Jonatha Rodrigues da Costa

Modelagem de Sistemas a Eventos Discretos

Maracanaú
2025

Jonatha Rodrigues da Costa

Modelagem de Sistemas a Eventos Discretos

Este livro trata sobre modelagem de sistemas a eventos discreotos aplicados à engenharia incluindo modelagem de sistemas utilizando a ferramenta de análise gráfica CPN IDE® .

Instituto Federal de Educação, Ciência e Tecnologia do Ceará

Maracanaú
2025

Dedico este trabalho a Deus, a minha família, aos colegas acadêmicos e aos discentes, especialmente os discentes dos cursos de engenharia.

Agradecimentos

"Estudar é uma dádiva divina!"

Resumo

Sumário

Sumário 6
Lista de ilustrações 8
1 Sistemas de Manufatura: Fabricando um Produto, Modelagem e Problemas de Controle 9
1.1 Perceptiva fabril 10
1.2 Conceitos de Modelagem de Sistemas de Manufatura 11
1.2.1 Principais Ferramentas de Modelagem de Sistemas 12
2 Redes de Petri: conceitos iniciais 14
2.1 Representações Formais da Rede de Petri 14
2.2 Representação Gráfica 15
2.2.1 Funcionamento conceitual de uma RP 16
2.3 Poder de Expressão das Redes de Petri 16
2.3.1 Sequência 16
2.3.2 Concorrência 17
2.3.3 Conflito 17
2.3.4 Sincronismo 17
2.3.5 Caminhos Alternativos 18
2.3.6 Repetição 18
2.3.7 Alocação de Recursos 18
2.4 Dinâmica e Propriedades das Redes de Petri 19
2.4.1 Alcance 19
2.4.2 Conservação 19
2.4.3 Vivacidade 20
2.4.4 Reversibilidade 20
2.4.5 Segurança 21
2.5 Conclusão 22
3 Redes de Petri: modelos iniciais 23
3.1 Caso Prático Industrial: representação em RP 23
3.1.1 Estrutura da Rede de Petri 23
3.2 Matrizes de Pré, Pós e de Incidência de uma Rede de Petri 24
3.2.1 Modelagem das Matrizes de uma RP 25
3.2.2 Matriz de Pré-Incidência (C^-) 26
3.2.3 Matriz de Pós-Incidência (C^+) 26
3.2.4 Matriz de Incidência (C) 26
4 RP temporizadas 28
4.1 Redes de Petri Temporais 28
4.1.1 Definição 28
4.1.2 Propriedades das Redes de Petri Temporais 29
4.2 Redes de Petri Temporalizadas 29

4.2.1 Definição . . . 29
4.2.2 Vantagens das Redes de Petri Temporalizadas . . . 29
4.3 Comparativo entre RP-T e RP-Tz . . . 30
4.4 Conclusão . . . 30
5 Redes de Petri Hierárquicas . . . 31
5.1 Definição . . . 31
6 CPN IDE® . . . 32
6.1 CPN IDE: Interface . . . 32
6.2 Funções básicas do CPN IDE . . . 34
6.3 Linguagem do CPN IDE: CPN ML . . . 34
6.4 Conjuntos de Cores . . . 35
6.4.1 Conjuntos de Cores Simples . . . 35
6.4.1.1 Conjunto de cores UNIT . . . 35
6.4.1.2 Conjunto de cores BOOLEAN . . . 36
6.4.1.3 Conjunto de cores INTEGER . . . 37
6.4.1.4 Conjunto de cores STRING . . . 38
6.4.1.5 Conjunto de cores ENUMERATED . . . 39
6.4.1.6 Conjunto de cores INDEXED . . . 40
6.4.2 Conjuntos de Cores Compostos . . . 41
6.4.2.1 Conjunto de cores PRODUCT . . . 41
6.4.3 Conjunto de cores RECORD . . . 42
6.4.3.1 Conjunto de Cores `union` . . . 44
6.4.3.2 Conjunto de cores LIST . . . 45
6.5 Modelo de Rede: Conto da Cinderela . . . 47

Lista de ilustrações

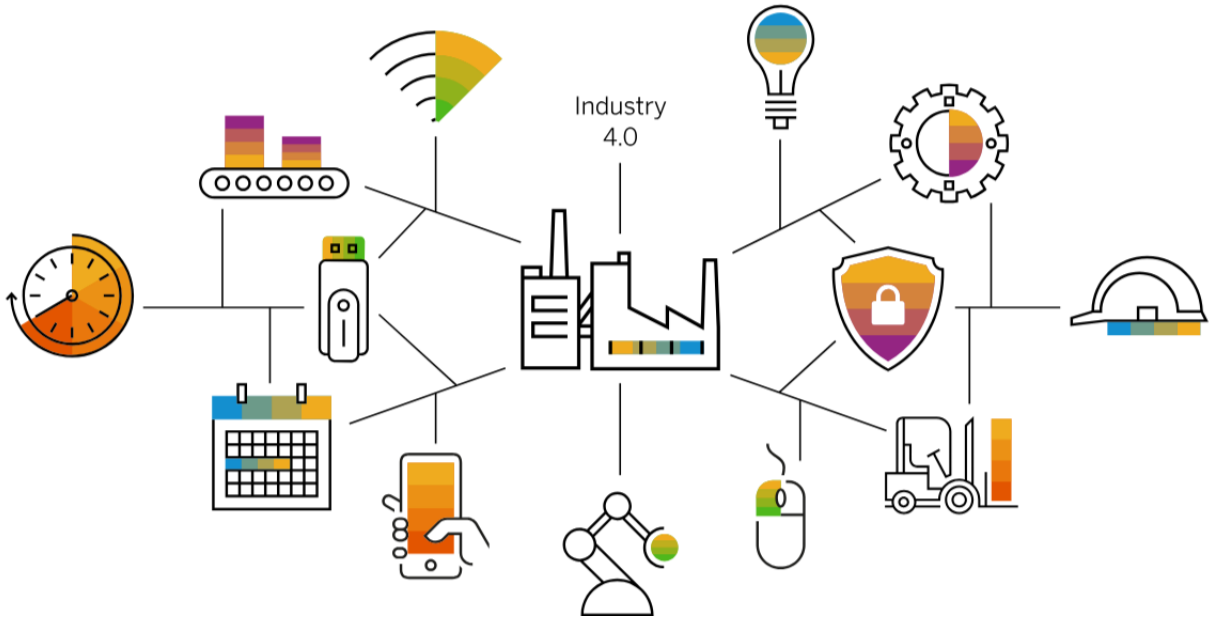
Figura 1 – Percepção da indústria 4.0 9
Figura 2 – Exemplo de Rede de Petri simples 16
Figura 3 – Grafo de marcações ilustrando a evolução do sistema 16
Figura 4 – Expressão de processo em sequência numa RP 17
Figura 5 – Expressão de processo em concorrência numa RP 17
Figura 6 – Expressão de processo em conflito numa RP 17
Figura 7 – Expressão de processo em sincronismo numa RP 18
Figura 8 – Expressão de processo em Caminhos Alternativos numa RP 18
Figura 9 – Expressão de processo de Repetição numa RP 18
Figura 10 – Expressão de Alocação de Recursos numa RP 19
Figura 11 – Alcançabilidade de uma RP 19
Figura 12 – Conservação de uma RP 20
Figura 13 – Vivacidade de uma RP 20
Figura 14 – Reversibilidade de uma RP 21
Figura 15 – Segurança de uma RP 21
Figura 16 – Diagrama gráfico da RP representando o fluxo de produção industrial 24
Figura 17 – Representação gráfica de um RP-T. 28
Figura 18 – Representação gráfica de um RP-Tz. 29
Figura 19 – Rede de Petri Hierárquica - Nível Superior 31
Figura 20 – Rede de Petri Hierárquica - Nível de sub-rede 31
Figura 21 – Interface de desenvolvimento integrado do *software* CPN IDE® 33
Figura 22 – Área de Declarações no CPN IDE® 33
Figura 23 – Modelagem do conjunto de cores `unit` no CPN IDE® 36
Figura 24 – Modelagem do conjunto de cores `bool` no CPN IDE® 37
Figura 25 – Modelagem do conjunto de cores `int` no CPN IDE® 38
Figura 26 – Modelagem do conjunto de cores `string` no CPN IDE® 39
Figura 27 – Modelagem do conjunto de cores `enumerated` no CPN IDE® 40
Figura 28 – Modelagem do conjunto de cores `indexed` no CPN IDE® 41
Figura 29 – Modelagem do conjunto de cores `product` no CPN IDE® 42
Figura 30 – Modelagem do conjunto de cores `record` no CPN IDE® 43
Figura 31 – Modelagem do conjunto de cores `union` no CPN IDE® 45
Figura 32 – Modelagem do conjunto de cores `LIST` no CPN IDE® 46
Figura 33 – Modelagem de caso Cinderela no CPN IDE® 47
Figura 34 – Modelagem de caso Cinderela - Resultado da simulação no CPN IDE® 49

1 Sistemas de Manufatura: Fabricando um Produto, Modelagem e Problemas de Controle

A manufatura tem desempenhado um papel crucial como um dos pilares fundamentais para o desenvolvimento econômico e tecnológico ao longo de toda a história da humanidade. Desde os primeiros artefatos rudimentares, fabricados manualmente nas antigas civilizações, até os modernos e complexos sistemas automatizados de produção que caracterizam a indústria de ponta nos dias de hoje, a evolução dos processos de manufatura reflete diretamente o avanço da sociedade. Esse progresso se manifesta de forma clara na busca constante pela maior eficiência, qualidade e inovação no modo como os produtos são criados e entregues ao consumidor. Essa manufatura, mais do que uma simples atividade produtiva, se consolidou como uma força propulsora no desenvolvimento econômico, sendo também um reflexo das mudanças sociais e tecnológicas que moldaram o mundo moderno, conforme Smith (2010). No contexto atual, a manufatura está imersa numa revolução, impulsionada pela Indústria 4.0, que, ao incorporar novas tecnologias como a *Artificial Intelligence* (Inteligência Artificial) (AI), a robótica, *Internet of Things* (Internet das coisas) (IoT) e a computação em nuvem, tem transformando as formas de produção e, consequentemente, as bases da economia global, conforme Jones (2018).

Ilustrando esse cenário, na Figura 1 é apresentada uma percepção basilar de elementos distintos que, ao serem integrados em um único sistema, oferecem uma visão abrangente e interconectada de como diferentes componentes podem interagir de maneira síncrona. Esse arranjo dos elementos não só permite uma melhor compreensão da dinâmica interna do sistema, mas também amplia a percepção sobre suas funcionalidades e possibilidades. A integração dos diversos componentes oferece uma abordagem mais holística e eficaz, que facilita a concepção e o entendimento de sistemas complexos, permitindo uma análise mais detalhada e uma visão mais clara de como cada parte contribui para o todo.

Figura 1 – Percepção da indústria 4.0



Fonte:Adaptado de (SAP, 2025)

O conceito de sistemas de manufatura, portanto, pode ser definido como um conjunto organizado de processos físicos e lógicos que interagem para transformar matérias-primas em produtos acabados, atendendo às demandas específicas de consumidores e mercados. Historicamente, pode-se identificar quatro grandes fases no desenvolvimento dos sistemas de manufatura. A primeira fase, a manufatura artesanal, era caracterizada pela produção manual e altamente personalizada de produtos, em que cada item produzido era único, feito sob medida de acordo com as necessidades do cliente. Já a segunda fase, marcada pela revolução industrial e pela introdução da linha de montagem, impulsionou a produção em massa, que permitiu a fabricação de grandes quantidades de produtos padronizados a custos muito mais baixos, representando um marco na história da produção. A terceira fase, a manufatura enxuta, focou na redução de desperdícios e na melhoria contínua dos processos, buscando a máxima eficiência na utilização de recursos e tempo, sem sacrificar a qualidade, conforme Anderson (2015). Essa estrutura sistêmica converge na Indústria 4.0, que integra a digitalização dos processos produtivos, automação avançada e sistemas ciberfísicos, possibilitando um nível de personalização e eficiência que antes era impensável. Essa nova fase tem o potencial de transformar profundamente todos os aspectos da produção, tornando-a mais ágil, inteligente e conectada, segundo Mellor (2014).

1.1 PERCEPTIVA FABRIL

Dentro desse vasto panorama, os sistemas de manufatura podem ser classificados conforme a abordagem adotada para a produção em: discreto, contínuo e flexível. No caso dos sistemas de produção discreta, cada item produzido é individualmente contável e tratado separadamente, como ocorre na fabricação de automóveis, eletrodomésticos e componentes eletrônicos. Já os sistemas de produção contínua operam com fluxos ininterruptos de materiais, como no caso das indústrias química e petroquímica, em que a produção de produtos como petróleo, produtos químicos e alimentos exige a constante movimentação de materiais e processos sem interrupções. Uma categoria intermediária são os sistemas flexíveis de manufatura, que têm a capacidade de se adaptar rapidamente a diferentes tipos de produção, ajustando-se a novos produtos e exigências de mercado. Nesse contexto, *CAM - Computer Aided Manufacturing*) (Manufatura Assistida por Computador) (CAM) surge como uma tecnologia crucial para otimizar processos, empregando simulações e controle digital para maximizar a eficiência e minimizar erros durante a produção, conforme apresenta Baskin (2017). Isso implica em produtos manufaturados com ciclos de vida distintos entre si, tais que podem ou não resultar de projetos com variações de qualidade e segurança.

O ciclo de vida de um produto, desde sua concepção até sua disposição final, envolve uma série de etapas interdependentes , as quais podem ser definidas como: fase inicial, projeto, fabricação e reciclagem. A fase inicial, que abrange o projeto e o desenvolvimento do produto, é de extrema importância, pois é nessa fase que são definidos aspectos cruciais como a funcionalidade, os materiais a serem utilizados e os processos de fabricação mais adequados para atingir a qualidade desejada, conforme Harris (2002). Após o projeto, segue-se o planejamento da produção, em que são definidos os recursos necessários, como equipamentos, materiais e mão de obra, bem como os processos a serem seguidos. A etapa de fabricação é em que ocorre a transformação real da matéria-prima no produto acabado, seguindo-se, então, a distribuição e o suporte, garantindo a entrega do produto e a manutenção adequada ao longo de sua vida útil, conforme Wilson (2016). O ciclo se fecha com a reciclagem ou o descarte adequado do produto, uma fase cada vez mais importante, dada a crescente preocupação com a sustentabilidade e o reaproveitamento de materiais, conforme Brundtland (1987).

Para garantir a eficiência e o controle sobre esses processos, a modelagem de sistemas de manufatura desempenha um papel essencial. Modelos formais, como diagramas de fluxo, máquinas de estados finitos e Redes de Petri, são ferramentas amplamente utilizadas para a análise, otimização e simulação dos fluxos produtivos. Essas ferramentas permitem que engenheiros e gestores analisem de maneira detalhada o comportamento dos sistemas e identifiquem pontos críticos, como gargalos e potenciais falhas. Com isso, é possível antecipar problemas e otimizar a sincronização entre as diversas etapas da produção, resultando em uma maior eficiência operacional, conforme Murata (1989).

No entanto, mesmo com a utilização dessas ferramentas de modelagem, os sistemas de manufatura enfrentam desafios consideráveis no controle de suas operações. A sincronização de processos, o balanceamento da linha de produção e a gestão eficaz de estoques e logística são questões complexas que exigem soluções criativas e eficientes por parte dos engenheiros e gestores. A teoria de controle supervisório, por exemplo, oferece uma abordagem robusta para enfrentar esses desafios. Baseada na modelagem com Redes de Petri, essa teoria permite a supervisão e coordenação eficaz dos sistemas produtivos, possibilitando uma maior flexibilidade e uma resposta rápida às mudanças nas condições de produção, conforme Chase e Aquilano (2006). A integração de diferentes abordagens de controle também é fundamental para a adaptação das fábricas às exigências da produção moderna, cada vez mais dinâmicas e variáveis,conforme Cassandras (2008).

