tokens P1) 1 )
                 (increase (number-of-tokens P3) 1 )
  )
)
```

Listagem 5.14: Exemplo de tradução para a transição t_1 da Figura 5.1.

Quando a descrição da ação é feita desta maneira, não há necessidade de descrever os arcos da rede, já que cada ação que representa o disparo da transição, conhece quais

lugares deverão receber e perder marcas.

No estado inicial do problema de planejamento, cada lugar da RdP deve ser descrito pelo operador de igualdade sobre a variável numérica que define a quantidade de marcas existente no lugar: `(= (number-of-tokens P1) 1)`. Já a descrição do estado objetivo, que é a parte mais importante do modelo proposto, deve definir um estado em que todas as transições da rede estejam desabilitadas. Para isso, cada transição é descrita por uma restrição que define o seu estado de desabilitada. Quando todas as restrições forem satisfeitas então significa que foi encontrado o estado em que a RdP entra em bloqueio. O estado objetivo do problema de alcançabilidade é dado pela *inequação de bloqueio* apresentada por Melzer e Römer [48]:

$$\forall\, t \in T, \quad \sum_{p \in {}^\bullet t} M(p) < |{}^\bullet t| \tag{5.2}$$

O Código 5.15 exemplifica a descrição do objetivo para a RdP da Figura 5.1. É importante observar que, apesar da condição de bloqueio da transição t_3 ser a mesma de t_2, caso fosse diferente, t_3 não deveria ter sua condição de bloqueio no estado objetivo, pois é uma transição que nunca pode disparar.

```
1   (:goal (and  ( > 2 (+(number-of-tokens P2)
                          (number-of-tokens P1) )  )
                 ( > 1 (number-of-tokens P3) )
           )
5   )
```

Listagem 5.15: Exemplo de modelagem do estado objetivo para a Figura 5.1.

A verificação de bloqueios em RdPT também pode ser feita utilizando este mesmo modelo. No entanto, há duas formas distintas de efetuar esta tradução. Na primeira, a RdPT é traduzida para um problema clássico de planejamento onde cada transição deve ser transformada em um conjunto de 3 ações sem tempo, conforme apresentado por Halsey *et al.* [14]. Na segunda maneira, a tradução é feita através da análise da estrutura da rede onde são ignoradas todas as transições em conflito cujo limite inferior de disparo seja maior que o limite superior de outras transições e, posteriormente, as informações temporais também são ignoradas.

5.3.1 Resultados preliminares

Para a realização dos experimentos, o presente trabalho contou com a coleção de problemas apresentada por Corbett [49], que se caracteriza por modelar problemas de comunicação em máquinas de estado. Todas as RdP testas são seguras e possuem um grau elevado de estados concorrentes. Os resultados foram obtidos com a utilização do planejador Metric-FF [50]. Para fins comparativos, foram utilizados os resultados apresentados por Edelkamp e Jabbar [4], referidos aqui por "EJmodel", e os resultados obtidos com a execução da ferramenta *MCSMODELS 1.7* [51], que utiliza a técnica de desdobramento e é uma ferramentas normalmente utilizada pela comunidade de RdP. Os experimentos foram realizados em um computador com processador Intel Pentium 4 de 3.0 GHz e 512 MB de memória, executando o sistema operacional GNU-Linux. Os resultados de Edelkamp e Jabbar foram obtidos em um computador semelhante: Intel Pentium 4 de 3.2 GHz e 2 GB de memória.

Dentre 70 problemas testados, foram escolhidos 26 para serem apresentados neste trabalho. A Tabela 5.1 apresenta, para cada problema, a quantidade de lugares (P), a quantidade de transições (T) e se a RdP possui ou não bloqueios (DL: S/N). A seguir, é apresentado o tamanho do plano (Plano), a quantidade de estados explorados (Expl.) e o tempo (Tempo), em segundos, necessário para encontrar o bloqueio utilizando a abordagem PMDC, EJmodel e MCSMODELS, respectivamente. "–" indica que o problema não foi resolvido e "*" indica que nenhum bloqueio foi encontrado. Os tempos apresentados pelas ferramentas PMDC e MCSMODELS são a média de 5 execuções para cada problema enquanto o tempo de EJmodel foi extraído diretamente do trabalho de Edelkamp e Jabbar [4]. Testes com RdPT não foram efetuados, pois não foi encontrado nenhum corpo de prova específico para este tipo de rede.

