Sequential Function Charts

Apesar da programação de CLPs ter sido realizada intuitivamente por muitos anos, existem algumas razões para a consideração de métodos formais quando se programa aplicações complexas:

- o crescimento da complexidade de sistemas automatizados requer a minimização do tempo de programação e, se possível, a possibilidade de reutilização de códigos para CLPs;

- em aplicações críticas relacionadas à sistemas de segurança, existe a necessidade de procedimentos de verificação e de validação das aplicações, ou seja, uma metodologia para se provar características estáticas e dinâmicas de programas de CLPs, tal como tempos de resposta.

A figura seguinte representa o processo de criação de um programa de controle para CLP. Sem o uso de técnicas formais, o processo seria limitado ao anel externo do diagrama. O programa é implementado diretamente da especificação informal do problema e validado informalmente com bases nestas especificações. Entende-se por especificação informal do problema a descrição do sistema automático em linguagem não formal (descrição do problema) e todo o conjunto de documentação técnica do sistema. A validação neste caso é possível posteriormente à implementação do projeto e quando se dispõe de uma equipe de testes, porém este é um processo lento e caro.

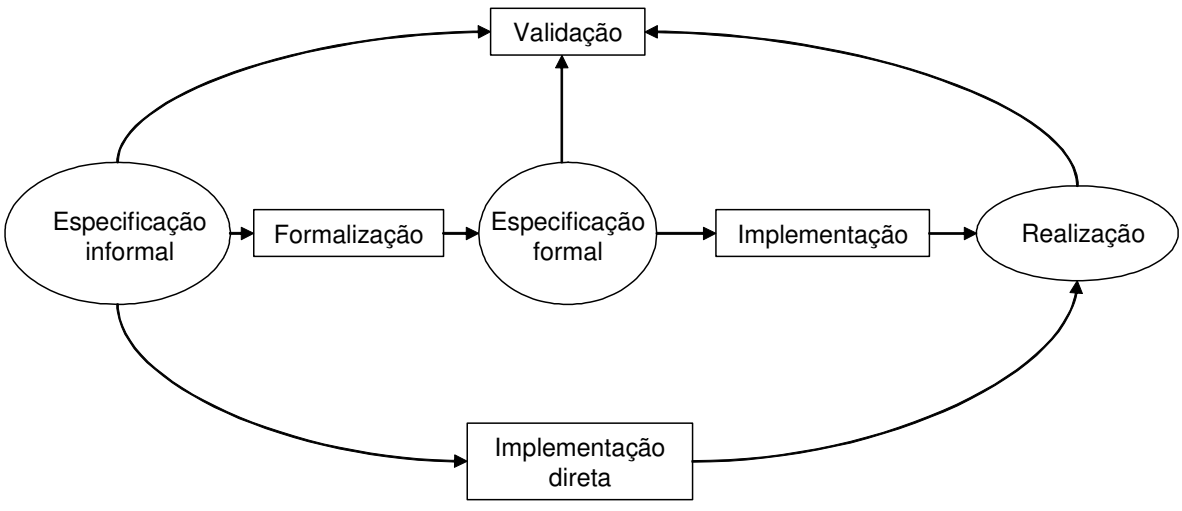


Figura 6.1. Processo de implementação de programas em CLPs

Como alternativa a esta metodologia de projeto, pode-se utilizar técnicas formais para a programação e a validação de programas em CLPs. Nesta metodologia, adicionam-se as etapas de formalização, especificação formal e implementação.

A formalização consiste da conversão da especificação informal em uma especificação formal, ou seja, que segue regras de representação e de comportamento. Esta etapa pode ter o auxílio de softwares nas nunca é automática, pois depende da interpretação do projetista. Uma especificação formal deve ser capaz de:

1 - capturar as características estáticas e dinâmicas do sistema;

2 - acomodar o estilo de modelagem dos estados e de atividades de uma forma natural;

3 - permitir a sua integração em qualquer outra linguagem de modelagem;

4 - ser capaz de representar sistemas sobre múltiplos níveis de abstração.

Já para o rastreamento de erros, verificação e validação, a técnica de especificação deve ser capaz de: a - permitir simulação e animação;

b - deve disponibilizar técnicas de análise eficientes para verificar as propriedades do modelo;

c - deve possuir ferramentas de análise de fácil acesso.

A implementação depende da plataforma tecnológica que se utiliza (software e hardware do CLP), neste caso pode-se contar com ferramentas de geração de código automático.

De acordo com a norma IEC 61131 descrita anteriormente, existem atualmente cinco linguagens padronizadas: IL (lista de instruções), ST (texto estruturado), FBD (diagramas de blocos funcionais), SFC (sequential function charts) e LD (diagramas ladder). No capítulo anterior foram apresentadas as linguagens LD e FBD, neste capítulo será apresentada a linguagem SFC.

Derivada das Redes de Petri (técnica descrita mais a diante neste capítulo) e do seu subconjunto destinado à modelagem de sistemas seqüenciais IEC 848, o denominado Grafcet (lançado em 1988), o SFC se presta a uma especificação formal de alto nível adequada para projetos de maior complexidade baseados em CLPs. A vantagem a destacar é a possibilidade de se traduzir um código SFC para as demais linguagens da norma.

Quando o projeto requer o processamento paralelo de duas ou mais tarefas em um controlador, ao contrario da programação direta em ladder (por exemplo), a técnica do Grafcet ou SFC torna-se mais adequada, pois baseia-se em diagramas gráficos de estrutura seqüencial adequada à modelagem de sistemas orientados a eventos, através de uma abordagem de máquina de estados.

Atualmente o Grafcet é adotado por alguns fabricantes de CLP como linguagem direta de programação. Para CLPs que não possuem essa característica, o Grafcet pode ser traduzido para ladder ou para qualquer outra linguagem IEC 1131, tornando-se assim uma ferramenta para elaboração de comandos seqüenciais, segundo uma abordagem *top down* para estruturação das ações do programa.

A sua filosofia consiste em partir da descrição informal das funções de automação a se projetar e decompô-las em passos e transições.

Nos passos e só neles são realizadas ações (por exemplo, ligar um contator de acionamento de um motor) ou eventualmente pode não se realizar qualquer ação (quando o controlador está em repouso). Em cada instante, numa dada seqüência só um passo está ativo.

Para haver a transição de um passo para outro é preciso que se verifique uma ou mais condições de transição. Por exemplo, para que um elevador em movimento do 2º para o 3º andar pare neste último, é preciso que um fim de curso indique a chegada da cabine a este andar.

Exemplo de representação de um SFC:

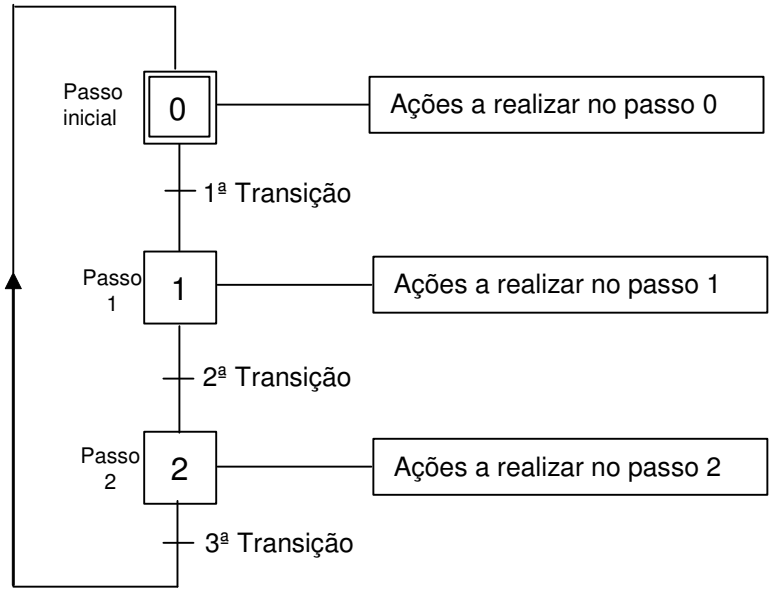
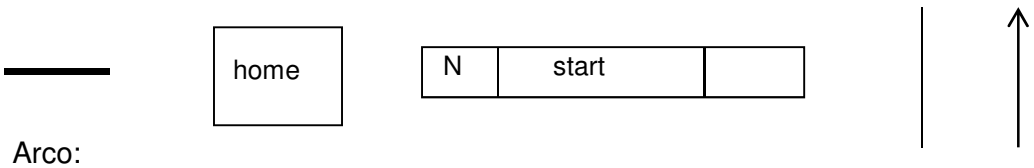


Figura 6.2. Exemplo de um SFC

Elementos de um Grafcet: passos, transições, arcos, ações e regras de evolução.