Dessa forma, compreender a estrutura e o funcionamento dos sistemas de manufatura é essencial para os profissionais da engenharia moderna. A modelagem eficiente e o controle adequado desses sistemas não apenas otimizam a produção, mas também impulsionam a inovação, a competitividade e o avanço tecnológico, permitindo que as empresas se destaquem no mercado globalizado (THOMPSON, 2019). As organizações que adotam essas tecnologias e abordagens têm uma vantagem estratégica significativa, pois conseguem produzir com mais eficiência, atender de forma mais precisa às necessidades dos consumidores e se adaptar rapidamente às mudanças do mercado.

1.2 CONCEITOS DE MODELAGEM DE SISTEMAS DE MANUFATURA

A modelagem de sistemas de manufatura é uma técnica essencial para representar, analisar e otimizar processos complexos dentro das fábricas. Modelar um sistema envolve criar uma abstração que descreve seus componentes, interações e comportamentos. Essas representações são fundamentais para entender e prever como o sistema opera, identificar falhas potenciais, otimizar o desempenho e apoiar decisões de planejamento e controle. A modelagem de sistemas pode ser dividida em diferentes abordagens, como modelagem gráfica, matemática e computacional.

Entre os conceitos-chave em modelagem, destacam-se:

- **Abstração**: Reduzir a complexidade de um sistema real, considerando apenas os aspectos mais relevantes para a análise ou melhoria do desempenho.
- **Simulação**: Técnica usada para imitar o comportamento do sistema sob diferentes condições, o que ajuda a prever os resultados de mudanças no processo sem a necessidade de experimentos reais.
- **Otimização**: Processo de encontrar a melhor solução para um problema de manufatura, seja em termos de custo, tempo ou recursos, ajustando variáveis dentro de um modelo matemático.

- **Estudo de Fluxo**: Análise dos caminhos que os materiais percorrem ao longo do sistema, buscando melhorar a fluidez e reduzir os tempos de espera e os gargalos.

1.2.1 PRINCIPAIS FERRAMENTAS DE MODELAGEM DE SISTEMAS

Diversas ferramentas e técnicas são utilizadas para modelar sistemas de manufatura. Cada ferramenta possui características distintas, adequando-se a diferentes tipos de problemas e abordagens de análise. Na Tabela 1 são apresentadas as principais ferramentas de modelagem, seus critérios de classificação e recomendações de utilização.

Tabela 1 – Principais ferramentas de modelagem de sistemas de manufatura.

Ferramenta	Critérios de Classificação	Recomendações de Utilização
Diagrama de Fluxo	Simplicidade e clareza na representação de processos	Ideal para sistemas simples, em que o foco é entender a sequência de atividades e decisões.
Máquinas de Estados Finitos	Modelagem de sistemas discretos com estados bem definidos	Útil para sistemas discretos e sequenciais, como linhas de montagem e processos repetitivos.
Redes de Petri	Representação gráfica de processos e controle de fluxos	Indicada para modelar sistemas complexos e concorrentes, como sistemas de controle de produção.
Simulação de Monte Carlo	Análise probabilística de incertezas nos processos de produção	Usada em sistemas com variabilidade, como aqueles sujeitos a falhas e incertezas em tempos de ciclo.
Programação Linear	Modelagem matemática e otimização de recursos	Aplicada para otimizar processos com restrições de recursos, como balanceamento de linhas de produção.
Algoritmos Genéticos	Métodos de busca e otimização evolutiva	Recomendados para problemas complexos de otimização em que soluções exatas são difíceis de encontrar.
Modelagem Baseada em Agentes	Análise de interação entre componentes autônomos do sistema	Ideal para sistemas dinâmicos e complexos, em que há interação entre diferentes agentes, como robôs em uma fábrica inteligente.

Fonte: Autor, (2025)

A escolha da ferramenta de modelagem adequada depende diretamente do tipo de sistema a ser analisado, das características dos processos envolvidos e dos objetivos da análise. Por exemplo, para sistemas mais simples, em que os processos são claros e as variáveis poucas, diagramas de fluxo podem ser suficientes. No entanto, quando lidamos com sistemas mais dinâmicos, com muitas variáveis e interações complexas, ferramentas mais sofisticadas, como Redes de Petri (RP) e simulações de Monte Carlo, podem ser mais eficazes. Isso nos leva à questão central de como essas ferramentas nos ajudam a entender e otimizar sistemas de manufatura, especialmente quando se trata de analisar eventos discretos.

Em todos os cenários, a modelagem de sistemas de manufatura permite uma visão mais integrada da operação como um todo. Ao aplicar essas ferramentas, conseguimos identificar pontos críticos, como gargalos e falhas potenciais, além de descobrir oportunidades para melhorias. Essa abordagem, ao mesmo tempo analítica e estratégica, é fundamental para o desenvolvimento de sistemas de produção mais

eficientes e adaptáveis, alinhados às necessidades da Indústria 4.0. E, ao falarmos da Indústria 4.0, nos deparamos com uma transformação profunda na forma como os sistemas de manufatura são gerenciados, o que torna a escolha da ferramenta certa ainda mais relevante.

No entanto, quando o foco está na modelagem de sistemas de manufatura baseados em eventos discretos, as RP se destacam como uma ferramenta particularmente poderosa. Sua capacidade de representar a dinâmica de sistemas complexos, em que múltiplos componentes interagem de maneira concorrente e assíncrona, torna-as ideais para ambientes em que a sincronização e a comunicação entre eventos são fundamentais. Ao contrário de abordagens mais tradicionais, as Redes de Petri nos fornecem uma representação visual clara e intuitiva dessas interações, o que facilita o entendimento da complexidade dos sistemas. Isso é especialmente importante em sistemas de manufatura modernos, como os encontrados em fábricas automatizadas ou ambientes de produção flexível, em que a variabilidade e a incerteza desempenham um papel significativo.

Portanto, ao nos aprofundarmos nesse tema, vemos que as RP não são apenas uma ferramenta teórica, mas uma solução prática e altamente adaptável para modelar e otimizar sistemas com eventos discretos. Elas nos permitem simular e analisar o comportamento de sistemas sujeitos a mudanças e falhas, oferecendo insights valiosos para melhorar a eficiência e a produtividade. Por essas razões, no próximo capítulo, adotaremos as RP como a principal ferramenta de estudo. Com isso, seremos capazes de realizar uma análise aprofundada das interações dentro dos sistemas de manufatura, identificando pontos críticos que impactam diretamente seu desempenho, ao mesmo tempo em que propomos melhorias para alcançar um processo de produção mais robusto e eficiente.

2 Redes de Petri: conceitos iniciais

As RPs constituem uma ferramenta matemática e gráfica de alta expressividade, empregada na modelagem, análise e simulação de Sistemas a Eventos Discretoss (SEDs). Introduzidas por Carl Adam Petri em 1962, essas redes têm sido amplamente adotadas para representar sistemas concorrentes, tanto síncronos quanto assíncronos, além de sistemas distribuídos. Seu emprego se destaca em áreas como automação industrial, protocolos de comunicação, controle de manufatura e gerenciamento de fluxos de trabalho em sistemas de informação.

2.1 REPRESENTAÇÕES FORMAIS DA REDE DE PETRI

Para formalizar a modelagem, a Rede de Petri pode ser descrita por diferentes estruturas matemáticas, dependendo do nível de detalhe necessário. Em literatura, a definição formal de uma Rede de Petri pode variar conforme o nível de detalhamento e os aspectos que se deseja enfatizar na modelagem. Assim, podem ser apresentadas diferentes definições, tais como no conceito de uma n-tupla. O termo n-upla refere-se ao número de elementos considerados na definição da RP, podendo assumir diferentes configurações:

- **Dupla (2 elementos)**: estrutura básica da rede;
- **Trípla (3 elementos)**: inclui conexões entre os componentes;
- **Quádrupla (4 elementos)**: adiciona a função de peso dos arcos;
- **Quíntupla (5 elementos)**: incorpora a marcação inicial da rede;
- **Séxtupla (6 elementos)**: inclui informações complementares, como rótulos e tempos.

RP COMO UMA DUPLA

Para incluir a noção de pesos associados aos arcos, pode-se definir:

$$RP = (P, T)$$

em que:

- P é o conjunto finito de *lugares*;
- T é o conjunto finito de *transições*;

RP COMO UMA TRIPLA

Para incluir a noção de pesos associados aos arcos, pode-se definir:

$$RP = (P, T, F)$$

em que:

- P, T possuem os mesmos significados anteriores;
- $F \subseteq (P \times T) \cup (T \times P)$ é o conjunto de *arcos*, que conecta lugares a transições e vice-versa;

RP COMO UMA QUADRUPLA

Para incluir a noção de pesos associados aos arcos, pode-se definir:

$$RP = (P, T, F, W)$$

em que:

- P, T e F possuem os mesmos significados anteriores;
- $W : F \rightarrow \mathbb{N}^+$ é uma função que atribui um *peso* a cada arco, representando, por exemplo, a quantidade mínima de *fichas* necessários para que uma transição seja disparada.

RP COMO UMA QUÍNTUPLA

A definição clássica de uma Rede de Petri integra tanto os pesos quanto a marcação inicial:

$$RP = (P, T, F, W, M_0)$$

- P, T, F e W possuem os mesmos significados anteriores;
- $M_0 : P \rightarrow \mathbb{N}$ é a *marcação inicial*, definindo a distribuição inicial de *fichas* em cada lugar.

RP COMO UMA SÉXTUPLA

Em contextos que demandam uma modelagem mais rica, como em sistemas temporizados ou hierárquicos, pode-se estender a definição para:

$$RP = (P, T, F, W, M_0, \Lambda)$$

em que:

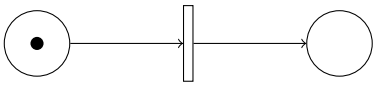
- P, T, F, W e M_0 : possuem os mesmos significados anteriores;
- Λ é uma função adicional que pode representar rótulos, restrições, tempos de disparo, prioridades ou quaisquer outras informações complementares necessárias para descrever aspectos específicos do sistema modelado.

Cada uma dessas definições é útil em contextos distintos, permitindo que o modelador/projetista escolha a abordagem que melhor se adequa à complexidade e aos requisitos do sistema a ser analisado.

2.2 REPRESENTAÇÃO GRÁFICA

Uma RP pode ser representada graficamente conforme a Figura 2. Os lugares são representados por círculos, as transições por retângulos e as fichas (*fichas*) por pontos dentro dos círculos.

Figura 2 – Exemplo de Rede de Petri simples



Fonte: Autor, (2025)

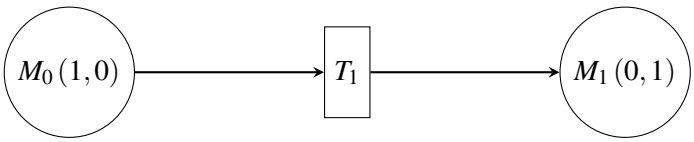
2.2.1 FUNCIONAMENTO CONCEITUAL DE UMA RP

Na Figura 3 ilustra-se a evolução de uma Rede de Petri simples. Considere uma rede com dois lugares (P_1 e P_2) e uma transição (T_1). A marcação inicial possui um ficha em P_1:

1. Estado inicial: $M_0 = (1,0)$ (um ficha em P_1, nenhum em P_2).
2. A transição T_1, estando habilitada, pode disparar, removendo um ficha de P_1 e adicionando um ficha em P_2.
3. Novo estado: $M_1 = (0,1)$ (o ficha é transferido para P_2).

Esse processo ilustra a evolução do sistema conforme os eventos ocorrem, permitindo uma análise dinâmica e detalhada do comportamento da rede.

Figura 3 – Grafo de marcações ilustrando a evolução do sistema



Fonte: Autor, (2025)

Nesse contexto, podem ser realizadas a modelagem e análise de sistemas de manufatura, a simulação de protocolos de comunicação, a modelagem de processos de *workflow* (fluxo de trabalho) e *Business Process Management - BPM* (Gerenciamento de Processos de Negócio), sistemas embarcados e de tempo real, e automação de processos industriais, dentre outros. Esses sistemas contém mais recursos, eventos e condicionantes para que cada evento ocorra.

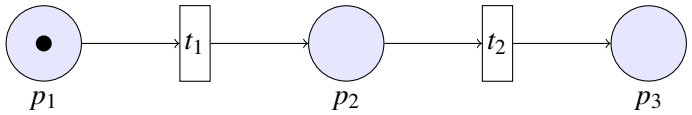
2.3 PODER DE EXPRESSÃO DAS REDES DE PETRI

As RPs possuem um alto poder de expressão, sendo capazes de modelar diversos comportamentos de sistemas concorrentes. A seguir, ilustramos alguns desses comportamentos utilizando grafos de RPs.

2.3.1 SEQUÊNCIA

Um processo sequencial é aquele no qual uma transição só pode ocorrer após a execução da anterior, conforme ilustrado na Figura 4.

Figura 4 – Expressão de processo em sequência numa RP

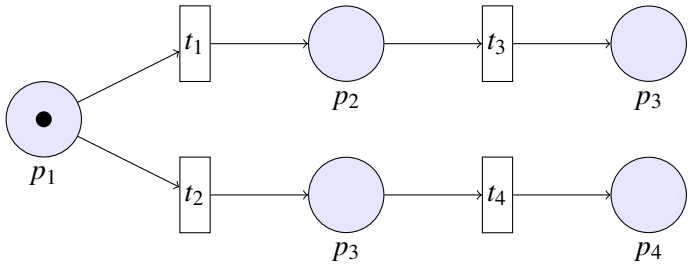


Fonte: Autor, (2025)

2.3.2 CONCORRÊNCIA

Duas transições independentes podem ocorrer simultaneamente: t_1 e t_2, conforme ilustrado na Figura 5.

Figura 5 – Expressão de processo em concorrência numa RP

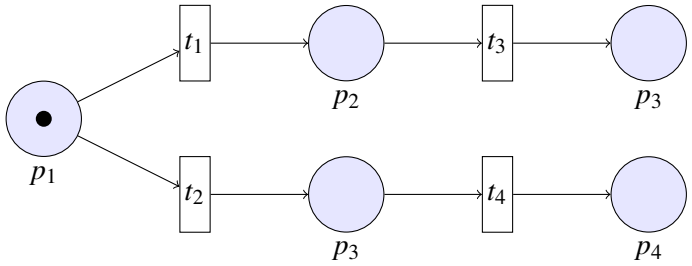


Fonte: Autor, (2025)

2.3.3 CONFLITO

Duas transições disputam o mesmo recurso e apenas uma pode ser disparada: t_1 ou t_2, conforme ilustrado na Figura 6.

Figura 6 – Expressão de processo em conflito numa RP

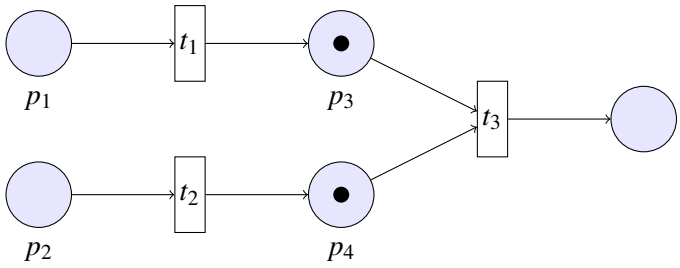


Fonte: Autor, (2025)

2.3.4 SINCRONISMO

Dois lugares (p_3 e p_4) habilitam simultaneamente uma mesma transição t_3, conforme ilustrado na Figura 7.