Observando a Tabela 5.1 e analisando a qualidade do plano obtido é possível notar que, para os problemas analisados, o modelo PMDC obtém planos com tamanho menor ou igual aos planos gerados pelo EJmodel. Constatou-se um problema na análise realizada pela ferramenta MCSMODELS, pois esta não garante que o problema está livre de bloqueio, tal como pode ser evidenciado no problema Q(1).

Tabela 5.1: Resultados experimentais para detecção de bloqueios em RdP.

Problema				PMDC			EJmodel			MCSMODELS	
Nome(tam.)	P	T	DL	Plano	Expl.	Tempo	Plano	Expl.	Tempo	Plano	Tempo
DP(6)	36	24	S	6	13	0,004	6	11	0,080	6	0,002
DP(8)	48	32	S	8	17	0,012	8	15	0,080	8	0,002
DP(10)	60	40	S	10	21	0,016	10	19	0,080	10	0,004
DP(12)	72	48	S	12	25	0,020	12	23	0,080	12	0,005
DPH(4)	39	46	N	*	512	0,052	–	–	–	*	0,005
DPH(5)	48	67	N	*	3112	0,336	–	–	–	*	0,028
DPH(6)	57	92	N	*	18264	2,620	–	–	–	*	1,345
DPH(7)	66	121	N	*	104680	20,250	–	–	–	*	44,664
ELEV(1)	63	99	S	9	15	0,042	11	45	0,030	9	0,002
ELEV(2)	146	299	S	12	25	0,318	16	74	0,200	12	0,016
ELEV(3)	327	783	S	15	37	2,264	18	106	2,080	15	0,238
ELEV(4)	736	1939	S	18	51	13,088	–	–	–	18	6,838
HART(25)	127	77	S	26	28	0,040	26	27	0,110	26	0,002
HART(50)	252	152	S	51	53	0,132	51	52	0,280	51	0,006
HART(75)	377	227	S	76	78	0,328	76	77	0,710	76	0,011
HART(100)	502	302	S	101	103	0,682	101	102	1,450	101	0,020
MMGT(1)	50	58	S	6	23	0,024	–	–	–	6	0,001
MMGT(2)	86	114	S	8	32	0,056	–	–	–	8	0,010
MMGT(3)	122	172	S	10	45	0,110	10	15	0,130	10	0,582
MMGT(4)	158	232	S	12	62	0,212	12	24	0,200	12	109,134
Q(1)	163	194	S	21	362	0,240	21	258	0,250	*	0,042
SENT(25)	104	55	S	3	5	0,024	3	5	0,090	37	0,003
SENT(50)	179	80	S	3	5	0,040	3	5	0,100	62	0,002
SENT(75)	254	105	S	3	5	0,064	3	5	0,140	87	0,002
SENT(100)	329	130	S	3	5	0,098	3	5	0,180	112	0,002
SPD(1)	329	130	S	4	7	0,010	4	5	0,080	16	0,426

Quanto ao tempo de processamento, a ferramenta MCSMODELS quase sempre é mais rápida que os outros modelos, como também é possível observar nas Figuras 5.12(a), 5.12(b) e 5.12(c). Nos poucos casos em que nosso modelo é mais rápido que a MCS-MODELS a diferença de tempo é muito grande, como pode ser visto na Figura 5.12(d). Quando comparado ao EJmodel os tempos resultantes do PMDC são menores na maioria dos casos. Entre os 20 problemas avaliados por EJmodel em [4], o modelo PMDC foi mais rápido em 75% dos problemas, mesmo sendo testado em um equipamento de menor capacidade que aquele utilizado por Edelkamp e Jabbar.