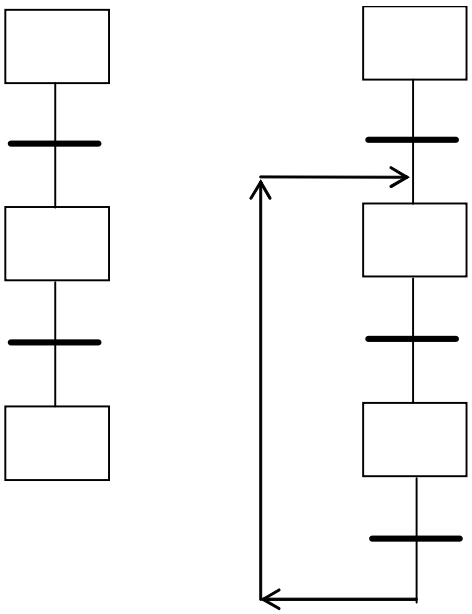
Elementos de um diagrama SFC ou Grafcet

Transição Passo (step) Ação Arco orientado

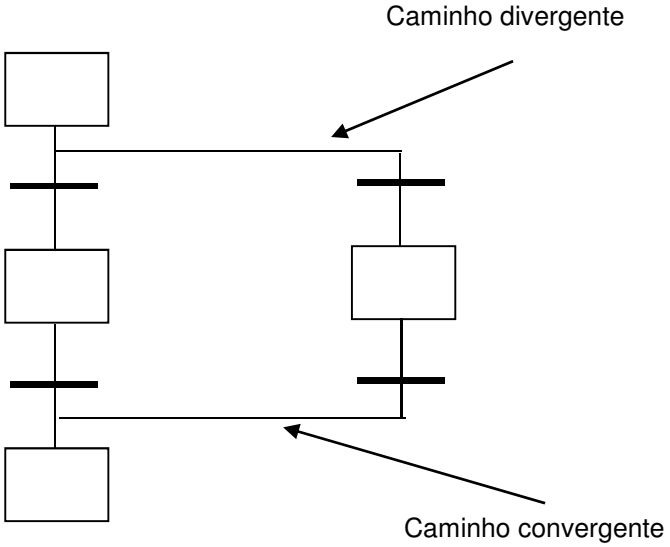


Arco:

- Um arco é representado por uma linha vertical.
- Um arco é orientado de cima pra baixo (default)
- No caso em que se projeta um arco orientado de baixo para a cima, o sentido deve ser representado através uma seta
- Uma seqüência é definida por uma série de passos, transições e passos ligados por arcos orientados



- É possível realizar seqüências alternativas através de caminhos divergentes e convergentes. As seqüências são executadas em mútua exclusão.

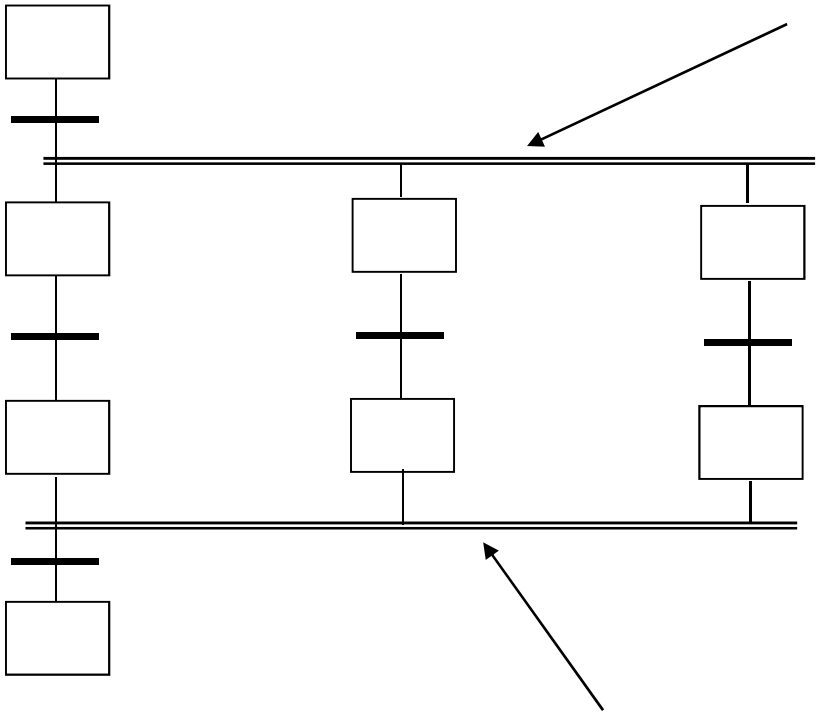


- Regras de validação de seqüências alternativas:

    1. As condições booleanas associadas a transições são estimadas da esquerda para direta.
    2. Se uma ou mais transições tem condições TRUE, a precedência é garantida à transição mais a esquerda
    3. É possível modificar a regra de precedência, atribuindo uma prioridade cada transição.
    4. É boa norma (hábito) associar à transições presentes em seqüências alternativas, condições lógicas mutuamente exclusivas

- Seqüência Simultâneas: é possível realizar seqüências simultâneas através de “Simultaneous Sequence Divergence” e “Simultaneous Sequence Convergence”.

Divergencia de seqüências simultâneas



Convergencia de seqüências simultâneas

- As seqüências simultâneas são consideradas todas em paralelo, pode-se por exemplo utilizar uma seqüência para o controle do processo e outra para realizar o monitoramento.

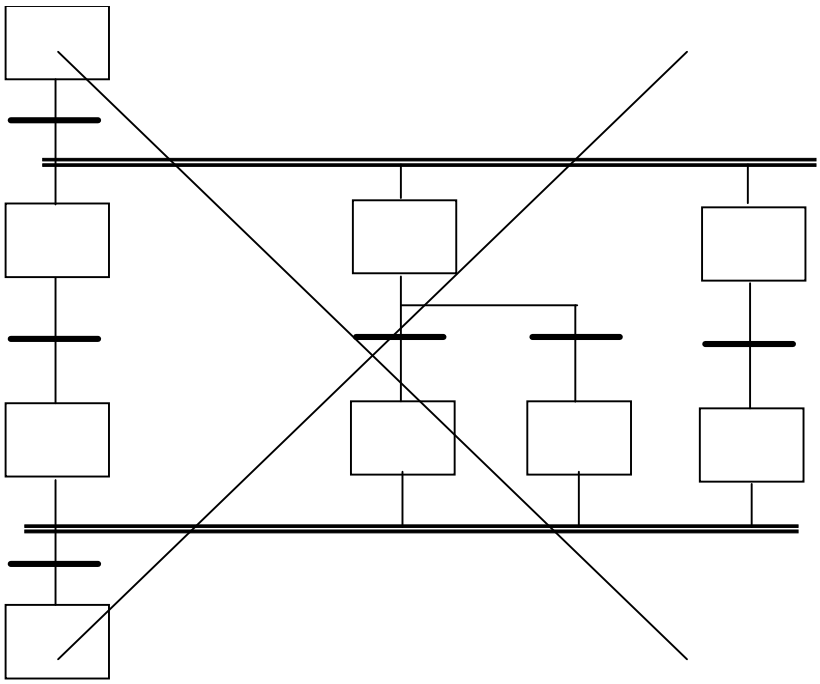
- Em cada seqüência simultânea, somente um passo de cada vez pode ser ativado

- A convergência de seqüências simultâneas acontece somente quando todos os últimos passos de cada seqüência forem ativados.

Regras de Programação de uma seqüência simultânea:

1. É necessário que a convergência de seqüências simultâneas possa ser atuada.
2. Todos os passos que se inserem em tal convergência devem poder ficar todos ativos

Exemplo de convergência de seqüências simultâneas errada:



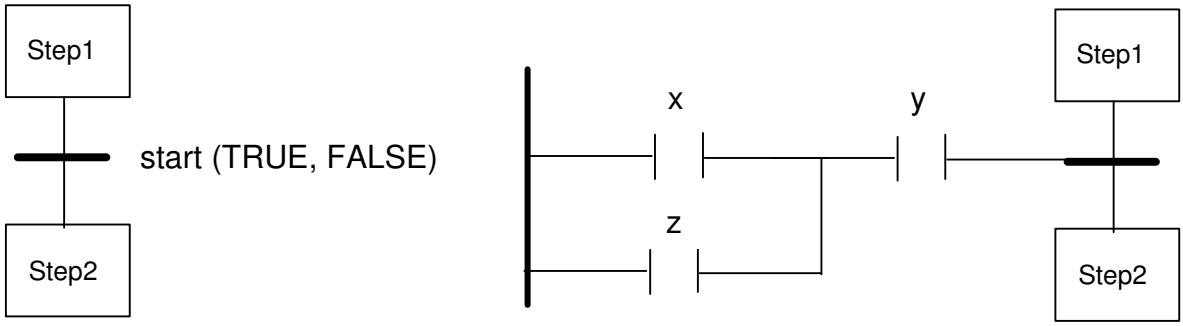
Transição:

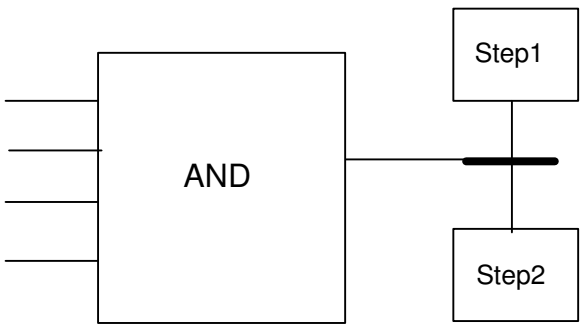
- Uma transição é representada por uma barra horizontal.
- A cada transição é possível associar pelo menos um passo precedente e pelo menos um passo seguinte (os passos precedentes e seguintes podem ser mais de um)
- Para cada transição vem associada uma condição. As condições podem ser constituídas de:
    1. Uma variável booleana (TRUE, FALSE)

2. Uma linha em linguagem ladder. Se a linha for ativada, a condição associada à transação torna-se verdadeira.
3. Uma saída binária de qualquer FB. Se a saída é verdadeira então as condições associadas à transição tornam-se verdadeiras.
4. Um nome de um programa. O programa deve ser definido separadamente e pode ser escrito em uma das linguagens IEC 1131-3. O programa deve fornecer como saída um valor booleano que deve ser associado ao nome do próprio programa.

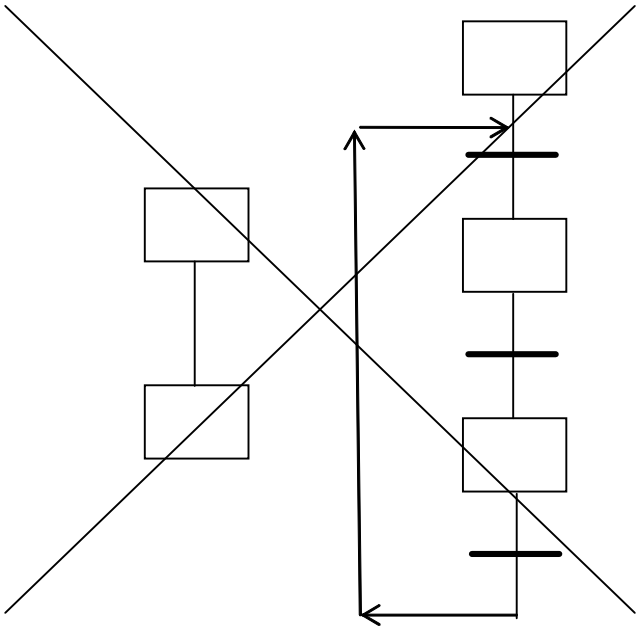
- A condição associada a uma transição é estimada **SOMENTE** quando todos os passos precedentes a essa estiverem ativos;
- Quando todos os passos precedentes a uma transição estiverem ativos e a condição associada à esta é verdadeira, todos os passos precedentes desativam-se e os passos seguintes tornam-se ativos;
- Se a uma transição está associada a um nome, este deve ser singular (ou único) em todo o programa;
- Os nomes associados às transições são variáveis locais.

