Apostila
Sistemas a Eventos Discretos

Eduard Montgomery Meira Costa, DSc
UFBA

Introdução

Nos últimos anos houve um desenvolvimento significativo do estudo dos sistemas dinâmicos feitos pelo homem, denominados de sistemas dinâmicos a eventos discretos, ou simplesmente, sistemas a eventos discretos. Estes tipos de sistemas são encontrados em muitas aplicações do cotidiano, como as redes de computadores, os sistemas de manufatura, os sistemas de comunicação os sistemas de tráfego aéreo e ferroviário e os sistemas operacionais.

Com o atual desenvolvimento tecnológico, houve um aumento considerável no processo produtivo. Tal crescimento de produtividade está associada aos estudos desenvolvidos sobre esses sistemas. Especialmente na indústria, onde a automação é determinante para a ampliação da produtividade, esses sistemas encontram sua aplicação.

A automação determina não só aumento da produtividade, como também segurança. Isto é devido a possibilidade de automação em processos onde haja perigo de vida ao ser humano. Assim, essa modernização necessita da ampliação do nível da automação industrial, o que gera em conseqüência, uma proliferação destes sistemas. Neste contexto, a investigação de ferramentas matemáticas adequadas para análise e projeto desses sistemas adquire um elevado grau de importância. Inclusive pelo fato de que as ferramentas ditas convencionais de estudo, como os modelos formulados por equações diferenciais, não são suficientes para viabilizar o estudo destes sistemas.

Os sistemas de automação construídos pelo homem são denominados de Sistemas Dinâmicos a Eventos Discretos, ou Sistemas a Eventos Discretos (SEDs) [1]. Este tipo de sistema surge, geralmente, quando se realiza a modernização de um processo industrial substituindo, onde possível, a tecnologia analógica pela tecnologia digital. Estes sistemas modernizados não são representados através de equações diferenciais, as quais são normalmente utilizadas para descrever os sistemas dinâmicos de variáveis contínuas (SDVCs). De fato, os SEDs exibem características específicas em sua evolução dinâmica que exigem outros paradigmas de representação. Assim, para estudar os SEDs, é necessário numa primeira etapa localizar este tipo de sistema numa classificação genérica de sistemas com o intuito de destacar suas características peculiares.

Sistemas são definidos como uma parte limitada do universo que pode ser caracterizada através de um conjunto finito de variáveis que podem ser associadas às grandezas físicas que a identifica univocamente. Num contexto mais amplo, sistemas são conjun-

tos de elementos, materiais ou imateriais, entre os quais se pode definir uma relação e que operam com uma estrutura organizada.

Os sistemas físicos apresentam em sua estrutura, componentes que interagem entre si. Cada componente também pode ser considerado um sistema com características próprias, ou seja, um subsistema. O comportamento de cada subsistema pode ser descrito independentemente dos demais componentes do sistema, exceto pelas suas interações. Cada componente tem seu próprio conjunto de estados, ou configuração interna. Este conjunto de estados é uma informação necessária para descrever o comportamento futuro do sistema. Freqüentemente, o estado de um componente depende dos seus estados passados. Logo, os estados de um componente definem uma trajetória no tempo e proporcionam uma informação a respeito de sua descrição.

Os componentes de um sistema, em geral, são concorrentes. A concorrência é definida pelas atividades paralelas que podem ser realizadas pelos vários componentes do sistema. Dessa forma, num mesmo instante de tempo podem ocorrer mudanças de estado em vários subsistemas.

A temporização das atividades realizadas nos diversos componentes pode ser muito complexa e, conseqüentemente, a descrição de suas interações pode se tornar difícil. Dessa forma, o estudo dos sistemas, requer o desenvolvimento de modelos que permitam analisar o comportamento das variáveis e definir suas características.

Sistemas a Eventos Discretos

Um SED é definido como um sistema cuja evolução dinâmica depende da ocorrência de eventos. Um evento pode ser identificado com uma ação proposital (ligar um interruptor), uma ocorrência espontânea (perda de conexão com o provedor de internet) ou o resultado da verificação de uma condição (a temperatura do reator excedeu o limite de segurança). Os eventos produzem mudanças de estado e, de modo geral, ocorrem em instantes de tempo irregulares.

Na Figura 0.1, é apresentado um diagrama que representa uma evolução dinâmica de um SED. Nessa figura os eventos são etiquetados com os símbolos α, β, γ, δ e ε, e os estados do sistema são representados por q_0, q_1, q_2, q_3 e q_4. O estado inicial é q_0 e, os símbolos α, β, γ, δ e ε, representam os eventos que mudam o estado do sistema para q_1, q_2, q_3 e q_4, respectivamente. Quando não há a ocorrência de nenhum evento,

o sistema permanece no mesmo estado.

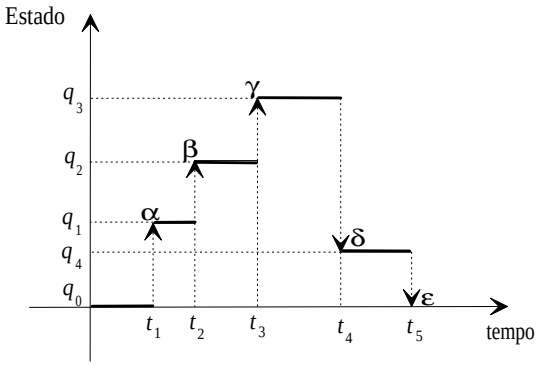

Figura 0.1: Evolução dinâmica de um sistema a eventos discretos: ocorrências de eventos assíncronos e mudanças instantâneas de estado.

Como mencionado anteriormente, um sistema é definido como uma parte limitada do universo que interage com um ambiente através de suas fronteiras. Uma diferença básica entre um SED e um SDVC é a forma pela qual interage com o ambiente. No caso de um SDVC, as mudanças de estado decorrem da troca de energia com o ambiente. No caso de um SED, os eventos gerados no ambiente causam mudanças na sua configuração interna e determinam sua evolução dinâmica. Desse modo, um SED é definido por três características básicas, que são:

1. **Ciclo de funcionamento descrito através do encadeamento de eventos** que determinam as tarefas realizadas ou em realização;

2. **Ocorrência de eventos simultâneos**, isto é, vários eventos podem ocorrer ao mesmo tempo para alterar o estado do sistema;

3. **Necessidade de sincronização**, pois para que a evolução dinâmica seja correta, o início de certas atividades requer o término de outras.

A seguir são apresentados alguns exemplos típicos de SEDs, para um maior esclarecimento de suas características.

Exemplo 1 *Considere os dois computadores interligados, como mostrado na Figura 0.2. Este sistema pode ser considerado um SED quando se deseja estudar a comunicação entre os computadores e o ambiente, que é descrita por ações que podem ser*

caracterizadas como eventos. A interação com o ambiente em qualquer um dos computadores efetua-se através da apresentação de dados no vídeo, aquisição de dados através do teclado ou de um "scanner".

- *As características desse sistema são típicas de um SED, desde que há eventos ocorrendo simultaneamente: um computador recebe dados pelo teclado, no mesmo instante em que o outro apresenta dados no vídeo. Seu funcionamento é descrito por seqüências de eventos: "apresentar dados no vídeo", "solicitar novos dados", "ler novos dados", "processar dados" e "apresentar novos dados no vídeo" é uma possível seqüência de eventos nesse sistema.*

- *Um computador somente avalia os novos dados após sua leitura: isto implica em sincronização. Enquanto não é fornecida uma entrada solicitada, o computador se mantém em estado de espera e, ao pressionar a tecla <**ENTER**>, ou o botão do "mouse", o estado do sistema é mudado para o estado de processamento dos dados. A comunicação entre os computadores, é semelhante, desde que o envio de uma mensagem mantém um dos computadores num estado de espera.*

- *Deve existir um sincronismo entre os computadores para garantir que um dado enviado seja corretamente recebido. Desse modo, quando um deles começa a enviar dados, o outro imediatamente inicia um processo de leitura e vice-versa.*

- *Percebe-se também, que o estado do sistema muda instantaneamente com a ocorrência de qualquer um dos eventos citados.*

Figura 0.2: Redes de computadores podem ser consideradas SEDs.

Exemplo 2 *Na Figura 0.3, é apresentado um diagrama simplificado de uma estação ferroviária, que também pode ser considerada um SED. Alguns eventos desse sistema*

são: "abrir portas", "fechar portas", "aviso de partida", "partida do trem", "aviso de chegada" e "parada na estação". Assim, a evolução dinâmica do sistema é representada pelo encadeamento de eventos.

- *A ocorrência de cada um destes eventos, muda o estado do SED abruptamente.*
- *A interação com o ambiente, efetua-se através de eventos do tipo: "entrada de passageiros", "saída de passageiros", "comunicação com a central" e "recepção de avisos".*
- *Vários eventos podem ocorrer simultaneamente, tais como: "aviso de chegada" e "abrir portas" ou "recebimento de aviso de estação ocupada" e "desaceleração para parada".*
- *Há sincronismo: Essa característica é vista na comunicação de um trem com a central que informa o estado da estação (livre ou ocupada). Neste caso, o trem deve responder e manter o movimento, caso a estação esteja livre, ou desacelerar e parar, caso a estação esteja ocupada. No segundo caso, o trem se mantém num estado de espera até receber um novo aviso de estação livre, e iniciar seu movimento, entrar na estação, parar, avisar da chegada e abrir as portas para saída de passageiros.*

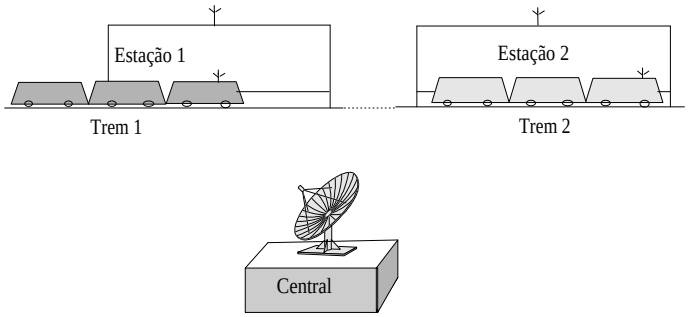

Figura 0.3: Sistema de supervisão de tráfego ferroviário.

Exemplo 3 *Um diagrama simplificado de um sistema de manufatura com recursos compartilhados composto por uma esteira, um braço robótico e uma máquina é mostrado na Figura 0.4. Este sistema também é um SED, pois seu comportamento pode ser*

descrito através de ações tais como "mover esteira", "colocar peça na máquina", "processar peça na máquina" e "retirar peça da máquina". Estes eventos em determinadas seqüências determinam tarefas.

- *Alguns desses eventos podem ocorrer em paralelo, tais como: "mover esteira" e "processar peça na máquina", ou "mover esteira" e "retirar peça da máquina". Entretanto deve haver sincronismo tal como: "colocar peça na máquina" requer que a máquina tenha terminado o processamento da peça e que o evento "retirar peça da máquina" tenha ocorrido, ou "mover esteira" somente pode ocorrer se não houver uma peça em sua extremidade, o que implica em dizer que a mesma somente deve ser movimentada depois do evento "colocar peça na máquina".*

- *A cada ocorrência de um destes eventos, o estado do sistema é modificado. Assim, no estado "esteira com peça na extremidade" e "braço robótico livre", a ocorrência do evento "colocar peça na máquina", muda o estado para "esteira livre para movimentação" e "braço robótico ocupado".*

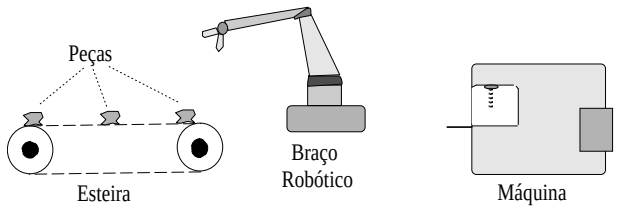

Figura 0.4: Um simples sistema de manufatura.

De modo geral, considera-se que a quantidade de eventos diferentes de um SED é finita. Uma etapa fundamental de qualquer estudo de SEDs é a definição de quais são os eventos e os estados relevantes. Essa definição juntamente com o paradigma de modelagem escolhido proporcionam uma ferramenta matemática para analisar o comportamento dos SEDs.

Quando é definido o paradigma de modelagem, elabora-se um modelo de um SED, a partir do qual, determina-se o espaço de estados. Neste espaço de estados, em muitos casos, torna-se necessário definir um ou mais estados para os quais o SED deva retornar ao completar uma tarefa. Estes estados são conhecidos por estados recorrentes (*home state*). Em algumas aplicações, um destes estados pode ser o estado inicial do sistema. Nesse caso, denomina-se este processo de *reinicialização*.

O problema de controle

Quando um SDVC não apresenta um comportamento dinâmico satisfatório projeta-se um controlador que interligado ao sistema numa configuração realimentada garante que o comportamento desejado seja efetivamente obtido. O projeto desse tipo de sistema de controle começa pela formulação de um problema de controle para o qual um controlador é a solução. Uma situação semelhante ocorre com os SEDs. Entretanto, nesse caso, convenciona-se que a solução do problema de controle de um SED é um supervisor.

O problema de controle de SEDs é, de modo geral, resolvido através da Teoria de Controle Supervisório (TCS) [2, 3]. Uma característica importante da TCS é a separação explícita do sistema a ser controlado (*open loop dynamics*) do controlador propriamente dito (*feedback control*).

A formulação do problema de controle dos SEDs é definida em três etapas:

1. **Modelagem**, na qual se utiliza um determinado paradigma que represente o SED matemática ou visualmente, e que permita determinar seus estados e sua evolução dinâmica;

2. **Especificação de comportamento**, onde se define qual a tarefa que o sistema deve realizar. Essa tarefa é expressa em linguagem natural e formalizada numa linguagem de lógica matemática compatível com o paradigma de modelagem;

3. **Síntese do supervisor**, em que se soluciona o problema de controle e define-se um supervisor. Este supervisor observa os eventos gerados pelo SED e define uma seqüência de ações de controle que garante que a seqüência de eventos que determina o comportamento especificado, se possível, será executada corretamente.

A modelagem de SEDs

Um modelo é uma representação, freqüentemente em termos matemáticos, através da qual são analisadas importantes características do objeto ou sistema que se deseja estudar. Desse modo, se o modelo representa adequadamente o sistema, vários estudos e análises podem ser feitas utilizando-se o próprio modelo, sem o risco, custo ou inconveniência da manipulação direta do sistema real.

Em várias áreas do conhecimento é necessário evitar que estudos sejam realizados diretamente num sistema real. Exemplos são os sistemas astronômicos que podem ser descritos através de modelos nos quais a idade, o nascimento, a interação e a morte das estrelas exigiriam longos períodos de tempo de observação e gerariam imensos amontoados de matéria e energia para qualquer estudo útil. Da mesma forma, os sistemas da física nuclear, que podem ser representados por modelos que descrevem as interações entre partículas subatômicas que criam condições de alto risco para os seres humanos envolvidos no estudo.

Dependendo do grau de detalhamento da representação do sistema, um modelo matemático pode ser muito complexo. De modo geral, este grau de detalhamento define um nível de abstração adotado na elaboração do modelo e determina sua complexidade. O nível de abstração é uma escolha de quem estuda o sistema. Entretanto, nessa escolha é importante considerar qual a utilização que será feita do modelo. Um modelo extremamente detalhado contribui para uma melhor exatidão do estudo mas pode requerer um tempo de computação elevado, não destacar as características mais importantes e apresentar dificuldades para sua caracterização. Por outro lado, um modelo simplificado, isto é, com elevado grau de abstração permite que alguns tipos de sistemas possam ser modelados por paradigmas matemáticos que apresentam um grafo associado. Estes tipos de sistema geralmente apresentam características de tempo discreto.

Na modelagem de SEDs vários paradigmas podem ser considerados. Entretanto, até o presente momento, nenhum desses se tornou aceito como paradigma universal, de forma a solucionar todos os problemas referentes aos SEDs. Os paradigmas mais utilizados na representação de SEDs são: Cadeias de Markov [4]; Teoria das filas [4]; Processos semi-Markovianos generalizados [4]; Álgebra de processos [5]; Álgebra de dióides [6, 7, 8, 9, 10], e suas formalizações específicas como a álgebra Max-Plus [11, 12, 13], álgebra Min-Plus [14, 15, 16, 17] e álgebra de caminhos [18]; Redes de Petri [19, 20, 21, 22, 23, 24, 25, 26]; Teoria de linguagens formais e autômatos (máquinas de estados finitos) [27, 28, 29, 30, 31].

Aqui são utilizados os autômatos e linguagens formais para representar e formular o problema de controle de SEDs.

A especificação de comportamento

Considerando que o paradigma de modelagem e o nível de abstração da representação do SED estão definidos, é necessário especificar qual o comportamento desejado, ou seja, qual a tarefa que o SED deve realizar. Esta etapa é a especificação de comportamento que intuitivamente pode ser feita em linguagem natural, de forma que numa frase se define o que o SED deve realizar. Contudo, uma linguagem natural é potencialmente ambígüa e desse modo é mais adequado utilizar linguagens matemáticas que são formais e permitem superar esta dificuldade.

Dois formalismos matemáticos podem ser utilizados para definir a especificação de comportamento. Estes formalismos matemáticos são: Linguagens formais [29, 28], em que é definido um alfabeto de símbolos que representam os eventos, e sua concatenação define palavras (*seqüências de símbolos*) ou seqüências de eventos, que representam tarefas a ser executadas; Lógica temporal [32, 33, 34, 35, 36, 37, 38], que utiliza proposições, operadores lógicos e operadores temporais para construção de fórmulas lógicas. A avaliação da veracidade ou falsidade da fórmula define se a tarefa é executada ou não.

A síntese do supervisor

A síntese do supervisor ou solução do problema de controle de SEDs é realizada para um dado modelo com o objetivo de satisfazer uma especificação de comportamento desejada.

A partir dos eventos gerados pelo SED o supervisor define quais as ações de controle que devem ser implementadas para que seqüências específicas de eventos sejam observadas. Este ciclo representa a operação de um sistema de controle em malha fechada, como mostrado na Figura 0.5.

Na síntese do supervisor considera-se que os eventos gerados pelo SED podem ser, numa primeira análise, classificados em duas classes distintas e disjuntas, as quais são designadas por: Eventos controláveis: Eventos cuja ocorrência pode ser alterada pela ação de controle, isto é, podem ser habilitados ou desabilitados em qualquer momento; Eventos não-controláveis: Eventos cuja ocorrência não pode ser alterada pela ação de controle, isto é, estão permanentemente habilitados em toda a evolução dinâmica

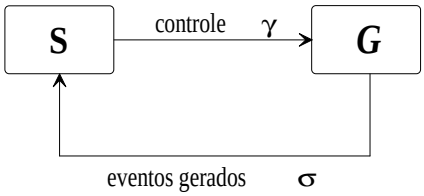

Figura 0.5: Controle supervisório em malha fechada: o gerador gera os eventos σ para o supervisor que determina a estratégia de controle que o controlador deve aplicar ao sistema para realizar a especificação desejada.

do SED. Exemplos de eventos controláveis são: a) o início de uma atividade e b) a parada de uma esteira. Exemplos de eventos não-controláveis são: a) a quebra de uma máquina, b) a falta de energia elétrica e c) o término do processamento de uma peça.

Uma outra classificação do problema de controle de SEDs pode ser feita quando se considera a observação parcial de eventos. Nesse caso, o supervisor recebe os eventos gerados pelo SED que são mapeados em uma etapa intermediária, denominada de observador [39, 40, 41, 42, 43, 44, 45, 46] e a supervisão é feita como mostrado na Figura 0.6.

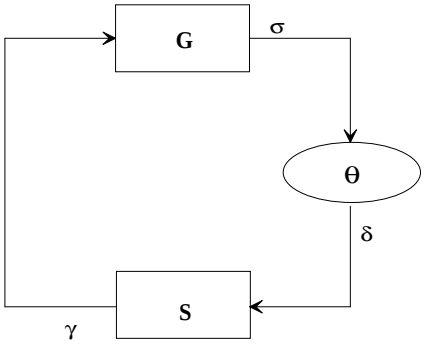

Figura 0.6: Controle supervisório em malha fechada: o gerador gera os eventos σ que são mapeados pelo observador θ, passando-os ao supervisor que determina a estratégia de controle que o controlador deve aplicar ao sistema para realizar a especificação desejada.

O observador classifica os eventos gerados pelo SED em dois tipos: Eventos observáveis, que são observáveis em qualquer instante na evolução dinâmica do sistema, e são mapeados num alfabeto reconhecível pelo supervisor; Eventos não-observáveis,

que são vistos pelo supervisor como palavras vazias, isto é, não sinalizam nenhuma transição de estado. Eventos relacionados com a dinâmica interna dos SEDs, são exemplos típicos.