Na maioria dos casos o espaço de busca explorado aumentou, porém os resultados do modelo PMDC ficaram, em média, 3 vezes mais rápidos para encontrar o plano do que o modelo proposto por Edelkamp e Jabbar. Além disso, é possível notar que, em muitos casos, a diminuição do espaço de busca explorado não produz necessariamente tempos menores. Este fato pode ser observado no problema ELEVATOR onde o espaço de busca explorado foi reduzido a um terço, mas o tempo para encontrar o plano continua sendo equivalente.

O conjunto de problemas usado também contém problemas livres de bloqueio, como é

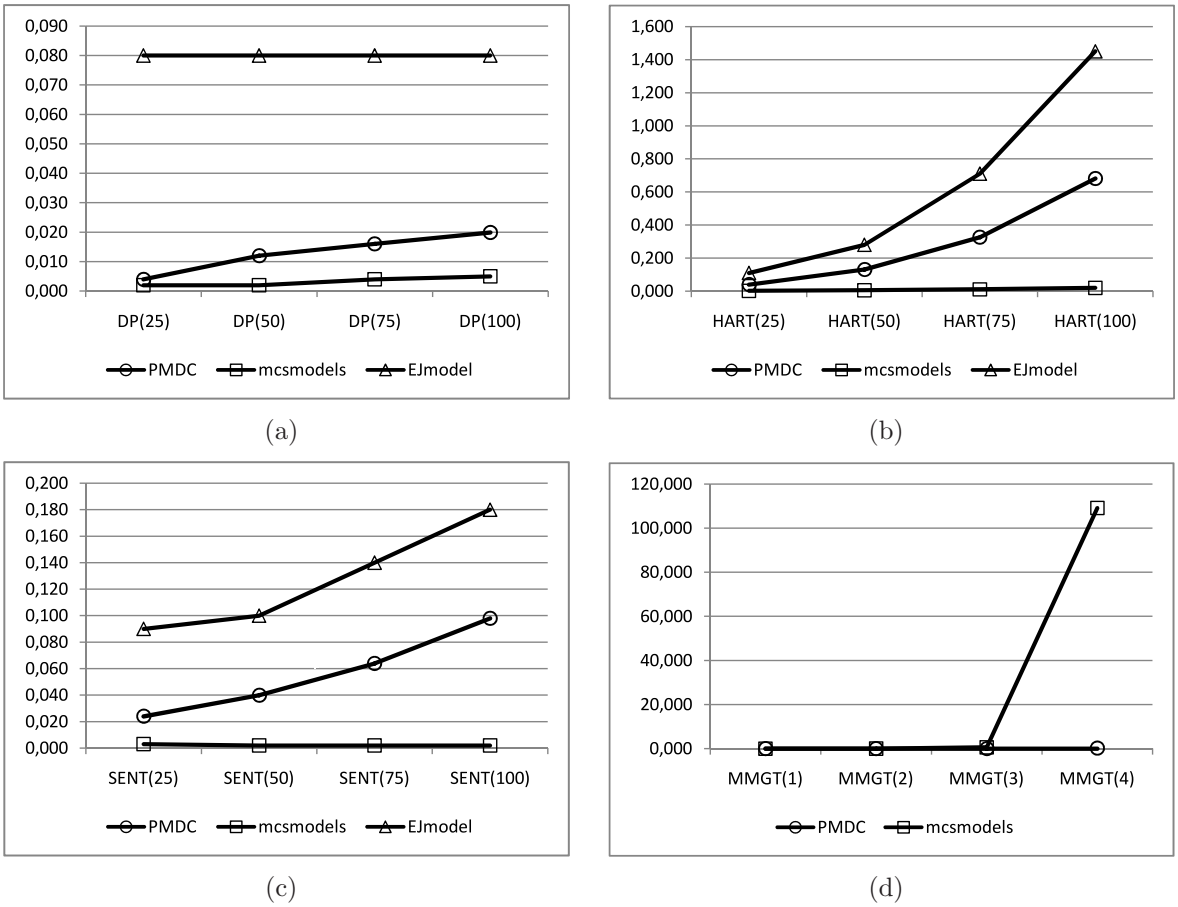

Figura 5.12: Comparação entre os tempos obtidos por cada abordagem (em segundos) para os problemas DP, HART, SENT e MMGT.