Exemplos de Transição:

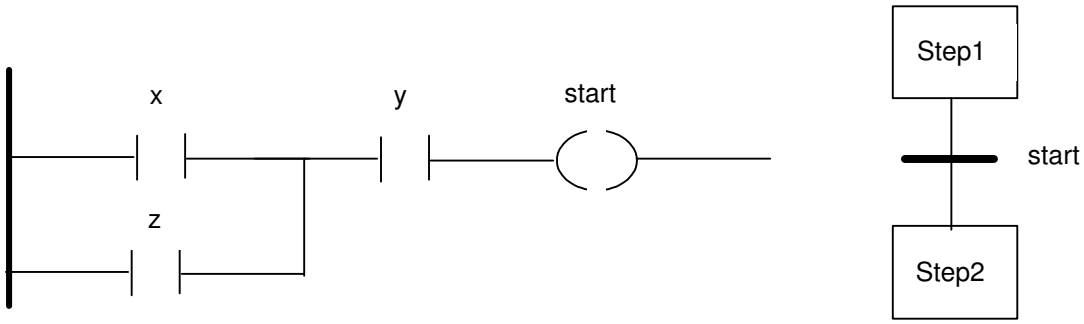




Exemplo de diagramas errados:



Programa "start" codificado em Ladder:



Passo (Step):

- Dois tipos de passo: Passo Normal e Passo Inicial (representado por caixa dupla)

- Pode existir somente um Passo Inicial, que vai ser ativado em um *cold-start*. Na linguagem Grafcet podem existir diversos passos iniciais (esta é uma diferença importante entre ambas as linguagens)

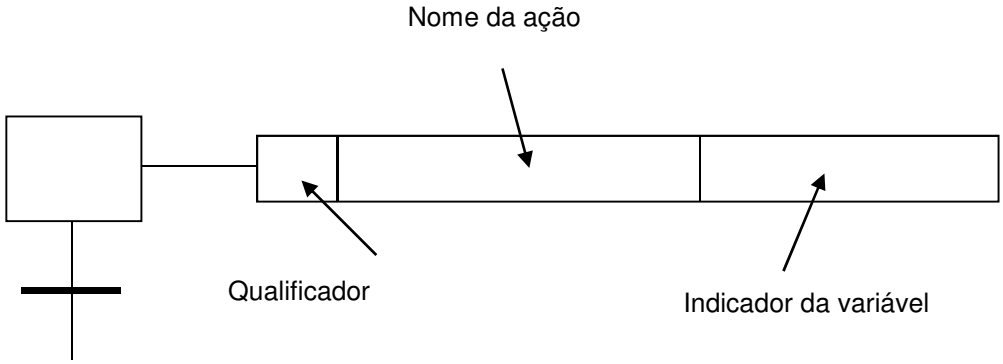
- Cada passo tem associado um nome único em todo o programa

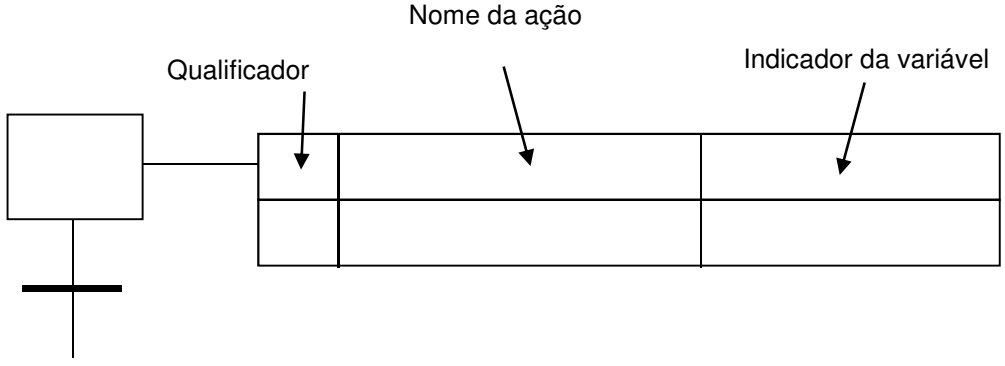
- Os nomes associados aos passos são variáveis locais

- Para cada passo é associado um estado: ativo ou não ativo. O estado é ativo quando a transição precedente ao passo é "liberada", isso significa também que a condição associada a esta transição tornou-se verdadeira.

Ações:

- A cada passo é possível se associar uma ou mais ações que descrevem o que se fazer quando o relativo passo é ativado
- Cada ação é representada por um retângulo conectado ao passo





- Qualificador: Este parâmetro especifica as modalidades de execução da ação. Pode assumir os valores: N, S, R, L, D, P, SD, DS, SL

- Nome da ação: Este parâmetro é único em todo o programa. Ele corresponde ao nome do programa que realiza a ação. O programa pode ser escrito em qualquer uma das linguagens IEC 1131

- Indicador da variável: Este parâmetro é opcional e permite indicar a variável que é modificada pela ação e, quando modificada, indica o fim da execução.

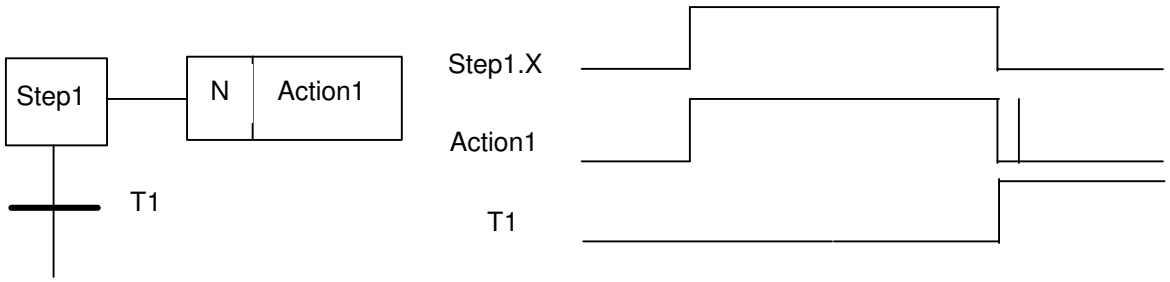
- Uma mesma ação pode ser associada a mais de um passo.

- É possível que a um passo não seja associada nenhuma ação. Nesse caso quando o Passo fica ativo, nada é executado, e se espera que o passo seja desativado pela liberação da transição de saída.

Observação: A norma IEC 1131-3 assume que qualquer ação associada a um passo vai ser executada ainda mais uma vez após a desativação do passo. Pode-se ou não adotar tal hipótese. Vantagem: permite o reset de eventuais parâmetros ou variáveis usados durante a execução da ação.

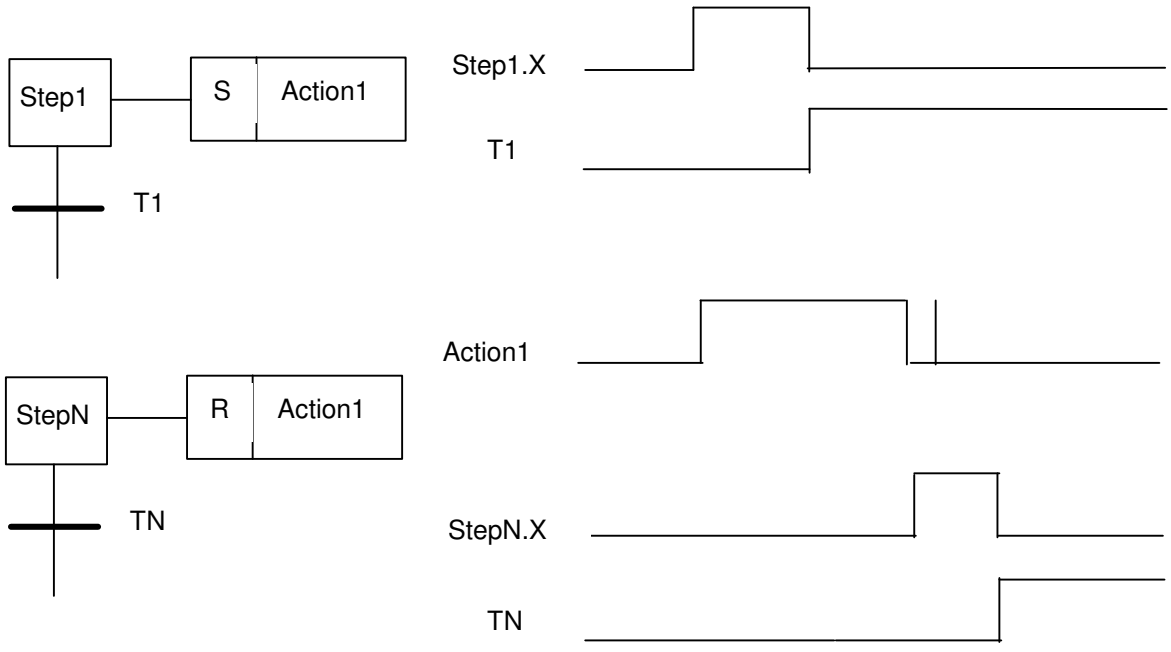
Qualificadores:

- Qualificador 'N' Non stored



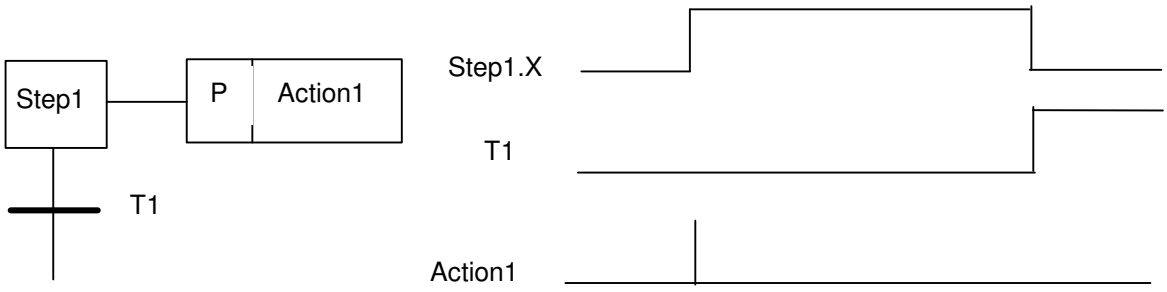
A ação "Action1" é executada continuamente enquanto o Step1 é ativo. Quando o Step1 é desativado, a ação é executada somente mais uma vez

- Qualificador 'S', 'R', Set e Reset



A ação "Action1" começa a ser executada quando o Step1 ficar ativo. A ação é memorizada e continua a ser executada enquanto o passo StepN ficar ativo. Neste caso a ação "Action1" é executada somente mais vez. No caso em que uma ação não seja resetada através do qualificador 'R', esta será executada indefinidamente.

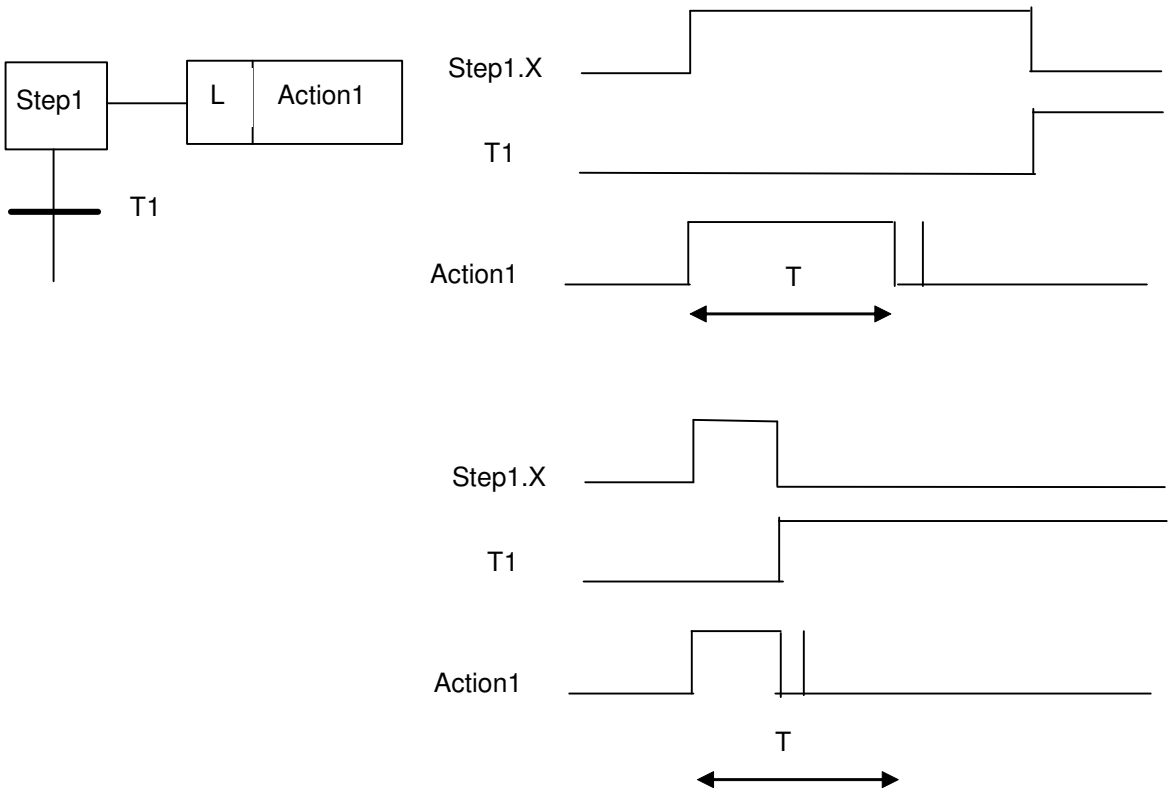
- Qualificador 'P' pulse



Quando o Step1 é ativado, a ação Action1 é executada somente uma vez.

Nota: em algumas implementações do padrão, a execução depois da desativação do Step1 pode não ser prevista.