Figura 7 – Expressão de processo em sincronismo numa RP

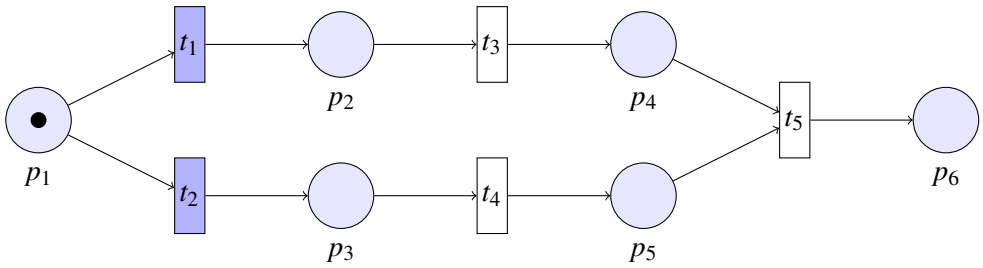


Fonte: Autor, (2025)

2.3.5 CAMINHOS ALTERNATIVOS

O sistema pode seguir diferentes fluxos dependendo da transição escolhida: t_1 ou t_2, conforme ilustrado na Figura 8.

Figura 8 – Expressão de processo em Caminhos Alternativos numa RP

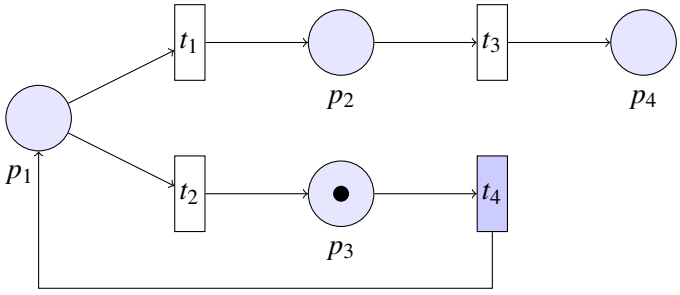


Fonte: Autor, (2025)

2.3.6 REPETIÇÃO

Um ciclo em que a execução retorna a um estado anterior disparando a transição t_4, conforme ilustrado na Figura 9.

Figura 9 – Expressão de processo de Repetição numa RP

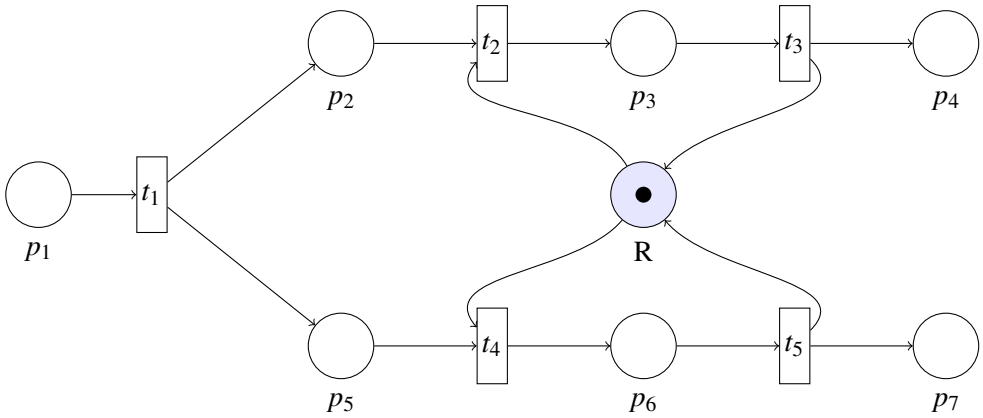


Fonte: Autor, (2025)

2.3.7 ALOCAÇÃO DE RECURSOS

Um recurso compartilhado precisa ser liberado antes de ser reutilizado. Na Figura **??** as transições t_2 ou t_4 alocam o recurso R no processo ora modelado. O recurso é liberado pelas transições t_3 ou t_5, respectivamente. Este processo é ilustrado na Figura 10.

Figura 10 – Expressão de Alocação de Recursos numa RP



Fonte: Autor, (2025)

2.4 DINÂMICA E PROPRIEDADES DAS REDES DE PETRI

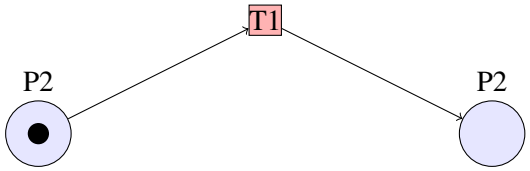
Conforme descrito anteriormente, as RP são modelos matemáticos poderosos para a descrição e análise de sistemas distribuídos e concorrentes, e suas propriedades fundamentais são essenciais para a compreensão do comportamento dinâmico desses sistemas. A seguir, exploramos as propriedades mais importantes das RPs, ilustrando com exemplos práticos e abordagens analíticas.

2.4.1 ALCANCE

O alcance de uma Rede de Petri refere-se ao conjunto de todas as marcações atingíveis a partir da marcação inicial M_0. Em outras palavras, é a coleção de todos os estados do sistema que podem ser alcançados por uma sequência de disparos de transições. O alcance pode ser representado visualmente por meio de um grafo de marcações, em que cada nó do grafo representa uma marcação e cada aresta entre os nós indica a transição disparada para alcançar a próxima marcação.

Exemplo: Considere uma RP simples com dois lugares P_1 e P_2, e uma transição T_1 que conecta esses lugares. Se o sistema começa com uma ficha em P_1, a única marcação atingível após um disparo de T_1 será P_2, e a marcação inicial (P_1) terá sido consumida enquanto uma ficha foi colocada em P_2. O alcance pode ser representado graficamente, com um nó para cada marcação possível. Esse processo está representado na Figura 11.

Figura 11 – Alcançabilidade de uma RP



Fonte: Autor, (2025)

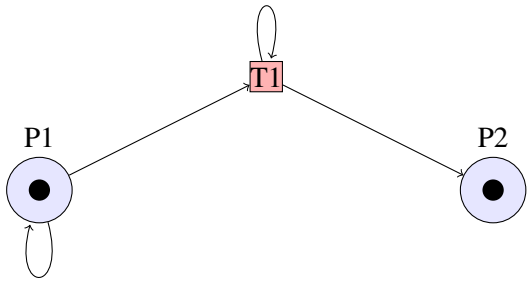
2.4.2 CONSERVAÇÃO

Uma Rede de Petri é dita conservativa se, ao longo de todas as transições disparadas no sistema, a soma total de fichas nos lugares (ou seja, a quantidade total de fichas presentes) permanece constante. Esta

propriedade é crucial em sistemas que envolvem conservação de recursos, como sistemas de produção, em que os recursos (fichas) não podem ser criados ou destruídos.

Exemplo: Em um sistema de manufatura modelado por uma RP, a quantidade total de matérias-primas ou produtos deve ser conservada durante o processo de produção. Se a rede é conservativa, a quantidade de fichas nos lugares de entrada e saída será sempre a mesma, independentemente de quantas transições tenham ocorrido. Esse processo está representado na Figura 12.

Figura 12 – Conservação de uma RP



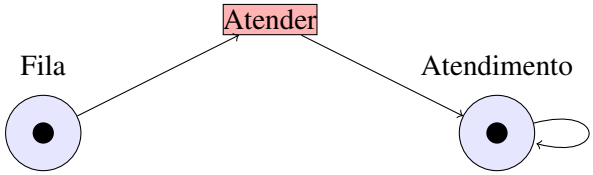
Fonte: Autor, (2025)

2.4.3 VIVACIDADE

A vivacidade de uma transição indica que ela pode, eventualmente, ser disparada, independentemente da sequência de transições já disparadas. Em termos mais formais, uma transição é viva se sempre existir uma sequência futura de disparos de transições que a habilitarão novamente. Esse conceito é fundamental em sistemas em que é necessário garantir que todas as operações possam ser realizadas, mesmo em cenários complexos de concorrência.

Exemplo: Em um sistema de fila, a transição que representa o atendimento de clientes deve ser viva, o que significa que, mesmo que os clientes cheguem em ordens diferentes, sempre haverá uma sequência de disparos que permitirá o atendimento de todos, sem que o sistema entre em um estado de "travamento". Esse processo está representado na Figura 13.

Figura 13 – Vivacidade de uma RP



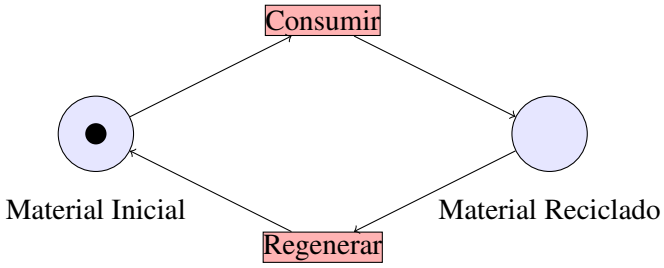
Fonte: Autor, (2025)

2.4.4 REVERSIBILIDADE

Uma rede é reversível se, para qualquer marcação atingível, é possível retornar à marcação inicial M_0. Ou seja, a reversibilidade implica que o sistema pode retornar ao seu estado inicial após um número finito de transições. Esta propriedade é especialmente importante em sistemas de controle e sistemas em que o restabelecimento de condições iniciais é necessário, como em processos de reinicialização ou recuperação de falhas.

Exemplo: Em um processo de reciclagem modelado por uma RP, em que materiais são consumidos e posteriormente regenerados, a reversibilidade garantiria que o sistema possa sempre retornar ao estado inicial de recursos, após a execução de um ciclo completo de transições. Esse processo está representado na Figura 14.

Figura 14 – Reversibilidade de uma RP



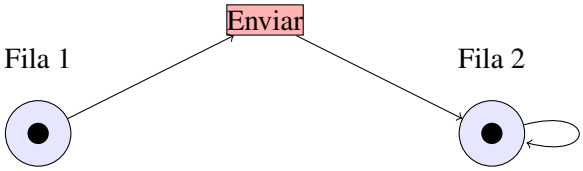
Fonte: Autor, (2025)

2.4.5 SEGURANÇA

A segurança de uma Rede de Petri refere-se à restrição de que nenhum lugar pode conter mais do que um número máximo específico de fichas durante a execução. Este conceito é vital para sistemas em que a quantidade de recursos em cada lugar deve ser controlada para evitar sobrecarga ou uso excessivo, como no controle de inventários ou sistemas de comunicação com limites de capacidade.

Exemplo: Em um sistema de controle de fluxo de mensagens, em que cada lugar representa uma fila de mensagens, a segurança garantiria que nenhuma fila contenha mais de um número máximo de mensagens, evitando congestionamentos e perdas de dados. Esse processo está representado na Figura 15.

Figura 15 – Segurança de uma RP



Fonte: Autor, (2025)

Além dessas propriedades fundamentais, as Redes de Petri podem ser caracterizadas por outros aspectos mais avançados, como a *liveness-preserving* (preservação de vivacidade) e a análise de dependências temporais e causais entre transições. Em muitos casos, a análise formal dessas propriedades requer ferramentas de verificação automáticas, como as oferecidas por simuladores como o CPN Tools, ou através de técnicas analíticas que permitem garantir a correção e a confiabilidade do sistema modelado.

O estudo dessas propriedades é crucial para garantir que a modelagem de sistemas complexos utilizando Redes de Petri seja efetiva, permitindo não apenas uma descrição precisa dos comportamentos do sistema, mas também uma análise e otimização do desempenho, segurança e confiabilidade do sistema como um todo.

2.5 CONCLUSÃO

As Redes de Petri se apresentam como uma ferramenta poderosa para a modelagem de sistemas dinâmicos discretos, permitindo a análise de concorrência, sincronização e recursos compartilhados. O formalismo matemático empregado facilita a verificação de propriedades cruciais para a confiabilidade de sistemas computacionais e industriais. Ao combinar o uso de invariantes de lugar e de transição, por exemplo, é possível realizar uma verificação formal de propriedades como segurança e vivacidade. Este capítulo estabeleceu os fundamentos teóricos das Redes de Petri, explorando diversas definições formais que variam conforme o nível de detalhamento, e apresentou suas principais aplicações e propriedades, servindo como base para estudos mais avançados, incluindo modelos estendidos, como Redes de Petri Coloridas e Temporizadas.

3 Redes de Petri: modelos iniciais

3.1 CASO PRÁTICO INDUSTRIAL: REPRESENTAÇÃO EM RP

Considere uma fábrica de montagem em que o processo produtivo é dividido em cinco etapas, cada uma representada por um *lugar* na rede. As etapas são:

- P_1: Recebimento de matéria-prima;
- P_2: Processamento inicial;
- P_3: Inspeção de qualidade;
- P_4: Montagem final;
- P_5: Expedição do produto acabado.

Os *eventos* (ou transições) que conectam essas etapas são:

- T_1: Inicia o processamento da matéria-prima, direcionando-a tanto para o processamento inicial quanto para a inspeção preliminar (um fluxo concorrente);
- T_2: Conclui o processamento inicial e encaminha o material para uma etapa de inspeção ou preparação adicional;
- T_3: Realiza uma inspeção complementar, garantindo a qualidade antes da montagem;
- T_4: Coordena a montagem final do produto;
- T_5: Finaliza o processo, liberando o produto para expedição e, eventualmente, retornando uma parte dos recursos para reinício do ciclo.

A rede em questão modela, portanto, o fluxo de fichas (representando, por exemplo, lotes ou unidades de produto) através dessas etapas, permitindo identificar gargalos, verificar a eficiência do fluxo e garantir que a qualidade seja mantida ao longo do processo produtivo.

3.1.1 ESTRUTURA DA REDE DE PETRI

A rede que descreve o processo acima é composta por:

- **Lugares:** $P = \{P_1, P_2, P_3, P_4, P_5\}$;
- **Transições:** $T = \{T_1, T_2, T_3, T_4, T_5\}$;
- **Arcos:** A conexão entre os elementos é dada por:

$$\begin{aligned} F \quad = \quad & \{(P_1,T_1),\ (T_1,P_2),\ (T_1,P_3),\\ & (P_2,T_2),\ (T_2,P_4),\ (P_3,T_3),\ (T_3,P_4),\\ & (P_4,T_4),\ (T_4,P_5),\ (P_5,T_5),\ (T_5,P_1)\}. \end{aligned}$$

Note que a transição T_1 distribui o fluxo de matéria-prima simultaneamente para duas rotas: uma para processamento inicial e outra para uma inspeção preliminar.

- **Marcação Inicial:** A fábrica inicia com a chegada de um lote de matéria-prima:

$$M_0: \quad M_0(P_1) = 1, \quad M_0(P_2) = M_0(P_3) = M_0(P_4) = M_0(P_5) = 0.$$

- **Função de Peso:** Assume-se que cada arco possui peso unitário:

$$W : F \to \mathbb{N}^+, \quad \text{com } W(e) = 1 \text{ para todo } e \in F.$$

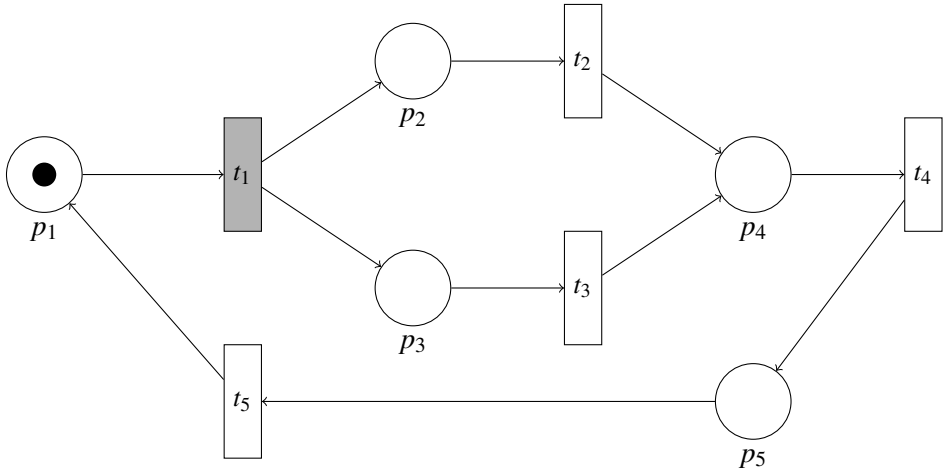
- **Função de Rótulo:** Para complementar a modelagem, associamos rótulos às transições:

$$\Lambda : T \to \{\text{descrições}\}, \quad \text{por exemplo, } \Lambda(T_i) = \text{“Etapa } T_i\text{”}, \; i = 1, \ldots, 5.$$

REPRESENTAÇÃO GRÁFICA DA RP PROPOSTA

Na Figura 16 é apresentada graficamente a RP em questão. Nesta figura, os lugares são representados por círculos e as transições por barras (retângulos):

Figura 16 – Diagrama gráfico da RP representando o fluxo de produção industrial



Fonte: Autor,(2025)

Note, na Figura 16, que os nós (lugares p_i e transições t_i) são conectados por arcos, seguindo fielmente o processo industrial descrito no início desta seção. Além disso, observe a representação da marcação inicial da RP, contendo uma ficha única no lugar p_1(círculo denso de cor preta), a ausência de fichas nos demais lugares p_i. Perceba também que todos os arcos têm peso $W = 1$. A transição t_1 destacada por uma moldura cinza sinaliza sua condição de habilitação para disparo, portanto, de ocorrência do evento.