A formulação genérica do problema de controle no contexto da TCS visa a determinação do supervisor a partir do modelo do sistema controlador e da especificação de comportamento desejada para o SED. Uma dificuldade encontrada na síntese do supervisor é que a tarefa especificada pode incluir ou não eventos do tipo não-controláveis. A ocorrência desses eventos pode levar o sistema a situações de bloqueio ou fazê-lo alcançar estados indesejáveis. Desse modo, a função do supervisor é assegurar que a seqüência de eventos gerada pela sua associação com o SED, implemente a especificação de comportamento desejada, ou seja, fazer com que o SED realize uma tarefa específica.

Exemplo 4 *Pode-se ilustrar como um exemplo típico de bloqueio, a situação no Exemplo 3, em que o braço robótico coloca uma peça na máquina para processamento e, de imediato pegue outra peça na esteira. Esta situação define um bloqueio no sistema, desde que o braço robótico fica com a peça sem poder devolvê-la para a esteira, nem colocá-la na máquina e nem retirar a peça processada da máquina. Por outro lado, é desejável que cada peça processada seja consumida. Caso contrário, a produção deve ser interrompida após ter sido armazenada uma determinada quantidade de peças processadas num depósito. Caso essas condições não sejam satisfeitas, haverá um aumento ilimitado de peças no depósito.*

A metodologia de construção do supervisor também pode ser feita através da divisão do sistema em níveis hierarquicos [47, 48, 49, 50, 51], definidos da seguinte maneira:

- **Baixo nível** que é o sistema sob controle, ou sistema físico real;
- **Alto nível**, o qual é um modelo abstrato e agregado do baixo nível.

A troca de informações entre estes dois níveis é feita através de canais de informação e controle que permitem observar o comportamento do sistema e também transmitir as ações de controle tal como ilustrado no diagrama da Figura 0.7.

Existem várias alernativas disponíveis para resolução do problema de controle supervisório. Aqui é enfatizada a **teoria de controle supervisório** com **observação total e parcial de eventos**, baseada em **autômatos** e **linguagens formais**.

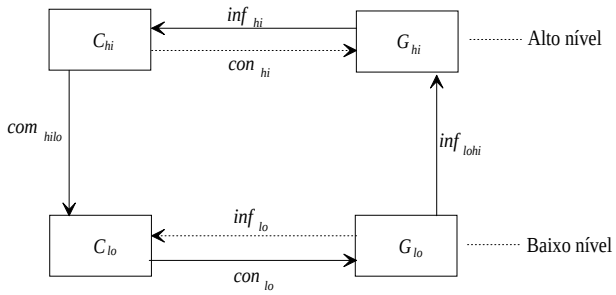

Figura 0.7: Controle hieráquico.

Linguagens Formais e Autômatos

As linguagens formais, são definidas a partir de um alfabeto que é um conjunto de símbolos (letras ou dígitos), representado geralmente pela letra grega maiúscula Σ. Por exemplo: $\Sigma = \{\alpha, \beta, \gamma, \delta\}$ e $\Sigma = \{a, b, c, d, e\}$.

Para um dado alfabeto, define-se uma palavra s como a concatenação de seus elementos, com ou sem repetição. Por exemplo, sendo $\Sigma = \{\alpha, \beta, \gamma, \delta\}$, então $\alpha\alpha, \beta, \alpha\beta, \gamma\delta$, $\gamma\alpha\delta, \gamma\delta\alpha\beta\beta, \beta\delta\delta\delta\alpha\alpha\gamma\delta$, são exemplos de palavras.

O comprimento de uma palavra s é igual ao número de símbolos que a compõe, isto é, é igual à sua cardinalidade, que é representada por $|s|$.

Na teoria de linguagens formais, a palavra vazia é a única palavra de comprimento nulo. Essa palavra é representada pelo símbolo ϵ. Desse modo, $\epsilon \notin \Sigma$, pois ϵ é uma palavra, e não um símbolo do alfabeto Σ.

Definem-se os conjuntos de palavras de mesmo comprimento $|s| = k$ como Σ^k.

Exemplo 5 *Se* $\Sigma = \{\gamma, \zeta\}$ *é um alfabeto, então*

$$\begin{aligned}
\Sigma^0 &= \{\epsilon\} \\
\Sigma^1 &= \{\gamma, \zeta\} \\
\Sigma^2 &= \{\gamma\gamma, \gamma\zeta, \zeta\gamma, \zeta\zeta\} \\
&\vdots
\end{aligned}$$

Note que sendo ϵ a única palavra de comprimento nulo, ela é também o único elemento de Σ^0.

A união dos conjuntos Σ^k gera os conjuntos:

- $\Sigma^+ = \bigcup_{k=1}^{\infty} \Sigma^k = \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots$, e
- $\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \Sigma^3 \cup \cdots = \Sigma^0 \cup \Sigma^+ = \{\epsilon\} \cup \Sigma^+$,

onde Σ^+ é interpretado como o conjunto de todas as palavras que podem ser formadas com os símbolos do alfabeto Σ. No caso de Σ^*, vê-se que este conjunto difere de Σ^+ apenas pela inclusão da palavra nula ϵ.

Uma linguagem, denotada por L, é definida como sendo um conjunto de palavras formadas através de uma concatenação específica de símbolos do alfabeto Σ. Logo, conclui-se que L é uma linguagem sobre o alfabeto dado, Σ, se e somente se $L \subseteq \Sigma^*$. Deve-se observar que a linguagem vazia $L = \{\}$, que não contém nenhuma palavra, é diferente da linguagem formada pela palavra nula $\Sigma^0 = \{\epsilon\}$, a qual contém um único elemento.

Também define-se a concatenação de palavras. Sendo s_1 e s_2 duas palavras formadas a partir dos símbolos de um alfabeto Σ, com

$$s_1 = \sigma_1\sigma_2\sigma_3...\sigma_k \text{ e}$$
$$s_2 = \sigma_{k+1}\sigma_{k+2}...\sigma_n,$$

sua concatenação, representada por s_1s_2, resulta numa nova palavra definida por

$$s = s_1s_2 = \sigma_1\sigma_2\sigma_3...\sigma_k\sigma_{k+1}\sigma_{k+2}...\sigma_n.$$

A concatenação de palavras pode ser estendida para definir a concatenação de linguagens. Se L_1 e L_2 são duas linguagens sobre um mesmo alfabeto, então L_1L_2, que é a linguagem concatenada, é definida por

$$L_1L_2 = \{s_1s_2 | s_1 \in L_1 \wedge s_2 \in L_2\}.$$

Define-se a parte inicial de uma palavra por prefixo dessa palavra. Assim, um prefixo de uma palavra s sobre um alfabeto Σ é qualquer palavra $s_1 \in \Sigma^*$ que pode ser completada com outra palavra $s_2 \in \Sigma^*$ para formar a palavra s. Ou seja, desde que s_1 concatenada com s_2, forma a palavra s, vê-se que s_1 é um prefixo de s. Desta maneira, tem-se que a palavra nula ϵ e a própria palavra s são prefixos da palavra s e assim, uma palavra s tem $|s| + 1$ prefixos. Denota-se por $Pref(s)$, o conjunto de todos os prefixos de uma palavra s.

O prefixo-fechamento, ou simplesmente fechamento da linguagem L, é uma linguagem associada a L, a qual é formada por suas palavras e por todos os seus prefixos. O prefixo-fechamento de L é denotado por $\overline{L}$ e é formalmente definido por

$$\overline{L} = \{s_1 : \exists s_2 \in \Sigma^* \wedge s_1 s_2 \in L\},$$

de onde se vê que $L \subseteq \overline{L}$.

Exemplo 6 *Dada uma linguagem* $L = \{\epsilon, \alpha\beta, \alpha\gamma\beta\}$ *com palavras formadas a partir do alfabeto* $\Sigma = \{\alpha, \beta, \gamma\}$*, o fechamento de* L *é* $\overline{L} = \{\epsilon, \alpha, \alpha\beta, \alpha\gamma, \alpha\gamma\beta\}$*, e portanto* $L \subset \overline{L}$*.*

Exemplo 7 *Dada uma linguagem* $L = \{\epsilon, \alpha, \alpha\alpha, \alpha\alpha\alpha, ...\}$ *com palavras formadas a partir do alfabeto* $\Sigma = \{\alpha\}$*, o fechamento de* L *é* $\overline{L} = \{\epsilon, \alpha, \alpha\alpha, \alpha\alpha\alpha, ...\}$ *e, neste caso,* $L = \overline{L}$*.*

Define-se uma linguagem como prefixo-fechada, ou simplesmente fechada, se e somente se $L = \overline{L}$. Assim, no Exemplo 7, L é prefixo-fechada. Disto decorre que, se L é prefixo-fechada, então, para cada palavra s pertencente à linguagem L, tem-se que $Pref(s) \subseteq L$.

Outras operações definidas para as linguagens formais são: fechamento-*Kleene*, união, interseção e complemento.

O fechamento-*Kleene* representado por L^* é definido como a mesma operação de fechamento usual para o conjunto Σ (alfabeto). Só que neste caso, utilizam-se palavras, as quais podem ter comprimentos maiores que 1. Desta forma, um elemento de L^* é formado pela concatenação de um número finito de elementos de L. O mesmo é definido como

$$L^* = \{\epsilon\} \cup L \cup LL \cup LLL \cup \cdots$$

Esta operação é idempotente, isto é, $(L^*)^* = L^*$.

A união de linguagens é definida por

$$L_a \cup L_b := \{s_a, s_b \in \Sigma^* : (s_a, s_b) \wedge (s_a \in L_a) \wedge (s_b \in L_b)\},$$

a interseção é definida por

$$L_a \cap L_b := \{s_a, s_b \in \Sigma^* : (s_a = s_b) \wedge (s_a \in L_a) \wedge (s_b \in L_b)\}$$

e o complemento (L^c), é definido por

$$L^c := \{s \in \Sigma^* : s \notin L\}.$$

Exemplo 8 *Dadas* $L_1 = \{\beta, \alpha\beta, \alpha\alpha, \beta\alpha\}$ *e* $L_2 = \{\epsilon, \alpha, \beta\alpha, \alpha\beta\}$, *a união de* L_1 *e* L_2 *é*

$$L_1 \cup L_2 = \{\epsilon, \alpha, \beta, \alpha\beta, \alpha\alpha, \beta\alpha\}$$

e sua interseção é

$$L_1 \cap L_2 = \{\beta\alpha, \alpha\beta\}.$$

Exemplo 9 *Se* $\Sigma = \{\alpha\}$ *e* $L = \{\alpha\alpha, \alpha\alpha\alpha, \alpha\alpha\alpha\alpha, ...\}$ *seu complemento é*

$$L^c = \{\epsilon, \alpha\}.$$

Alguns tipos de linguagens podem ser representados de forma compacta através de expressões regulares. A construção de expressões regulares segue as seguintes regras básicas:

1. $\emptyset$ é uma expressão regular denotando o conjunto vazio, ϵ é uma expressão regular denotando o conjunto $\{\epsilon\}$, e σ também é uma expressão regular denotando o conjunto $\{\sigma\}$ para todo $\sigma \in \Sigma$.

2. Se r e s são expressões regulares, então rs, $(r+s)$, r^* e s^* também são expressões regulares.

3. Não há outras expressões regulares que possam ser construídas através da aplicação das regras 1 e 2 um número finito de vezes.

Logo, para representar de forma compacta linguagens complexas utiliza-se σ^* e s^* a repetição do símbolo σ e da palavra s por um número arbitrário de vezes e o símbolo $+$ representando o operador lógico "*ou*", indicando uma opção entre duas possibilidades.

Exemplo 10 *A linguagem*

$$L = \{\epsilon, \alpha, \alpha\beta, \alpha\beta\beta, \alpha\beta\beta\alpha, \alpha\beta\beta\alpha\beta, \alpha\beta\beta\alpha\beta\beta, ...\}$$

pode ser representada de forma compacta pela expressão regular

$$L = \left\{\left(\alpha\beta^2\right)^* ((\alpha + \alpha\beta) + \epsilon)\right\}.$$

Exemplo 11 *A expressão regular* $(\alpha\beta)^* + \gamma$ *representa a linguagem*

$$L = \{\epsilon, \gamma, \alpha\beta, \alpha\beta\alpha\beta, \alpha\beta\alpha\beta\alpha\beta, ...\}.$$

Definição 1 *Qualquer linguagem que pode ser denotada por uma expressão regular é denominada de* Linguagem Regular.

Esta Definição formaliza o conceito de linguagem regular, que é de fundamental importância para o estudo dos autômatos seqüenciais finitos [27, 28, 29, 30].

Autômatos

Um autômato é um dispositivo que pode reconhecer uma determinada linguagem e que também pode ser usado para representar certos tipos de sistemas dinâmicos. Um autômato têm estados que representam as condições operacionais do dispositivo ou as situações do sistema dinâmico. Se o número de estados é finito o autômato é denominado de autômato finito ou máquina de estado finito e se o número de estados é infinito o autômato é denominado de autômato infinito ou máquina de estado infinito.

Para reconhecer as palavras de uma determinada linguagem o autômato lê sequencialmente os símbolos e esta leitura produz, eventualmente, uma mudança de estado. Uma palavra é dita reconhecida se o estado final alcançado após a leitura do último símbolo pertence a um conjunto de *estados marcados*. O estado em que um autômato se encontra antes da leitura do primeiro símbolo é denominado de *estado inicial*. Desse modo, um autômato pode ser visto como uma entidade de controle que internamente tem uma variável que representa o estado do autômato. Logo, cada símbolo lido atualiza esta variável de acordo com uma função de transição que associa um novo estado a cada par (*símbolo*, *estado*) ou (*evento*, *estado*). Se a função de transição leva a mais de um estado quando certos símbolos são lidos, o autômato é denominado de autômato não determinístico. Por outro lado, se a função de transição leva a um único estado para cada símbolo lido, o autômato é dito determinístico.

Definição 2 *Um autômato determinístico finito, ou simplesmente um autômato é uma quíntupla* $A = (Q, \Sigma, \delta, q_0, Q_m)$*, onde:* Q *é um conjunto finito de estados* q*;* Σ *é o alfabeto ou conjunto de símbolos* σ*;* $\delta : \Sigma \times Q \rightarrow Q$ *é a função de transição de estados,*

onde

$$\delta(\epsilon, q) = q \qquad e$$
$$\delta(\sigma, q) = q', \quad \textit{para } q,\ q' \in Q \textit{ e } \sigma \in \Sigma;$$

$q_0 \in Q$ *é o estado inicial;* $Q_m \subseteq Q$ *é o conjunto de estados marcados.*

Observando a função de transição de estados δ, vemos que q' somente será um estado do autômato A, se o símbolo σ for uma entrada aceita por ele.

Graficamente, um autômato pode ser representado por um diagrama de transição de estados, que é um grafo direcionado, onde seus vértices são os estados e os arcos representam as funções de transição. Logo, um autômato definido por $A = (Q, \Sigma, \delta, q_0, Q_m)$ pode ser representado graficamente pelo grafo orientado $G = \{V, W\}$, no qual

$$V = Q,\ W = \{(q, q', \sigma) : q, q' \in Q \wedge \sigma \in \Sigma \wedge \delta(\sigma, q) = q',$$

os vértices ou estados são representados por círculos, os estados marcados, por círculos em linha dupla (dois círculos concêntricos) e o estado inicial é indicado por uma seta que não é saída de nenhum vértice.

Exemplo 12 *O autômato representado graficamente na Figura 0.8, é definido por:*

- $\Sigma = (\alpha, \beta, \gamma)$;
- $Q = (0, 1, 2)$;
- $\delta(\alpha, 0)$=1, $\delta(\alpha, 2)$=2, $\delta(\beta, 0)$=2, $\delta(\beta, 1)$=0, $\delta(\beta, 2)$=1, $\delta(\gamma, 1)$=1, $\delta(\gamma, 2)$=0;
- $q_0 = 0$ *e*
- $Q_m = \{1, 2\}$.

onde os círculos são estados, os círculos concêntricos são estados marcados, a seta que não é saída de nenhum vértice (*estado*) *é o estado inicial e os arcos indicam as funções de transição de estados.*

. *Nesta figura, observa-se que, do estado inicial* (*estado 0*)*, através do evento* α (*arco* α) *há a mudança para o estado marcado 1, que é representado pela função de transição* $\delta(\alpha, 0) = 1$. *Neste estado, a ocorrência do símbolo* γ *não produz mudança de estado* (*função de transição* $\delta(\gamma, 1) = 1$). *Contudo, no estado inicial,*

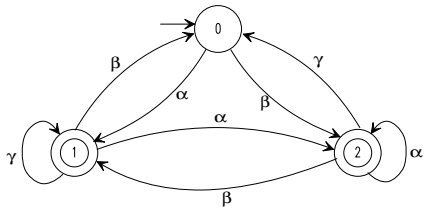

Figura 0.8: Autômato.

se houver a ocorrência do símbolo β, o autômato segue para o estado marcado 2 (função de transição $\delta(\beta, 0) = 2$) e assim por diante.

Observa-se que quanto maior o número de estados do autômato, mais difícil é sua visualização e sua compreensão através de sua representação gráfica.

Exemplo 13 *Considere um setor de uma fábrica que é constituído de um braço robótico, uma máquina que processa peças-brutas e um déposito para armazenar peças-trabalhadas* (buffer)*. O braço robótico transporta as peças-brutas e as peças-trabalhadas. O buffer pode armazenar uma quantidade máxima de três peças-trabalhadas. O buffer de peças-brutas e a retirada de peças-trabalhadas do* buffer *para outro setor da fábrica não são representados nesse exemplo. Os eventos desse sistema são: início de processamento de uma peça-bruta (α), braço robótico retira uma peça-trabalhada na máquina (β), braço robótico coloca uma peça-trabalhada no* buffer *(γ) e recarregamento da máquina com duas peças-brutas (η). Desse modo, os estados do autômato são:*

estado	*máquina*	*braço*	buffer
1	*livre*	*livre*	*vazio*
2	*processando*	*livre*	*vazio*
3	*livre*	*carregando peça*	*vazio*
4	*processando*	*carregando peça*	*vazio*
5	*processando*	*livre*	*1 peça*
6	*livre*	*carregando peça*	*1 peça*
7	*processando*	*carregando peça*	*1 peça*
8	*processando*	*livre*	*2 peças*
9	*livre*	*carregando peça*	*2 peças*
10	*livre*	*livre*	*3 peças*

e o estado inicial é máquina carregada com três peças-brutas estando o braço robótico livre e o buffer *vazio. O estado marcado é o estado* 10*. Este autômato está representado graficamente na Figura 0.9.*

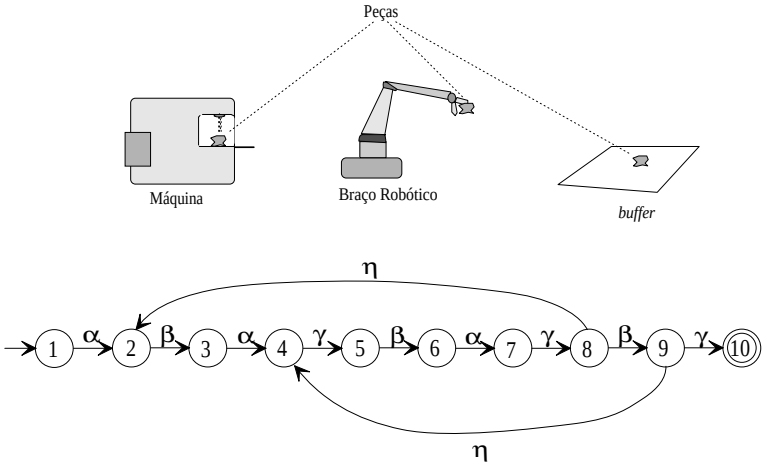

Figura 0.9: Sistema e autômato.

Neste segundo exemplo, a função de transição não é definida para todos os estados para evitar situações absurdas do tipo, *braço robótico retira peça-trabalhada do buffer e braço robótico coloca uma peça-trabalhada na máquina.* De modo geral, quando se utiliza um autômato para representar o comportamento de um SED, é necessário restringir a função de transição, como será apresentado na próxima seção.

As seqüências de símbolos $\{\beta\alpha, \alpha\gamma\}$ e $\{\alpha\beta\alpha\gamma\beta\alpha\gamma\beta\gamma, \alpha\beta\alpha\gamma\beta\alpha\gamma\beta\eta\gamma\beta\alpha\gamma\beta\gamma\}$ dos exemplos anteriores constituem palavras definidas para o alfabeto do autômato. Isso sugere que a função de transição pode ser estendida e definida para processar seqüências de símbolos.

Definição 3 *Seja um autômato* $A = (Q, \Sigma, \delta, q_0, Q_m)$ *e seja uma função de transição estendida* δ^**, que é a função* $\delta^* : \Sigma^* \to Q$*, de tal forma que:*

$$\begin{aligned} &\delta^*(\epsilon, q) = q \qquad\qquad e \\ &\delta^*(s\sigma, q) = \delta\ \ (\sigma, \delta^*(s, q)) \text{ para } q \in Q \text{ e } s \in \Sigma^*. \end{aligned}$$

Desde que $\delta^*(\sigma, q) = \delta(\sigma, \delta^*(s, q)) = \delta(\sigma, q)$ para o caso em que $s = \epsilon$, pode-se utilizar δ ao invés de δ^*, por uma questão de conveniência.