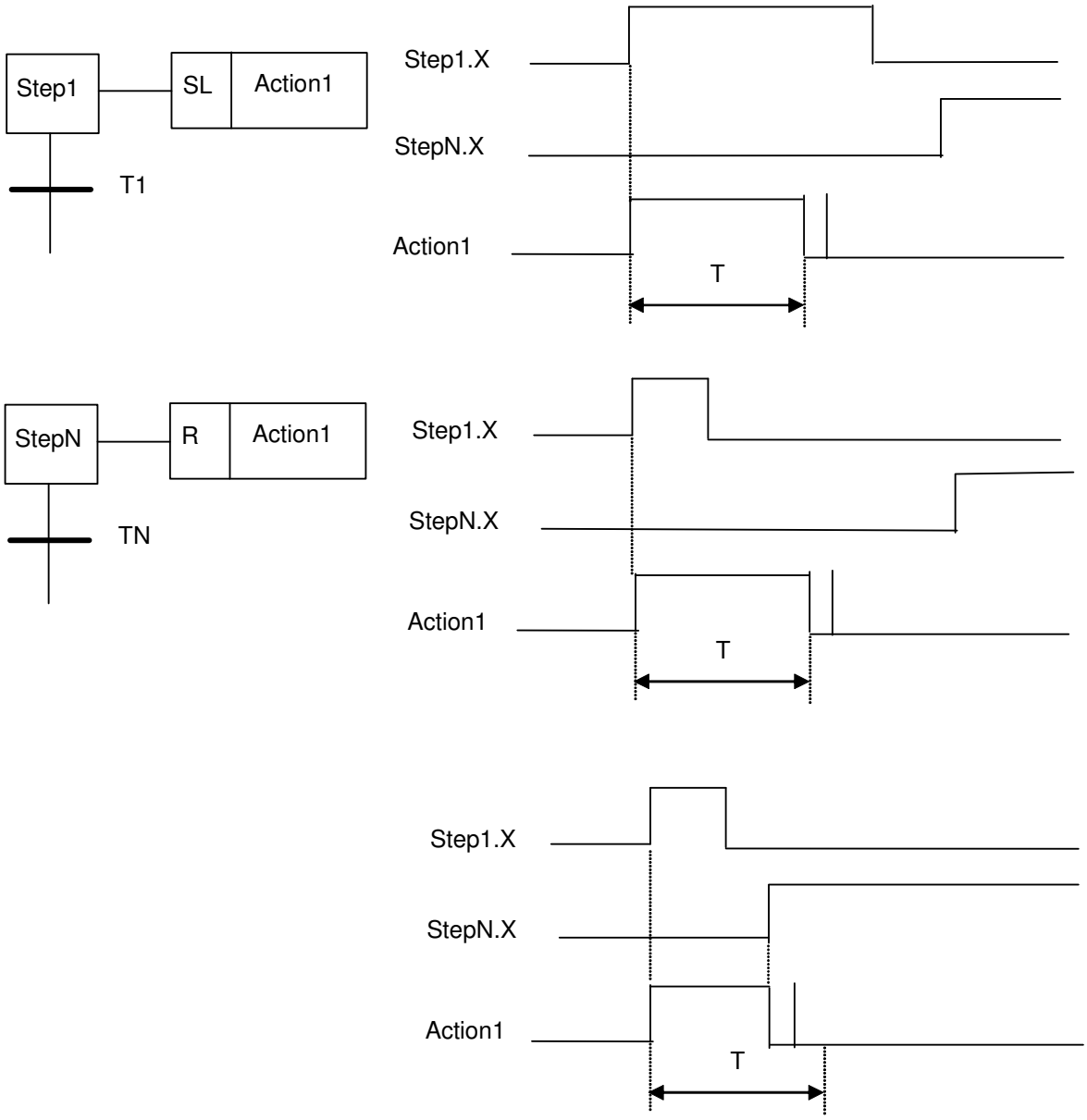
- Qualificador 'L' Time Limited



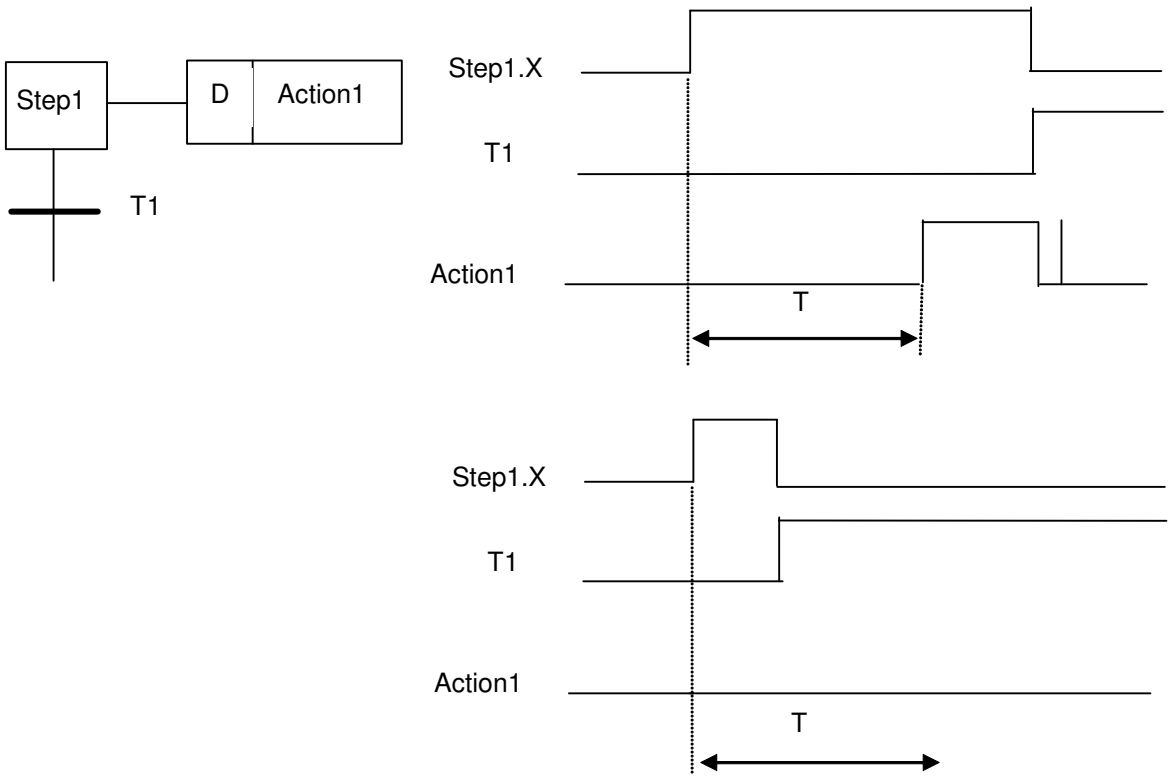
A ação "Action1" começa a ser executada quando o Step1 fica ativo. A ação é executada em um determinado intervalo de tempo T (associado ao qualificador L). Ao fim de tal intervalo, a ação é terminada. Se o passo é desativado antes que do final do tempo T, então a ação "Action1" é terminada.

Quando a ação é terminada, em ambos os casos, essa é executada pela a última vez e somente uma vez.

- Qualificador 'SL' stored and time limited

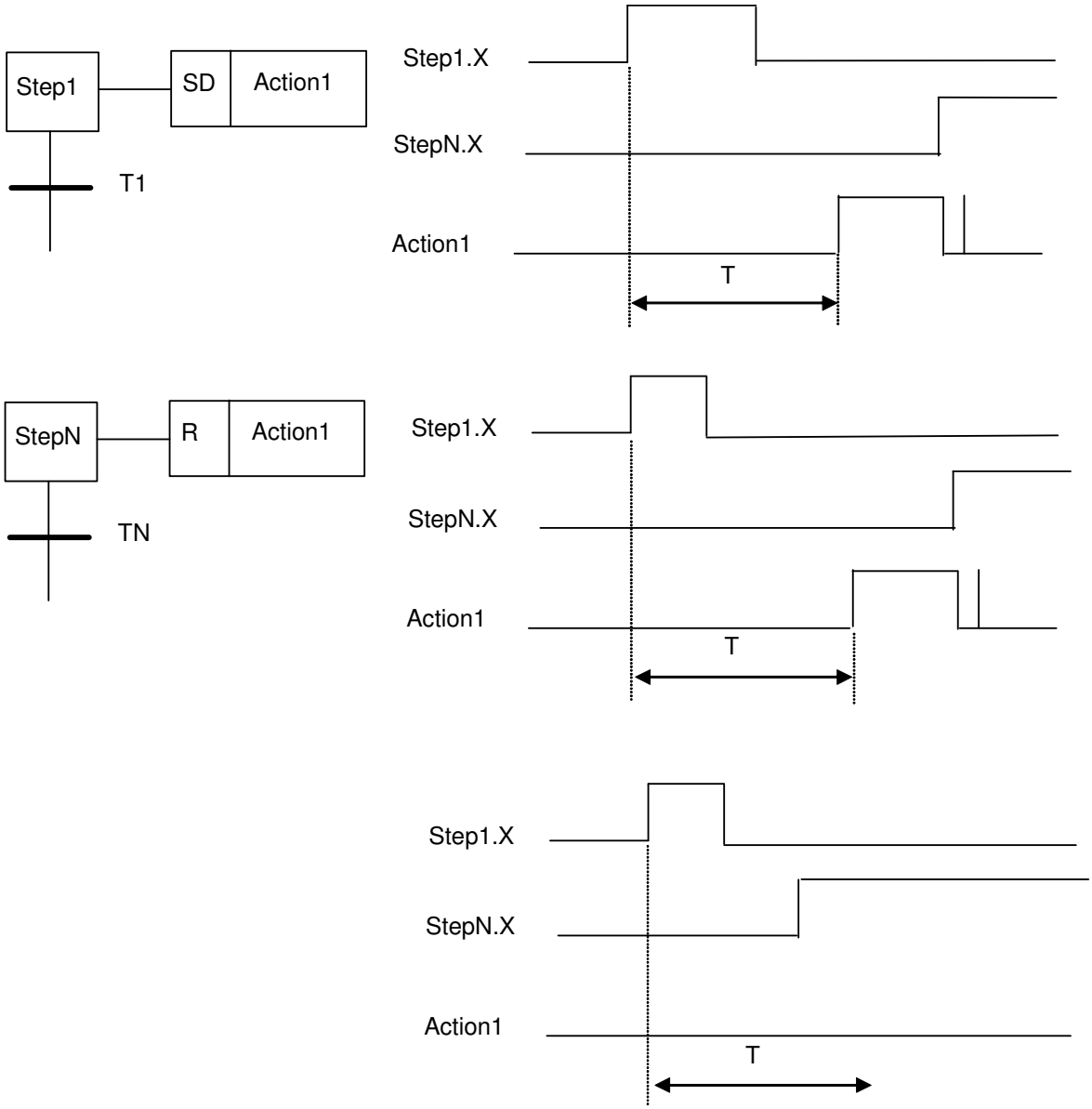

Step1
SL
Action1
T1
Step1.X
StepN.X
Action1
T
StepN
R
Action1
TN
Step1.X
StepN.X
Action1
T
Step1.X
StepN.X
Action1
T

- Qualificador 'D' time delayed



A ação "Action1" começa a ser executada depois do intervalo temporal T (associado ao qualificador D) a partir do instante no qual o Step1 fica ativo. A ação é executada enquanto o Step1 não é desativado. Nesse caso a ação "Action1" é executada somente uma vez. Se o Step1 não permanece ativo por um tempo pelo menos igual a T, a ação nunca é executada.

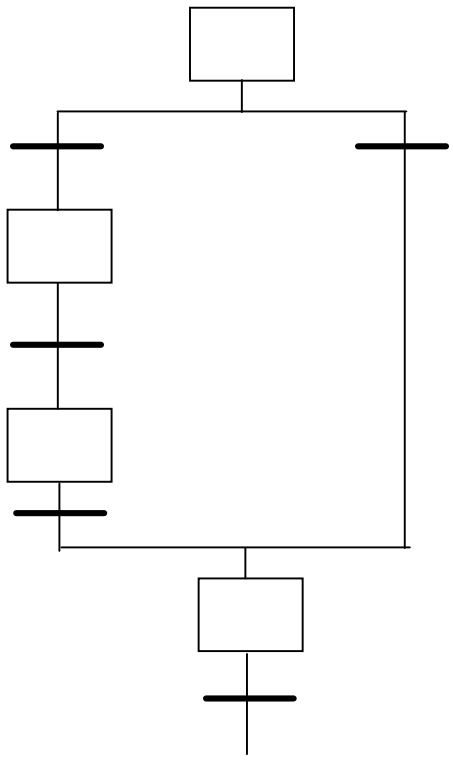
- Qualificador 'SD' stored and time delayed



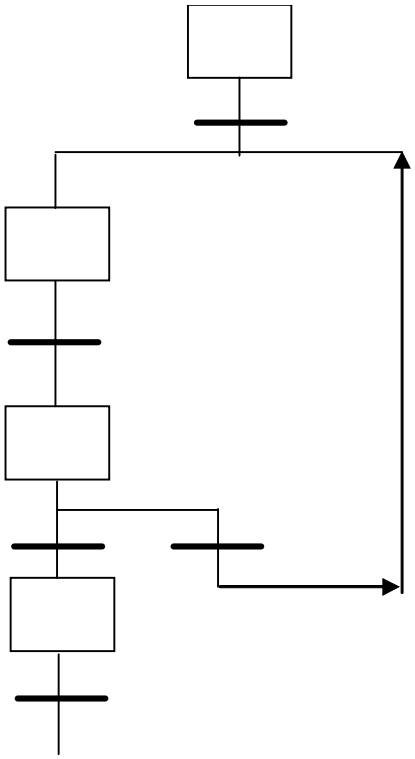
Quando o Step1 é ativo, a ação "Action1" é memorizada, mas não é executada até que o intervalo de duração T termine. A ação é executada enquanto não é resetada através do qualificador R. Se a ação "Action1" é chamada com o qualificador R antes que o intervalo T acaba-se, a ação não será executada.