3.2 MATRIZES DE PRÉ, PÓS E DE INCIDÊNCIA DE UMA REDE DE PETRI

As redes de Petri são uma poderosa ferramenta para modelar sistemas dinâmicos e discretos, como processos de produção, controle de tráfego, e até sistemas computacionais. Elas são compostas por lugares, transições e arcos, em que os lugares representam estados ou recursos, as transições representam eventos ou ações, e os arcos indicam como os lugares e transições se interconectam.

Uma rede de Petri pode ser analisada por meio de suas matrizes de incidência, que fornecem informações cruciais sobre as relações entre lugares e transições. As três matrizes mais importantes em uma rede de Petri são:

- **Matriz de Pré-Incidência** (C^-), que descreve as conexões entre os lugares e as transições para as quais esses lugares servem de entrada.
- **Matriz de Pós-Incidência** (C^+), que descreve as conexões entre as transições e os lugares que recebem fichas após a execução de uma transição.
- **Matriz de Incidência** (C), que é a diferença entre as matrizes de pós e pré-incidência, e descreve de forma completa a interação entre lugares e transições.

Essas matrizes são fundamentais para a análise de comportamento dinâmico de sistemas modelados por redes de Petri, especialmente em termos de controle e verificação de propriedades como alcance de lugares e sincronização de transições.

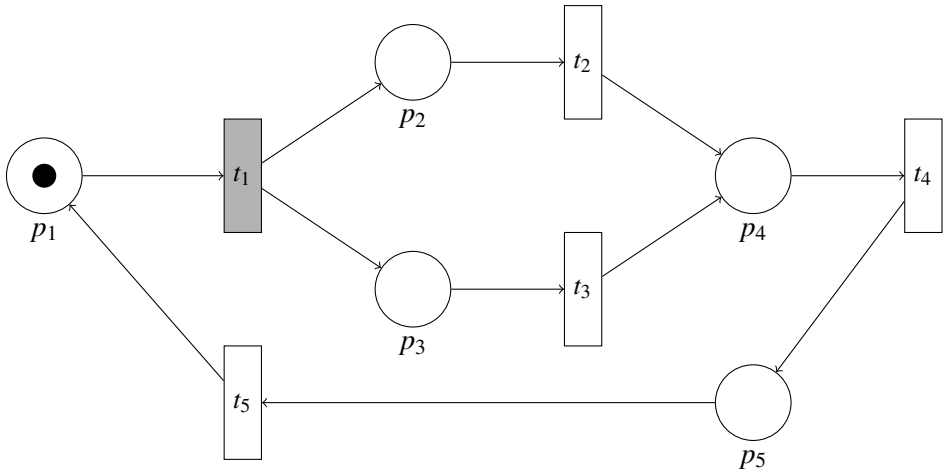
3.2.1 MODELAGEM DAS MATRIZES DE UMA RP

Consideremos a RP da Figura 16, apresentada na Secção 3.1.1, cujas características básicas são:

- p_1 é um lugar de entrada que tem uma ficha inicial.
- p_2 e p_3 são lugares que representam dois caminhos paralelos a partir de T_1.
- p_4 e p_5 são lugares de saída.
- As transições t_1, t_2, t_3, t_4, t_5 são eventos que conectam os lugares.

A referida figura é novamente posta a seguir ,visando facilitar a percepção das matrizes de incidência.

Diagrama gráfico da RP representando o fluxo de produção industrial (Figura 16 da seção 3.1.1.)



Fonte: Autor,(2025)

3.2.2 MATRIZ DE PRÉ-INCIDÊNCIA (C^-)

A matriz de pré-incidência C^- é uma matriz $p \times t$, em que p é o número de lugares e t é o número de transições. Ela descreve a relação entre os lugares e as transições na direção dos arcos que vão dos lugares para as transições. Em outras palavras, cada elemento C^-_{ij} indica o número de fichas que um lugar P_i envia para a transição T_j.

Para a RP apresentada acima, a matriz de pré-incidência C^- seria apresentado conforme (1):

$$C^- = \begin{pmatrix} & t_1 & t_2 & t_3 & t_4 & t_5 \\ p_1 & 1 & 0 & 0 & 0 & 0 \\ p_2 & 0 & 1 & 0 & 0 & 0 \\ p_3 & 0 & 0 & 1 & 0 & 0 \\ p_4 & 0 & 0 & 0 & 1 & 0 \\ p_5 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{1}$$

Cada linha representa um lugar, e cada coluna representa uma transição. Os valores 1 indicam que há um arco conectando o lugar à transição correspem quente.

3.2.3 MATRIZ DE PÓS-INCIDÊNCIA (C^+)

A matriz de pós-incidência C^+ também é uma matriz $p \times t$, mas ela descreve as conexões entre as transições e os lugares que recebem fichas após a execução de uma transição. Em outras palavras, cada elemento C^+_{ij} indica o número de fichas que a transição T_j envia para o lugar P_i.

Para a rede apresentada, a matriz de pós-incidência C^+ seria apresentado conforme (2):

$$C^+ = \begin{pmatrix} & t_1 & t_2 & t_3 & t_4 & t_5 \\ p_1 & 0 & 0 & 0 & 0 & 1 \\ p_2 & 1 & 0 & 0 & 1 & 0 \\ p_3 & 1 & 0 & 0 & 0 & 0 \\ p_4 & 0 & 1 & 1 & 0 & 0 \\ p_5 & 0 & 0 & 0 & 1 & 0 \end{pmatrix} \tag{2}$$

Assim como na matriz de pré-incidência, os valores 1 indicam a direção de fluxo de fichas das transições para os lugares.

3.2.4 MATRIZ DE INCIDÊNCIA (C)

A matriz de incidência C é dada pela diferença entre as matrizes de pós-incidência e pré-incidência. Ela descreve a relação completa entre lugares e transições, considerando tanto os arcos de entrada (pré-incidência) quanto os de saída (pós-incidência), conforme apresentado conforme (3):

$$C = C^+ - C^- = \begin{pmatrix} & t_1 & t_2 & t_3 & t_4 & t_5 \\ p_1 & -1 & 0 & 0 & 0 & 1 \\ p_2 & 1 & -1 & 0 & 0 & 0 \\ p_3 & 1 & 0 & -1 & 0 & 0 \\ p_4 & 0 & 1 & 1 & -1 & 0 \\ p_5 & 0 & 0 & 0 & 1 & -1 \end{pmatrix} \tag{3}$$

Na matriz de incidência, o valor -1 indica um arco de entrada, o valor 1 indica um arco de saída, e o valor 0 indica que não há arco entre o lugar e a transição.

As matrizes de pré, pós e de incidência são fundamentais para a análise quantitativa de RPs. Elas permitem a descrição das interações entre os lugares e transições e são usadas em diversas aplicações, como análise de alcançabilidade, verificações de segurança e determinação de ciclos de eventos em sistemas dinâmicos. Por meio dessas matrizes, é possível modelar e compreender o comportamento de sistemas complexos de maneira eficaz e matemática.

4 RP temporizadas

Conforme já definido anteriormente, as RPs são uma poderosa ferramenta para modelagem e análise de sistemas concorrentes e distribuídos. No entanto, em muitas aplicações práticas, a dimensão temporal do sistema é crucial, como em sistemas de controle industrial, redes de comunicação e sistemas de produção. Para incorporar aspectos temporais às RP clássicas, surgiram extensões como as *Redes de Petri Temporais* e as *Redes de Petri Temporalizadas*.

4.1 REDES DE PETRI TEMPORAIS

4.1.1 DEFINIÇÃO

Uma Rede de Petri Temporal (RP-T) é uma extensão das RPs clássicas em que são atribuídos intervalos de tempo mínimo e/ou máximo para a condição de disparo das transições. Isso significa que uma transição, uma vez habilitada, deve esperar um tempo mínimo antes de disparar, mas não pode ultrapassar um tempo máximo.

Formalmente, uma RP-T é definida como um par $T : T \to \mathbb{R}^+$ em que, para cada transição $t \in T$, há um tempo associado (α, β), em que:

- α é o tempo mínimo que deve decorrer antes que a transição possa disparar.
- β é o tempo máximo permitido para que a transição dispare.

EXEMPLO DE REDE DE PETRI TEMPORAL

Considere um sistema de produção com duas etapas: **montagem** e **inspeção**. A montagem leva entre 3 e 6 segundos, e a inspeção entre 2 e 4 segundos.

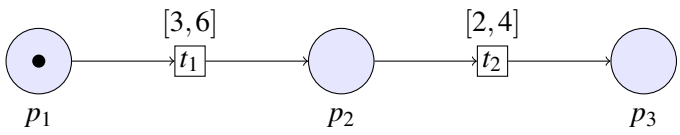
Representamos essa dinâmica por uma RP-T:

- t_1: Montagem, com intervalo de tempo $[3,6]$.
- t_2: Inspeção, com intervalo de tempo $[2,4]$.

Isso significa que, após a habilitação de t_1, deve-se esperar pelo menos 3 segundos antes de dispará-la, mas não mais que 6 segundos. O mesmo ocorre para t_2.

O processo acima descrito é representado graficamente conforme Figura 17.

Figura 17 – Representação gráfica de um RP-T.



Fonte: Autor,(2025)

4.1.2 PROPRIEDADES DAS REDES DE PETRI TEMPORAIS

As RP-Ts preservam muitas propriedades das RP clássicas, mas também introduzem novos desafios, como:

- **Bloqueio Temporal**: Se uma transição perder seu intervalo de disparo, o sistema pode ficar preso.
- **Concorrência Temporizada**: O tempo influencia quais transições disparam primeiro em cenários concorrentes.

4.2 REDES DE PETRI TEMPORALIZADAS

4.2.1 DEFINIÇÃO

As Rede de Petri Temporizadas (RP-Tzs) diferem das RP-Ts por associarem tempos **não às transições**, mas às fichas (*tokens*). Isso permite modelar fenômenos em que o tempo é uma propriedade das entidades que fluem no sistema, como processos industriais ou redes de transporte.

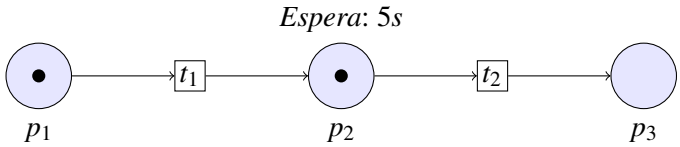
EXEMPLO DE REDE DE PETRI TEMPORALIZADA

Seja um sistema de entrega de pacotes no qual cada pacote tem um tempo de processamento antes de ser enviado. Podemos modelar isso com fichas que carregam *timestamps*(marca temporal)[1]:

- p_1: Local de coleta
- p_2: Processamento (com tempo de retenção de 5 segundos)
- p_3: Entrega

Aqui, quando uma ficha chega a p_2, ela deve aguardar 5 segundos antes que t_2 possa movê-la para p_3. Esse processo descrito é representado graficamente na Figura 18.

Figura 18 – Representação gráfica de um RP-Tz.



Fonte: Autor,(2025)

4.2.2 VANTAGENS DAS REDES DE PETRI TEMPORALIZADAS

- Modelam sistemas em que a temporalidade é inerente às entidades.
- Evitam bloqueios causados por tempos fixos nas transições.

[1] Um *timestamp* (marca temporal) representa um valor de tempo associado a um evento ou elemento do sistema, indicando quando algo ocorreu ou quando será processado.

4.3 COMPARATIVO ENTRE RP-T E RP-TZ

Característica	RP-T	RP-Tz
Tempo associado a	Transições (t)	Fichas (*tokens*)
Controle de fluxo	Baseado em intervalos fixos	Baseado no tempo de vida da ficha
Aplicabilidade	Processos industriais, workflow	Redes de comunicação, transporte

Tabela 2 – Comparativo entre RPT e RPTz

4.4 CONCLUSÃO

Redes de Petri Temporais e Temporalizadas são ferramentas fundamentais para modelagem de sistemas dinâmicos com restrições temporais. Enquanto as primeiras são mais indicadas para processos de controle e produção, as segundas são essenciais para modelar sistemas onde o tempo é uma propriedade das entidades que fluem.

No próximo capítulo, exploraremos algoritmos de análise para essas redes e suas aplicações na otimização de sistemas reais.

5 Redes de Petri Hierárquicas

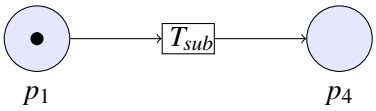
As Redes de Petri Hierárquicas (RPH) estendem as Redes de Petri convencionais permitindo a modelagem em diferentes níveis de abstração. Essa abordagem facilita a decomposição de sistemas complexos em componentes mais gerenciáveis.

5.1 DEFINIÇÃO

Uma RPH consiste em uma estrutura em que certos componentes (sub-redes) podem ser representados como transições abstratas em um nível superior. Isso permite uma visão modular e organizada do sistema modelado.

Sejam as figuras 19 e 20 a seguir. A Figura 19 ilustra uma RP em que é representado um processo em nível supeiror como entrada e saída de um sub-processo que ocorre num nível de sub-rede. A Figura 20, por sua vez, representa esse processo no sub-nível da RP.

Figura 19 – Rede de Petri Hierárquica - Nível Superior

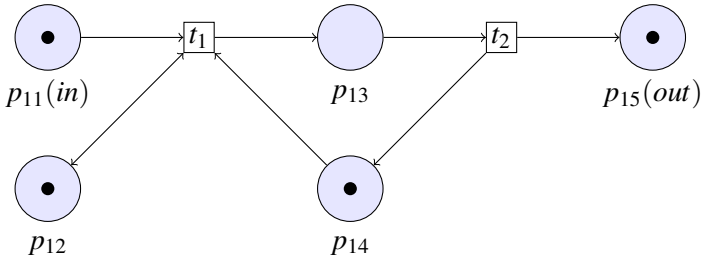


Fonte: Autor,(2025)

Aqui, a transição T_{sub} representa uma sub-rede detalhada em outro nível, cuja a estrutura interna da transição T_{sub} exibe sua funcionalidade interna.

Perceba, na Figura 20, que os lugares p_{11} e p_{15} estão referenciados com as inscrições *in* e *out*, respectivamente. Essas inscrições referem-se aos lugares de entrada e saída da transição T_{sub} que estão representados na Figura 19 por p_1 e p_4. Isso significa que para cada lugar de entrada e saída p_i na transição T_{sub}, haverá um lugar p_j correspondente dentro da sub-rede modelada ($\forall\, i, j\ \in \mathbb{N}$).

Figura 20 – Rede de Petri Hierárquica - Nível de sub-rede



Fonte: Autor,(2025)

As Redes de Petri Hierárquicas permitem uma organização mais clara e a reutilização de componentes em sistemas de grande porte.