o caso dos problemas DPH. No entanto, foi verificado que a maior dificuldade do modelo PMDC é encontrar uma resposta eficiente para este tipo de problema, apesar de que em alguns poucos casos ele é mais rápido, como pode ser observado no problema DPH(7). Isto ocorre porque a máquina de busca do planejador verifica todos os estados alcançáveis da rede para chegar à conclusão de que não existe bloqueio.

5.4 Considerações

Esta seção tratou de apresentar métodos de tradução entre os problemas de planejamento temporal em inteligência artificial e problemas de alcançabilidade em RdP com tempo. No entanto, esta dissertação é um segmento dos trabalhos de Silva [1], Hickmott et al. [2], Petry [3] e Edelkamp e Jabbar [4] quando se fala em tradução de formalismos, na qual são consideras as informações temporais dos modelos, o que torna esta tarefa mais

complexa.

Na verificação de bloqueios em RdP utilizando o planejador FF-Metric, a tradução apresentada nas seções 5.1.3 e 5.1.4 não obtiveram resultados satisfatórios quando a descrição do objetivo do problema de planejamento era composto por um número grande de disjunções. Sendo assim, outra tradução é apresentada na seção 5.3, onde foram utilizados predicados numéricos para descrever as transições. Os resultados obtidos foram, em média, três vezes melhores em relação aos resultados apresentados por Edelkamp e Jabbar [4], mesmo com o uso de uma máquina com configuração inferior.

A simplificação do modelo PMDC em relação ao EJmodel, deve-se ao descarte da descrição de todos os arcos da rede e da existência de um único tipo de ação: a de disparo. Além disso, toda a descrição da RdP é feita utilizando apenas operações aritméticas básicas enquanto para o EJmodel são utilizados quantificadores existenciais e universais, disjunções, efeitos condicionais e pré-condições negadas. Outra vantagem está relacionada ao fato de não existir ações que estabelecem o estado da transição (habilitada ou desabilitada) já que a própria pré-condição da ação de disparo verifica isso. Contudo, como cada transição é traduzida em uma ação, o arquivo de domínio do problema de planejamento torna-se mais extenso, o que não afeta o desempenho do planejador.

Pode-se concluir que o desempenho do modelo PMDC está entre o EJmodel e a ferramenta MCSMODELS, isto porque quando comparado ao primeiro a vantagem obtida é o tempo menor para computar o plano. Já em relação ao segundo, apesar do tempo de processamento maior, o modelo PMDC garante uma resposta correta, mesmo nos casos onde a ferramenta MCSMODELS falhou.

Os resultados obtidos não são suficientes para demonstrar todas as vantagens do modelo proposto. No entanto, em trabalhos futuros, fazem-se necessários testes com RdP maiores, pois em alguns domínios acredita-se que, quanto maior o tamanho da rede a ser analisada, maior é o ganho obtido pelo modelo aqui proposto. Também cabe a trabalhos futuros realizar testes com outros algoritmos de busca e a utilizar objetivos métricos para minimizar o espaço de estados. Outra sugestão ainda é abordar estratégias rápidas para obter respostas em redes livres de bloqueios.

CAPÍTULO 6

CONCLUSÕES E TRABALHOS FUTUROS

O propósito geral da pesquisa desenvolvida por este trabalho é possibilitar o relacionamento entre planejamento e RdP com o intuito de explorar as vantagens que cada área do conhecimento pode proporcionar. Sendo assim, o trabalho de pesquisa decorrente desta dissertação resultou na apresentação de métodos de tradução entre problemas de planejamento temporal em inteligência artificial e RdP com tempo visando resolver problemas de alcançabilidade. Além disso, todas as traduções apresentadas, com exceção daquela para verificação de bloqueios em RdP, podem ser utilizadas para qualquer fim e não apenas para a verificação de problemas de alcançabilidade.