Elementos Fundamentais de Controle através de SFC

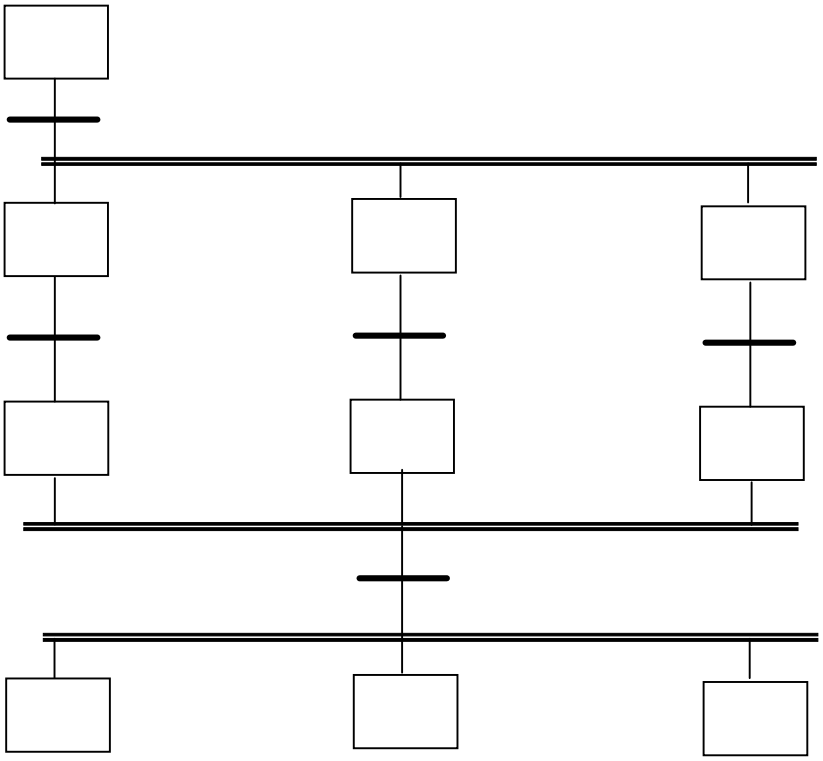
- “Bypass” ou salto de uma seqüência:



- Loop de uma sequencia



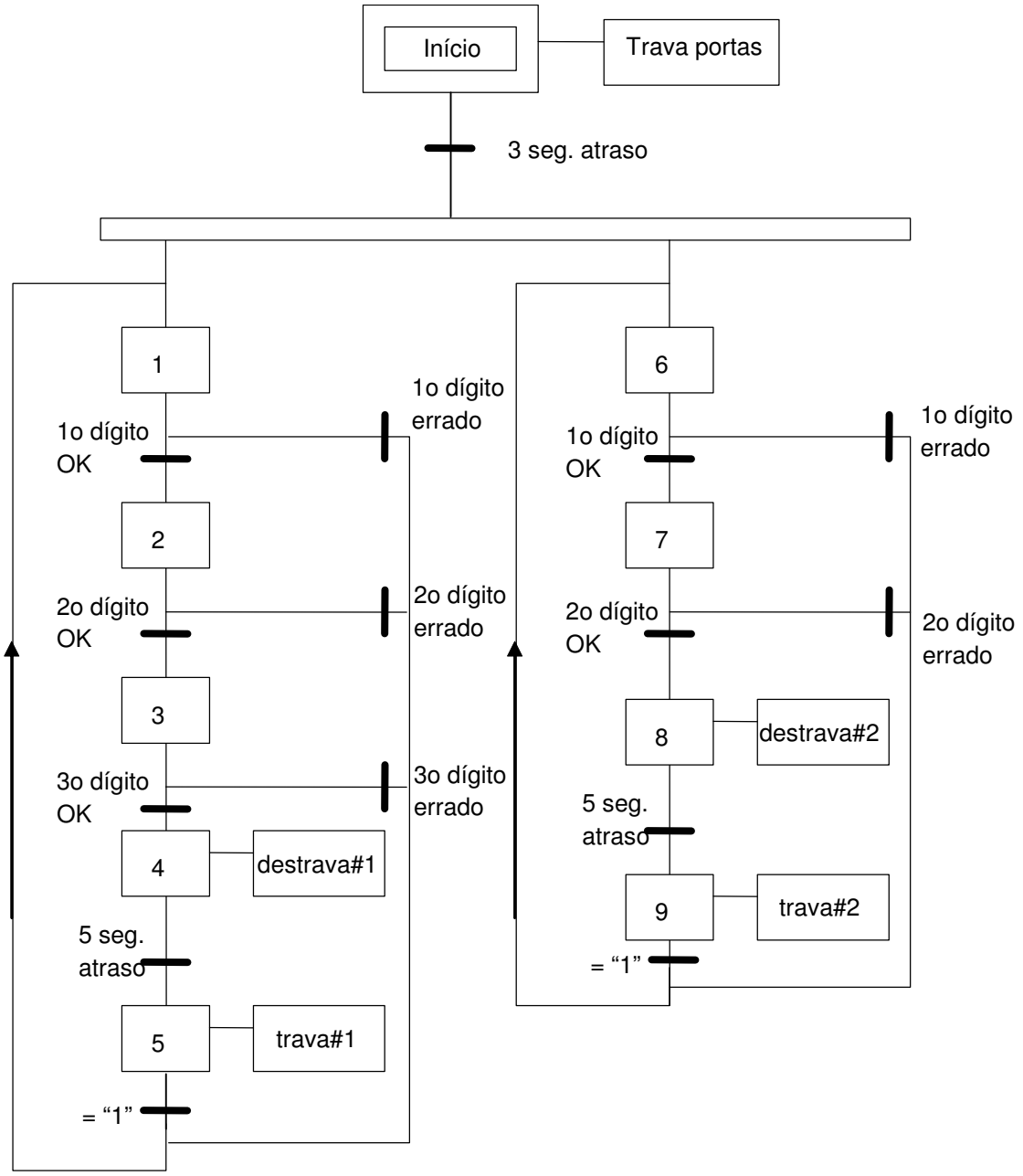
- Rendezvous:



Exemplo 5.1. Controle de Acesso

O exemplo a seguir representa um SFC para controle de segurança de duas portas. Uma porta requer um código de acesso de dois dígitos, a segunda porta requer um código de três dígitos. A execução do sistema inicia-se no topo do diagrama, no passo inicial, quando o sistema é ligado. Existe uma ação associada ao passo inicial que trava a porta (Nota: na prática, em SFC se utiliza lógica ladder para acionamento de saídas e leitura de entradas, recurso não apresentado no diagrama).

Após a inicialização do diagrama, sua execução é dividida em dois processos e ambos os passos 1 e 6 são ativados. Quando uma condição lógica de digitação do código é satisfeita, a transição correspondente desativa o passo anterior e aciona o passo seguinte. Enquanto o passo 1 estiver ativo, duas transições podem ser disparadas. No caso da digitação correta do dígito, o passo 1 ficara inativo e o passo 2 ativo. Caso a digitação seja incorreta, então o a transição leva o sistema novamente para o passo 1. O passo 1 não tem nenhuma ação associada, portanto nada deve acontecer neste período. A lógica para ambas as portas deve se repetir após o final do ciclo de “combinação-abertura-atraso-travamento”.

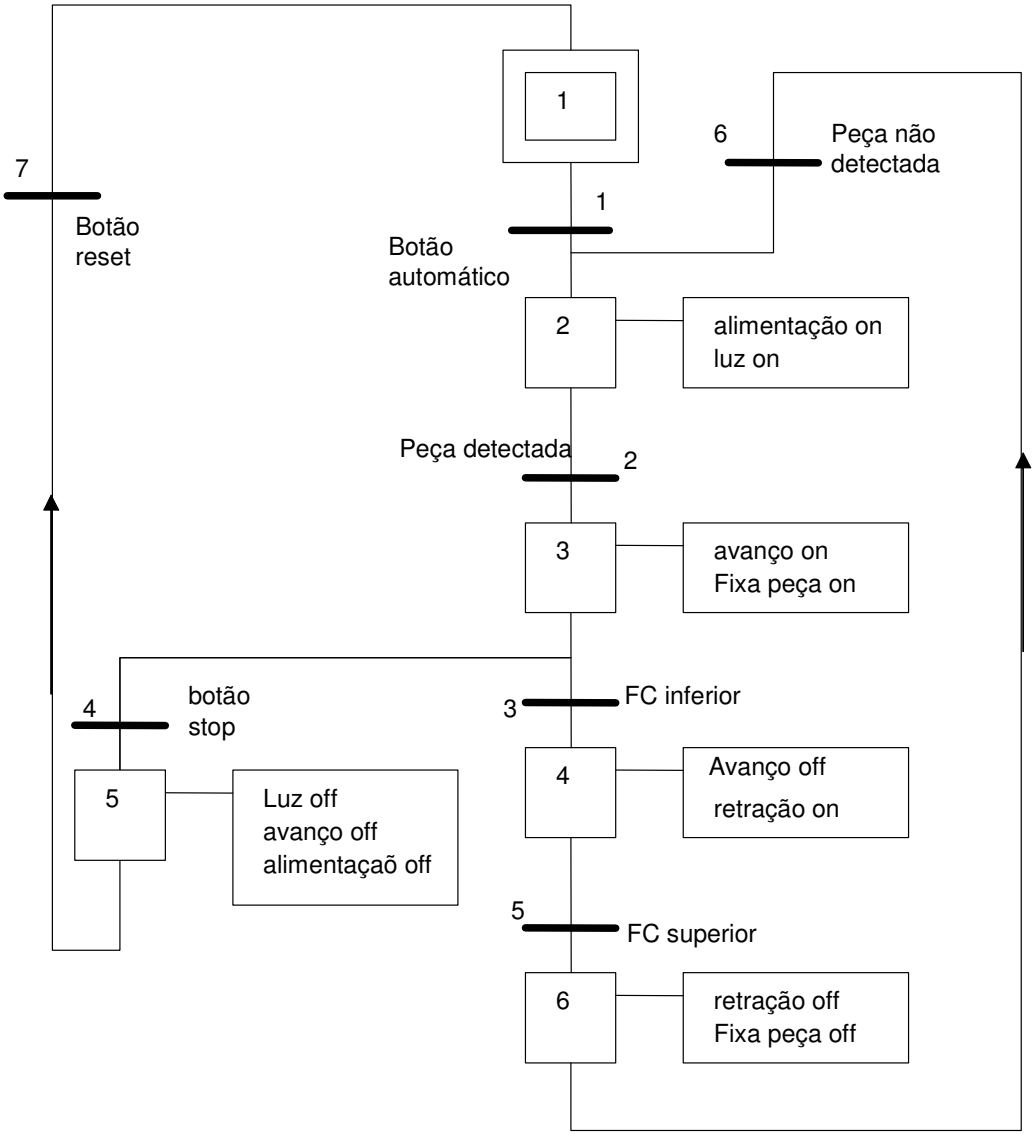


Exemplo 5.2. Controle de Prensa

Um SFC para o controle de uma prensa de estampagem é apresentado a seguir. Note que o processo é composto por uma única seqüência, portanto poderia ser facilmente implementado diretamente em ladder.