6 CPN IDE®

Com o avanço das tecnologias de modelagem e simulação, a análise de sistemas complexos tornou-se uma tarefa cada vez mais acessível e eficiente. Nesse contexto, o CPN Tools® consolidou-se como uma ferramenta amplamente reconhecida na comunidade de Redes de Petri, oferecendo um ambiente robusto para a construção, simulação e análise de modelos de Redes de Petri Coloridas (RPC) (*Coloured Petri Nets* – CPN). No entanto, a manutenção e expansão dessa ferramenta tornaram-se progressivamente mais desafiadoras devido à sua implementação na linguagem BETA e à utilização do sistema de desenvolvimento Mjølner, ambos pouco adotados atualmente, conforme publicizado no sitio do desenvolvedor disponível em <https://cpnide.org/>.

Para superar essas limitações e garantir a evolução contínua da ferramenta, foi introduzido o CPN IDE®, que substitui o CPN Tools® como uma solução moderna para edição e simulação de modelos de RPCs. A principal vantagem do CPN IDE® é sua extensibilidade, permitindo a incorporação de novas funcionalidades e garantindo compatibilidade com sistemas operacionais mais recentes. Diferentemente do CPN Tools®, que dependia de um editor baseado em BETA, o CPN IDE® adota uma abordagem modular e moderna, utilizando um editor baseado em JavaScript®, que se comunica com um controlador Java por meio de uma interface REST[2] . Esse controlador, por sua vez, utiliza o Access/CPN, que encapsula o simulador ML do CPN Tools® dentro de uma implementação em Java.

Conforme destacam os desenvolvedores, a transição do CPN Tools para o CPN IDE foi motivada pela necessidade de garantir a longevidade da ferramenta, especialmente para aplicações na área de mineração de processos. No Departamento de Matemática e Ciência da Computação da Eindhoven University of Technology (TU/e), onde o CPN Tools era amplamente utilizado no ensino e na pesquisa, a preocupação com a obsolescência tecnológica levou ao desenvolvimento do CPN IDE como uma alternativa mais sustentável e expansível. Além de manter as funcionalidades do CPN Tools, o CPN IDE facilita a integração com ferramentas como o **ProM**[3], possibilitando, a descoberta de uma Rede de Petri a partir de *logs* de eventos, sua edição e posterior análise de conformidade por meio da simulação.

6.1 CPN IDE: INTERFACE

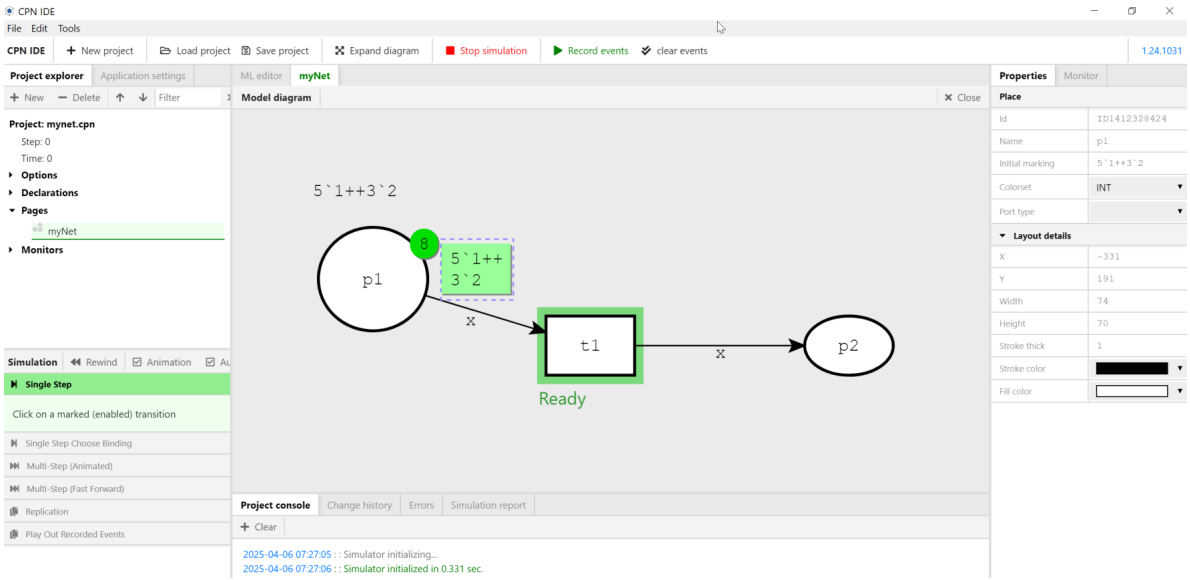
Na Figura 21 é apresentada a área interface de desenvolvimento integrado do *software* CPN IDE®. Perceba, ao centro do figura em questão, a área de trabalho contendo três elementos em destaque: a) p_1; b) t_1 e c) p_2. Note que a transição t_1 contém uma moldura verde (cor utilizada por padrão neste *software* em modo de animação) indicando seu estado atual - habilitada para o disparo. Além disso, perceba que há oito fichas no lugar p_1, sendo cinco fichas de valor 1 e outras três fichas de valor 2. Finalmente, note os arcos que conecam p_1 a t_1 e t_1 a p_2. Esses arcos contém uma definição de variável x que identifica o peso correspondente ao valor de cada ficha presente em p_1. Isso é possível, porque no CPN IDE®, cada lugar

[2] REST (*Representational State Transfer* - Transferência Representacional de Estado) é uma arquitetura de desenvolvimento de sistemas distribuídos, baseada em operações como GET, POST, PUT e DELETE, permitindo a comunicação entre aplicações através de padrões HPPT (*HyperText Transfer Protocol* - Protocolo de Transferência de Hipertexto).

[3] ProM é uma plataforma de mineração de processos que oferece uma coleção extensível de algoritmos para análise de *logs* de eventos, permitindo descobrir, monitorar e melhorar processos de negócios com base em dados reais extraídos de sistemas de informação. Disponível em: <https://www.promtools.org/>.

está definido com um tipo de cor (conjunto de ficha), neste caso, os lugares p_1 e p_2 estão definidos como tipo inteiros (INT).

Figura 21 – Interface de desenvolvimento integrado do *software* CPN IDE®

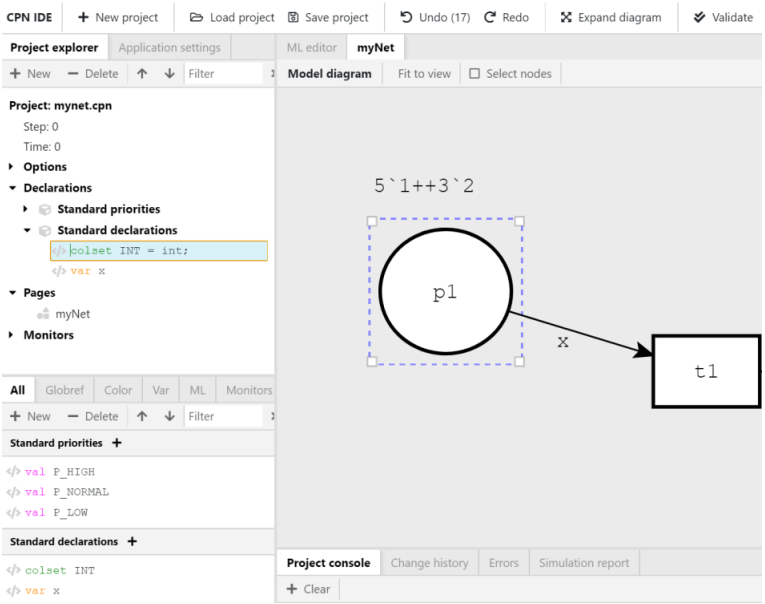


Fonte: Autor,(2025)

Um conjunto de cores foi definido: INT, do tipo inteiro. Assim, todos os lugares da rede possuem o conjunto de cores INT associado. Além disso, foi também definida a variável x, que é do tipo INT.

Esse processo de inserção de declarações é realizado no menu *Project explorer* do CPN IDE®, conforme destacado na Figura 22, o qual será detalhado na Seção 6.4.

Figura 22 – Área de Declarações no CPN IDE®



Fonte: Autor,(2025)

6.2 FUNÇÕES BÁSICAS DO CPN IDE

As principais funcionalidades do **CPN IDE** incluem:

- **Criação e edição de modelos**: realizada por meio de um editor gráfico especializado em RPCs. Esse editor permite desenhar a rede, definir atributos dos elementos e adicionar declarações na linguagem **CPN ML**. O modelo pode ser organizado em múltiplas páginas interconectadas, formando uma estrutura hierárquica.
- **Simulação do comportamento do modelo**: possibilita a depuração e a análise dinâmica. O CPN IDE oferece, assim como seu antecessor - CPN Tools, simulação passo a passo para testes detalhados e simulação automática com um número predefinido de passos. Para análises estatísticas, a simulação em grandes intervalos de tempo possibilita a estimativa de métricas como taxa de transferência e **QoS**[4].
- **Geração e análise do espaço de estados**: viável para modelos simples, produzindo relatórios contendo propriedades essenciais das RPs, como *boundedness* (limitação) e *liveness* (vivacidade). Além disso, disponibiliza uma linguagem baseada em **CPN ML** para consultas personalizadas sobre propriedades específicas do espaço de estados. No entanto, para modelos mais complexos, a explosão do espaço de estados pode tornar essa abordagem impraticável.

6.3 LINGUAGEM DO CPN IDE: CPN ML

O CPN IDE utiliza a linguagem de programação **CPN ML** para definir declarações e inscrições na rede. Essa linguagem permite especificar **conjuntos de cores** (tipos de dados), variáveis, funções e constantes. Cada **lugar** na rede é associado a um conjunto de cores específico, restringindo as fichas que ele pode conter. Já variáveis e funções são utilizadas nas inscrições de transições e arcos.

As declarações fazem parte da estrutura da rede e estão localizadas na área de **índice**. O **CPN ML** inclui conjuntos de cores padrão pré-definidos, como:

- **E** – Elementary
- **INT** – Inteiro
- **BOOL** – Booleano
- **STRING** – Caracter

Além dessas opções, o usuário pode adicionar suas próprias declarações por meio do **menu sensível ao contexto**. Para redes mais complexas, também é possível carregar declarações externas a partir de arquivos.

Quando uma rede é criada ou carregada na área de trabalho, o **CPN IDE** realiza automaticamente a verificação da sua **sintaxe**. O estágio dessa verificação pode ser identificado pela **cor da aura** dos elementos e pela **cor da linha sublinhando as declarações**. Essas cores aparecem na **área de índice**,

[4] **QoS** : conjunto de métricas que avaliam o desempenho de um sistema, incluindo latência, jitter, largura de banda e taxa de perda de pacotes.

destacando o nome da página à qual pertencem. Se a rede estiver aberta em uma das abas da área de trabalho, seu nome será exibido no topo da aba e também será sublinhado.

O processo de verificação de sintaxe segue a seguinte lógica:

- **Aura laranja**: indica que o elemento ainda não foi verificado.
- **Aura amarela**: indica que o processo de verificação está em andamento.
- **Sem aura**: a sintaxe foi verificada e está correta.
- **Aura vermelha**: há um erro de sintaxe no elemento.

O tempo de verificação pode variar, levando de alguns segundos a minutos, dependendo da complexidade da rede. Caso a **aura laranja** permaneça por um período prolongado, isso pode indicar um problema na verificação ou um erro no elemento.

As **declarações** são verificadas sequencialmente, do topo para a base. Se uma declaração depender de outra que ainda não foi definida, um erro será gerado. Sempre que uma alteração for feita em qualquer declaração, aquelas que apresentavam erro serão verificadas novamente.

Elementos com erro são sublinhados com uma **linha vermelha**, assim como o nome da rede e das páginas relacionadas. Quando um elemento exibe uma **aura vermelha**, significa que ele foi verificado, mas contém um erro. Nesse caso, o **CPN IDE** exibe uma **bolha de conversação** detalhando o problema identificado. Além disso, elementos conectados a um componente com erro não serão verificados até que o problema seja corrigido.

6.4 CONJUNTOS DE CORES

O CPN IDE, assim com o CPN Tools, fornece um conjunto de cores padrão simples e um conjunto de cores compostas.

6.4.1 CONJUNTOS DE CORES SIMPLES

O conjunto de cores padrão simples compreende as cores (conjuntos): `Unit, Boolean, Integer, String, Enumerated, Index.`

6.4.1.1 Conjunto de cores UNIT

O conjunto de cores `unit` compreende um único elemento. Esse conjuto é utilizado quando se deseja modelar um lugar que simplesmente indica presença de uma ficha (*token*). Sua declaração possui a seguinte sintaxe:

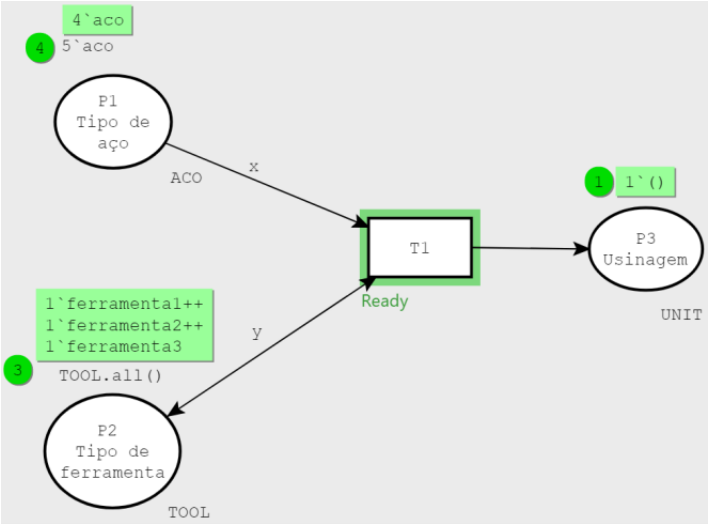
```
colset name = unit [with new unit];
```

Se a opção (*new unit*) não for utilizada, o nome da ficha coincide com o nome do conjunto de cores.

Na Figura 23 é apresentada uma rede que modela uma etapa de usinagem utilizando o conjunto de cores `unit`. O lugar p_1 representa a modelagem de material a ser usinado, o lugar p_2 representa o recurso de ferramental necessário à usinagem, e o lugar $p3$ representa a material já usinado. Perceba

que: as fichas em p_1 são do tipo `aco`, pertencentes ao `colset` ACO; as fichas em p_2 são do tipo `ferramenta1,ferramenta2,ferramenta3`, pertencentes ao `colset` TOOL; as fichas em p_3 são do tipo indefinido, porém pertencentes ao `colset` UNIT.

Figura 23 – Modelagem do conjunto de cores `unit` no CPN IDE®



Fonte: Autor, (2025)

Perceba que a marcação atual da rede corresponde a um estado aleatório de disparo da transição. As seguintes declarações foram utilizadas na rede em questão.

```
colset ACO = with 1..5;
colset TOOL = unit with ferramenta1 | ferramenta2 | ferramenta3 ;
colset UNIT = unit;
var x : ACO;
var y : TOOL;
```

A modelagem desta rede no CPN IDE® está disponível no endereço: <https://github.com/JonathaCosta-IA/MDES/tree/main/B-MDES_Nets/Nets_handbook>.