Exemplo 14 *Considerando o Exemplo 13 e as palavras* $s_1 = \beta\alpha, s_2 = \gamma\beta\alpha\gamma$, *o autômato fica representado como visto na Figura 0.10, que simplifica sua representação em termos de estados intermediários* (*quando estes não têm importância na análise que se deseja realizar*). *Essa representação simplificada é denominada de grafo de transição.*

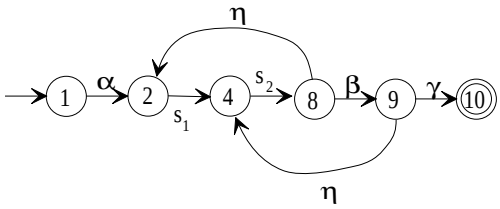

Figura 0.10: Grafo de transição.

Como se vê, conhecida a função de transição e o estado atual do autômato, é possível determinar seu estado após o processamento de um símbolo dado. Assim, processando uma palavra a partir de um dado estado, pode-se confirmar se o estado alcançado pertence ou não ao conjunto de estados marcados Q_m. A linguagem marcada ou reconhecida pelo autômato é definida por

$$L_m(A) = \{s : s \in \Sigma^* \wedge \delta(s, q_0) \in Q_m\},$$

ou seja, são palavras, que a partir do estado inicial q_0, levam o autômato a um estado marcado. Esta linguagem pode ser encontrada seguindo os arcos no grafo do mesmo.

Exemplo 15 *Seja* $\Sigma = \{\alpha, \beta\}$ *um conjunto de eventos* (*alfabeto*) *e as linguagens definidas por*

$$\mathcal{L}_\alpha = \{\epsilon, \alpha, \alpha\alpha, \beta\alpha, \alpha\alpha\alpha, \alpha\beta\alpha, \beta\alpha\alpha, \beta\beta\alpha, ...\}$$

e

$$\mathcal{L}_\beta = \{\epsilon, \beta, \beta\beta, \beta\alpha, \beta\beta\beta, \beta\alpha\beta, \alpha\beta\beta, \alpha\alpha\beta, ...\}$$

que consistem de todas as palavras formadas com símbolos de Σ *sempre seguidas do evento* α *e* β, *respectivamente. Observe que* $\mathcal{L}_\alpha \subset L_m(A)$ *e* $\mathcal{L}_\beta \subset L_m(A)$ *sendo* $L_m(A)$ *a linguagem marcada pelo autômato* $A = (Q, \Sigma, \delta, q_0, Q_m)$, *apresentado na Figura 0.11, com* $Q = \{0, 1\}$, $q_0 = 0$, $Q_m = \{0, 1\}$ *e* δ *é definida através de* $\delta(\alpha, 0) = 1$, $\delta(\beta, 0) = 0$, $\delta(\alpha, 1) = 1$, $\delta(\beta, 1) = 0$. *Note que nesse caso,* δ *é uma função completa e portanto, a linguagem gerada pelo autômato* A *é* $L(A) = \Sigma^*$.

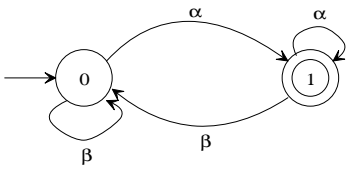

Figura 0.11: Autômato com linguagem marcada L_m=$\{\alpha,\ \alpha\alpha,\ \beta\alpha,\ \alpha\alpha\alpha,\ \alpha\beta\alpha,\ \beta\alpha\alpha,\ \beta\beta\alpha,\ ...\}$, consistindo de todas as palavras de α e β, seguidas por α, construída através dos símbolos do alfabeto Σ=$\{\alpha,\ \beta\}$.

Vários autômatos podem reconhecer uma mesma linguagem e podem ter uma mesma linguagem marcada L_m.

Definição 4 *Considere dois autômatos A_1 e A_2. Diz-se que A_1 e A_2 são equivalentes se $L(A_1) = L(A_2)$ e $L_m(A_1) = L_m(A_2)$.*

Exemplo 16 *Os autômatos da Figura 0.12 têm a mesma linguagem gerada*

$$L(A) = \{\epsilon, \alpha, \alpha\alpha, \beta\alpha, \alpha\alpha\alpha, \alpha\beta\alpha, \beta\alpha\alpha, \beta\beta\alpha, ...\}$$

e a mesma linguagem marcada

$$L_m(A) = \{\alpha, \alpha\alpha, \beta\alpha, \alpha\alpha\alpha, \alpha\beta\alpha, \beta\alpha\alpha, \beta\beta\alpha, ...\}$$

do autômato do Exemplo 15. Logo, estes autômatos são equivalentes.

Uma outra consideração a ser feita com relação aos autômatos, é a condição de bloqueio. Diz-se que um autômato é bloqueável se $L_m(A) \subset L(A)$.

Exemplo 17 *O autômato visto na Figura 0.13 tem $L(A) = (\alpha\beta)^* \alpha(\beta + (\kappa + \epsilon))$ e $L_m(A) = (\alpha\beta)^* \alpha$. Este autômato é bloqueável, desde que $L_m(A) \subset L(A)$, Esta condição pode ser observada no estado 3, que é não marcado. Quando o autômato se encontra nesse estado, nele permanece, não podendo mais atingir nenhum estado marcado.*

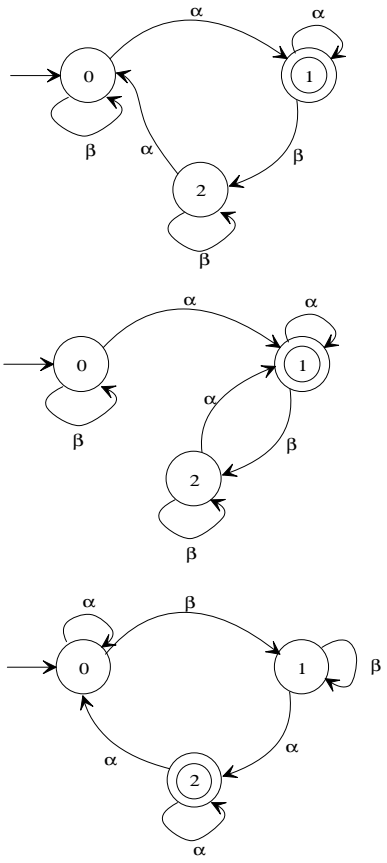

Figura 0.12: Autômatos com linguagem L=$\{\epsilon$, α, $\alpha\alpha$, $\beta\alpha$, $\alpha\alpha\alpha$, $\alpha\beta\alpha$, $\beta\alpha\alpha$, $\beta\beta\alpha$, ...$\}$ e linguagem marcada L_m=$\{\alpha$, $\alpha\alpha$, $\beta\alpha$, $\alpha\alpha\alpha$, $\alpha\beta\alpha$, $\beta\alpha\alpha$, $\beta\beta\alpha$, ...$\}$, construída através dos símbolos do alfabeto Σ=$\{\alpha$, $\beta\}$.

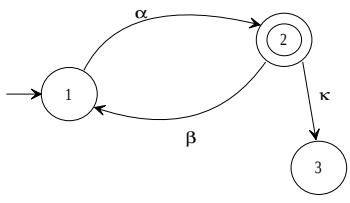

Figura 0.13: Autômato com bloqueio.

Geradores

Um gerador é um autômato no qual a função de transição é definida para um subconjunto próprio de Σ^*. Os geradores são mais compatíveis com a representação de SEDs.

Definição 5 *Um gerador é uma quíntupla $G = (Q, \Sigma, \delta, q_0, Q_m)$, onde os elementos Q, Σ, q_0, Q_m, e δ têm a mesma definição do autômato 2, com δ definido como a função, geralmente parcial, de transição de estados.*

A diferença existente entre os autômatos e os geradores é que, no caso dos autômatos, a função de transição não pode ser parcial (definida apenas para um subconjunto de eventos para cada estado do gerador).

Semelhantemente aos autômatos, tem-se o conjunto de eventos, a função de transição estendida, sua linguagem e sua linguagem marcada, como com as seguintes definições:

Definição 6 *Para um gerador $G = (Q, \Sigma, \delta, q_0, Q_m)$, associa-se a cada estado $q \in Q$ o conjunto de eventos definidos $\Sigma(q)$, definido por:*

$$\Sigma(q) = \{\sigma : \sigma \in \Sigma \wedge \delta(\sigma, q)!\} \tag{0.1}$$

com $\delta(\sigma, q)!$ identificando que δ é definido para o par (σ, q).

Definição 7 *Seja um gerador $G = (Q, \Sigma, \delta, q_0, Q_m)$. Sua função de transição estendida, denotada por δ^*, é uma função*

$$\delta^* : \Sigma^* \times Q \to Q$$

tal que

$$\delta^*(\epsilon, q) = q \text{ e } \delta^*(s\sigma, q') = \delta(\sigma, \delta^*(s, q)) \tag{0.2}$$

$$\forall\, q \in Q, s \in \Sigma^*, q' = \delta^*(s, q) \text{ e } \delta^*(\sigma, q')! \tag{0.3}$$

Definição 8 *Seja um gerador $G = (Q, \Sigma, \delta, q_0, Q_m)$. Sua linguagem gerada $L(G)$ é:*

$$L(G) = \{s : s \in \Sigma^* \wedge \delta(s, q_0)!\}. \tag{0.4}$$

A função de transição estendida é, de modo geral, representada por δ, ao invés de δ^* pelas mesmas razões dos autômatos, ou seja, por conveniência.

Também é observado que a linguagem gerada *L(G)* por um gerador é *prefixo-fechada*, isto é, $\forall G = (Q, \Sigma, \delta, q_0, Q_m), L(G) = \overline{L(G)}$.

Analogamente aos autômatos, a linguagem marcada do gerador é definida como a seguir:

Definição 9 *Dado um gerador* $G = (Q, \Sigma, \delta, q_0, Q_m)$, *a linguagem marcada de* G, *denotada por* $L_m(G)$, *é* $L_m(G) = \{s : s \in \Sigma^* \wedge \delta(s, q_0) \in Q_m\}$.

Como o comportamento de um SED é caracterizado pela ocorrência de eventos, isto é, um SED gera palavras de comprimento crescente, à medida que evolui, e de acordo com o alfabeto gerado pelo SED, é possível encontrar seqüências de símbolos, isto é, palavras que não representam seqüências de eventos fisicamente possíveis. Assim, a linguagem gerada pelo sistema é um subconjunto próprio de Σ^*. Desta forma, esta linguagem gerada inclui, para cada palavra, todos os seus prefixos.

A linguagem *prefixo-fechada* representa o comportamento lógico de um SED, em que não ocorrem eventos simultâneos. A mesma é definida formalmente, da seguinte maneira:

Definição 10 *A linguagem* prefixo-fechada *que representa o comportamento lógico de um SED é denominada de* linguagem gerada do sistema;

Definição 11 *A linguagem que representa o conjunto de todas as tarefas que podem ser executadas por um SED, é denominada* linguagem marcada do sistema.

Considerando que L é a linguagem gerada por um SED e que L_m, seja sua linguagem marcada, de acordo com estas Definições, tem-se que a linguagem marcada L_m contém as palavras geradas pelo SED que também gera todos os seus prefixos, ou seja, um SED produz as palavras contidas em $\overline{L_m}$. Dessa forma, a linguagem marcada do SED não é necessariamente *prefixo-fechada*. Assim, é visto que $L_m \subseteq \overline{L_m} \subseteq L = \overline{L}$.

Para representar o comportamento de um SED através de um autômato, é necessário restringir sua função de transição, definindo-a apenas para alguns pares (*evento*, *estado*) do conjunto $\Sigma \times Q$. Essa restrição é inerente na definição do gerador e desse

modo, é possível representar as linguagens gerada L e marcada L_m dos SEDs. Assim, um SED com linguagem gerada L e linguagem marcada L_m, é representado por um gerador, de tal forma que $L(G) = L$ e $L_m(G) = L_m$.

Semelhantemente ao autômatos, é necessário determinar se os estados do gerador são alcançáveis e se ele reconhece alguma palavra. Para isto, definem-se a acessibilidade e a coacessibilidade dos geradores.

Definição 12 *A componente acessível de um gerador $G = (Q, \Sigma, \delta, q_0, Q_m)$, denotada por $A_c(G)$ é:*

$$A_c(G) = (Q_{ac}, \Sigma, \delta_{ac}, q_0, Q_{ac,m}), \quad onde$$

$$Q_{ac} = \{q : \exists s \in \Sigma^* \wedge \delta(s, q_0) = q\};$$
$$Q_{ac,m} = Q_{ac} \cap Q_m;$$
$$\delta_{ac} = \delta \mid (\Sigma \times Q_{ac}).$$

Nesta Definição, δ_{ac} é a função δ restrita ao domínio $\Sigma \times Q_{ac}$. Assim, um gerador G é dito acessível quando $G = A_c(G)$. A componente acessível de um gerador é a garantia da existência de palavras que o mesmo pode processar a partir do estado inicial, embora que nenhuma seja reconhecida, isto é, marcada.

Definição 13 *A componente coacessível de um gerador $G = (Q, \Sigma, \delta, q_0, Q_m)$, denotada por $Co(G)$ é:*

$$Co(G) = (Q_{co}, \Sigma, \delta_{co}, q_0, Q_{co,m}), \quad onde$$

$$Q_{co} = \{q : q \in Q \wedge \exists\, s \in \Sigma^* \wedge \delta(s, q) = q_m\};$$
$$Q_{co,m} = Q_{co} \cap Q_m;$$
$$\delta_{co} = \delta \mid (\Sigma \times Q_{co}).$$

A componente coacessível de um gerador garante a existência de estados reconhecidos a partir de qualquer estado do autômato.

Definição 14 *Dado um gerador $G = (Q, \Sigma, \delta, q_0, Q_m)$, este será coacessível se e somente se, toda palavra em $L(G)$ for um prefixo de uma palavra em $L_m(G)$, isto é:*

$$L(G) \subseteq \overline{L_m(G)}. \tag{0.5}$$

Disto, vê-se que em um gerador coacessível, existe pelo menos uma seqüência de eventos que o leva a um estado marcado.

Os geradores que são, ao mesmo tempo, acessíveis e coacessíveis, são denominados ajustados ou *trim*.

Exemplo 18 *Na Figura 0.14(a), é visto um autômato G. Este gerador não é acessível devido ao fato de que o estado 6 não é alcançável a partir do estado 0. Na Figura 0.14(b), encontra-se um gerador semelhante mas sem o estado 6. Neste caso, o gerador é acessível pois todos os estados são alcançados. Por outro lado, para encontrar o gerador coacessível, é necessário identificar os estados de G que não são coacessíveis a partir do estado marcado 2. Estes estados são: 3, 4 e 5. Retirando esses estados e suas respectivas transições, o gerador resultante torna-se coacessível. Note que o estado 6 não é retirado, pois ele não é alcançável a partir do estado marcado 2. Este gerador coacessível é visto na Figura 0.14(c). Finalmente, o gerador* trim*, está apresentado na Figura 0.14(d), pois a ordem em que as operações de acessibilidade e de coacessibilidade são tomadas não afetam o resultado final.*

Composição de autômatos

Uma das alternativas de modelagem de SEDs requer a decomposição do sistemas em sub-sistemas e, para cada sub-sistema é definido um autômato que representa seu comportamento dinâmico. No entanto, para analisar o sistema completo é necessário remontar essa decomposição para simplificar o estudo. Neste caso, é necessário compôr os autômatos que representam os sub-sistemas para obter um autômato do sistema. Há duas operações de composição que podem ser realizadas com autômatos finitos: a composição por produto, representada por $\times$ e a composição paralela, representada por $||$. A composição paralela é denominada de composição síncrona e a composição por produto é denominada de composição completamente síncrona.

Definição 15 *Sejam* $A_1 = (Q_1, \Sigma_1, \delta_1, q_{0_1}, Q_{m_1})$ *e* $A_2 = (Q_2, \Sigma_2, \delta_2, q_{0_2}, Q_{m_2})$ *dois autômatos. O autômato* $A_3 = A_1 \times A_2$ *resultante da composição por produto é definido por* $A_3 := (Q_3, \Sigma_3, \delta, q_{0_3}, Q_{m_3})$ *com* $Q_3 = Q_1 \times Q_2$, $\Sigma_3 = \Sigma_1 \cup \Sigma_2$, $q_{0_3} = (q_{0_1}, q_{0_2})$,

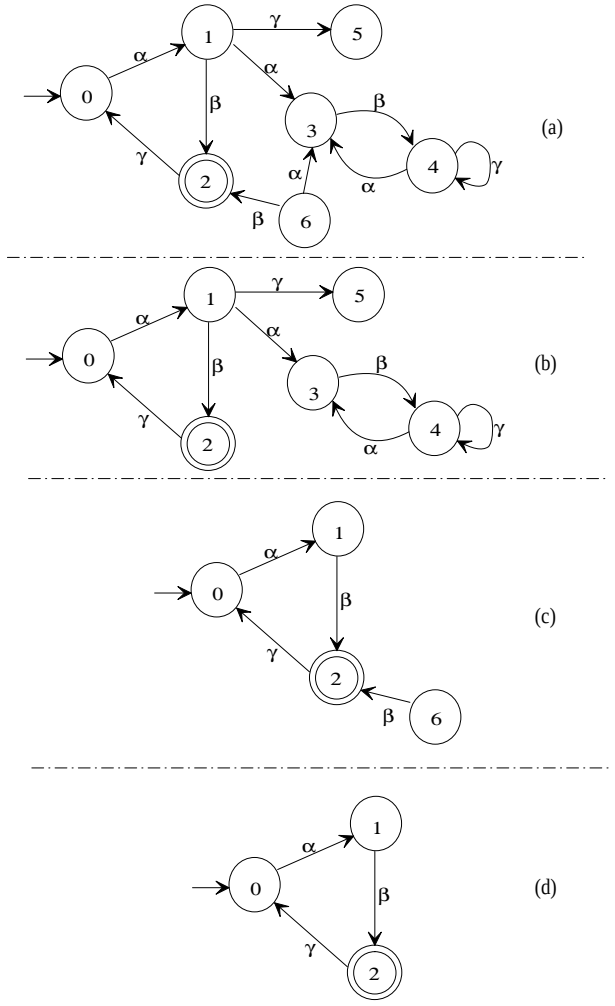

Figura 0.14: Gerador (a) e geradores acessível (b), coacessível (c) e *trim* (d).

$Q_{m_3} = Q_{m_1} \times Q_{m_2}$ *e*

$$\delta_3(\sigma,(q_{i_1},q_{i_2}))=\begin{cases} (\delta_1(\sigma,q_{i_1}),\delta_2(\sigma,q_{i_2}))=(q_{i'_1},q_{i'_2}) & se\ \exists\sigma,\delta_1(\sigma,q_{i_1})=q_{i'_1}\ e\ \delta_2(\sigma,q_{i_2})=q_{i'_2} \\ (\delta_1(\sigma,q_{i_1}),q_{i_2})=(q_{i'_1},q_{i_2}) & se\ \exists\sigma,\delta_1(\sigma,q_{i_1})=q_{i'_1}\ apenas\ em\ A_1 \\ (q_{i_1},\delta_2(\sigma,q_{i_2}))=(q_{i_1},q_{i'_2}) & se\ \exists\sigma,\delta_2(\sigma,q_{i_2})=q_{i'_2}\ apenas\ em\ A_2 \\ indefinido & caso\ contrário \end{cases}$$

Exemplo 19 *O produto dos autômatos* A_1 *e* A_2*, apresentados da Figura 0.15*(a) *e 0.15*(b)*, respectivamente, onde os elementos de* Σ_1 *são os mesmos elementos de* Σ_2*, está apresentado na Figura 0.15*(c)*. Pode-se ver que nos dois autômatos* A_1 *e* A_2*, só se encontra um único arco* (α) *que sai do estado* x *para o estado* x*, em* A_1 *e que, equivalentemente, sai do estado* 0 *para o estado* 1 *no autômato* A_2*. Por outro lado, quando o autômato* A_2 *encontra-se no estado* 1*, só há um arco saindo do mesmo, que é equivalente no autômato* A_1*, que também é o arco* α*. Também, o estado* $(x,0)$ *do autômato construído pelo produto de* A_1 *e* A_2*, não é marcado porque o estado inicial do autômato* A_2*, não é. Porém, o estado* $(x,1)$ *do autômato construído pelo produto de* A_1 *e* A_2 *é marcado, pois em ambos os autômatos, a função de transição* α*, leva de um estado marcado para outro estado marcado.*

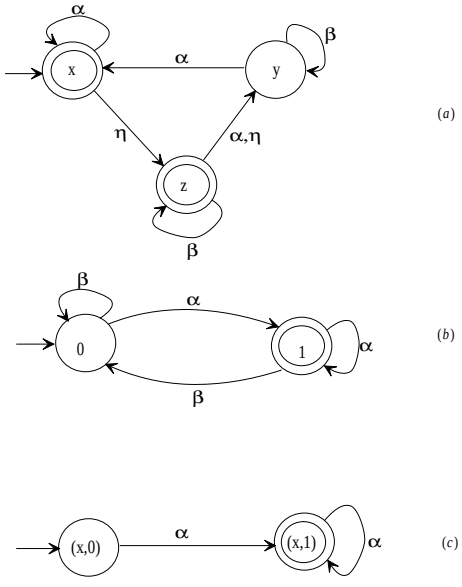

Figura 0.15: Autômato A_1 (a), autômato A_2 (b) e produto $A_1 \times A_2$ (c).