O processo da prensa se inicia em um estado ocioso, quando um botão "automático" é pressionado, a alimentação da prensa e uma luz de indicação são ligados. Frente à detecção de uma peça, o movimento é iniciado até a sensibilização de um fim de curso, então a prensa tem o movimento de retorno acionado até um fim

de curso superior ser acionado. Um botão de parada pode ser acionado somente durante o movimento de descida da prensa (note que este tipo de mecanismo de segurança pode ser ativo durante toda a operação). Quando o sistema é parado, deve-se pressionar um botão de reset antes de se iniciar novamente a operação automática. Após o passo 6, a prensa aguarda a não detecção da peça antes de aguardar por uma nova peça. Sem este mecanismo a prensa entraria em um ciclo interminável.



Exercícios

6.1. Crie um SFC para uma estação de trabalho para 2 pessoas. A estação tem duas prensas que podem ser utilizadas ao mesmo tempo. Cada prensa tem um botão de

ciclo que inicia o avanço. Um fim de curso inferior interrompe o avanço, e o cilindro retrai até o acionamento de um fim de curso superior.

6.2. Crie um SFC para o controle de um semáforo de trânsito. Devem-se considerar botões para a solicitação de travessia de pedestres em ambos os sentidos. Uma seqüência normal de operação deve ser: Sinal verde por 16 segundos, Amarelo por 4 segundos e Vermelho por 16 segundos. Se um dos botões de pedestres for acionado, uma luz de travessia de pedestre deve ser acesa por 10 segundos e o sinal verde estendido por 24 segundos.

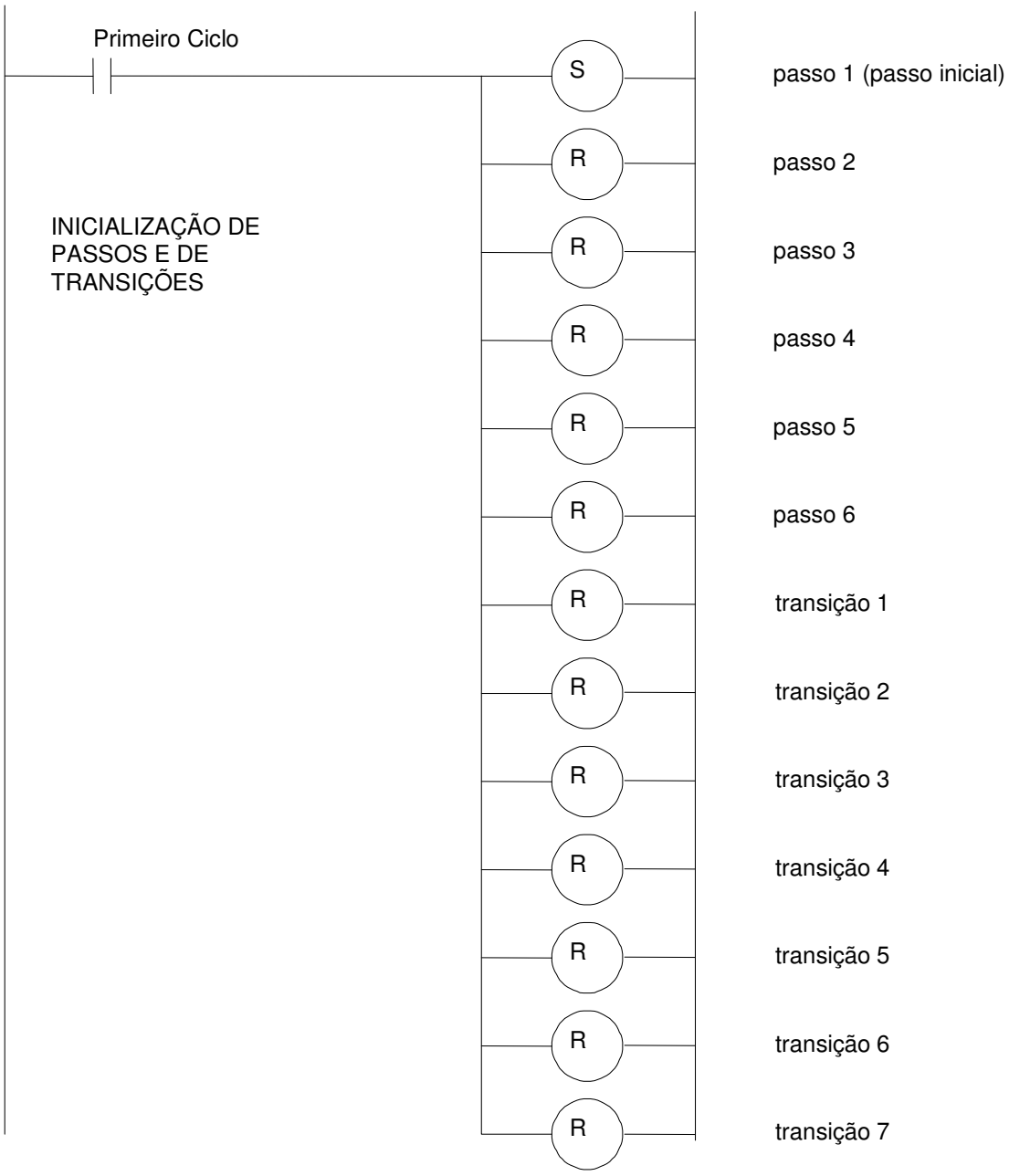
6.3. Desenhe um SFC para o processo de estampagem onde o avanço e a retração são disparados por um botão de ciclo único.

6.4. Projete o controle de um portão automático de garagem:
- existe um único botão na garagem e outro no controle remoto.
- quando o botão é pressionado, o portão abre ou fecha.
- se o botão é pressionado durante o movimento, o portão pára. Um Segundo acionamento do botão inicia o movimento no sentido contrário.
- existem fins de curso nas duas extremidades do movimento do portão.
- existe uma barreia ótica no curso do portão, que interrompe o fechamento e inicia a abertura quando o sinal ótico é bloqueado.
- existe uma sinalização luminosa que se ativa por 1 minuto após a abertura ou o fechamento do portão.

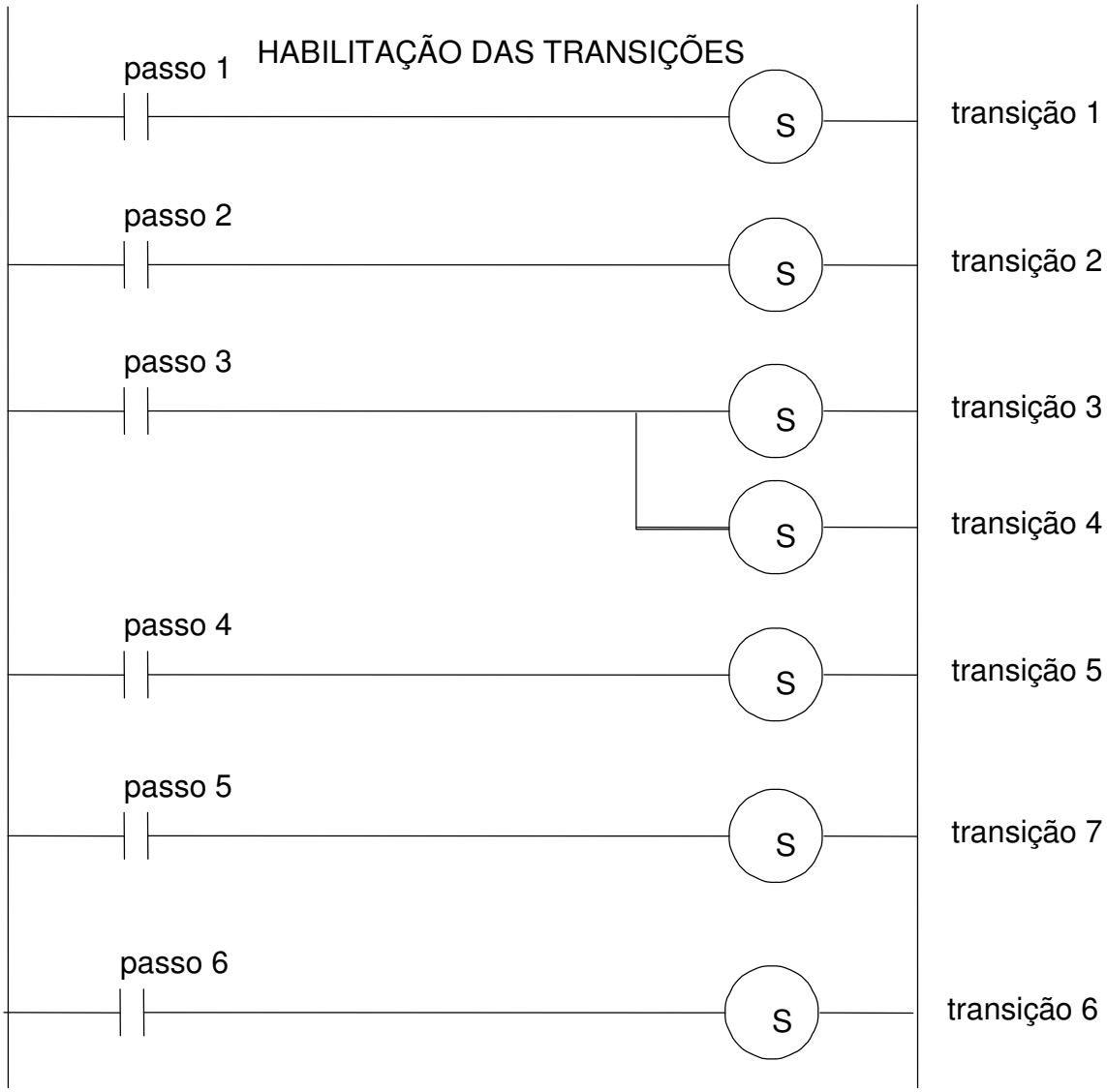
Conversão de diagramas SFC em linguagem Ladder

Um diagrama SFC pode ser convertido diretamente em ladder de algumas formas distintas. O método aqui apresentado deve ser utilizado em seqüência à elaboração do diagrama SFC. O programa ladder começa com uma seção de inicialização de passos e transições para valores únicos. Em uma segunda seção da lógica ladder ativam-se as transições subseqüentes aos passos. Em uma terceira seção, caso determinada condição de transição seja satisfeita, então o próximo passo é ativado e o passo precedente, bem como a própria transição em questão, são desativados. Segue-se a lógica ladder com a codificação da ação associada a cada passo, com as ativações de saídas e lógicas locais em uma última seção.