6.4.1.2 Conjunto de cores BOOLEAN

O conjunto de cores boolean compreende dois valores true e false. Sua declaração possui a seguinte sintaxe:

`colset name = bool [with new false, new true];`

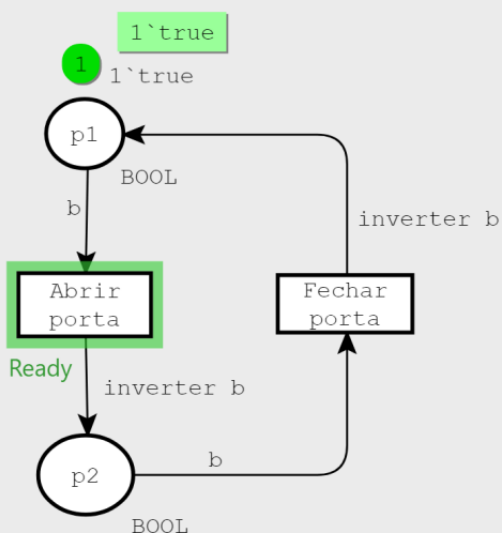
As opções (*new false, new true*) permitem novos nomes para *false* e *true*. Por exemplo, não e sim. `colset Pergunta = bool with (nao, sim);` As seguintes operações podem ser aplicadas às variáveis booleanas:

```
not b : negação do valor booleano de b;
b1 andalso b2 : conjunção booleana and;
b1 orelse b2 :  disjunção booleana or.
```

Na Figura 24, é apresentada uma rede de Petri colorida que modela o processo de abertura e fechamento automático de uma porta com base em um sensor de presença. O lugar p_1 representa o

estado inicial do sensor, indicando que há uma pessoa próxima à porta (valor booleano `true`). Já o lugar p_2 representa o estado final do sensor, indicando que a pessoa se afastou (valor `false`). A função `inverter b` é responsável por realizar a operação booleana de negação, alternando entre os estados `true` e `false`, conforme o comportamento esperado do sensor.

Figura 24 – Modelagem do conjunto de cores `bool` no CPN IDE®



Fonte: Autor, (2025)

Perceba que a marcação atual da rede corresponde a um estado aleatório de disparo da transição. As seguintes declarações foram utilizadas na rede em questão.

```
colset BOOL = bool;
var b : BOOL;
fun inverter b = not b;
```

A modelagem desta rede no CPN IDE® está disponível no endereço: <https://github.com/JonathaCosta-IA/MDES/tree/main/B-MDES_Nets/Nets_handbook>.

6.4.1.3 Conjunto de cores INTEGER

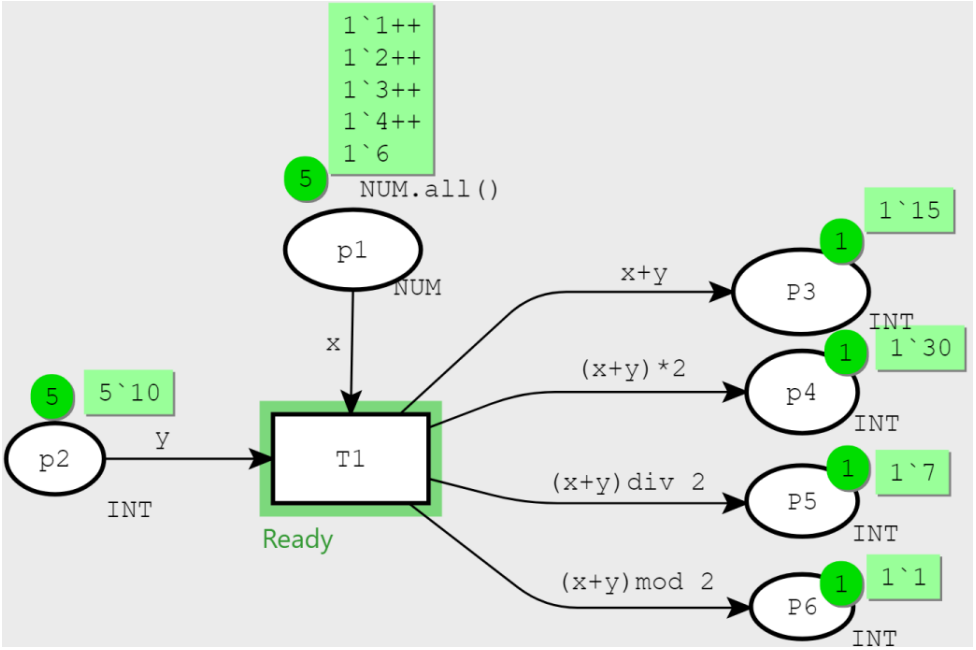
O conjunto de cores Integer compreende os valores inteiros. Sua declaração possui a seguinte sintaxe:

```
colset name = int with int-exp1 .. int-exp2;
```

Adicionalmente, a opção **with** permite restringir o conjunto de cores inteiro pelo *intervalo* determinado pelas duas expressões `int-exp1` e `int-exp2`.

Na Figura 25, é apresentada uma rede de Petri colorida que modela uma etapa de contagem de produtos, utilizando o conjunto de cores `int`. O lugar p_1 armazena os valores individuais dos produtos, enquanto o lugar p_2 representa um peso fixo de 10 unidades associado a cada operação de contagem.

Figura 25 – Modelagem do conjunto de cores `int` no CPN IDE®



Fonte: Autor, (2025)

Perceba que a marcação atual da rede corresponde a um estado aleatório de disparo da transição. As seguintes declarações foram utilizadas na rede em questão.

```
colset INT = int;
colset NUM = int with 1..6;
var x : NUM;
var y : INT;
```

Na marcação atual da rede, observam-se cinco fichas em p_1, com os valores $[1,2,3,4,5]$; cinco fichas de valor 10 em p_2; e uma ficha no lugar p_3, com valor 15. Essa ficha em p_3 resulta da soma valor 5, anteriormente presente em uma ficha de p_1, com o valor 10, proveniente de uma ficha de p_2. Essa operação é descrita no arco que liga T_1 a p_3 pela expressão $x+y$. O modo processo aplica-se às expressões dos demais arcos de modo que: p_4 recebe o produto de $(x+y)\times 2$, p_5 recebe o valor da divisão de $\frac{(x+y)}{2}$, e p_5 recebe o valor do resto de divisão $(x+y)\times 2$. Assim, a rede executa uma operação de adição entre os valores de entrada, simulando o processo de contagem ponderada de produtos.

As seguintes operações podem ser aplicadas às variáveis inteiras: `+, -, div, mod, abs, Int.min, Int.max`. A modelagem desta rede no CPN IDE® está disponível no endereço: <https://github.com/JonathaCosta-IA/MDES/tree/main/B-MDES_Nets/Nets_handbook>.

6.4.1.4 Conjunto de cores STRING

O conjunto de cores **String** é especificado por uma sequência de caracteres ASCII entre aspas. Sua declaração possui a seguinte sintaxe:

```
colset name = string [with string-exp1 ..string-exp2
[and int-exp1..int-exp2]];
```

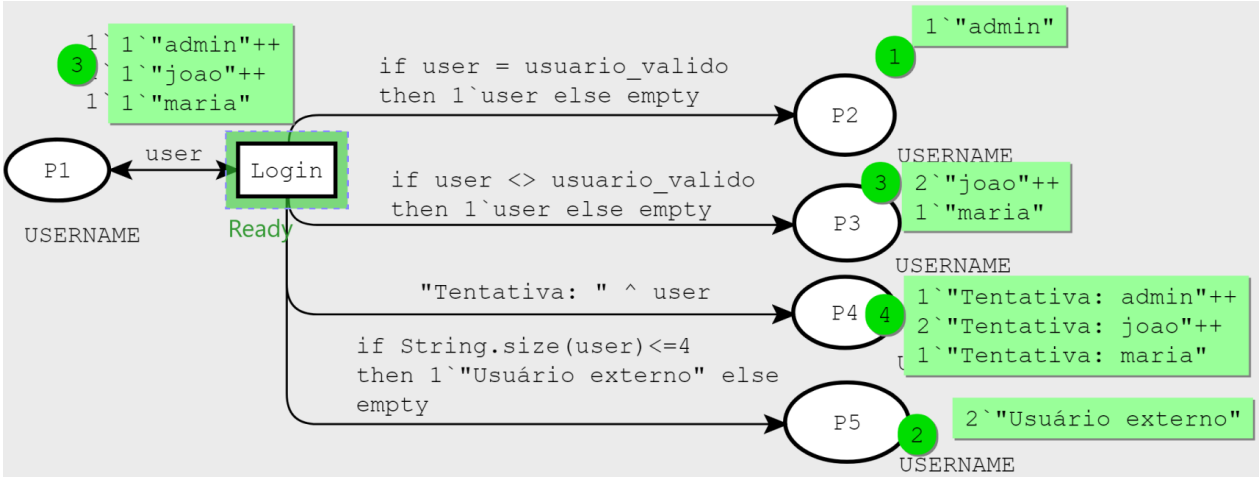
A opção `with` especifica o intervalo de caracteres válidos, por exemplo:

```
colset minusculas = string with "a".."z";
```

Na Figura 26, é apresentada uma rede de Petri colorida que modela uma etapa de acesso (*login*) de um sistema digital, utilizando o conjunto de cores `string`. O lugar p_1 armazena os valores individuais dos usuários que tentam acessar o sistema, o lugar p_2 representa os acessos de usuários autorizados *admin*, o lugar p_3 armazena o usuários não autorizados, o lugar p_4 armazena todas as tentativas de acesso, e o lugar p_5 armazena as fichas cujo valor tamanho da `string` seja menor ou igual a 4, identificando como `usuário externo`.

A distinção entre os usuários é realizada por uma função *if* sobre a variável *user*, tendo *admin* como *usuário válido*.

Figura 26 – Modelagem do conjunto de cores `string` no CPN IDE®



Fonte: Autor, (2025)

Perceba que a marcação atual da rede corresponde a um estado aleatório de disparo da transição.

As seguintes declarações foram utilizadas na rede em questão.

```
colset USERNAME = string;
var user : USERNAME;
val usuario_valido = "admin";
```

As seguintes operações podem ser aplicadas às variáveis string: `^(concatenação), String.size, substring`.

A modelagem desta rede no CPN IDE® está disponível no endereço: <https://github.com/JonathaCosta-IA/MDES/tree/main/B-MDES_Nets/Nets_handbook>.

6.4.1.5 Conjunto de cores ENUMERATED

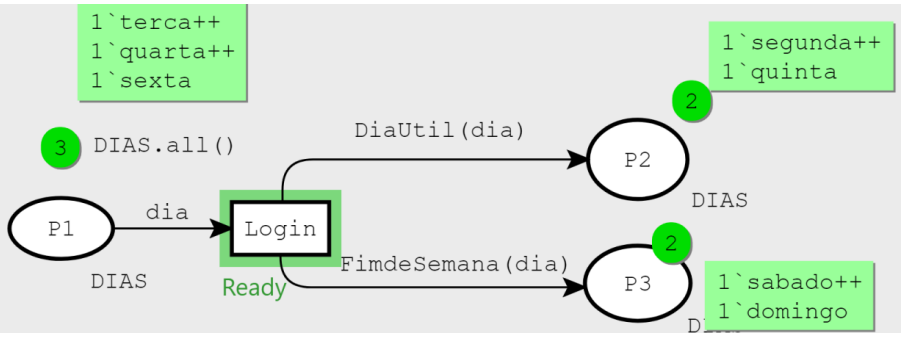
O conjunto de cores **Enumerated** explicita todos os identificadores na sua declaração. Sua sintaxe é assim definida:

```
colset name = with id0 | id1 | ...| idn;
```

Na Figura 27, é apresentada uma Rede de Petri Colorida que modela uma etapa relacionada aos dias da semana, utilizando um conjunto de cores do tipo `enumerated`. O lugar p_1 armazena fichas re-

presentando os dias da semana; o lugar p_2 armazena os dias úteis, filtrados pela função `DiaUtil(dia)`; e o lugar p_3 contém os dias de sábado e domingo, identificados pela função `FimdeSemana(dia)`.

Figura 27 – Modelagem do conjunto de cores `enumerated` no CPN IDE®



Fonte: Autor, (2025)

Perceba que a marcação atual da rede corresponde a um estado aleatório de disparo da transição. As seguintes declarações foram utilizadas na rede em questão.

```
colset DIAS = with segunda | terca | quarta
| quinta | sexta | sabado | domingo;
var dia : DIAS;
fun DiaUtil dia =  if dia <> sabado andalso dia <> domingo
then 1`dia else empty;
fun FimdeSemana dia =  if dia = sabado orelse dia = domingo
then 1`dia else empty;
```

A modelagem desta rede no CPN IDE® está disponível no endereço: <https://github.com/JonathaCosta-IA/MDES/tree/main/B-MDES_Nets/Nets_handbook>.

6.4.1.6 Conjunto de cores INDEXED

Os conjuntos de cores **Indexed** são sequências de valores que contêm um inteiro e um índice especificador. Sua declaração possui a seguinte sintaxe:

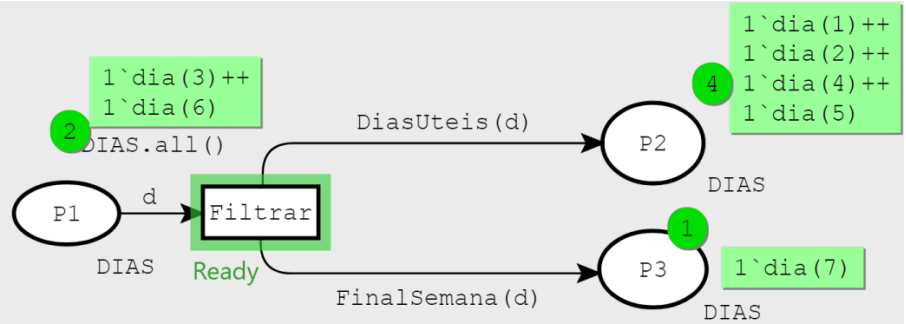
`colset name = index id with int-exp1 ..int-exp2;`

Valores indexados possuem o seguinte formato: `id i` ou `id(i)`, em que

`i` é um inteiro e `int-exp1` $\leq$ `i` $\leq$ `int-exp2`.

Na Figura 28, é apresentada uma Rede de Petri Colorida que modela uma etapa relacionada aos dias da semana, utilizando um conjunto de cores do tipo `indexed`. O lugar p_1 armazena fichas representando os dias da semana; o lugar p_2 armazena os dias úteis, filtrados pela função `DiasUteis(d)`; e o lugar p_3 contém os dias de sábado e domingo, identificados pela função `FinalSemana(d)`.

Figura 28 – Modelagem do conjunto de cores `indexed` no CPN IDE®



Fonte: Autor, (2025)

Perceba que a marcação atual da rede corresponde a um estado aleatório de disparo da transição. As seguintes declarações foram utilizadas na rede em questão.

```
colset DIAS = index dia with 1..7;
var d: DIAS;
fun DiasUties (dia(i)) =  if i <=5 then 1`dia(i) else empty;
fun FinalSemana (dia(i)) =  if i >5 then 1`dia(i) else empty;
```

A modelagem desta rede no CPN IDE® está disponível no endereço: <https://github.com/JonathaCosta-IA/MDES/tree/main/B-MDES_Nets/Nets_handbook>.

6.4.2 CONJUNTOS DE CORES COMPOSTOS

Os *conjuntos de cores compostos* são construídos a partir da combinação de conjuntos de cores simples. A linguagem CPN ML oferece suporte aos seguintes tipos de conjuntos compostos: `product`, `record`, `union`, `list`, `subset` e `alias`.

Dentre esses, os conjuntos `product` e `record` representam dados estruturados formados pelo produto cartesiano entre os elementos de outros conjuntos. A principal diferença entre eles está na nomeação dos componentes: enquanto os elementos do conjunto `product` são referenciados apenas por sua posição (sem nomes explícitos), os componentes do conjunto `record` são identificados por nomes, o que favorece a legibilidade e organização dos dados.

Essa distinção é análoga à encontrada entre tipos de dados em linguagens de programação tradicionais, como o tipo `record` na linguagem Pascal e a estrutura `struct` na linguagem C.