Apesar dos testes somente terem ocorridos com a tradução das RdP para verificação de bloqueios em planejamento, apresentada na seção 5.3, o resultado obtido por esta tradução foi bastante expressivo, principalmente quando comparado com os resultados de Edelkamp e Jabbar [4], autores que foram uma das principais referências deste trabalho. Dentre as demais traduções a que mais se destaca é aquela referente ao relacionamento que vai de planejamento temporal para RdP temporais. Tal destaque é atribuído a este relacionamento porque qualquer ação durativa descrita em PDDL pode ser simulada pela RdP obtida através do processo de tradução, mesmo sabendo das limitações das transições temporais apresentadas pelo teorema 1.

Por fim, foi possível concluir que o poder de expressividade das transições temporais e temporizadas de uma RdP é inferior à de uma ação durativa descrita pela linguagem PDDL, o que limitada sua representação na modelagem de problemas de planejamento, tal como é o caso de alguns problemas que necessitam do uso do conceito de concorrência requerida sobre uma mesma ação. Além disso, trabalhos que relacionam as duas áreas do conhecimento e que utilizam informações temporais para descrever seus problemas não foram encontrados na literatura.

Sob outra ótica, já que são apresentadas traduções entre ambas as áreas, outra contribuição importante deste trabalho é permitir o uso do *benchmark* da área de planejamento temporal para efetuar testes com as RdP com tempo, que até o presente momento não existia.

Trabalhos futuros que visam dar continuidade ao desenvolvido desta dissertação podem levar em consideração as seguintes sugestões:

- Apesar da tradução apresentada na seção 5.2 cobrir a maior parte dos problemas utilizados pela competição de planejadores, ainda existem várias especificações da linguagem PDDL que não foram traduzidas, tais como efeitos condicionais, quantificadores universais e existenciais, disjunções, predicados tipados e predicados numéricos, não são abordadas, pois o escopo principal deste trabalho é a tradução das restrições temporais;

- Implementar as traduções citadas no capítulo 5, já que a única efetivamente testada foi aquela para verificação de bloqueios;

- Criar e implementar o algoritmo para a fase de pós-processamento para corrigir o erro descrito pelo item b) da seção 5.2.5;

- Efetuar testes com as RdP temporais geradas pela tradução dos problemas de planejamento temporal, assim como utilizar a técnica de desdobramento da RdP, conhecido como *unfolding* [52], para analisar as suas propriedades estruturais e comportamentais;

- Investigar a viabilidade real de análise de RdP em planejamento e vice-versa.

A perspectiva deste trabalho é possibilitar novas técnicas eficientes de análise, tanto para a área de RdP quanto para a de planejamento, assim como já foi comprovado naquela para a verificação de bloqueios em RdP apresentada na seção 5.3. Também há indícios de que a RdP temporal gerada pela tradução de um problema de planejamento, apresentada na seção 5.2, pode ser simplificada, reduzindo o tamanho da RdP e, consequentemente, a sua complexidade de análise.

BIBLIOGRAFIA

[1] SILVA, F. *Rede de Planos: Uma proposta para a Solução de Problemas de Planejamento em Inteligência Artificial usando Redes de Petri*. Tese (Doutorado) — Centro Federal de Educação Tecnológica do Paraná - CEFET-PR, Curitiba, PR, Brasil, fev. 2005.

[2] HICKMOTT, S. L. et al. Planning via petri net unfolding. In: VELOSO, M. M. (Ed.). *IJCAI*. [s.n.], 2007. p. 1904–1911. Disponível em: <http://dblp.uni-trier.de/db/conf/ijcai/ijcai2007.html>.

[3] PETRY, F. C. *Planejamento Aplicado à Verificação de Bloqueios em Redes de Petri*. Dissertação (Mestrado) — Universidade Federal do Paraná - UFPR, Curitiba, PR, Brasil, ago. 2008.

[4] EDELKAMP, S.; JABBAR, S. Action planning for directed model checking of petri nets. *Electronic Notes in Theoretical Computer Science*, v. 149, n. 2, p. 3–18, 2006.

[5] SMITH, D. E. *Planning Formalisms Commentary*. set. 2007. ICAPS-07 Planning Formalisms Session Commentary. Disponível em: <http: //ti.arc.nasa.gov/ people/ de2smith/ publications.html>.