A seguir, demonstra-se a codificação em ladder do exemplo 6.2. Controle de Prensa.

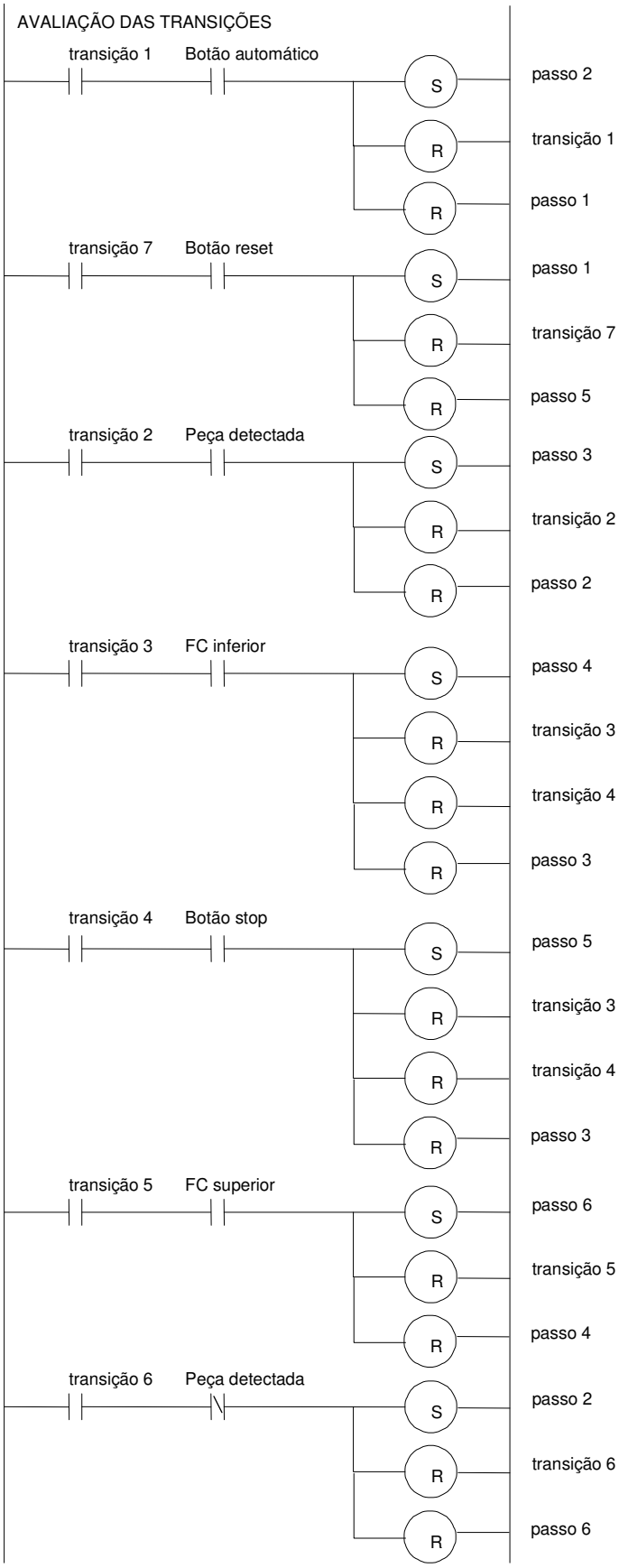


Nesta primeira seção do ladder, inicializam-se todas as variáveis relacionadas a passo e transições. Note que são seis passos e sete transições nomeados de forma única no programa. Deve-se sempre nomear passos e transições de forma única. Em geral os CLPs possuem uma variável de sistema que pode ser utilizada para a ativação das inicializações somente no primeiro ciclo de scan do programa. Caso o controlador não tenha este recurso, é necessária a utilização de uma lógica para realizar tal função.

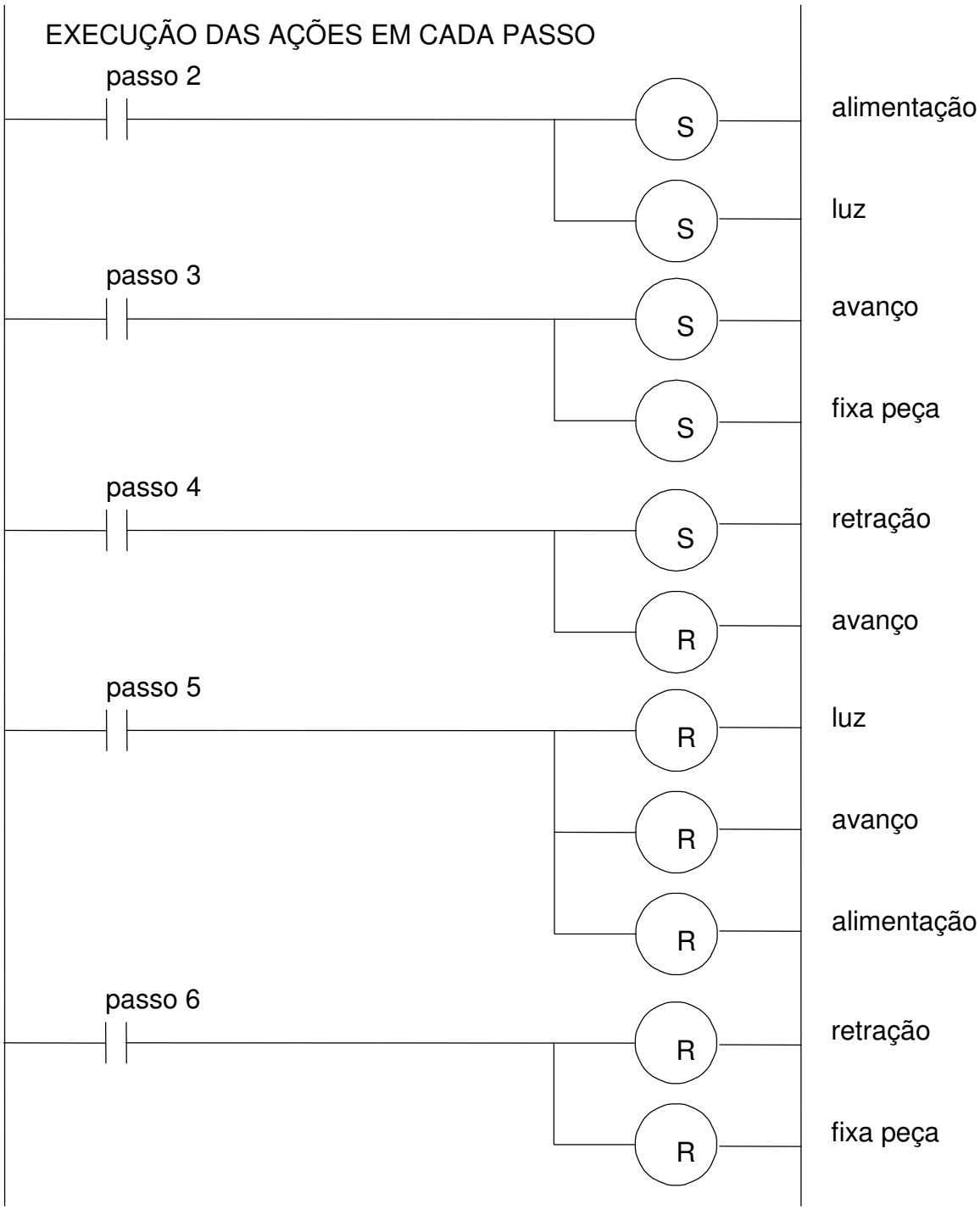


Na seção de habilitação das transições, deve-se reproduzir a estrutura do SFC em relação às seqüências passos-transições.

Na seção seguinte, cada transição deve ter sua condição de disparo avaliada e, caso a avaliação seja positiva, deve-se ativar os passos seguintes, desativar o passo anterior e desativar a própria transição em questão.


AVALIAÇÃO DAS TRANSIÇÕES
transição 1
Botão automático
S
passo 2
R
transição 1
R
passo 1
transição 7
Botão reset
S
passo 1
R
transição 7
R
passo 5
transição 2
Peça detectada
S
passo 3
R
transição 2
R
passo 2
transição 3
FC inferior
S
passo 4
R
transição 3
R
transição 4
R
passo 3
transição 4
Botão stop
S
passo 5
R
transição 3
R
transição 4
R
passo 3
transição 5
FC superior
S
passo 6
R
transição 5
R
passo 4
transição 6
Peça detectada
S
passo 2
R
transição 6
R
passo 6

Por fim, executam-se as ações atribuídas a cada passo. Note que nesta seção do diagrama ladder o passo 1 não foi considerado por não ter nenhuma ação associada.



Alguns CLPs permitem a programação gráfica diretamente do SFC. Mesmo assim, códigos em ladder, lista de instruções, blocos funcionais ou texto estruturado

devem ser programados para cada transição ou passo. Cada código então deve ser separado em um programa distinto nesta lógica.

Exercícios

5.5. a 5.9. Codifique em ladder os SFC´s projetados nos exercícios 5.1 a 5.4 e no exemplo 5.1 respectivamente.