Exemplo 20 *A composição entre os autômatos A_1 e A_2, apresentados da Figura 0.16(a) e 0.16(b), respectivamente, onde somente o elemento $\eta \in \Sigma_1 \cap \Sigma_2$, está apresentado na Figura 0.16(c).*

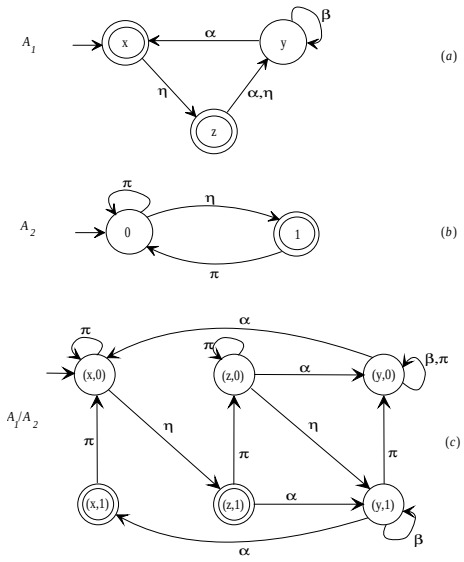

Figura 0.16: Autômato A_1 (a), autômato A_2 (b) e produto $A_1 \times A_2$ (c).

Quando $\Sigma_1 \cap \Sigma_2 = \emptyset$, o procedimento para a construção da composição síncrona de dois autômatos A_1 e A_2 define um autômato que gera ambas linguagens $L(A_1)$ e $L(A_2)$, independentemente. Esta operação é conhecida como composição *shuffle*, e é formalizada de acordo com a Definição 15 de produto síncrono, onde a função de transição δ_3 só é definida para os dois últimos casos, ou seja,

$$\delta_3(\sigma,(q_{i_1},q_{i_2}))=\begin{cases} (\delta_1(\sigma,q_{i_1}),q_{i_2})=(q_{i'_1},q_{i_2}) & \text{se } \exists\sigma,\delta_1(\sigma,q_{i_1})=q_{i'_1} \text{ apenas em } A_1 \\ (q_{i_1},\delta_2(\sigma,q_{i_2}))=(q_{i_1},q_{i'_2}) & \text{se } \exists\sigma,\delta_2(\sigma,q_{i_2})=q_{i'_2} \text{ apenas em } A_2. \end{cases}$$

Exemplo 21 *A composição $shuffle$ dos autômatos A_1 e A_2, apresentados nas Figuras 0.17(a) e 0.17(b), respectivamente, onde todos os elementos de Σ_1 são diferentes dos elementos de Σ_2, está apresentado na Figura 0.17(c). Pode-se ver que esta composição se apresenta com os dois autômatos A_1 e A_2 produzindo suas linguagens independentemente.*

O produto de autômatos, é uma das operações utilizada na Teoria de Controle Supervisório para fundamentar a composição entre o supervisor e o gerador, determinando assim, o acoplamento que define o sistema supervisionado.

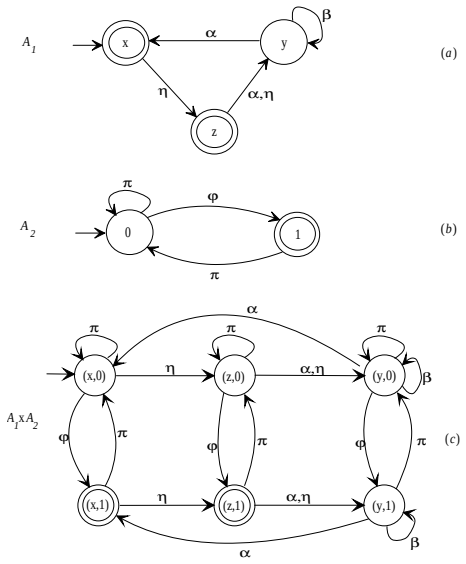

Figura 0.17: Autômato A_1 (a), autômato A_2 (b) e produto composição $shuffle$ de A_1 e A_2 (c).

Modelagem de Sistemas a Eventos Discretos

Para um determinado SED e, definidas as propriedades que se deseja estudar, deve-se escolher um conjunto de estados possíveis de serem alcançados quando o sistema evolui dinamicamente, um conjunto de estados marcados, que representem tarefas concluídas, e um conjunto de transições, que representam eventos ou ações que determinam a evolução dinâmica. Essas escolhas dependem do nível de abstração desejado que é definido no detalhamento da descrição textual do SED. Desse modo, uma etapa preliminar na construção do modelo de um SED por meio de um gerador, exige, inicialmente, a definição dos seguintes conjuntos:

1. O conjunto de estados do SED, deve incluir o estado inicial, de onde o sistema inicia sua evolução e todos os outros estados alcançáveis a partir deste.

2. Com o conjunto de estados, deve-se definir quais os estados que definem completamente a execução das tarefas, ou seja, os estados marcados.

3. O conjunto de transições do SED é formado por todas as ações que definem mudanças de estado no sistema.

A partir destes conjuntos, efetua-se à construção gráfica do modelo, de acordo com os seguintes passos:

1. Desenhe um círculo com uma seta que não é saída de nenhum estado, para representar o estado inicial. Esse estado deve ser etiquetado com um número ou com o nome que descreve a situação do sistema no estado inicial. Quando utiliza-se uma etiquetação numérica, o estado inicial é etiquetado com o número 0.

2. Desenhe outros círculos para representar os demais estados do sistema, marcados ou não marcados. Esses estados também devem ser etiquetados com uma numeração crescente, a partir do número 1, ou com termos que descrevam a situação do sistema em cada estado.

3. Para os estados que representam tarefas completadas (estados marcados), desenhe um outro círculo interno;

4. Conecte os círculos desenhados e numerados, através de arcos etiquetados com a função de transição que descreve a ação ou evento que promove a mudança de estado do sistema, caso exista alguma função de transição possível entre esses estados. Se não houver transição possível não deve ser desenhado nenhum arco.

Com esta formulação, constrói-se um modelo de um SED por meio de um gerador.

Exemplo 22 *Considere um sistema simplificado de processamento de material reciclável mostrado na Figura 0.18: há um local de armazenamento para materiais a serem reciclados* (la)*, inicialmente com sua capacidade completa de 2 itens. Há um braço robótico* (br) *que transporta as peças desse local para uma máquina que processa peças* (mp)*. Este mesmo braço transporta a peça trabalhada na máquina para um local de consumo* (lc)*. Sempre que uma peça é processada ou trabalhada, ela é transportada de volta ao local de armazenamento para materiais reciclados, pelo mesmo braço robótico. Este sistema apresenta 11 estados diferentes, os quais são descritos como na tabela a*

seguir:

Estados	*Situação do sistema*
estado inicial (0)	*la com 2 peças, lc vazio, mp livre, br livre;*
estado 1	*br com peça, la com 1 peça, lc vazio, mp livre;*
estado 2	*br livre, la com 1 peça, lc vazio, mp ocupada (não processando);*
estado 3	*br livre, la com 1 peça, lc vazio, mp ocupada (processando);*
estado 4	*br com peça, la com 1 peça, lc com 1 peça, mp livre;*
estado 5	*br livre, la com 1 peça, lc com 1 peça, mp livre;*
estado 6	*br com peça, la vazio, lc com 1 peça, mp livre;*
estado 7	*br livre, la vazio, lc com 1 peça, mp ocupada (não processando);*
estado 8	*br livre, la vazio, lc com 1 peça, mp ocupada (processando);*
estado 9	*br com peça, la vazio, mp livre;, lc com 1 peça;*
estado 10	*br livre, la vazio, lc com 2 peças, mp livre.*

Definindo como estados marcados lc com pelo menos uma peça processada pronta para o consumo com sistema parado (br e mp livres) e, também, toda peça processada já consumida e pronta para ser reciclada, tem-se que os estados marcados são os estados 0, 4 e 10. Definindo as funções de transição tem-se: α - br pegar peça; β - br soltar peça; λ - mp processar peça; μ - peça ser consumida, constrói-se o modelo como mostrado seqüencialmente nas Figuras 0.19(a), 0.19(b) e 0.19(c), onde esta última é o modelo do SED.

Exemplo 23 *Considere um sistema simplificado de redes de computadores mostrado na Figura 0.20: há dois computadores (c_1 e c_2) e uma impressora (i). Cada computador pode ser ligado/desligado separadamente. Considera-se que a impressora sempre é ligada quando um dos computadores é ligado. Há uma fila de impressão com uma capacidade máxima de dois trabalhos. Estando ligado, qualquer computador pode enviar no máximo dois trabalhos para uma fila de impressão. Assim, se um computador envia um trabalho, ou ele envia outro, ou o outro computador envia seu trabalho. A fila de impressão obedece a prioridade de quem primeiro envia, é o primeiro que imprime. Assim, identificando os estados, tem-se ao todo 24 estados, os quais são descritos como:* estado inicial (0) - c_1 *e* c_2 *desligados;* estado 1 - c_1 *ligado e* c_2 *desligado;* estado 2 - c_2

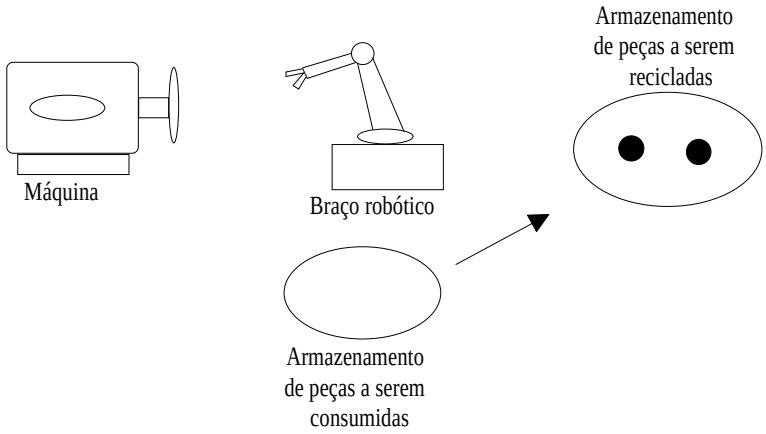

Figura 0.18: Sistema simplificado de reciclador de material com consumidor.

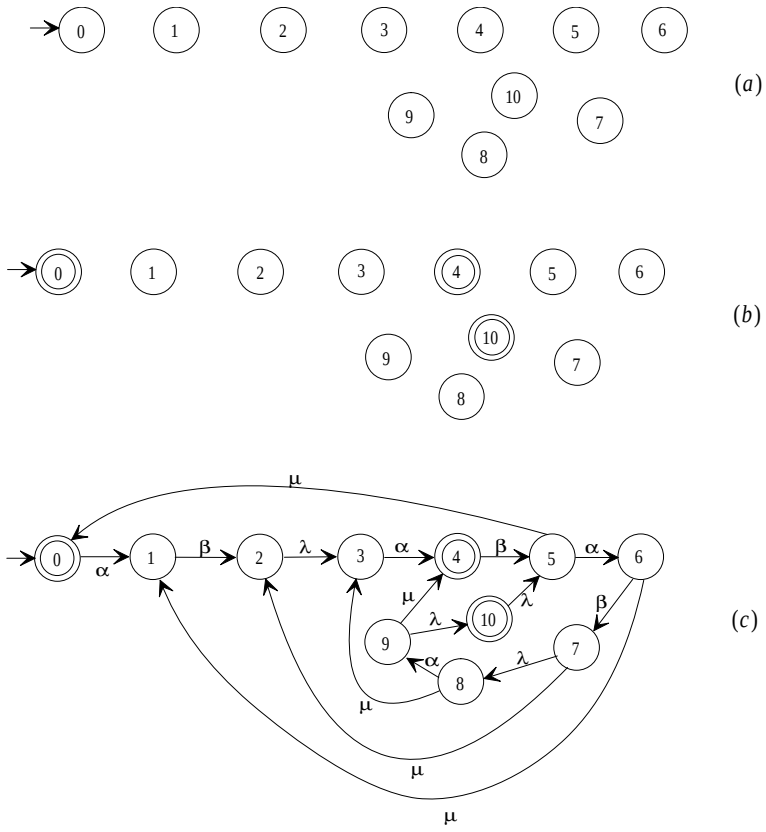

Figura 0.19: (a) Construção dos estados; (b) definição dos estados marcados e (c) modelo do SED.

ligado e c_1 desligado; estado 3 - *c_1 e c_2 ligados;* estado 4 - *c_1 envia trabalho t_1 para a fila com c_2 desligado;* estado 5 - *c_1 envia trabalho t_2 para a fila com c_2 desligado;* estado 6 - *i imprimindo t_1 de c_1 com c_2 desligado;* estado 7 - *i imprimindo t_2 de c_1 com c_2 desligado;* estado 8 - *c_2 envia trabalho t_1 para a fila com c_1 desligado;* estado 9 - *c_2 envia trabalho t_2 para a fila com c_1 desligado;* estado 10 - *i imprimindo t_1 de c_2 com c_1 desligado;* estado 11 - *i imprimindo t_2 de c_2 com c_1 desligado;* estado 12 - *c_1 envia trabalho t_1 com c_2 ligado;* estado 13 - *c_1 envia trabalho t_2 com c_2 ligado;* estado 14 - *c_2 envia trabalho t_1 com c_1 ligado;* estado 15 - *c_2 envia trabalho t_2 com c_1 ligado;* estado 16 - *i imprimindo t_1 de c_1 com c_2 ligado;* estado 17 - *i imprimindo t_2 de c_1 com c_2 ligado;* estado 18 - *i imprimindo t_1 de c_2 com c_1 ligado;* estado 19 - *i imprimindo t_2 de c_2 com c_1 ligado;* estado 20 - *c_2 envia trabalho t_2 com fila contendo trabalho t_1 de c_1;* estado 21 - *i imprimindo t_1 de c_1, fila com t_2 de c_2;* estado 22 - *c_1 envia trabalho t_1 com fila contendo trabalho t_2 de c_2;* estado 23 - *i imprimindo t_1 de c_2, fila com t_2 de c_1. Definindo como estados marcados: computadores desligados e computadores ligados (só ou em conjunto), apenas os estados 0, 1, 2, 3 e 4 são marcados, pois sempre que qualquer tarefa de impressão é executada, eles voltam a condição de computadores ligados e fila sem trabalhos, ou desligam-se os computadores. Definindo as funções de transição, tem-se: α - liga c_1; β - liga c_2; θ_1 - desliga c_1; θ_2 - desliga c_2; λ_1 - i imprime trabalho de c_1; λ_2 - i imprime trabalho de c_2; μ_1 - c_1 envia um trabalho para a impressão; μ_2 - c_2 envia um trabalho para a impressão; ν_1 - i termina impressão de trabalho de c_1 e ν_2 - i termina impressão de trabalho de c_2. Assim, constrói-se o modelo como mostrado seqüencialmente nas Figuras 0.21(a), 0.21(b) e 0.21(c), onde esta última é o modelo do SED.*

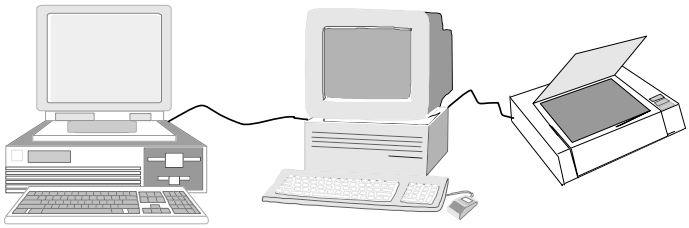

Figura 0.20: Simples rede de computadores com uma impressora.

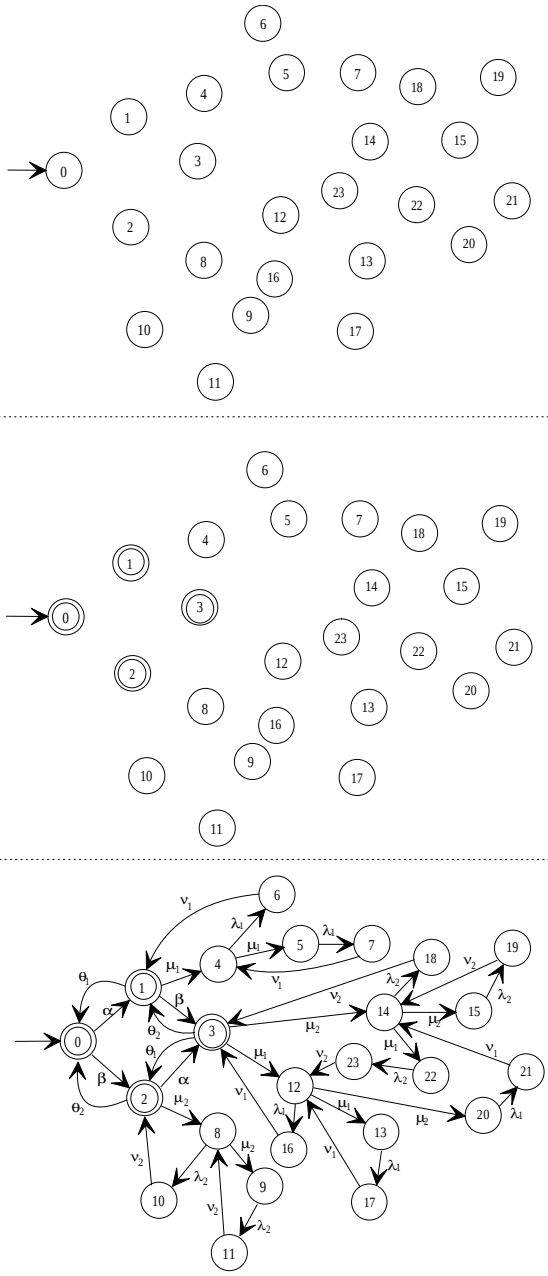

Figura 0.21: (a) Construção dos estados; (b) definição dos estados marcados e (c) modelo do SED.

A teoria de controle supervisório

A Teoria de Controle Supervisório (TCS) utiliza como paradigma de modelagem dos SEDs as linguagens formais e os autômatos. A idéia dessa teoria é de modelar o SED por um gerador e sua linguagem e, a partir daí, especificar um comportamento através de uma linguagem formal, avaliando as possibilidades desta fundamentar um outro autômato (denominado supervisor) que possa realizar esta tarefa requerida, quando acoplado ao gerador que modela o SED. Esta teoria também é conhecida como modelo ou abordagem **R-W** (Ramadge e Wonham [1, 2, 52, 3]).

Os geradores no modelo R-W

Como visto anteriormente, os geradores representam um SED desde que sua linguagem gerada L e sua linguagem marcada L_m sejam tais que, $L(G) = L$ e $L_m(G) = L_m$. Contudo, dada uma linguagem $L \subseteq \Sigma^*$, nem sempre existe um gerador finito tal que $L(G) = L$. Uma das causas disto decorre da teoria de linguagens formais, que define que a classe de linguagens representáveis por autômatos determinísticos finitos é exatamente a classe de linguagens regulares.

Como os resultados estabelecidos na abordagem **R-W** são formulados em termos da teoria de linguagens formais, o modelo **R-W** não é limitado à classe dos SEDs representáveis por geradores finitos, embora seus resultados úteis, geralmente o sejam. Devido a isto, na maioria dos casos, tratam-se de sistemas representáveis por geradores finitos.

Exemplo 24 *A Figura 0.22 mostra um gerador G que modela uma máquina com três estados: R (em repouso), A (em atividade) e M (em manutenção). Há quatro transições possíveis, identificadas pelos eventos do alfabeto $\Sigma = \{\alpha, \beta, \lambda, \mu\}$. O modelo permite concluir que a máquina parte do estado de repouso, de onde pode passar ao estado ativo (α) e deste voltar ao repouso (β) ou então sofrer uma pane (λ) e entrar em manutenção, de onde voltará eventualmente à condição de repouso (μ). A linguagem gerada, $L(G)$, é o conjunto de todas as palavras obtidas partindo-se do estado inicial R e seguindo o grafo. A expressão regular que representa esta linguagem pode ser escrita*

como:

$$L\left(G\right) = \left(\alpha\beta + \alpha\lambda\mu\right)^{*}\left(\epsilon + \alpha + \alpha\lambda\right).$$

Note que o único estado marcado de G é o estado inicial. Isto significa que a linguagem marcada $L_m(G)$ compreende as palavras que representam um ciclo completo no grafo, seja pelo caminho $\alpha\beta$ ou pelo caminho $\alpha\lambda\mu$:

$$L_m\left(G\right) = \left(\alpha\beta + \alpha\lambda\mu\right)^{*}.$$

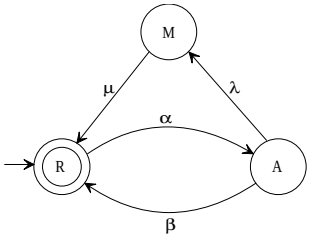

Figura 0.22: Gerador representando uma máquina com três estados.