6.4.2.1 Conjunto de cores PRODUCT

O conjunto de cores do tipo `product` segue a seguinte sintaxe:

```
colset <nome> = product <nome_1> * <nome_2> * ...* <nome_n>;
```

Os valores pertencentes a esse conjunto possuem a forma:

$$(v_1, v_2, \ldots, v_n),$$

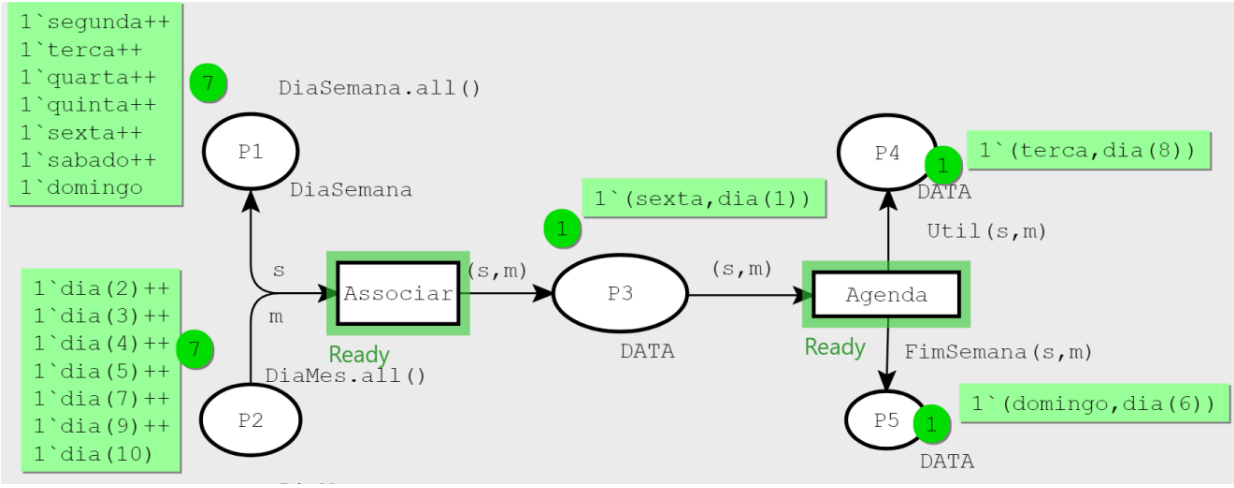
em que cada v_i é do tipo correspondente $<nome_i>$, para $1 \leq i \leq n$.

Para acessar o i-ésimo elemento de uma variável do tipo `product`, utiliza-se a seguinte notação:

```
#<i> <nome>;
```

Na Figura 29, apresenta-se uma Rede de Petri Colorida que modela uma etapa relacionada aos dias da semana. O lugar p_1 armazena fichas representando os dias da semana, enquanto o lugar p_2 contém fichas correspondentes a 10 dias úteis. O lugar p_3 representa a associação entre as fichas provenientes de p_1 e p_2, utilizando um conjunto de cores do tipo `product`. Por fim, os lugares p_4 e p_5 armazenam, respectivamente, as fichas classificadas como dias úteis e fins de semana, com base nas funções `Util` e `FimSemana` definidas nos arcos correspondentes.

Figura 29 – Modelagem do conjunto de cores `product` no CPN IDE®



Fonte: Autor, (2025)

Perceba que a marcação atual da rede corresponde a um estado aleatório de disparo da transição. As seguintes declarações foram utilizadas na rede em questão.

```
colset DiaSemana = with segunda | terca | quarta | quinta | sexta | sabado |
    domingo;
colset DiaMes = index dia int 1 .. 10;
colset DATA = product DiaSemana * DiaMes;
var s : DiaSemana;
var m :DiaMes;
fun Util(s,m) = if s<>sabado andalso s<>domingo then 1`(s,m) else empty;
fun FimSemana(s,m) = if s=sabado orelse s=domingo then 1`(s,m) else empty;
```

A modelagem desta rede no CPN IDE® está disponível no endereço: <https://github.com/JonathaCosta-IA/MDES/tree/main/B-MDES_Nets/Nets_handbook>.

6.4.3 CONJUNTO DE CORES RECORD

O conjunto de cores do tipo **record** possui a seguinte sintaxe:

```
colset nome = record id1:name1 * id2:name2 * ... * idn:namen;
```

Os valores pertencentes a esse conjunto de cores têm a forma:

$$(id1 = v1, id2 = v2, \ldots, idn = vn)$$

em que cada v_i é um valor do tipo $name_i$, para $1 \leq i \leq n$.

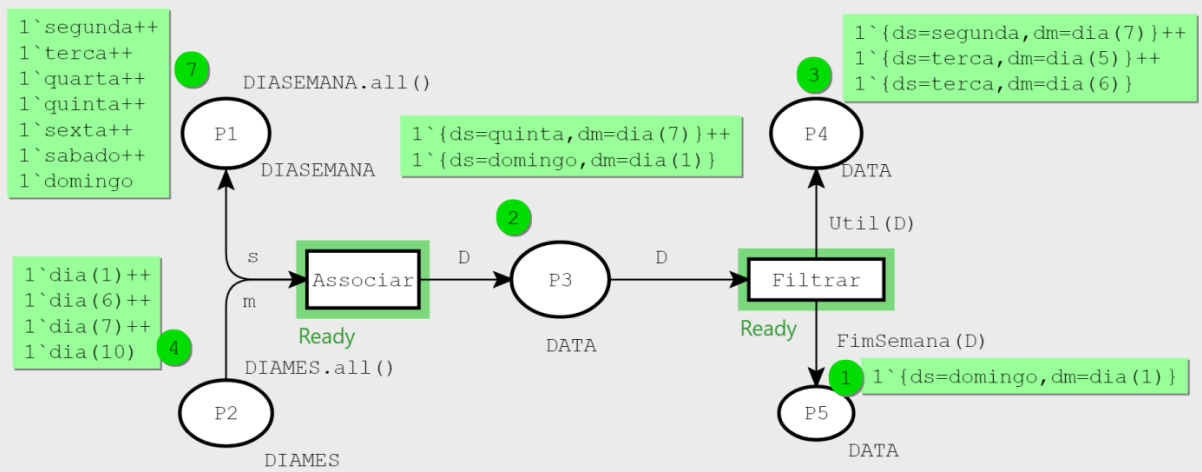
Para acessar o i-ésimo campo de um **record**, utiliza-se a seguinte operação:

```
#idi nome;
```

Considere o mesmo exemplo apresentado para o conjunto de cores `product` representado na Figura 29. Perceba a mesma rede agora modificada para o tipo `record` e representado na Figura 30.

Percebe que a Rede de Petri Colorida que modela, **de modo símile**, uma etapa relacionada aos dias da semana. O lugar p_1 armazena fichas representando os dias da semana, enquanto o lugar p_2 contém fichas correspondentes a 10 dias úteis. O lugar p_3 representa a associação entre as fichas provenientes de p_1 e p_2, utilizando um conjunto de cores do tipo `record`. Por fim, os lugares p_4 e p_5 armazenam, respectivamente, as fichas classificadas como dias úteis e fins de semana, com base nas funções `Util` e `FimSemana` definidas nos arcos correspondentes.

Figura 30 – Modelagem do conjunto de cores `record` no CPN IDE®



Fonte: Autor, (2025)

Perceba que a marcação atual da rede corresponde a um estado aleatório de disparo da transição.

As seguintes declarações foram utilizadas na rede em questão:

```
colset DIASEMANA = with segunda | terca | quarta | quinta | sexta | sabado |
    domingo;
colset DIAMES = index dia int 1 .. 10;
colset DATA= record ds:DIASEMANA * ds:DIAMES;
var s : DIASEMANA;
var m : DIAMES;
var D : DATA;
fun Util(D: DATA) = if #ds D <>sabado andalso #ds D<>domingo
then 1`D else empty;
fun FimSemana(D: DATA) = if #ds D = sabado orelse #ds D = domingo
then 1`D else empty;
```

A modelagem desta rede no `CPN Tools`® está disponível no repositório: <https://github.com/JonathaCosta-IA/MDES/tree/main/B-MDES_Nets/Nets_handbook>.

Observe que o `colset record` permite a declaração de identificadores nomeados diretamente associados aos tipos de dados do conjunto. Isso pode ser verificado no trecho destacado abaixo, extraído

das declarações utilizadas na modelagem da rede apresentada anteriormente, em que os identificadores `ds` e `dm` são definidos como parte integrante da estrutura do conjunto de cores:

```
colset DATA = record ds:DIASEMANA * dm:DIAMES;
```

A principal vantagem dessa abordagem é a melhoria na legibilidade e na clareza semântica do modelo. Ao utilizar identificadores nomeados, como `ds` e `dm`, torna-se mais intuitivo compreender o significado de cada componente do *token*, reduzindo a ambiguidade e a necessidade de memorizar posições específicas, como ocorre nos conjuntos de cores do tipo `product`. Essa estrutura favorece a manutenção e a escalabilidade do modelo, especialmente em redes complexas, em que o número de campos e a reutilização de estruturas são frequentes.

Além disso, embora os conjuntos de cores `product` e `record` apresentem diversas semelhanças estruturais, a forma de acesso aos seus elementos é distinta. No caso de `record`, o acesso é mais intuitivo e se assemelha ao uso de `structs` na linguagem de programação C, onde os campos são acessados por identificadores nomeados, utilizando-se a notação `#`. Em contraste, no tipo `product`, o acesso é posicional, geralmente realizado por meio de funções como `proj_i`.

6.4.3.1 Conjunto de Cores `union`

O conjunto de cores `union` permite combinar dois ou mais conjuntos de cores distintos em um único tipo. Em condições normais, cada lugar da rede aceita fichas de apenas um conjunto de cores. No entanto, o uso de `union` supera essa limitação ao possibilitar que diferentes tipos de dados sejam associados a um mesmo lugar, proporcionando maior flexibilidade na modelagem de comportamentos heterogêneos.

A sintaxe geral para a declaração de um conjunto `union` é a seguinte:

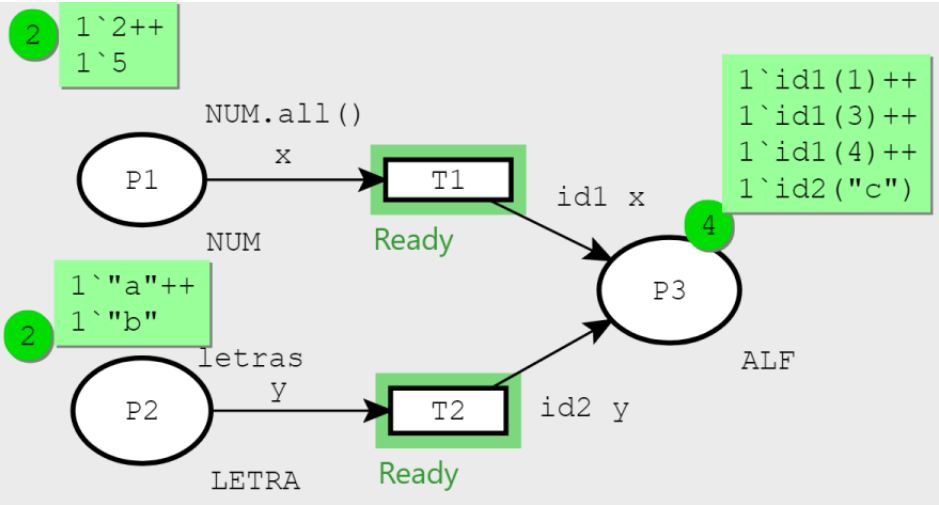
```
colset Nome = union id1[:Tipo1] + id2[:Tipo2] + ... + idn[:Tipon];
```

Cada `id` atua como um construtor que identifica qual tipo está sendo utilizado em um determinado token. Caso o tipo (`Tipo`) seja omitido, o identificador correspondente é tratado como um valor simbólico (constante), podendo ser referenciado diretamente por seu nome.

Operações de correspondência de padrões (*pattern matching*) podem ser utilizadas para identificar e extrair valores das fichas conforme o construtor utilizado, facilitando o tratamento seletivo de informações dentro das transições.

Na Figura 31 é apresentada uma rede basilar com o evento de união de dois conjuntos de fichas distintas.

Figura 31 – Modelagem do conjunto de cores `union` no CPN IDE®



Fonte: Autor, (2025)

As seguintes declarações foram utilizadas na rede em questão.

```
colset NUM = int with 1..5;
colset LETRA = string;
colset ALF = union id1 : NUM + id2: LETRA;
var p : ALF;
var x : NUM;
var y : LETRA;
val letras = 1`"a"++ 1`"b"++1`"c";
```

A modelagem desta rede no CPN IDE® está disponível no endereço: <https://github.com/JonathaCosta-IA/MDES/tree/main/B-MDES_Nets/Nets_handbook>.

6.4.3.2 Conjunto de cores LIST

O conjunto de cores `list` possui tamanho variável e se constitui de uma sequência de elementos de um mesmo conjunto de cores previamente definido. Funções padrão permitem acessar o primeiro e o último elemento de uma lista. Para acessar elementos do interior da lista, funções recursivas têm que ser usadas.

O conjunto de cores list possui a seguinte sintaxe:

`colset name = list name0 [with int-exp1 .. int-exp2];`

A cláusula `with` especifica o menor e o maior tamanho da lista. Os elementos de uma lista possuem a seguinte forma:

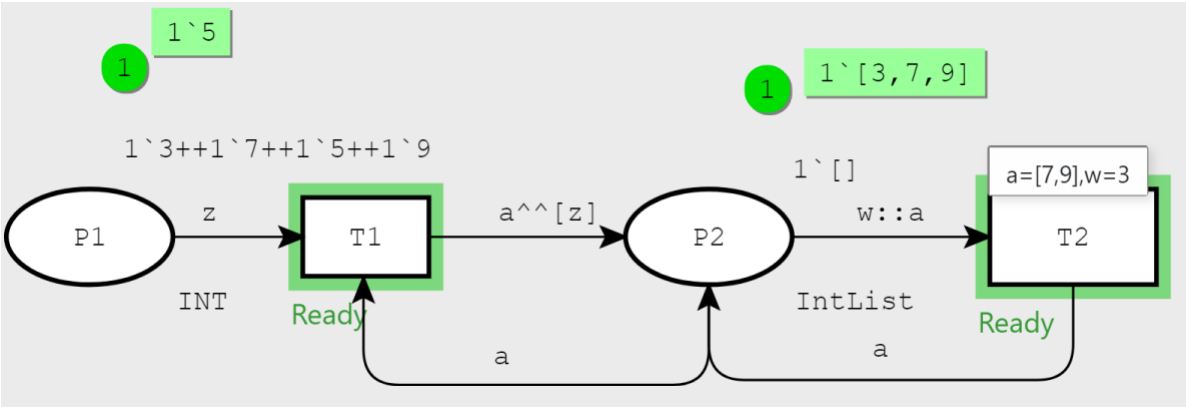
- `nil` - lista vazia (o mesmo que `[]`)
- `e::l` - coloca o elemento `e` como cabeça da lista `l`
- `l1^^l2` - concatena as duas listas `l1` e `l2`
- `hd l` - primeiro elemento da lista `l`
- `tl l` - toda a lista `l`, exceto o primeiro elemento

- `length l` - retorna o tamanho da lista `l`
- `rev l` - retorna uma lista inversa à lista `l`
- `map f l` - aplica a função `f` em todos os elementos da lista `l`
- `mem l x` - retorna verdadeiro se `x` pertence à lista `l`
- `List.nth(l,n)` - retorna o n-ésimo elemento da lista `l`, onde $0 \leq n <$ `length l`
- `List.take(l,n)` - retorna os primeiros `n` elementos da lista `l`
- `List.drop(l,n)` - retorna o que resta na lista após remover os primeiros `n` elementos
- `List.exists p l` - retorna verdadeiro se `p` for verdadeiro para algum elemento da lista `l`
- `List.null l` - retorna verdadeiro se a lista `l` é vazia

Normalmente, no disparo de uma transição em uma RPC qualquer, retiram-se fichas dos lugares de entrada da transição de forma aleatória, ou seja, a ligação entre as variáveis dos arcos e as fichas é feita de forma aleatória, desde que essa ligação habilite a transição. Entretanto, em alguns setores do conhecimento, tais como telecomunicações, linha de montagem, etc., existe um ordenamento de chegada e saída de informação ou produtos. Existe, então, a necessidade de políticas de prioridade. Algumas destas políticas são, por exemplo, a FIFO (*First In First Out*), ou primeira que chega é a primeira que sai; e a LIFO (*Last In First Out*), ou última que chega é a primeira que sai.