[6] CUSHING, W. et al. When is temporal planning really temporal? In: VELOSO, M. M. (Ed.). *IJCAI*. [s.n.], 2007. p. 1852–1859. Disponível em: <http://dblp.uni-trier.de/db/conf/ijcai/ijcai2007.html>.

[7] COLES, A. I. et al. Planning with problems requiring temporal coordination. In: *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI 08)*. [S.l.: s.n.], 2008.

[8] GEREVINI, A. Automated planning in temporal domains: Some recent advances and current research topics. In: *TIME*. IEEE Computer Society, 2007. p. 3–4. ISBN 978-0-7695-2836-6. Disponível em: <http://dblp.uni-trier.de/db/conf/time/time2007.html>.

[9] NILSSON, N. J.; FIKES, R. E. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artifical Intelligence*, v. 2, n. 3-4, p. 189–208, 1971.

[10] VILA, L. A survey on temporal reasoning in artificial intelligence. *AI Communications*, v. 7, n. 1, p. 4–28, 1994. Disponível em: <citeseer.ist.psu.edu/vila94survey.html>.

[11] PENBERTHY, J. S.; WELD, D. S. Temporal planning with continuous change. In: *Proceedings of the 12th National Conference on Artificial Intelligence*. Seattle, WA, USA: AAAI, 1994. v. 2, p. 1010–1015. Disponível em: <http://dblp.uni-trier.de/db/conf/aaai/aaai94-2.html>.

[12] YOUNES, H. L. S.; SIMMONS, R. G. VHPOP: Versatile Heuristic Partial Order Planner. *J. Artif. Intell. Res. (JAIR)*, v. 20, p. 405–430, 2003. Disponível em: <http://dblp.uni-trier.de/db/journals/jair/jair20.html>.

[13] DO, M. B.; KAMBHAMPATI, S. Sapa: A multi-objective metric temporal planner. *J. Artif. Intell. Res. (JAIR)*, v. 20, p. 155–194, 2003. Disponível em: <http://dblp.uni-trier.de/db/journals/jair/jair20.html>.

[14] HALSEY, K.; LONG, D.; FOX, M. CRIKEY - A Planner Looking at the Integration of Scheduling and Planning. In: *Proceedings of the Workshop on Integration Scheduling Into Planning at 13th International Conference on Automated Planning and Scheduling (ICAPS'03)*. [S.l.: s.n.], 2004. p. 46–52.

[15] MCDERMOTT, D. et al. *PDDL - The Planning Domain Definition Language*. New Haven, CN, USA, 1998.

[16] RINTANEN, J. Complexity of concurrent temporal planning. In: BODDY, M.; FOX, M.; THIÉBAUX, S. (Ed.). *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS'07)*. Providence, Rhode Island, USA: [s.n.], 2007. v. 17, p. 280–287.

[17] MURATA, T. Petri nets: Properties, analysis and applications. In: *Proceedings of the IEEE*. Los Alamitos, CA, USA: IEEE, 1989. v. 7, n. 4, p. 541–580.

[18] PETERSON, J. L. Petri nets. *ACM Comput. Surv.*, v. 9, n. 3, p. 223–252, 1977. Disponível em: <http://dblp.uni-trier.de/db/journals/csur/csur9.html>.

[19] ALLEN, J. F. Planning as temporal reasoning. In: ALLEN, J. F.; FIKES, R.; SANDEWALL, E. (Ed.). *KR'91: Principles of Knowledge Representation and Reasoning*. San Mateo, California: Morgan Kaufmann, 1991. p. 3–14. Disponível em: <citeseer.ist.psu.edu/allen91planning.html>.

[20] LAVALLE, S. M. *Planning Algorithms*. Cambridge, U.K.: Cambridge University Press, 2006. Available at http://planning.cs.uiuc.edu/.

[21] SMITH, D.; FRANK, J.; JÓNSSON, A. Bridging the gap between planning and scheduling. *Knowledge Engineering Review*, v. 15, n. 1, p. 47–83, 2000.