Assim, vê-se que $L\left(G\right)$ é interpretada como uma representação do comportamento fisicamente possível do sistema e $L_m\left(G\right)$ como uma representação do conjunto de tarefas que o mesmo é capaz de executar. Logo, isto nos permite utilizar a coacessibilidade de um gerador como critério para a ausência de bloqueio, ou seja, num gerador coacessível

$$L\left(G\right) = \overline{L_m\left(G\right)},$$

significando que toda palavra gerada é prefixo de alguma palavra marcada. Então, toda seqüência de eventos fisicamente possível tem pelo menos uma continuação que leva a uma tarefa completa. Assim, nesta interpretação, tem-se que um gerador coacessível e o sistema por ele representado são ditos *não bloqueantes*.

Geradores com Entradas de Controle

Na TCS, é considerado que em um dado gerador $G = (Q, \Sigma, \delta, q_0, Q_m)$, o alfabeto gerado Σ é dividido em dois subconjuntos: Σ_c e Σ_{uc}. O subconjunto de eventos $\Sigma_c \subseteq \Sigma$, cujos elementos são denominados eventos controláveis, são os eventos que podem ser inibidos ou impedidos de ocorrer através de um agente externo. Os eventos que

formam o conjunto Σ_{uc} são denominados eventos não controláveis, os quais estão sempre habilitados, não podendo sofrer ação de controle.

As relações válidas para estes conjuntos são:

$$\Sigma = \Sigma_{uc} \cup \Sigma_c \qquad \text{e}$$
$$\Sigma_{uc} \cap \Sigma_c = \varnothing.$$

Com esta partição do alfabeto de eventos, podem-se descrever características de sistemas físicos. Assim, como visto anteriormente, o início da operação de uma máquina e o envio de uma mensagem num sistema de comunicação são eventos controláveis, enquanto que uma pane, a perda de uma mensagem, ou o término de operação de uma máquina são eventos não controláveis.

Para que seja possível interferir no funcionamento do gerador, este precisa ser dotado de uma interface, através da qual se possa informar quais eventos devem ser habilitados e quais devem ser inibidos. Dessa forma, denomina-se *entrada de controle* o conjunto de eventos habilitados em um determinado estado. Logo, como uma especificação não deve tentar inibir eventos não controláveis, uma entrada de controle só é válida se o gerador contiver o conjunto de eventos não controláveis. Formalmente, o conjunto de entradas de controle é definido como:

Definição 16 *Dado um gerador* $G = (\Sigma, Q, \delta, q_0, Q_m)$*, cujo alfabeto é particionado em* $\Sigma = \Sigma_c \cup \Sigma_{uc}$*, o conjunto de entradas de controle associado a* G *é dado por:*

$$\Gamma_c = \{\gamma | \Sigma_{uc} \subseteq \gamma \subseteq \Sigma\}. \tag{0.6}$$

As entradas de controle definem quais os eventos que devem estar habilitados num determinado estado. Assim, em um estado q do gerador G, onde há eventos controláveis e eventos não controláveis habilitados, uma entrada de controle γ aplicada nesse estado define quais os únicos eventos que podem ocorrer. Como os eventos não controláveis não sofrem ação de controle, todos os eventos não controláveis definidos neste estado estão, por definição, sempre habilitados. Apenas os eventos controláveis definidos em γ podem ocorrer (os demais eventos são inibidos).

Dado o conjunto de entradas de controle Γ_c associado a um gerador G, este passa a ser visto como se suas funções de transição deixassem de ser definidas para os eventos inibidos pela entrada de controle aplicada em um determinado estado. Com isto pode-se definir um gerador controlado.

Definição 17 *Dado $\Gamma_c \subseteq 2^{\Sigma}$ como sendo o conjunto de entradas de controle, define-se um gerador controlado G_c como um par $\langle G, \Gamma_c \rangle$ onde G é um gerador com alfabeto Σ, particionado em eventos controláveis Σ_c e eventos não controláveis Σ_{uc}, equipado com um conjunto de entradas de controle Γ_c.*

Denomina-se *planta*, o modelo do sistema a ser controlado, semelhantemente à teoria clássica de controle. O comportamento do sistema na ausência de qualquer ação de controle é definida como *linguagem da planta*, a qual representa o comportamento do sistema.

Como dito anteriormente, desde que seja definida uma entrada de controle γ a uma planta, esta irá se comportar como se os eventos inibidos fossem eliminados de sua estrutura de transição. Dessa forma, em um gerador cujo alfabeto é particionado em eventos controláveis e não controláveis, a função de transição deste gerador não está definida para os eventos inibidos por uma dada entrada de controle que seja aplicada ao gerador em um determinado instante. Logo, como este é o mecanismo de controle adotado pela TCS, necessita-se, então, chavear as entradas de controle para especificar, a partir da linguagem gerada, determinadas tarefas a serem realizadas. Então, usando a entrada de controle, a linguagem do gerador pode ser modificada.

Exemplo 25 *No gerador apresentado na Figura 0.23(a), a linguagem é*

$$L(G) = ((\alpha\beta + \eta)\nu + \epsilon)^* .$$

Considerando que α e η são eventos não controláveis e β e ν são eventos controláveis, mantendo a permanente desabilitação de β, a linguagem torna-se

$$L(G) = \alpha + (\eta\nu + \epsilon)^* ,$$

que é o gerador visto na Figura 0.23(b).

Este mecanismo de controle é quem define um chaveamento nas entradas de controle, determinando que a seqüência especificada que define a tarefa requerida, se possível, seja seguida. Com este mecanismo apresentado, torna-se necessário encontrar as condições necessárias e suficientes para a existência de um controlador que faça o chaveamente da planta de modo a satisfazer uma especificação de comportamento definida, bem como o procedimento de síntese para obter tal controlador. Este controlador para uma planta é denominado supervisor.

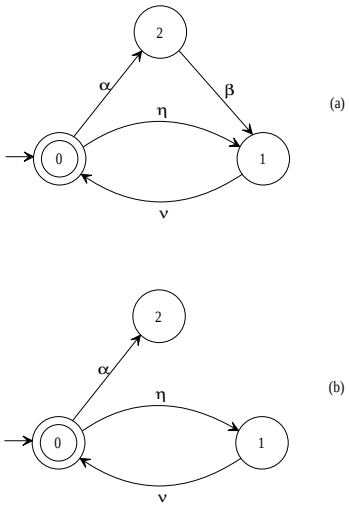

Figura 0.23: Gerador com todos os eventos habilitados e gerador com o evento β inibido.

Supervisores e condições de existência

Com a fundamentação teórica dos geradores controlados, deve-se determinar como chavear a entrada de controle em resposta à cadeia de eventos previamente gerada pelo sistema. Este chaveamento é feito pelo supervisor, que é o agente externo que determina a ação de controle a ser aplicada ao sistema.

Um supervisor é definido formalmente como:

Definição 18 *Um supervisor para um gerador controlado* $G_c = \langle G, \Gamma_c \rangle$ *é um par* $\mathbf{S} = \langle S, \Theta \rangle$*, composto de um gerador* $S = (\Sigma, X, \xi, x_0, X_m)$ *e de um mapa de controle* Θ*, em que:*

- Σ *é o mesmo alfabeto de* G*;*
- X *é um conjunto de estados;*
- $\xi : \Sigma^* \times X \rightarrow X$ *é uma função de transição parcial estendida;*
- $x_0 \in X$ *é o estado inicial;*
- $X_m \subseteq X$ *é o conjunto de estados marcados;*

- $\Theta : X \rightarrow \Gamma_c$ *é uma função que associa a cada* $x \in X$ *uma entrada de controle* $\gamma \in \Gamma_c$.

Na Figura 0.24, está representado um sistema composto pelo gerador controlado G_c supervisionado pelo supervisor $\mathbf{S}$, em malha fechada. Nesta Figura pode-se ver que o gerador controlado recebe a ação de controle do supervisor em resposta aos eventos gerados pela planta modelada por G. Desta forma, o gerador segue a especificação dada pelo supervisor.

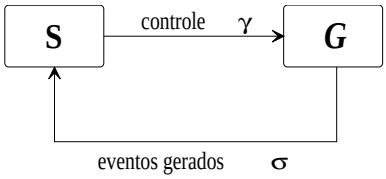

Figura 0.24: Supervisão de um SED.

De acordo com a Definição de supervisor, tem-se que a ação de controle modifica a linguagem associada ao gerador, pois, como dito na seção anterior, a mesma pode inibir seqüências de eventos que antes podiam ocorrer. Logo, sua linguagem, pode ser assim definida:

Definição 19 *Dados um gerador controlado* G_c *e um supervisor* $\mathbf{S}$*, a linguagem gerada pelo sistema supervisionado, denotada por* $L(\mathbf{S}/G)$*, é tal que*

$$\begin{array}{l} \epsilon \in L(\mathbf{S}/G) \qquad e \\ s\sigma \in L(\mathbf{S}/G) \text{ se e somente se } s \in L(\mathbf{S}/G) \wedge s\sigma \in L(G) \wedge \sigma \in \Theta(\xi(s, x_0)), \end{array} \tag{0.7}$$

onde Θ *é o mapa de controle que associa a cada estado* $x \in X$ *uma entrada de controle* $\gamma \in \Gamma_c$*, a ser aplicada a* G.

A composição síncrona entre o supervisor e o gerador define a linguagem do sistema supervisionado. Esta linguagem representa a trajetória que o sistema deve seguir. Deve-se observar que os eventos habilitados em cada estado do supervisor são os únicos eventos que podem ocorrer no estado correspondente do gerador. Os outros eventos possíveis de ocorrerem no respectivo estado do gerador são inibidos, o que é determinado pela entrada de controle do supervisor. Esta condição pode ser vista na composição

síncrona do supervisor com o gerador, observando os pares $q_{\mathbf{S}/G} = (q_{\mathbf{S}}, q_G)$, onde $q_{\mathbf{S}/G}$ é o estado q da composição síncrona $\mathbf{S}/G$, $q_{\mathbf{S}}$ é o estado q do supervisor $\mathbf{S}$ e q_G é o estado q do gerador G. Isto é visto na Definição 19, em que somente os eventos que estão habilitados em cada estado, são os que ocorrem sob supervisão.

Deve-se observar que, dada uma palavra s que pertençe a $L(\mathbf{S}/G)$, também pertence a $L\,(G)$, o que determina que a linguagem do sistema supervisionado satisfaz

$$L(\mathbf{S}/G) \subseteq L\,(G)\,. \tag{0.8}$$

Então, vê-se que $L(\mathbf{S}/G)$ é uma linguagem prefixo-fechada, ou seja,

$$L(\mathbf{S}/G) = \overline{L(\mathbf{S}/G)} \tag{0.9}$$

visto que uma palavra $s\sigma \in L(\mathbf{S}/G)$ somente se $s \in L(\mathbf{S}/G)$.

Define-se a linguagem controlada do sistema supervisionado como a seguir:

Definição 20 *Dados um gerador controlado G_c e um supervisor $\mathbf{S}$, a linguagem controlada do sistema sob supervisão, denotada por $L_c(\mathbf{S}/G)$, é definida como:*

$$L_c(\mathbf{S}/G) = L(\mathbf{S}/G) \cap L_m(G). \tag{0.10}$$

A linguagem controlada, $L_c(\mathbf{S}/G)$, é a parte da linguagem marcada original sob ação de controle que representa as tarefas que são completadas sob supervisão. Em outros termos, a linguagem controlada é simplesmente a parte da linguagem original que "sobrevive" sob a ação de controle.

Assim, é determinado que, se $L_m\,(G)$ representa as tarefas que podem ser completadas pela planta, então $L_c\,(\mathbf{S}/G)$ representa as tarefas que podem ser completadas sob supervisão. Logo, isto implica nas seguintes condições:

$$L_c(\mathbf{S}/G) \subseteq L(\mathbf{S}/G) \tag{0.11}$$

e

$$L_c(\mathbf{S}/G) \subseteq L_m(G) \tag{0.12}$$

Disto decorre que a linguagem marcada do sistema sob supervisão é

$$L_m(\mathbf{S}/G) = L_c(\mathbf{S}/G) \cap L_m(\mathbf{S}) \tag{0.13}$$

Assim, conclui-se que

$$L_m(\mathbf{S}/G) \subseteq L_c(\mathbf{S}/G) \subseteq L(\mathbf{S}/G) \subseteq L(G), \tag{0.14}$$

ou seja, a linguagem $L(\mathbf{S}/G)$, que é gerada pelo sistema composto pelo supervisor e pelo gerador controlado, $\mathbf{S}/G_c$, pode ser interpretada como o conjunto de todas as possíveis seqüências finitas de eventos que têm possibilidade de ocorrer no sistema.

Supervisores Próprios

É preciso garantir que os eventos no supervisor $\mathbf{S}$ só devam ocorrer, quando eles também ocorrerem em G_c e estiverem habilitados por Θ.

Definição 21 *Um supervisor* $\mathbf{S}$ *é dito ser* completo*, em relação a um gerador* G_c*, quando o seguinte é verdadeiro: para todo* $s \in \Sigma^*$ *e* $\sigma \in \Sigma$ *as três condições*

$$\begin{aligned} &s \in L(\mathbf{S}/G), \\ &s\sigma \in L(G) \qquad e \\ &\sigma \in \Theta\left(\xi\left(s, x_0\right)\right), \end{aligned} \tag{0.15}$$

juntas implicam em

$$\xi\left(s, x_0\right)!, \ \textit{isto é,}\ \xi\left(s, x_0\right) \ \textit{é definido.} \tag{0.16}$$

Esta é a condição necessária para considerar o supervisor $\mathbf{S}$ como completo em relação a um gerador controlado G_c. Logo, se s é uma palavra que pode ocorrer no sistema supervisionado e o evento σ é uma continuação fisicamente possível desta palavra, se σ está habilitado, então a palavra $s\sigma$ deve estar definida na função de transição do supervisor.

Torna-se necessário estabelecer duas restrições a serem satisfeitas pelas linguagens $L(\mathbf{S}/G)$, $L_c(\mathbf{S}/G)$ e $L_m(\mathbf{S}/G)$, para controlar um SED de maneira satisfatória. Estas restrições são definidas a seguir:

Definição 22 *Um supervisor* $\mathbf{S}$ *é dito não bloqueável se e somente se*

$$\overline{L_c(\mathbf{S}/G)} = L(\mathbf{S}/G) \tag{0.17}$$

Definição 23 *Um supervisor* $\mathbf{S}$ *é dito não rejeitável se e somente se*

$$\overline{L_m(\mathbf{S}/G)} = \overline{L_c(\mathbf{S}/G)}. \tag{0.18}$$

Destas duas definições, pode-se ver que um supervisor é bloqueável se existir pelo menos uma palavra fisicamente possível em $L(\mathbf{S}/G)$ que não é prefixo de qualquer palavra em $L_c(\mathbf{S}/G)$, e portanto, através desta palavra o sistema nunca pode completar uma tarefa especificada. Por outro lado, um supervisor é rejeitável caso exista pelo menos uma palavra em $\overline{L_c(\mathbf{S}/G)}$ representando uma tarefa completada, contudo, que não pertence à linguagem marcada $L_m(\mathbf{S}/G)$. Neste caso, é possível atingir um estado em que nenhuma tarefa seja reconhecida, isto é, que não pertence à especificação.

Estas situações indesejáveis são ilustradas no Exemplo a seguir:

Exemplo 26 *Seja o gerador* $G = (Q, \Sigma, \delta, q_0, Q_m)$*, com* $\Sigma = \{\alpha, \beta, \lambda\}$ *mostrado na Figura 0.25*(a)*, cuja linguagem marcada é*

$$L_m(G) = (\alpha\lambda^*\beta)^*$$

com $\Sigma_c = \{\beta\}$*. Considere primeiramente o supervisor da Figura 0.25*(b)*. Quando acoplado através do produto de autômatos ao gerador da Figura 0.25*(a)*, o comportamento do sistema fica restrito ao representado pelo gerador da Figura 0.25*(c)*. Vê-se que*

$$L(\mathbf{S}/G) = (\alpha\beta)^*(\epsilon + \alpha\lambda^*)$$

e que

$$L_c(\mathbf{S}/G) = (\alpha\beta)^*,$$

de modo que a condição de ausência de bloqueio não é satisfeita. Disto decorre que, nenhuma seqüência incluindo o evento λ *leva a uma tarefa completada sob supervisão, pois o evento* β *é permanentemente inibido logo após a primeira ocorrência de* λ*. Por outro lado, considerando o supervisor da Figura 0.25*(d) *acoplado através do produto de autômatos ao gerador da Figura 0.25*(a)*, cujo comportamento resultante é visto na Figura 0.25*(e)*, tem-se que*

$$L_c(\mathbf{S}/G) = L_m(G) = (\alpha\lambda^*\beta)^*.$$

Isto significa que todas as tarefas fisicamente possíveis podem ser completadas sob supervisão. Entretanto,

$$L_m\left(\mathbf{S}/G\right) = \left(\alpha\beta\right)^*,$$

que viola a condição de ausência de rejeição. Logo, a partir da primeira ocorrência do evento λ, todas as tarefas completadas deixam de ser marcadas.

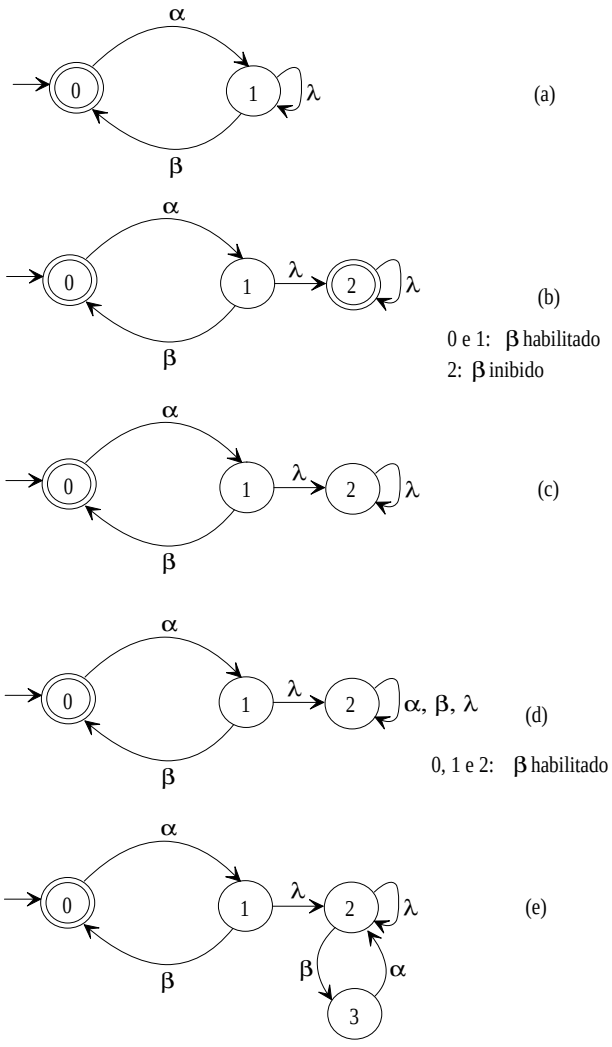

Figura 0.25: Gerador (a) e supervisores com bloqueio (b) e rejeição (d) e comportamentos resultantes (c) e (e), respectivamente.

Quando um supervisor é não bloqueável e não rejeitável, diz-se que ele é um supervisor completo.

Para a síntese do supervisor ser completa, determina-se que ele não deva apresentar rejeição nem bloqueio, o que leva a definição de supervisor próprio, que é a seguinte:

Definição 24 *Um supervisor completo, isto é, não bloqueável e não rejeitável é dito supervisor próprio se*

$$\overline{L_m(\mathbf{S}/G)} = \overline{L_c(\mathbf{S}/G)} = L(\mathbf{S}/G). \tag{0.19}$$

Tendo isto em vista, torna-se necessário definir o problema de controle supervisório, e determinar sua solução. Isto é visto a seguir.

Formulação e resolução do problema de controle supervisório

O problema principal da TCS, está definido em determinar mudanças no comportamento de um SED. As linguagens apresentadas anteriormente permitem a formulação de problemas abstratos de síntese de supervisores. De um modo geral, um problema desse tipo supõe que se represente o comportamento fisicamente possível do sistema e o comportamento desejado sob supervisão por linguagens, sendo o objetivo construir um supervisor para a planta tal que o comportamento do sistema em malha fechada se limite ao comportamento desejado. Para tanto, definem-se os seguintes conceitos:

Definição 25 *Sejam duas linguagens $K, L \subseteq \Sigma^*$. K é dita fechada em relação a L, ou* $L-$fechada *se e somente se*

$$K = \overline{K} \cap L \tag{0.20}$$

Definição 26 *Sejam duas linguagens $K, L \subseteq \Sigma^*$ e um alfabeto $\Sigma = \Sigma_c \cup \Sigma_u$. K é dita* $L-$controlável *se e somente se*

$$\overline{K}\Sigma_{uc} \cap L \subseteq \overline{K}. \tag{0.21}$$

Estas Definições são os conceitos conhecidos por *fechamento* e *controlabilidade*, respectivamente, onde a linguagem K é definida como a linguagem da especificação de comportamento, ou o que se deseja que o SED realize, e a linguagem L é a linguagem gerada pelo SED.