Na Figura 32 é apresentado um exemplo de FIFO. Primeiro, valores inteiros são retirados do lugar `p1` de forma aleatória e armazenados por ordem de chegada em uma lista no lugar `p2 (1`[ ])`, definido como uma lista vazia. Quando `t2` estiver habilitada, o disparo da mesma vai retirar sempre a ficha que chegou primeiro ao lugar `p2`. Na figura, após `t2` disparar duas vezes, as fichas `9` e `7` são retiradas consecutivamente de `p2`.

Figura 32 – Modelagem do conjunto de cores `LIST` no CPN IDE®



Fonte: Autor, (2025) adaptado de (BARROSO, 2006)

As seguintes declarações foram utilizadas na rede em questão.

```
colset INT = int ;
colset IntList = list int with 1`[];
var a,z : INT;
var w : IntList;
```

Perceba, ainda na Figura 32, a existência uma moldura na transição t_2 contendo a seguinte informação `a=[7,9],w=3`. Significa que ao disparar t_2 a ficha de valor 3 será removida para p_2, depois a ficha de valor 7 e assim sucessivamente, exatamente mantendo a ordem de chegada. A modelagem desta rede no CPN IDE® está disponível no endereço: <https://github.com/JonathaCosta-IA/MDES/tree/main/B-MDES_Nets/Nets_handbook>.

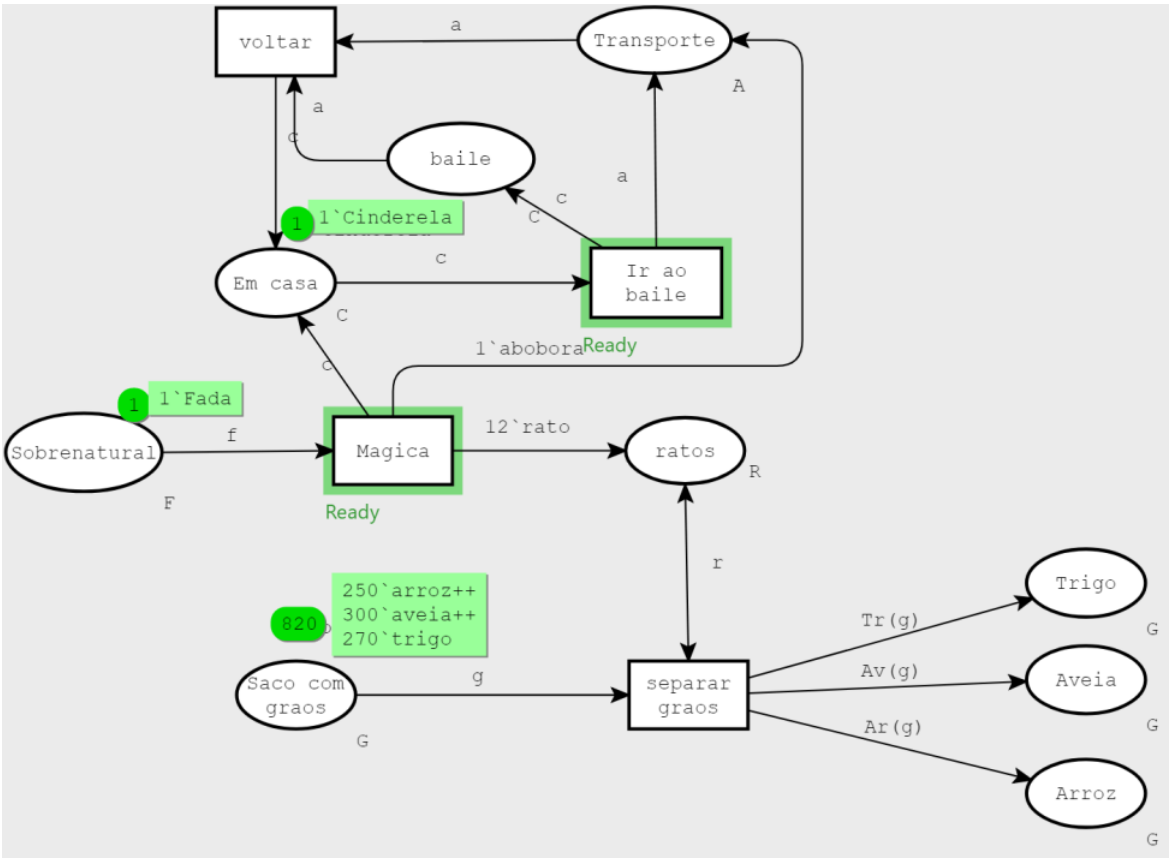
6.5 MODELO DE REDE: CONTO DA CINDERELA

O exemplo a seguir apresenta a modelagem de uma situação clássica da literatura infantil em RP, adaptada de Zaitsev (2006, *apud* (BARROSO, 2006)).

O CASO CINDERELA

A madrasta de Cinderela mandou que ela separasse grãos de diferentes tipos, mas no exemplo, camundongos amigos de Cinderela separam os grãos enquanto ela vai ao baile. Este exemplo está ilustrada no Figura 33.

Figura 33 – Modelagem de caso Cinderela no CPN IDE®



Fonte: Autor, (2025) adaptado de (BARROSO, 2006)

Neste modelo de RPC, as seguintes declarações de conjuntos de cores e variáveis foram utilizadas:

```
colset A = unit with abobora;
colset C = unit with Cinderela;
colset G = with arroz | aveia | trigo;
colset R = unit with rato;
colset F = unit with Fada;
var a : A;
var c : C;
var g : G;
var r : R;
var f : F;
fun Tr(g) = if (g=trigo) then 1`trigo else empty;
fun Av(g) = if (g=aveia) then 1`aveia else empty;
fun Ar(g) = if (g=arroz) then 1`arroz else empty;
val graos = 250`arroz++300`aveia++270`trigo;
```

Nesta modelagem, foram definidos cinco conjuntos de cores: **A**, contendo a ficha *abóbora*; **C**, com a ficha *Cinderela*; **G**, composto por três fichas (*arroz*, *aveia* e *trigo*); **R**, com a ficha *rato*; e **F**, com a ficha *Fada*. Note que inicialmente, apenas a transição *Mágica* está habilitada, destacada por uma moldura na cor verde. Ao conversar com *Cinderela*, a fada cria doze ratos, uma abóbora e desaparece, acionando a transição *Mágica*. Note também que a viagem de *Cinderela* até o baile e a separação dos grãos ocorrem de forma concorrente, podendo acontecer simultaneamente e em qualquer ordem. A abóbora funciona como um recurso essencial para habilitar e disparar as transições *Ir ao baile* e *Voltar*, que levam *Cinderela* ao baile e a trazem de volta. Para isso, utiliza-se um auto-laço, representado por um arco bidirecional conectando o lugar *Transporte* à transição *Ir ao baile* (Figura 33). Os ratos, por sua vez, servem como recursos para habilitar e disparar a transição *Separar* os tipos de grãos.

Considerando o sentido dos arcos e suas respectivas inscrições, perceba que o disparo da transição *Mágica* não altera a marcação do lugar *Em casa*, que contém uma ficha de cor *Cinderela*. Da mesma forma, o disparo da transição *Ir ao baile* não modifica a marcação do lugar *Transporte* (ficha de cor *Abóbora*), pois esse mesmo recurso é utilizado para trazer *Cinderela* de volta para casa ao acionar a transição *Voltar*.

Note que os demais arcos são unidirecionais. Desse modo, um arco que conecta um lugar a uma transição indica que, no momento do disparo, uma ou mais fichas serão removidas do lugar, conforme definido pela inscrição do arco de entrada da transição. No exemplo, todas as inscrições são variáveis pertencentes aos respectivos conjuntos de cores.

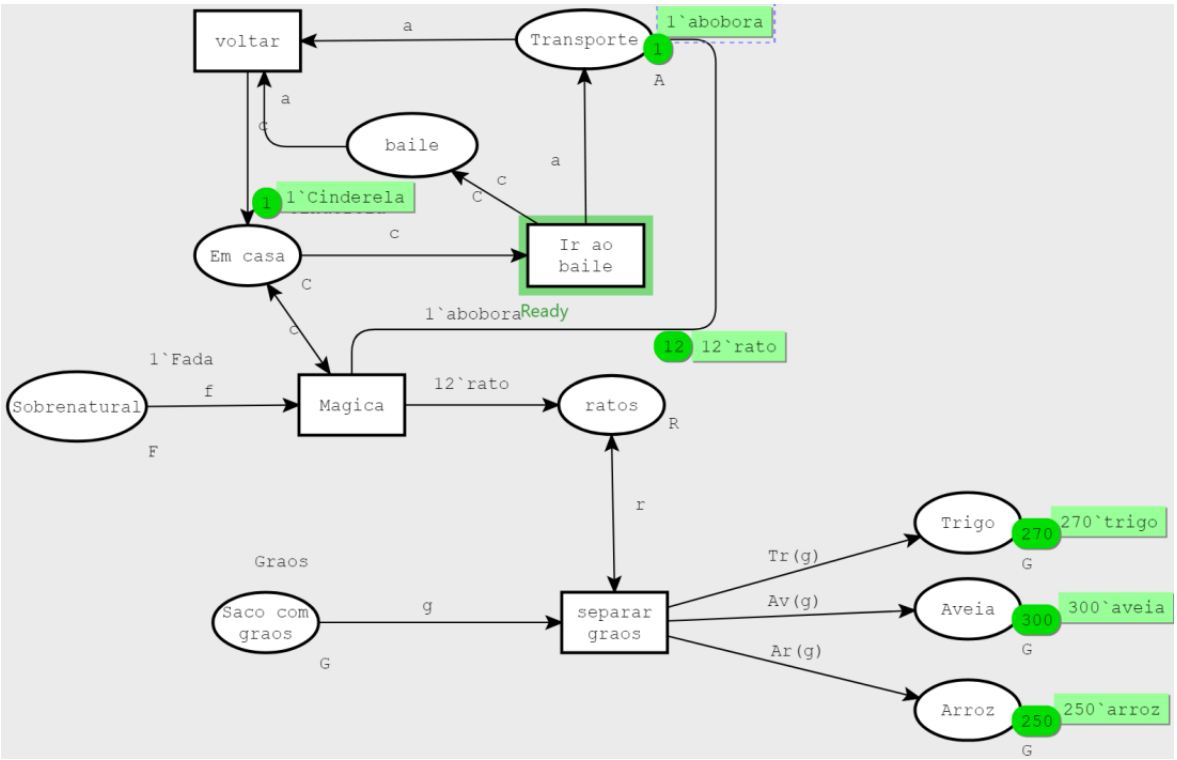
Como ilustração disso, perceba que o arco de entrada da transição *Separar grão* possui a inscrição *g*, que representa uma variável do conjunto de cores **G**. Assim, quando a transição *Separar* é disparada, um grão é retirado aleatoriamente do lugar *Saco com grãos*. Inscrições de arco mais complexas serão exploradas nas próximas seções. Os arcos de saída das transições, ao serem disparadas, geram novas fichas, que podem ou não coincidir com as retiradas dos lugares de entrada. Isso significa que novas fichas podem ser criadas no sistema. Outro modo, no disparo da transição *Ir ao baile*, a ficha *Cinderela* é

removida do lugar *Em casa* por meio da variável *c*, associada ao arco que conecta esse lugar à transição. Em seguida, uma ficha de mesmo valor (*Cinderela*) é inserida no lugar *Baile*, conforme a inscrição *c* no arco que liga a transição *Ir ao baile* a esse lugar.

Finalmente, a transição *Mágica*, por sua vez, possui uma lógica mais complexa. Ao ser disparada, a ficha *Fada* é removida do lugar *Sobrenatural* pela variável *f* e desaparece, pois essa variável não está associada a nenhum arco de saída da transição. Entretanto, a marcação do lugar *Em casa* permanece inalterada devido ao auto-laço (arco bidirecional) associado à variável *c*.

Como resultado do disparo dessa transição, são criadas doze fichas *Rato* e uma ficha *Abóbora*, conforme as inscrições especificadas nos arcos de saída. Veja na Figura 34 a marcação final do modelo após 823 passos de simulação.

Figura 34 – Modelagem de caso Cinderela - Resultado da simulação no CPN IDE®



Fonte: Autor, (2025) adaptado de (BARROSO, 2006)

Nesta marcação, Cinderela já retornou do baile e está em casa. A fada desapareceu, e seus amigos ratos finalizaram a separação dos grãos, distribuindo 270 grãos de trigo, 300 de arroz e 250 de aveia em seus respectivos lugares. Note que o disparo **separar grão** retira aleatoriamente uma ficha do saco de grãos por meio da variável *g*. No entanto, esse grão só pode ser colocado em um único destino entre os lugares de saída da transição (arroz, aveia ou trigo). Os arcos de saída da transição possuem funções associadas que garantem a seleção apenas do grão correspondente ou de uma ficha especial chamada *empty*. Quando uma ficha *empty* é escolhida, isso significa que nenhum grão será colocado no lugar de saída da transição.

O modelo RPC estudado não contempla diversas características do conto de fadas. Por exemplo, o tempo não é considerado, o que significa que o aviso da Fada para Cinderela sobre a meia-noite não está representado no modelo. Por isso, recomenda-se uma análise detalhada para identificar outras

características relevantes do conto. Após um estudo mais aprofundado do CPN IDE®, pode-se construir um modelo que represente de forma mais fiel a história original. A modelagem desta rede no CPN IDE® está disponível no endereço: <https://github.com/JonathaCosta-IA/MDES/tree/main/B-MDES_Nets/Nets_handbook>.

REFERÊNCIAS BIBLIOGRÁFICAS

ANDERSON, D. **Produção em Massa e a Evolução da Manufatura**. [S.l.]: Editora XY, 2015.

BARROSO, J. M. S. G. C. **Introdução às Redes de Petri Coloridas usando a ferramenta CPN Tools**. 2006. Acesso em: 11 mar. 2025. Disponível em: <https://www.scribd.com/document/496849470/redespetritutorialcpntools-160625125824>.

BASKIN, D. **Sistemas Flexíveis de Manufatura**. [S.l.]: Editora Flex, 2017.

BRUNDTLAND, G. H. **Nosso Futuro Comum**. [S.l.]: Editora Mundial, 1987.

CASSANDRAS, C. **Introduction to Discrete Event Systems**. [S.l.]: Springer, 2008.

CHASE, R. B.; AQUILANO, N. J. **Gestão de Operações: Produção e Cadeia de Suprimentos**. [S.l.]: Editora McGraw-Hill, 2006.

HARRIS, F. **Gestão da Produção e Operações**. [S.l.]: Editora Pearson, 2002.

IDE, C. **From CPN Tools via Access/CPN to CPN IDE**. 2025. Acesso em: 9 abr. 2025. Disponível em: <https://cpnide.org/>.

JONES, M. Indústria 4.0: O futuro da manufatura. **Revista de Tecnologia**, v. 15, n. 2, p. 123–135, 2018.

MELLOR, P. **Indústria 4.0: A Revolução Digital**. [S.l.]: Editora Tech, 2014.

MURATA, T. Petri nets: Properties, analysis, and applications. **Proceedings of the IEEE**, v. 77, n. 4, p. 541–580, 1989.

SAP. **What is Industry 4.0?** 2025. Acesso em: 11 mar. 2025. Disponível em: <https://www.sap.com/products/scm/industry-4-0/what-is-industry-4-0.html>.

SMITH, J. **História da Manufatura**. [S.l.]: Editora ABC, 2010.

THOMPSON, D. **Inovação e Competitividade na Manufatura Global**. [S.l.]: Springer, 2019.

WILSON, J. M. **Planejamento e Controle da Produção**. [S.l.]: Editora Cengage Learning, 2016.