[22] RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 2nd edition. ed. Englewood Cliffs, NJ, USA: Prentice-Hall, 2003. 1132 p. ISBN 0-13-790395-2.

[23] PEDNAULT, E. P. D. ADL: Exploring the Middle Ground Between STRIPS and the Situation Calculus. In: *KR*. [s.n.], 1989. p. 324–332. Disponível em: <http://dblp.uni-trier.de/db/conf/kr/kr89.html>.

[24] FOX, M.; LONG, D. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research (JAIR)*, v. 20, p. 61–124, 2003. Disponível em: <http://dblp.uni-trier.de/db/journals/jair/jair20.html>.

[25] VELOSO, M. M. *Learning By Analogical Reasoning in General Problem Solving*. Tese (PhD) — Carnegie Mellon University, Pittsburgh, PA, U.S.A., 1992.

[26] CUSHING, W. et al. Evaluating temporal planning domains. In: BODDY, M.; FOX, M.; THIÉBAUX, S. (Ed.). *Proceedings of the Seventeenth International Conference on Automated Planning and Scheduling (ICAPS'07)*. Providence, Rhode Island, USA: [s.n.], 2007. v. 17.

[27] WULLINGER, P. *Transformation of Temporally Expressive Into Temporally Simple Planning Problems*. Dissertação (Mestrado) — University of Bamberg, Bamberg, Bavaria, Alemanha, set. 2007.

[28] GHALLAB, M.; NAU, D.; TRAVERSO, P. *Automated Planning: Theory and Practice*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004. 663 p. ISBN 1-55860-856-7.

[29] ALLEN, J. F. Maintaining knowledge about temporal intervals. *Communications of the ACM*, v. 26, p. 832–843, 1983.

[30] LEVER, J.; RICHARDS, B. *A planning architecture using temporal constraint solving*. London, Greater London, Inglaterra, jan. 1993.

[31] VILAIN, M. B.; KAUTZ, H. A. Constraint propagation algorithms for temporal reasoning. In: *AAAI*. [s.n.], 1986. p. 377–382. Disponível em: <http://dblp.uni-trier.de/db/conf/aaai/aaai86-1.html>.

[32] ZAIDI, A. K. On temporal logic programming using petri nets. *IEEE Transactions on Systems, Man, and Cybernetics, Part A*, v. 29, n. 3, p. 245–254, maio 1999. Disponível em: <http://dblp.uni-trier.de/db/journals/tsmc/tsmca29.html>.

[33] DECHTER, R.; MEIRI, I.; PEARL, J. Temporal constraint networks. *Artificial Intelligence*, v. 49, n. 1-3, p. 61–95, 1991.

[34] MEIRI, I. Combining qualitative and quantitative constraints in temporal reasoning. In: DEAN, T.; MCKEOWN, K. (Ed.). *Proceedings of the Ninth National Conference on Artificial Intelligence*. Menlo Park, California: AAAI Press, 1991. p. 260–267. Disponível em: <citeseer.ist.psu.edu/meiri95combining.html>.

[35] BARBER, F. Reasoning on interval and point-based disjunctive metric constraints in temporal contexts. *Journal of Artificial Intelligence Research*, v. 12, p. 35–86, 2000. Disponível em: <citeseer.ist.psu.edu/barber00reasoning.html>.

[36] SMITH, D. E.; WELD, D. S. Temporal planning with mutual exclusion reasoning. In: DEAN, T. (Ed.). *IJCAI*. Morgan Kaufmann, 1999. p. 326–337. ISBN 1-55860-613-0. Disponível em: <http://dblp.uni-trier.de/db/conf/ijcai/ijcai99.html>.

[37] COLES, A. et al. Managing coordination in temporal planning using planner-scheduler interaction. Preprint submitted to Elsevier Science. nov. 2007. Disponível em: <http://personal.cis.strath.ac.uk/ ac/52426/crikeyaij>.

[38] PETRI, C. A. *Kommunikation mit Automaten.* Tese (Doutorado) — Bonn: Institut für Instrumentelle Mathematik, Schriften des IIM Nr. 2, 1962. Second Edition:, New York: Griffiss Air Force Base, Technical Report RADC-TR-65–377, Vol.1, 1966, Pages: Suppl. 1, English translation: Communication with Automata.