Torna-se necessário determinar as condições de existência do supervisor para a realização de uma tarefa específica. É dada, então, a seguinte proposição:

Proposição 1 *Seja* $\mathbf{S}$ *um supervisor completo para* G_c*. Então* $L(\mathbf{S}/G)$ *é prefixo-fechada e* $L(G)-$*controlável.*

Também é necessário estabelecer as condições de existência de supervisores para os problemas formulados em termos de linguagens geradas e marcadas. Para isto, apresentam-se os teoremas a seguir:

Teorema 1 *Dados um gerador* G *tal que* $L(G)$ *represente seu comportamento fisicamente possível e uma linguagem especificada* $K \subseteq L(G)$*, existe um supervisor completo* $\mathbf{S}$ *tal que* $L(\mathbf{S}/G) = K$ *se e somente se* K *for prefixo-fechada e* $L(G)$*-controlável.*

Teorema 2 *Dados um gerador* G *tal que* $L_m(G)$ *represente as tarefas que podem ser completadas pelo sistema na ausência de qualquer ação de controle e uma linguagem especificada* $K \subseteq L_m(G)$ *então*

1. *Existe um supervisor completo* $\mathbf{S}$ *tal que* $L_c(\mathbf{S}/G) = K$ *se e somente se* K *for* $L_m(G)$-fechada *e existir uma linguagem prefixo-fechada e* $L(G)$-controlável K' *tal que* $K' \cap L_m(G) = K$.

2. *Existe um supervisor não bloqueável* $\mathbf{S}$ *tal que* $L_c(\mathbf{S}/G) = K$ *se e somente se* K *for* $L(G)$-fechada *e* $L(G)$-controlável*;*

3. *O supervisor* $\mathbf{S}$ *será próprio somente se o gerador* $S = (\Sigma, X, \xi, x_0, X_m)$ *for tal que* $X_m = X$.

Para problemas formulados em termos de linguagens marcadas, se K satifizer as condições do Teorema 2, então o supervisor será tal que $L(\mathbf{S}/G) = \overline{K}$, visto que nesse caso,

$$L_c(\mathbf{S}/G) = \overline{K} \cap L_m(G) = K. \tag{0.22}$$

Estes resultados somente podem ser empregados quando a linguagem especificada K satisfaz as condições exigidas. Quando a linguagem K não satisfaz as condições exigidas à existência do supervisor, ou seja, a linguagem especificada K não é $L_m(G)$-*fechada*, nem $L(G)$-*controlável*, é necessário encontrar uma sublinguagem $K^{\uparrow} \subseteq K$, a qual é sempre possível e satisfaz todas as condições restritivamente. Esta sublinguagem é conhecida por *Suprema Sublinguagem Controlável* $K^{\uparrow}$, ou $\sup C(L)$, que soluciona o

problema do supervisor, contudo restringe seus resultados ao mínimo tolerável. Dessa maneira, a $\sup C(L)$ evita problemas na execução de uma tarefa especificada, eliminando as possibilidades de ocorrências de bloqueios ou de execução de outras tarefas que não estejam especificadas.

Sendo assim, se $K^{\uparrow}$ solucionar de forma satisfatória o problema dado, isto é, se $K \supseteq A$, onde A é uma linguagem que representa o comportamento mais restrito que pode ser tolerado, $K^{\uparrow}$ pode ser utilizada em substituição à linguagem anteriormente especificada. A solução para este problema é um supervisor que implementa $K^{\uparrow}$. Logo, para o caso dos geradores de estado finitos, $K^{\uparrow}$ é sempre computável [2].

Considerando $K \subseteq \Sigma^*$ uma linguagem especificada, onde Σ^* representa o conjunto de todas as linguagens definidas a partir do alfabeto Σ e sendo $C(K)$ a família das linguagens controláveis de K, então $C(K)$ é sempre não vazia pois a linguagem vazia é controlável.

Um importante resultado em relação à controlabilidade das linguagens é que a família $C(K)$ é fechada em relação à união de linguagens, ou seja, existe uma única sublinguagem controlável máxima $K^{\uparrow}$ tal que $K^{\uparrow} \subseteq K$. Assim, nota-se que $K^{\uparrow}$ pode ser a linguagem vazia.

Este problema pode ser enunciado da seguinte maneira: *Dados um gerador* G, *uma linguagem-alvo marcada* $E \subseteq \Sigma^*$ *e uma linguagem mínima admissível* $L_A \subseteq E$, *encontrar um supervisor próprio* $\mathbf{S}$ *tal que*

$$L_A \subseteq L_c(\mathbf{S}/G) \subseteq E. \tag{0.23}$$

Quando E é $L_m(G)$-*fechada* e $L(G)$-*controlável*, a existência de um supervisor tal que $L_c(S/G) = E$ é garantida, o que significa que o problema tem uma solução não restritiva, como visto anteriormente. Nos casos em que E não satisfaz estas condições, é possível obter uma solução minimamente restritiva, como dado pelo problema descrito anteriormente na equação (0.23).

Quando todos os estados do gerador são marcados, garante-se que qualquer supervisor obtido é não bloqueável.

O algoritmo a seguir soluciona o problema de controle supervisório, para o caso em que um gerador de estado finito G é descrito por L (comportamento em malha aberta) e K (comportamento desejado):

Algoritmo 1 Algoritmo para a Construção de $K^{\uparrow}$ *([2])*

- *Dados o gerador G trim e o gerador H (composição síncrona do gerador G com a especificação desejada), faça:*

1. *Construir a matriz de transições* $\mathbf{A}$ *do gerador H, onde*

$$\mathbf{A} = [a_{i,j}], a_{i,j} = \begin{cases} \sigma, & \text{se } \exists\, \sigma \text{ do estado } i \text{ para o estado } j; \\ - & \text{caso contrário;} \end{cases}$$

2. *Incluir ao lado direito da matriz de transições* $\mathbf{A}$ *o vetor coluna que representa $\Sigma\left(H\left(x\right)\right) \cap \Sigma_{uc}$, em que $\Sigma\left(H\left(x\right)\right)$ representa os eventos habilitados no estado q do gerador G correspondente no estado x da composição síncrona;*

3. *Inclua ao lado direito da tabela o vetor coluna $\Sigma\left(x\right)$, representando os eventos habilitados no estado x do gerador H;*

4. *Para cada estado x_i, em* $\mathbf{A}$ *que não satisfaz $\Sigma\left(H\left(x_i\right)\right) \cap \Sigma_{uc} \subset \Sigma\left(x_i\right)$, remover a linha da tabela e a coluna da matriz* $\mathbf{A}$ *referente ao estado x_i;*

5. *Encontrar a componente coacessível do gerador resultante;*

6. *Itere este processo até que todos os estados x_i restantes satisfaçam*

$$\Sigma\left(H\left(x_i\right)\right) \cap \Sigma_{uc} \subset \Sigma\left(x_i\right).$$

Exemplo 27 *Sejam $\Sigma = \{\alpha_1, \alpha_2, \beta\}$, $\Sigma_{uc} = \{\beta\}$, e em notação de expressões regulares*

$$L\left(G\right) = \overline{\left(\alpha_1\beta^2 + \alpha_2\right)\beta^*},$$
$$L\left(H\right) = \overline{\alpha_1\beta^2} + \alpha_2\beta^*,$$

que são a linguagem do gerador trim visto na Figura 0.26 e a especificação de comportamento H desejada para este gerador. Utilizando o algoritmo da $\sup C\left(L\right)$ apresentado por Ramadge e Wonham [2], inclui-se separadamente à matriz de transição do gerador da especificação de comportamento, visto na Figura 0.27, duas colunas: uma listando

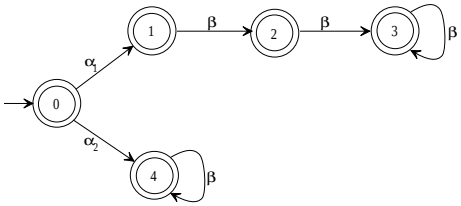

Figura 0.26: Autômato gerador de $L(G) = \overline{\left(\alpha_1\beta^2 + \alpha_2\right)\beta^*}$.

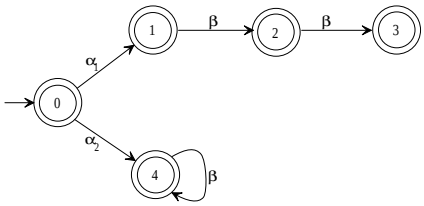

Figura 0.27: Especificação $L(H) = \overline{\alpha_1\beta^2} + \alpha_2\beta^*$.

$\Sigma(H(x)) \cap \Sigma_{uc}$, *outra listando* $\Sigma(x)$. *Isto define a seguinte tabela:*

H_0 :

	1 2 3 4 5	$\Sigma(H(x)) \cap \Sigma_{uc}$	$\Sigma(x)$
1	α_1 $\quad$ α_2		$\alpha_1\alpha_2$
2	β	β	β
3	β	β	β
4		β	
5	β	β	β

Desde que $\Sigma(H(4)) \cap \Sigma_{uc} \not\subset \Sigma(4)$, *remove-se o estado* 4 *da tabela e encontra-se que o gerador resultante é trim. O resultado é um gerador para a linguagem* $L(H_1)$, *apresentado na tabela a seguir:*

H_1 :

	1 2 3 5	$\Sigma(H(x)) \cap \Sigma_{uc}$	$\Sigma(x)$
1	α_1 $\quad$ α_2		$\alpha_1\alpha_2$
2	β	β	β
3		β	
5	β	β	β

onde sua linguagem é $L(H_1) = \overline{\alpha_1\beta} + \alpha_2\beta^*$. *Iterando esse procedimento é produzida a*

seguinte seqüência de tabelas:

H_2 :

	1 2 5	$\Sigma(H(x)) \cap \Sigma_{uc}$	$\Sigma(x)$
1	α_1 α_2		$\alpha_1\alpha_2$
2		β	
5	β	β	β

$L(H_2) = \overline{\alpha_1} + \alpha_2\beta^*$

H_3 :

	1 5	$\Sigma(H(x)) \cap \Sigma_{uc}$	$\Sigma(x)$
1	α_2		$\alpha_1\alpha_2$
5	β	β	β

$L(H_3) = \alpha_2\beta^*$

em que, $L(H_3)$ *é a* $\sup C(L)$*, cujo supervisor está apresentado na Figura 0.28.*

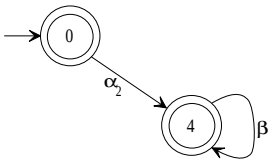

Figura 0.28: Supervisor para a $\sup C(L) = L(H_3) = \alpha_2\beta^*$.

Exemplo 28 *Considere o autômato apresentado na Figura 0.29 e a especificação de comportamento apresentada na Figura 0.30. A linguagem marcada do autômato é*

$$L_m(G) = (\alpha\beta)^* + (\alpha\kappa\eta)^*$$

e a linguagem marcada da especificação de comportamento é

$$L(H) = (\alpha + \kappa)^* \beta (\alpha + \kappa)^* \eta.$$

Pode-se ver que $L(H) \not\subset L_m(G)$*, pois* α^* *não existe em* $L_m(G)$*. Dessa forma, para calcular a* $\sup C(L)$*, constrói-se a composição síncrona* $H||G$*, a qual é vista na Figura 0.31. Com a linguagem* $L\left(H'\right) = L(H/G)$ *dessa composição, utiliza-se o algoritmo para construção da* $\sup C(L)$*, em que inclui-se separadamente à matriz de transição do gerador da especificação de comportamento e as duas colunas listando* $\Sigma(H'(x)) \cap \Sigma_{uc}$

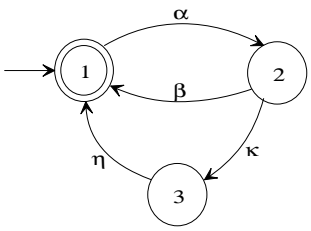

Figura 0.29: Autômato com linguagem $L_m(G) = (\alpha\beta)^* + (\alpha\kappa\eta)^*$.

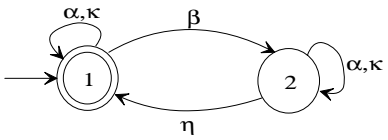

Figura 0.30: Especificação $L(H) = (\alpha + \kappa)^* \beta (\alpha + \kappa)^* \eta$.

e $\Sigma(x)$. *Isto define a seguinte tabela:*

H_0' :

	1 2 3 4 5 6	$\Sigma(H'(x)) \cap \Sigma_{uc}$	$\Sigma(x)$
1	α		α
2	κ β		$\kappa\beta$
3		η	
4	α		α
5	κ		κ
6	η	η	η

Desde que $\Sigma(H'(3)) \cap \Sigma_{uc} \not\subset \Sigma(3)$, *remove-se o estado* 3 *da tabela e encontra-se que o gerador resultante é trim. O resultado é um gerador para a linguagem* $L\left(H_1'\right)$, *apresentado na tabela a seguir:*

H_1' :

	1 2 4 5 6	$\Sigma(H'(x)) \cap \Sigma_{uc}$	$\Sigma(x)$
1	α		α
2	β		β
4	α		α
5	κ		κ
6	η	η	η

onde sua linguagem é $L\left(H_1'\right) = \alpha\beta\alpha\kappa\eta$. *Como todas as outras linhas satisfazem*

$$\Sigma(H'(x)) \cap \Sigma_{uc} \subset \Sigma(x),$$

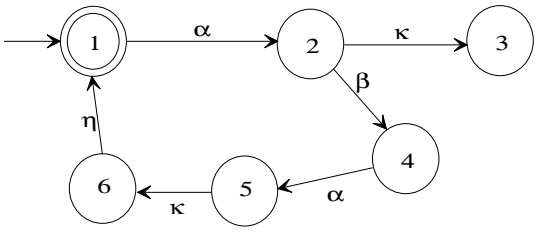

Figura 0.31: Composição síncrona $H||G$.

e H_1' *é coacessível, então,* $L\left(H_1'\right)$ *é a* $\sup C\left(L\right)$ *que é a linguagem do supervisor visto na Figura 0.32.*

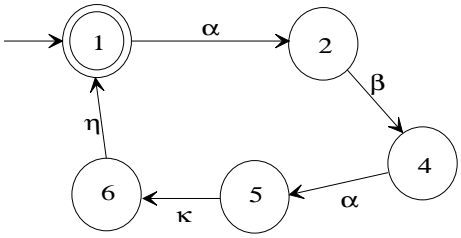

Figura 0.32: Supervisor para $\sup C\left(L\right) = L\left(H_1'\right) = \alpha\beta\alpha\kappa\eta$.

Exemplo 29 *Considere um gato e um rato dentro de um simples labirinto. Neste labirinto, em certos lugares, encontram-se passagens que são exclusivas para cada animal, dadas por* g_1 *e* g_2*, e* r_1 *e* r_2*, para o gato e para o rato, respectivamente. Este labirinto está apresentado na Figura 0.33. De acordo com as possibilidades de passagens dos animais para os lugares, constrói-se o autômato gerador correspondente, o qual é visto na Figura 0.34. Neste autômato, os estados são representados de acordo com a seguinte tabela:*

Animal\Estado	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*
Gato	q_1	q_1	q_1	q_2	q_2	q_2	q_3	q_3	q_3
Rato	q_2	q_3	q_1	q_2	q_3	q_1	q_2	q_3	q_1

que implica em dizer em quais quartos (q_i) *se encontram os animais, no respectivo estado. Nos estados 3, 4 e 8, o gato pode comer o rato. Esta é uma condição não desejada. Também, os arcos correspondentes, denotadas pelas letras* g_i *ou* r_i *representam os movimentos que os animais* (*gato e rato, respectivamente*)*, podem ter no labirinto. De certa forma, todas as passagens podem ser controladas para a passagem dos animais,*

exceto g_1, que implica em que o gato pode sempre atravessar esta passagem. Então, o comportamento desejado para este sistema é impedir que o gato se encontre com o rato, controlando a abertura das passagens no labirinto. Tendo isto em vista, tem-se que $\Sigma_{uc} = \{g_1\}$, e a especificação de comportamento é dada pelo autômato mostrado na Figura 0.35. Dessa especificação de comportamento, determina-se a seguinte tabela

Estado	*Localização* $(gato, rato)$	*evento*
1	$(1,2)$	g_1, r_2
2	$(1,3)$	r_2, g_2
6	$(2,1)$	r_1
5	$(2,3)$	r_1
7	$(3,2)$	g_1

a qual é construída através da localização possível dos animais, que satisfazem a especificação. Daí, utilizando o algoritmo da Suprema Sublinguagem Controlável, encontra-se a seguinte tabela:

	1	2	6	5	7	$\Sigma(H(x)) \cap \Sigma_{uc}$	$\Sigma(x)$
1		r_2			g_1	g_1	r_1, g_1
2	r_2			g_2		g_1	r_1, g_2
6	r_2			r_1			r_1
5		g_2	r_1				r_1, g_2
7	r_2					g_1	g_1

Como no estado 2, é violada a condição

$$\Sigma(H(x)) \cap \Sigma_{uc} \subset \Sigma(x)$$

este estado é removido, ficando, então, o autômato mostrado na Figura 0.36. Neste autômato, observa-se que os estados 5 e 6 não mais são alcançados, ficando a Suprema Sublinguagem Controlável definida por

$$\sup C(L) = g_1^*.$$

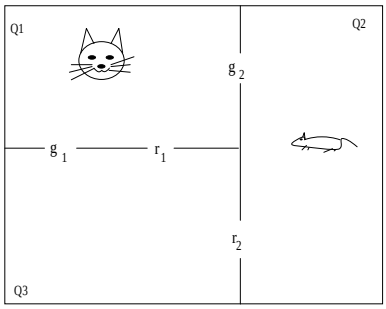

Figura 0.33: Gato e rato dentro do labirinto.

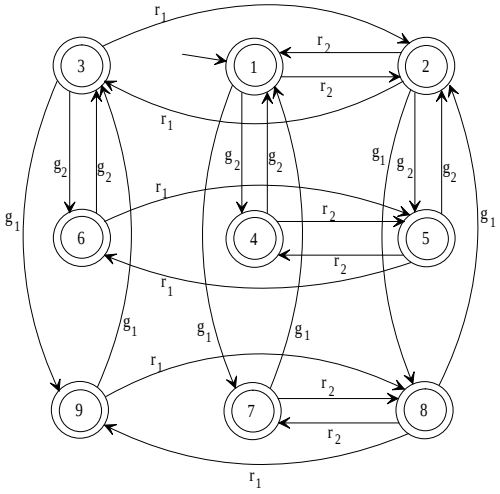

Figura 0.34: Autômato correspondente ao gato e rato dentro do labirinto.

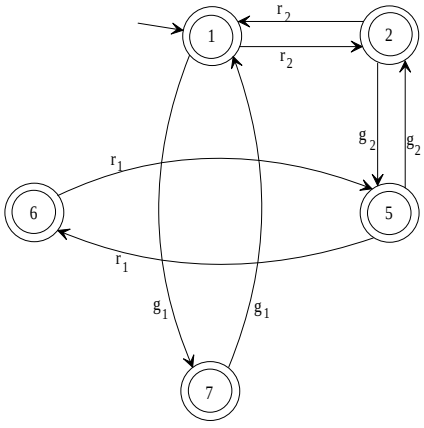

Figura 0.35: Especificação de comportamento para o sistema do labirinto para o gato e o rato.

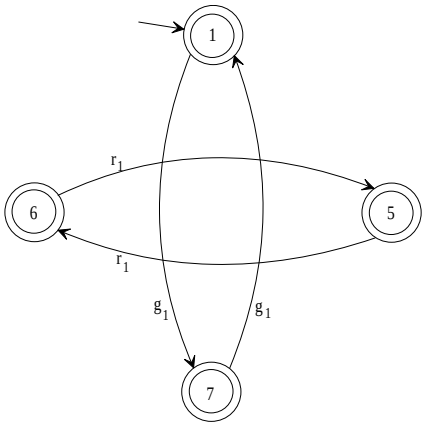

Figura 0.36: Autômato resultante da primeira iteração do Algoritmo da Suprema Sublinguagem Controlável, para o sistema do gato e o rato no labirinto.

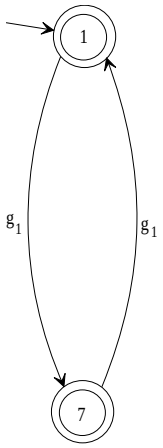

Figura 0.37: Supervisor para a $\sup C(L)$ do sistema do labirinto para o gato e o rato.

TCS e Observação Parcial de Eventos

A existência de eventos não observáveis em um sistema, exige uma avaliação das palavras que o reconhece, para determinar a igualdade entre elas. Isto é, dadas duas palavras distintas devido à presença de eventos não observáveis, estas podem realizar uma mesma tarefa. Diz-se que os sistemas que apresentam eventos não observáveis possuem observação parcial de eventos. Assim, dado um sistema que contém eventos não observáveis, com a ocorrência de um desses eventos, o sistema não percebe a mudança no ambiente. Para este caso, o problema de controle supervisório é similar ao problema anteriormente estudado. Entretanto, há a inclusão de um estágio de observação, em que o supervisor deve receber os eventos gerados pelo SED mapeados através do observador e determinar a estratégia de controle a ser implementada para que o SED siga a especificação desejada. Porém, como o modelo do SED inclui eventos não observáveis, o supervisor deve ser tal que a ação de controle a ser implementada aja no sistema prevendo as possíveis ocorrências desses eventos nos estados alcançados.

Observação de eventos

Os eventos não observáveis podem ser exemplificados por mudanças que ocorrem no interior de uma máquina. Em outras palavras, todos os eventos da dinâmica interna dos sistemas. Assim, considerando um sistema com uma máquina que perfura chapas de aço, dá-se como exemplos de eventos observáveis a chegada da chapa de aço na máquina, o início da perfuração, o término da perfuração e a retirada da chapa de aço perfurada da máquina. Os eventos não observáveis são todos os eventos que ocorrem dentro da máquina, os quais mudam seu estado interior, mas que não são percebidos exteriormente ao sistema. Isto é, polias que passam a girar ou param e peças que se encaixam são exemplos destes.

Eventos não observáveis também podem se apresentar em alguns modelos de SEDs devido à própria modelagem, bem como por serem de natureza não observáveis. Exemplo deste último caso pode ser dado por um sistema em que são enviadas peças a um buffer, onde o evento que gera o envio das peças não seja observável, mas que este evento pode ser inibido quando há um número determinado de peças no buffer.

Denota-se o conjunto dos eventos não observáveis por Σ_{uo} e o conjunto dos eventos observáveis por Σ_o. Com a introdução dessa partição no conjunto de eventos do SED,

tem-se, então:

$$\Sigma = \Sigma_o \cup \Sigma_{uo} = \Sigma_c \cup \Sigma_{uc}. \tag{0.24}$$

Assim, os eventos observáveis podem ser tanto controláveis como não controláveis. Da mesma forma, os eventos não observáveis.

A concepção dos eventos observáveis e não observáveis define que para a realização da síntese do supervisor para um SED com observação parcial de eventos haja um estágio de observação para que o supervisor receba apenas os eventos observáveis. Dessa forma, determina a ação de controle sobre a linguagem observada sem que os eventos não observáveis que ocorrem, impeçam que o comportamento especificado seja realizado. Este estágio de observação, citado anteriormente, é formalizado pelo conceito de projeção. Formalmente, define-se a projeção ou função de observação como a seguir:

Definição 27 *A projeção ou função de observação de eventos,* $\theta : \Sigma^* \to \Sigma_o^*$ *é um mapeamento tal que,*

$$\begin{array}{ll} \theta(\epsilon) = \epsilon & \\ \theta(\sigma) = \epsilon & se\ \sigma \in \Sigma - \Sigma_o = \Sigma_{uo} \\ \theta(\sigma) = \sigma & se\ \sigma \in \Sigma_o \\ \theta(s\sigma) = \theta(s)\,\theta(\sigma) & se\ s \in \Sigma^*, \sigma \in \Sigma. \end{array} \tag{0.25}$$

Logo, vê-se que o efeito da projeção θ sobre uma dada palavra s é o de apagar todos os símbolos que pertençam a Σ_{uo}.

Exemplo 30 *Sejam* $\Sigma = \{\alpha, \beta, \gamma, \eta\}$ *e* $\Sigma_{uo} = \{\beta\}$*. Então,*

$$\begin{aligned} &\theta(\alpha\beta\beta\gamma\alpha\beta\eta) = \alpha\gamma\alpha\eta, \\ &\theta(\beta\beta\alpha\beta^*) = \alpha, \\ &\theta(\beta) = \epsilon. \end{aligned}$$

Estende-se esta definição para as linguagens como:

$$\theta(L) = \{s' | s' = \theta(s) \wedge s \in L\}. \tag{0.26}$$

Exemplo 31 *Sejam* $\Sigma = \{\alpha, \beta, \lambda\}$ *e* $\Sigma_{uo} = \{\beta\}$*. Então, para um observador limitado a ver os eventos de*

$$\Sigma_o = \Sigma - \Sigma_{uo} = \{\alpha, \lambda\},$$

a planta representada pelo gerador da Figura 0.38(a) *tem o aspecto mostrado na Figura 0.38*(b)*.*

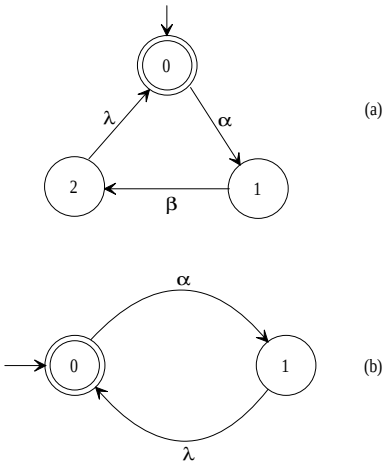

Figura 0.38: Gerador com eventos não observáveis (a) e gerador após a projeção (b).

Quando os eventos gerados pelo SED são mapeados pelo observador e enviados ao supervisor, o supervisor tem de prever a ocorrência dos eventos não observáveis para poder determinar a ação de controle e garantir que o SED realize a tarefa especificada. Esta situação é formalizada através da projeção inversa.

A projeção inversa determina que, para uma dada palavra s, sua projeção inversa irá obter o conjunto de todas as palavras que, quando vistas através da projeção, serão iguais a palavra s. Formalmente, tem-se:

Definição 28 *A projeção inversa, $\theta^{-1} : \Sigma^* \rightarrow \Sigma^*$ é um mapeamento tal que,*

$$\theta^{-1}(s) = \{s' | s' \in \Sigma^* \wedge \theta(s') = s\}, s \in \Sigma^*. \tag{0.27}$$

Exemplo 32 *Sejam $\Sigma = \{\alpha, \beta, \eta\}$, $\Sigma_{uo} = \{\alpha\}$ e $s = \beta\alpha\eta$. Então, $s' = \theta(s) = \beta\eta$ e $\theta^{-1}(s') = \{\alpha^*\beta\alpha^*\eta\alpha^*\}$.*

Igualmente, a projeção inversa pode ser estendida às linguagens $L \subseteq \Sigma^*$, como

$$\theta^{-1}(L) = \{s' | s' \in \Sigma^* \wedge \theta(s') \in L\}. \tag{0.28}$$

Exemplo 33 *Na Figura 0.39, é visto como é feita a projeção inversa graficamente. Dado um gerador G, obtém-se $\theta^{-1}(L(G))$ através da adição de auto-laços de todos os eventos de Σ_{uo} em todos os estados de G.*

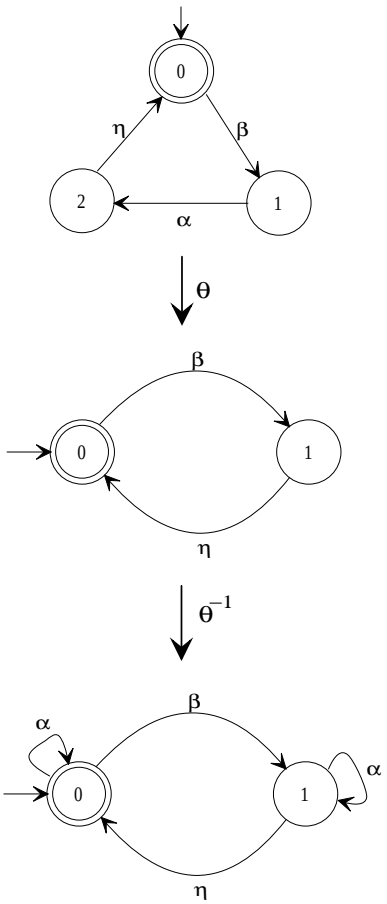

Figura 0.39: Exemplo de projeção inversa em um gerador.

Disto conclui-se que, para uma linguagem qualquer L,

$$L \subseteq \theta^{-1}(\theta(L)). \tag{0.29}$$

Em outras palavras, dizemos que θ é a projeção natural cujo efeito sobre uma palavra $s \in \Sigma^*$ é de apenas apagar elementos σ de s que não pertençam ao conjunto de eventos observáveis Σ_o.

Relações de Equivalência e Observabilidade

Desde que $L \subseteq \Sigma^*$, usa-se $\mathcal{E}(L)$ para denotar o conjunto de todas as relações de equivalência sobre L [29]. Então, cada elemento de $\mathcal{E}(L)$ é determinado pela relação de equivalência $\ker\theta$, o qual é definido como a seguir:

Definição 29 *Seja $L \subseteq \Sigma^*$ e $\mathcal{E}(L)$ o conjunto de todas as relações de equivalência sobre L. Então, cada elemento de $\mathcal{E}(L)$, é determinado pela relação de equivalência definida por*

$$\ker\theta = \{(s_1, s_2) \,|\theta(s_1) = \theta(s_2), s_1, s_2 \in L\}. \tag{0.30}$$

Pode-se ver que $\ker\theta \subseteq \Sigma^* \times \Sigma^*$.

A relação $\ker\theta$ implica em dizer que duas palavras pertencentes à linguagem $L \subseteq \Sigma^*$, são iguais, desde que seus mapeamentos também sejam iguais. Em outros termos, numa determinada linguagem, duas palavras são iguais, se elas têm a mesma seqüência de eventos observáveis e realizam uma mesma tarefa.

Exemplo 34 *Sendo $\Sigma = \Sigma_o \cup \Sigma_{uo}$, com $\Sigma = \{\alpha, \beta, \gamma\}$ e $\Sigma_o = \{\alpha, \beta\}$, então as palavras*

$$s_1 = \alpha\alpha\beta\gamma\alpha\gamma\gamma\gamma\beta\alpha \qquad e$$
$$s_2 = \gamma\alpha\alpha\beta\gamma\gamma\alpha\gamma\beta\gamma\gamma\alpha,$$

$s_1, s_2 \in L \subset \Sigma^*$, são tais que

$$(s_1, s_2) \in \ker\theta,$$

pois

$$\theta(s_1) = \alpha\alpha\beta\alpha\beta\alpha = \theta(s_2).$$

Exemplo 35 *Sendo* $\Sigma = \Sigma_o \cup \Sigma_{uo}$*, com* $\Sigma = \{\alpha, \beta, \gamma\}$ *e* $\Sigma_o = \{\alpha, \beta\}$*, então para a linguagem*

$$L = \{\alpha, \alpha\beta, \alpha\gamma\gamma\beta\alpha, \gamma\beta, \gamma\alpha\gamma\gamma, \alpha\gamma\beta, \gamma\alpha\beta\gamma\alpha, \gamma\gamma\beta\gamma\},$$

tem-se que

$$\ker\theta = \{(\alpha, \gamma\alpha\gamma\gamma), (\alpha\beta, \alpha\gamma\beta), (\alpha\gamma\gamma\beta\alpha, \gamma\alpha\beta\gamma\alpha), (\gamma\beta, \gamma\gamma\beta\gamma)\}.$$

Para definir observabilidade é conveniente associar com cada palavra s dois subconjuntos de eventos distintos, como a seguir:

Definição 30 *Seja* $K \subseteq \Sigma^*$*. Para* $s \in \Sigma^*$*, define-se o conjunto de eventos ativos por*

$$A_K(s) = \begin{cases} \left\{\sigma \in \Sigma | s\sigma \in \overline{K}\right\}, & \text{se } s \in \overline{K} \\ \varnothing, & \text{caso contrário} \end{cases}$$

e, o conjunto de eventos inativos por

$$IA_K(s) = \begin{cases} \left\{\sigma \in \Sigma | s\sigma \in L(G) - \overline{K}\right\}, & \text{se } s \in \overline{K} \\ \varnothing, & \text{caso contrário.} \end{cases}$$

Com essa definição, tem-se que o conjunto $A_K(s)$ consiste de todos os eventos, cuja ocorrência de um evento seguindo um prefixo de s de K preserva a propriedade de prefixo, enquanto o conjunto $IA_K(s)$ contém os eventos que poderiam ocorrer em G, mas que destróem a propriedade de prefixo. Assim, utilizando esses conjuntos, define-se a relação binária $K-$*ativa* sobre Σ^*, denotada por act_K, como a seguir:

Definição 31 act_K *é uma relação de tolerância sobre* Σ^**, se para toda palavra* $s, s' \in \Sigma^*$*, o par* $(s, s') \in act_K$ *se e somente se:*

1. $A_K(s) \cap IA_K(s') = \varnothing = A_K(s') \cap IA_K(s)$*, e*
2. $s \in \overline{K} \cap L_m(G) \wedge s' \in \overline{K} \cap L_m(G) \Rightarrow (s \in K \Leftrightarrow s' \in K)$.

Nessa definição, nota-se que um par $(s, s') \in act_K$ se s ou s' não pertence a $\overline{K}$, porque se $s \notin \overline{K}$, então $A_K(s) = IA_K(s) = \varnothing$. Caso contrário, os pares de palavras (s, s') em act_K significa que os prefixos de s e s' de K têm continuações (um evento posterior) idênticas com respeito aos pares de palavras em $\overline{K}$ e, se cada um par está em $L_m(G)$ e um de fato pertence a K, então, o outro também pertence.

Exemplo 36 *Considere* $\Sigma = \{\alpha, \beta, \lambda, \eta\}$, $\Sigma_{uo} = \{\lambda, \eta\}$,

$$L(G) = \overline{\lambda\alpha + \eta(\alpha + \beta)}$$

e

$$K_1 = \overline{\lambda\alpha + \eta\beta}.$$

Então, por exemplo

$$\begin{aligned} A_K(\epsilon) &= \{\lambda, \eta\}, \quad & IA_K(\epsilon) &= \varnothing \\ A_K(\lambda) &= \{\alpha\}, \quad & IA_K(\lambda) &= \varnothing \\ A_K(\eta) &= \{\beta\}, \quad & IA_K(\eta) &= \{\alpha\}. \end{aligned}$$

Assim, os pares de palavras $(\epsilon, \lambda), (\epsilon, \eta) \in act_{K_1}$, *mas* $(\lambda, \eta) \notin act_{K_1}$. *Agora, suponha*

$$L_m(G) = \epsilon + \lambda(\epsilon + \alpha) + \eta(\alpha + \beta)$$

e

$$\overline{K_2} = K_1,$$

porém com K_2 *sendo não igualmente* $L_m(G)$*-fechada, desde que*

$$\lambda \in \overline{K_2} \cap L_m(G), \lambda \notin K_2.$$

Dessa forma, tem-se que $\epsilon \in K_2 \cap L_m(G)$ *e* $\lambda \in \overline{K_2} \cap L_m(G)$, *mas* $\lambda \notin K_2$. *Assim,* $(\epsilon, \lambda) \notin act_{K_2}$

A Definição 31 pode ser escrita equivalentemente, como a seguir:

Definição 32 act_K *é uma relação de tolerância sobre* Σ^*, *se para toda palavra* $s, s' \in \Sigma^*$, *o par* $(s, s') \in act_K$ *se e somente se:*

1. $(\forall\sigma)\, s\sigma \in \overline{K} \wedge s' \in \overline{K} \wedge s'\sigma \in L(G) \Rightarrow s'\sigma \in \overline{K}$;
2. $s \in K \wedge s' \in \overline{K} \cap L_m(G) \Rightarrow s' \in K$;
3. *As duas condições anteriores se mostram com* s *e* s' *permutados.*

Deve-se observar que o act_K é uma operação reflexiva e simétrica. Porém, ela não é transitiva.

Exemplo 37 *Considere* $\Sigma = \Sigma_o \cup \Sigma_{uo}$, *com* $\Sigma = \{\alpha, \beta, \gamma\}$ *e* $\Sigma_o = \{\alpha, \beta\}$. *Sendo*

$$L(G) = \{\epsilon, \alpha, \alpha\gamma, \alpha\gamma\gamma, ..., \alpha\beta, \alpha\gamma\beta, \alpha\gamma\gamma\beta, ..., \alpha\beta\gamma, \alpha\beta\gamma\gamma, ...\},$$
$$L_m(G) = \{\epsilon, \alpha, \alpha\gamma, \alpha\gamma\gamma, ...\}$$

e

$$K = \overline{K} = \{\epsilon, \alpha, \alpha\gamma, \alpha\gamma\gamma\},$$

então, para as palavras $s = \alpha$ *e* $s' = \alpha\gamma$, *encontra-se que*

$$\begin{aligned} &s\gamma = \alpha\gamma \in \overline{K}, \\ &s' = \alpha\gamma \in \overline{K}, \\ &s'\gamma = \alpha\gamma\gamma \in L(G), \end{aligned}$$

que juntas implicam em que

$$s'\gamma = \alpha\gamma\gamma \in \overline{K},$$

o que pode ser visto na própria linguagem K. *Também,*

$$\begin{aligned} &s = \alpha \in K, \\ &s' = \alpha\gamma \in \overline{K} \cap L_m(G), \end{aligned}$$

que juntas implicam em que

$$s' = \alpha\gamma \in K.$$

Permutando s *e* s', *tem-se*

$$\begin{aligned} &s'\gamma = \alpha\gamma\gamma \in \overline{K}, \\ &s = \alpha \in \overline{K}, \\ &s\gamma = \alpha\gamma \in L(G). \end{aligned}$$

Logo

$$s\gamma = \alpha\gamma \in \overline{K}$$

e

$$\begin{aligned} &s' = \alpha\gamma \in K, \\ &s = \alpha \in \overline{K} \cap L_m(G), \end{aligned}$$

que juntas implicam em que

$$s = \alpha \in K.$$

Assim, o par $(\alpha, \alpha\gamma) \in act_K$.

Considerando as definições anteriores, e sendo $L_m(G)$ a linguagem marcada do SED, a seguinte definição determina quando uma linguagem especificada K é observável em relação à linguagem gerada do SED $L(G)$, a partir da definição de act_K:

Definição 33 *Uma linguagem $K \subseteq L_m(G)$ é observável se*

$$\ker \theta \leq act_K, \tag{0.31}$$

isto é,

$$(\forall s, s' \in \Sigma^*)\, \theta(s) = \theta(s') \Rightarrow (s, s') \in act_K. \tag{0.32}$$

Em outras palavras, $\theta(s)$ pode ser vista como a saída de G. Daí, um requerimento natural da observabilidade é que, se s e s' produzem a mesma saída, então s e s' devem ser consistentes, ou seja, operam numa mesma constância para produzirem um mesmo comportamento.

Exemplo 38 *A linguagem K do Exemplo 37 é observável, pois vê-se que a condição da Definição 33 é satisfeita.*

Controle de SEDs com Observação Parcial de Eventos

Quando se considera a existência de eventos não observáveis em um SED, o supervisor apenas vê os eventos observáveis que ocorrem em G. Assim, para este caso, o supervisor é um gerador $S = (\Sigma, X, \xi, x_0, X_m)$, definido como na Definição 18, tal que para uma palavra

$$t \in \theta(L(G)),$$

o supervisor $\mathbf{S} = \langle S, \Theta \rangle$ é definido de acordo com:

$$\mathbf{S}(t) = \Sigma_{uc} \cup \left\{ \sigma \in \Sigma_c | \exists s'\sigma \in \overline{K}\, (\theta(s') = t) \right\}. \tag{0.33}$$

Assim, o supervisor habilita após a ocorrência da palavra observada $t \in \theta(L(G))$:

i. Todos os eventos não controláveis e

ii. Todos os eventos que podem continuar em qualquer palavra s' (tal que $\theta(s') = t$) pertence a $\overline{K}$.

Esse formalismo refere-se a que o supervisor apenas vê os eventos observáveis (Σ_o) que são gerados em G.

Com a introdução da observação de eventos, o seguinte teorema garante a existência do supervisor:

Teorema 3 *Considere um SED* $G = (Q, \Sigma, \delta, q_0, Q_m)$*, em que*

$$\Sigma = \Sigma_c \cup \Sigma_{uc} = \Sigma_o \cup \Sigma_{uo}.$$

Coloque K *sendo uma sublinguagem não vazia de* $L_m(G)$*. Existe um supervisor próprio* $\mathbf{S}$ *para* G*, tal que*

$$L_m(\mathbf{S}/G) = K \tag{0.34}$$

e

$$L(\mathbf{S}/G) = \overline{K}$$

se e somente se K *é* $L_m(G)$*-fechada, controlável com respeito à* $L(G)$ *e* Σ_{uc} *e observável com respeito à* $L(G)$*,* θ *e* Σ_c*.*

Então, para existir um supervisor é necessário que as linguagens $L(\mathbf{S}/G)$ e $L_m(\mathbf{S}/G)$ que descrevem o comportamento do SED supervisionado, em que

$$L_m(\mathbf{S}/G) := L(\mathbf{S}/G) \cap L_m(G),$$

devem ser tais que

$$L_m(\mathbf{S}/G) = K$$

e sejam expressas pelos mesmos conceitos de observabilidade e controlabilidade. Ou seja, o supervisor só existe se a linguagem especificada K for controlável e observável ao mesmo tempo.