[39] BALBO, G. et al. Petri nets 2000: Introductory tutorial. In: NIELSEN, M.; SIMPSON, D. (Ed.). *21st International Conference on Application and Theory of Petri Nets (ICATPN)*. Aarhus, Denmark: Springer, 2000. (Lecture Notes in Computer Science, v. 1825), p. 26–30. ISBN 3-540-67693-7. DOS SLIDES. Disponível em: <http://www. informatik. uni-hamburg. de/ TGI/PetriNets/ introductions/ pn2000_introtut.pdf>.

[40] BUCHHOLZ, P. Petri nets. Course of International Center for Computational Logic - Technischen Universität Dresden. 2002.

[41] ZURAWSKI, R.; ZHOU, M. Petri nets and industrial applications: A tutorial. *Industrial Electronics, IEEE Transactions on*, v. 41, n. 6, p. 567–583, dez. 1994. Disponível em: <http://dx.doi.org/10.1109/41.334574>.

[42] CARDOSO, J.; VALETTE, R. *Redes de Petri*. Florianópolis, SC, Brasil: UFSC, 1997. 212 p.

[43] JúNIOR", N. M. *Redes de Petri Temporais: Método de Análise Baseado em Tempo Global*. Dissertação (Mestrado) — Universidade Federal do Paraná - UFPR, Curitiba, PR, Brasil, fev. 2008.

[44] RAMCHANDANI, C. *Analysis of Asynchronous Concurrent Systems by Timed Petri Nets*. Tese (Doutorado) — Cambridge, Mass.: MIT, Dept. Electrical Engineering, fev. 1974. Project MAC TR-120.

[45] SIFAKIS, J. Use of petri nets for performance evaluation. In: BEILNER, H.; GELENBE, E. (Ed.). *Performance*. Bad Godesberg, Bonn, Germany: North-Holland, 1977. p. 75–93. ISBN 0-444-85058-9.

[46] SIFAKIS, J. Performance evaluation of systems using nets. In: BRAUER, W. (Ed.). *Proceedings of the Advanced Course on General Net Theory of Processes and Systems*. London, UK: Springer-Verlag, 1980. (Lecture Notes in Computer Science, v. 84), p. 307–319. ISBN 3-540-10001-6. Disponível em: <http://dblp.uni-trier.de/db/conf/ac/nt.html>.

[47] MERLIN, P. M. *A Study of Recoverability of Computer Systems*. Tese (Doutorado) — University of California, Dep. Comput. Sci, Irvine, CA, USA, 1974.

[48] MELZER, S.; RÖMER, S. Deadlock checking using net unfoldings. In: GRUMBERG, O. (Ed.). *Computer Aided Verification, 9th International Conference, CAV '97*. Haifa, Israel: Springer, 1997. (Lecture Notes in Computer Science, v. 1254), p. 352–363. ISBN 3-540-63166-6. Disponível em: <http://dblp.uni-trier.de/db/conf/cav/cav97.html>.

[49] CORBETT, J. C. Evaluating deadlock detection methods for concurrent software. *IEEE Trans. Software Eng.*, v. 22, n. 3, p. 161–180, 1996. Disponível em: <http://dblp.uni-trier.de/db/journals/tse/tse22.html>.

[50] HOFFMANN, J. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *J. Artif. Intell. Res. (JAIR)*, v. 20, p. 291–341, 2003. Disponível em: <http://dblp.uni-trier.de/db/journals/jair/jair20.html>.

[51] HELJANKO, K.; SIMONS, P. *mcsmodels 1.4: a deadlock and reachability checker using net unfoldings*. 1999. Helsinki University of Technology, Laboratory for Theoretical Computer Science, Espoo, Finland. Software.

[52] ESPARZA, J.; RÖMER, S.; VOGLER, W. An improvement of McMillan's unfolding algorithm. *Form. Methods Syst. Des.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 20, n. 3, p. 285–310, 2002. ISSN 0925-9856.