Quando apenas a linguagem $L(\mathbf{S}/G)$ é de interesse, então o seguinte corolário é apresentado:

Corolário 1 *Considere um SED* $G = (Q, \Sigma, \delta, q_0, Q_m)$*, em que*

$$\Sigma = \Sigma_c \cup \Sigma_{uc} = \Sigma_o \cup \Sigma_{uo}.$$

Coloque K sendo uma sublinguagem não vazia de $L(G)$. Existe um supervisor $\mathbf{S}$*, tal que*

$$L(\mathbf{S}/G) = \overline{K} \tag{0.35}$$

se e somente se K é controlável com respeito à $L(G)$ e Σ_{uc} e observável com respeito à $L(G)$, θ e Σ_c.

Quando K não satisfaz as condições apresentadas, é necessário encontrar um supervisor que apresente uma sublinguagem de K, que seja controlável e observável. Assim, para enunciar o problema de observação e controle supervisório ($POCS$), considere duas linguagens fechadas $A, E \subseteq \Sigma_o^*$, tais que

$$A \subseteq E \subseteq L(G) \tag{0.36}$$

onde A é interpretado como o comportamento mínimo admissível, E é o comportamento legal (ou especificado) e Σ_o^* é o conjunto de todas as palavras formadas apenas por eventos observáveis. Assim, tem-se que o controle da planta G é tal que, uma linguagem menor que A é considerada inadequada.

Para a resolução do $POCS$, tem-se que construir um supervisor $\mathbf{S}$ para G, tal que

$$A \subseteq L(\mathbf{S}/G) \subseteq E. \tag{0.37}$$

Isto é, a linguagem do sistema supervisionado deve ser no mínimo a menor linguagem admissível e, no máximo, o comportamento especificado.

Se as linguagens A e E são fechadas, então tem-se:

Proposição 2 *Se A e E são $L_m(G)-$fechadas e* $\mathbf{S}$ *soluciona o* POCS*, então*

$$A \subseteq L_m(\mathbf{S}/G) \subseteq E. \tag{0.38}$$

Para discutir a solução do $POCS$, define-se para as linguagens fechadas $A, E \subseteq L(G)$, $A \neq \varnothing$, os seguintes conjuntos:

$$\underline{C}(E) \ : = \{K | K \subseteq E \wedge K \text{ é fechada e controlável}\} \tag{0.39}$$

$$\underline{O}(L) \ : = \{K | K \supseteq A \wedge K \text{ é fechada e observável}\}. \tag{0.40}$$

Utiliza-se aqui a notação $\underline{C}(E)$ em substituição à $CF(L)$, por conveniência.

Desses conjuntos, apresentam-se as seguintes proposições:

Proposição 3 *$\underline{C}(E)$ é uma classe de linguagens, não vazia, que é fechada sob uniões arbitrárias. Particularmente,* $\sup \underline{C}(E)$ *existe em* $\underline{C}(E)$.

Proposição 4 *$\underline{O}(A)$ é uma classe de linguagens, não vazia, que é fechada sob interseções arbitrárias. Particularmente,* $\inf \underline{O}(A)$ *existe em* $\underline{O}(A)$.

Da Proposição 3, tem-se que quaisquer uniões de linguagens fechadas dá uma linguagem fechada e $\sup \underline{C}(E)$ é a máxima linguagem controlável existente em $\underline{C}(E)$, contida na linguagem E. A Proposição 4 determina a existência da menor linguagem observável existente em $\underline{O}(A)$, contida na linguagem A. Naturalmente,

$$\inf \underline{O}(A) \subseteq L(G). \tag{0.41}$$

Destas duas proposições, tem-se que, a solução do *POCS* é provável a partir do teorema a seguir:

Teorema 4 *O* POCS *é solucionável se e somente se*

$$\inf \underline{O}(A) \subseteq \sup \underline{C}(E). \tag{0.42}$$

Visto que, $A \subseteq L(\mathbf{S}/G) \subseteq E$ é a condição para a resolução do *POCS*, isto implica que

$$\overline{A} \subseteq \overline{E}. \tag{0.43}$$

E pelas propriedades da linguagem para a solução do *POCS*, tem-se

$$\overline{A} \cap K \subseteq \overline{E} \cap K \tag{0.44}$$

em que $\overline{A} \cap K$ e $\overline{E} \cap K$ são fechadas e, observável e controlável, respectivamente. Como $\overline{A} \cap K$ é observável, ela faz parte de $\underline{O}(A)$. Como $\overline{E} \cap K$ é controlável, ela se encontra em $\underline{C}(E)$. Logo, o comportamento mínimo admissível A e a especificação desejada E caracterizam a solução do *POCS* pelo Teorema 4.

Exemplo 39 *Considere* $\Sigma = \{\alpha, \beta, \eta\}$, $\Sigma_o = \{\alpha, \beta\}$ *e* $\Sigma_c = \{\alpha, \eta\}$. *Sendo*

$$L(G) = \overline{\alpha\eta^*(\beta + \alpha(\eta + \beta))} \qquad e$$
$$L_m(G) = \alpha\eta^*(\beta + \alpha(\eta + \beta)),$$

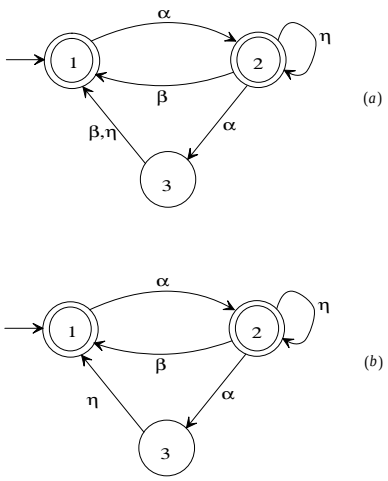

Figura 0.40: Gerador com eventos não observáveis e especificação de comportamento para exemplo da construção do supervisor.

e a especificação

$$K = L(H) = \alpha\eta^*(\beta + \alpha\eta)$$

vistos na Figura 0.40(a) e (b). Então, para construir o supervisor faz-se:

$$\theta(L(G)) = \overline{\alpha(\beta + \alpha\beta)} \qquad e$$
$$\theta(L_m(G)) = \alpha(\beta + \alpha\beta);$$

e calcula-se a $\sup C(E)$ *da mesma forma como visto no cálculo do suprevisor sem observação parcial, isto é, constrói-se a tabela a seguir*

K_0 :

	1 2 3	$\Sigma(H(x)) \cap \Sigma_{uc}$	$\Sigma(x)$
1	α		α
2	β η α	β	β, α
3	η		β

Como falha a condição $\Sigma(H'(3)) \cap \Sigma_{uc} \not\subset \Sigma(3)$*, então, eliminam-se a linha* 3 *e a coluna* 3 *da tabela, ficando com*

K_1 :

	1 2	$\Sigma(H(x)) \cap \Sigma_{uc}$	$\Sigma(x)$
1	α		α
2	β η	β	β

$L(H_1) = \alpha\eta^*\beta$

em que, $L(H_1)$ *é a* $\sup C(E)$*, pois não mais é violada a condição* $\Sigma(H(x)) \cap \Sigma_{uc} \subset \Sigma(x)$ *em nenhum estado (Figura 0.41(a)). O leitor pode ver que* $L(H_1)$ *é observável. Como o supervisor veja apenas os eventos observáveis do gerador* G*, sua linguagem é* $\theta(L(H_1)) = \alpha\beta$ *(Figura 0.41(b)), que em malha fechada com* G*, realiza* $L(H_1) = \alpha\eta^*\beta$*.*

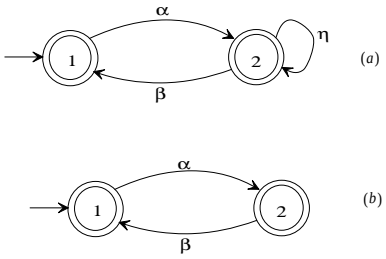

Figura 0.41: Supervisor para o gerador com eventos não observáveis.

Em resumo, a solução do *POCS* é determinada como a seguir: *Dado* K*, calcula-se o supervisor* $\mathbf{S}$ *sobre* $L(G)$*, e testa-se se* $\mathbf{S}$ *é observável (contém a* $\inf \underline{O}(A)$*), com* $\mathbf{S}$ *sendo o supervisor para a Suprema Sublinguagem Controlável.*

Observação: Em geral, se E não é fechada o *POCS* pode não ser solucionável simplesmente porque E tem apenas poucas sublinguagens.

Controle Supervisório e Normalidade

A definição de observabilidade é uma formalização para construir um supervisor para um SED com observação parcial de eventos. Contudo, apenas quando as linguagens relevantes são todas fechadas. Também, como a observabilidade não é preservada sob união, em geral um controle supervisório ótimo (minimamante restritivo) pode não existir. Entretanto, a definição de normalidade é mais forte que a definição de observabilidade e determina uma formalização mais consistente para sintetizar um supervisor. No entanto, esta solução restringe os resultados do controle supervisório por proibir a inibição de qualquer evento controlável não observável que ocorra. Isto é, apenas eventos observáveis podem ser inibidos.

Uma linguagem K é dita normal de acordo com a seguinte definição:

Definição 34 *Seja $K \subseteq L(G)$. K é dita ser normal com respeito à $L(G)$ e à θ, ou $L(G)$-normal, se*

$$K = L(G) \cap \theta^{-1}(\theta(K)). \tag{0.45}$$

K é normal apenas quando ela é a maior sublinguagem de $L(G)$, cuja projeção é $\theta(K)$. Desse modo, K é determinada unicamente pela sua projeção.

Exemplo 40 *Seja $\Sigma = \{\alpha, \beta, \kappa\}$, com $\Sigma_o = \{\alpha, \kappa\}$. Sendo $L(G) = \alpha + \beta^*\alpha\kappa + \beta\kappa^*$, a linguagem $K = \alpha + \beta\kappa^*$ é normal, pois*

$$\theta(K) = \alpha + \kappa^*,$$
$$\theta^{-1}(\theta(K)) = \beta^*\alpha\beta^* + \beta^*\kappa^*\beta^*,$$

e, conseqüentemente,

$$L(G) \cap \theta^{-1}(\theta(K)) = \alpha + \beta\kappa^* = K.$$

A seguinte proposição garante a observabilidade de uma linguagem normal:

Proposição 5 *Considere K sendo uma sublinguagem fechada ou $L_m(G)$-fechada de $L(G)$. Se K é normal, então K é observável.*

Considerando $E \subseteq L_m(G)$, o *POCS* é, então, enunciado como a seguir: *Encontrar um supervisor próprio* $\mathbf{S}$ *para G, tal que*

$$\varnothing \neq L_m(\mathbf{S}/G) \subseteq E. \tag{0.46}$$

A investigação do *POCS* requer a definição das seguintes classes de linguagens:

$$C(E) \;:\; = \{K \subseteq E | K \text{ é controlável}\}, \tag{0.47}$$
$$N(E, L_m(G)) \;:\; = \{K \subseteq E | K \text{ é } L_m(G) - \text{normal}\}, \tag{0.48}$$
$$\overline{N}(E, L(G)) \;:\; = \{K \subseteq E | \overline{K} \text{ é } L(G) - \text{normal}\}. \tag{0.49}$$

Cada família destas é não vazia e fechada sob uniões arbitrárias. Colocando

$$V(E) := C(E) \cap N(E, L_m(G)) \cap \overline{N}(E, L(G)), \tag{0.50}$$

então $V(E)$ é não vazio e fechado sob uniões arbitrárias. Assim, $\sup V(E)$ existe em $V(E)$.

A seguir, é apresentado uma condição suficiente para a solução do *POCS*:

Teorema 5 *Seja* $K \neq \varnothing$ *e* $K \in V(E)$. *Define-se*

$$\mathbf{S}(s) := \begin{cases} \Sigma_{uc} \cup \left\{\sigma \in \Sigma_c | \theta(s\sigma) \in \theta\left(\overline{K}\right)\right\}, & \theta(s) \in \theta\left(\overline{K}\right) \\ \Sigma_{uc}, & \theta(s) \in \theta(L(G)) - \theta\left(\overline{K}\right) \end{cases} \tag{0.51}$$

e

$$L_m(\mathbf{S}/G) := L(\mathbf{S}/G) \cap K. \tag{0.52}$$

Então, $\mathbf{S}$ *soluciona o* POCS, *com*

$$L_m(\mathbf{S}/G) = K. \tag{0.53}$$

Com esse teorema, garante-se a solução para o *POCS* quando uma linguagem é normal.

Bibliografia

[1] P.J.G. Ramadge and W.M. Wonham. Supervision of discrete event processes. *Procedings of 21st Conference on Decision and Control*, pages 1228–1229, 1982.

[2] P.J.G. Ramadge and W.M. Wonham. On the supremal controllable sublanguage of a given language. *SIAM Journal of Control and Optimization*, 25(3):637–659, May 1987.

[3] P.J.G. Ramadge and W.M. Wonham. The control of discrete event systems. *Proceedings of the IEEE*, 77(1):81–98, 1989.

[4] X.R. Cao and Y.C. Ho. Models of discrete event dynamic systems. *IEEE Control System Magazine*, 10(4):69–76, 1990.

[5] R. Milner. *A Calculus of Comunicating Systems*. Springer Verlag, New York, USA, 1980.

[6] S. Gaubert. *Théorie des Systèmes Linéaires dans les Dioïdes*. PhD thesis, École Nationale Supérieure des Mines de Paris, 1992.

[7] J.L. Boimond. Sur l'étude des systèmes à événements discrets dans l'algèbre des dioïdes: Identification, commande des graphes d'événements temporisés, représentation des graphes d'événements temporisés à paramètres variables. Technical report, l'Université d'Angers, 1999.

[8] S. Gaubert. On rational series in one variable over certain dioids. Technical report, Institut National de Recherche en Informatique et en Automatique, 1994.

[9] D. L'Her. *Modélisation du GRAFCET Temporisé et Vérification de Propriétés Temporelles*. PhD thesis, l'Université de Rennes 1, 1997.

[10] L. Libeaut. *Sur l'utilisation des Dioïdes pour la Commande des Systèmes a Événements Discrets*. PhD thesis, École Doctorale Sciences pour L'Ingenieur de Nantes, 1996.

[11] F. Baccelli, G. Cohen, G.J. Olsder, and J.P. Quadrat. *Synchronisation and Linearity. An Algebra for Discrete Event Systems.* John Wiley Sons, 1992.

[12] S. Gaubert. On the burnside problem for semigroups of matrices in the (max,+) algebra. *Semigroup Forum, Springer-Verlag New York Inc.*, 52:271–292, 1996.

[13] S. Gaubert and J. Mairesse. Task resource models and (max,+) automata. In J. Gunawardena Ed., editor, *Idempotency - Collection of the Isaac Newton Institute*, 1995.

[14] G. Cohen, D. Dubois, J.P. Quadrat, and M. Viot. A linear system theoretic view of discrete event process and its use for performance evaluation in manufacturing. *IEEE Transactions on Automatic Control*, 30(3):210–220, 1985.

[15] C.G. Cassandras and S. Lafortune. *Introduction to Discrete Event Systems.* Kluwer Academic Publishers, 1999.

[16] R.C. Green. *Minimax Algebra.* Lectures Notes in Economics and Mathematical Systems. Springer-Verlag, 1979.

[17] J.C. Terrasson, S. Gaubert, and J. Gunawardena. Dynamics of min-max functions. Technical report, HP Laboratories Technical Report, 1997.

[18] S. Gaubert. Introduction aux systèmes dynamiques à Événements discrets. Notes de Cours, 1999.

[19] T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[20] J.L. Peterson. *Petri Net Theory and Modeling of Systems.* Prentice Hall, 1981.

[21] W. Reisig. *Petri Nets: an Introduction.* Springer-Verlag Berlin Heidelberg, 1985.

[22] W. Reisig. *A Primer in Petri Net Design.* Springer-Verlag, 1992.

[23] K. Jensen. *Coloured Petri Nets - Basic Concepts, Analysis Methods and Practical Use*, volume 1 : Basic Concepts. Springer-Verlag, 1992.

[24] F. DiCesare, G. Harhalakis, J.M. Proth, M. Silva, and F.B. Vernadat. *Practice of Petri Nets in Manufacturing.* Chapman and Hall, 1993.

[25] J.M. Proth and X. Xie. *Petri Nets: A Tool for Design and Management of Manufacturing Systems.* John Wiley - Sons Ltd, 1996.

[26] M.C. Zhou and F. DiCesare. *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Kluwer Academic Publishers, 1993.

[27] A. Gill. *Introduction to the Theory of Finite-State Machines*. McGraw-Hill Electronic Sciences Series. McGRAW-HILL Book Company, 1962.

[28] J.E. Hopcroft and J.D. Ullman. *Formal Languages and Their Relation to Automata*. Addison-Wesley Publishing Company, 1969.

[29] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, USA, 1979.

[30] A. Salomaa. *Formal Languages*. ACM Monograph Series. Academic Press, Inc. New York, 1973.

[31] S. Pinchinat, H. Marchand, and M. Le Borgne. Symbolic abstractions of automata and their application to the supervisory control problem. Technical report, Institut National de Recherche en Informatique et en Automatique, 1999.

[32] A. Pnueli. The temporal semantics of concurrent programs. *18th Symposium on Foudations of Computer Science*, 1977.

[33] K.L. McMillan. *Symbolic Model Checking: An Approach to the State Explosion Problem*. PhD thesis, Carnegie Mellon University, 1992.

[34] J.C. Bradfield. *Verifying Temporal Properties of Systems*. Progress in Theoretical Computer Science. Birkhäuser, 1992.

[35] A. Galton. *Temporal Logics and their Applications*. Academic Press, 1987.

[36] K. Heljanko. Model checking the branching time temporal logic ctl. Technical Report 45, Helsinki University of Technology - Otaniemi, Finland, May, 1997.

[37] J.S. Ostroff. *Temporal Logic for Real-Time Systems*. Research Studies Press Ltd., 1989.

[38] P. Racloz and D. Buchs. Symbolic proof of ctl formulae over petri nets. Technical report, C.U.I. University of Geneva, 1997.

[39] F. Lin and W.M. Wonham. On observability of discrete event systems. *Information Sciences*, pages 173–198, 1988.

[40] R. Cieslak, C. Desclaux, A.S. Fawaz, and P. Varaiya. Supervisory control of discrete-event processes with partial observations. *IEEE Transactions on Automatic Control*, 33(3):249–260, 1988.

[41] A. Haji-Valizadeh and K.A. Loparo. Minimizing the cardinality of an events set for supervisors of discrete-event dynamical systems. *IEEE Transactions on Automatic Control*, 41(11):1579–1593, 1996.

[42] E.M.M. Costa. *Observabilidade: Uma Ferramenta para a Síntese de Supervisores para Sistemas a Eventos Discretos*. Relatório de Projeto e Pesquisa. Universidade Federal da Paraíba - Campus II, Campina Grande, PB, Brasil, Fevereiro, 2000.

[43] M. Heymann and F. Lin. Discrete-event control of nondeterministic systems. *IEEE Transactions on Automatic Control*, 43(1):3–17, 1998.

[44] R. Kumar, V. Garg, and S.I. Marcus. On supervisory control of sequential behaviors. *IEEE Transactions on Automatic Control*, 37(12):1978–1985, 1992.

[45] M. Lawford, W.M. Wonham, and J.S. Ostroff. State-event observers for labeled transition systems. Technical report, University of Toronto, USA, 1997.

[46] C.M. Özveren and A.S. Willsky. Observability of discrete event systems. *IEEE Transactions on Automatic Control*, 35(7):797–806, 1990.

[47] H. Zhong and W.M. Wonham. On the consistency of hierarchical supervision in discrete-event systems. *IEEE Transactions on Automatic Control*, 35(10):1125–1134, 1990.

[48] Y. Guan. Implementation of hierarchical observer theory. Master's thesis, University of Toronto, 1997.

[49] Y. Brave and M. Heymann. Control of discrete event systems modeled as hierarchical state machines. *IEEE Transactions on Automatic Control*, 38(12):1803–1819, 1993.

[50] S. Gaubert and J. Gunawardena. A non-linear hierarchy for discrete event dynamical systems. In *Proc. Of the Fourth Workshop on Discrete Event Systems*, 1998.

[51] K.C. Wong. *Discrete-Event Control Architecture: An Algebraic Approach*. PhD thesis, Departament of Electrical Engineering, University of Toronto, Toronto, 1994.

[52] P.J.G. Ramadge and W.M. Wonham. Supervisory control of a class of discrete event processes. *SIAM Journal of Control and Optimization*, 25(1):206–230, 1987.