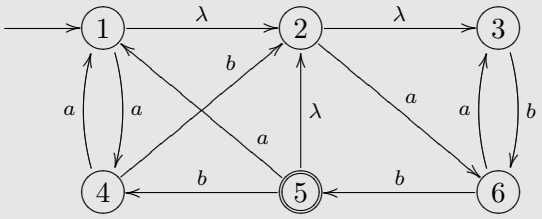
Teoria da Computação

Exercícios resolvidos

Luís Monteiro

Lic. Eng. Informática, 2007/08

Exercício 1 Construa um AFD equivalente ao seguinte AFND:



(Exame de Recurso de 2005/06)

A construção envolve três passos. O primeiro passo consiste em determinar, para cada estado q e cada símbolo a do alfabeto de entrada, todos os estados q' atingíveis a partir de q "consumindo" apenas a. Isto quer dizer que se percorre exactamente uma transição $\xrightarrow{a}$ e um número arbitrário, possivelmente zero, de transições $\xrightarrow{\lambda}$, antes e depois da transição com a, como se ilustra a seguir:

$$q \xrightarrow{\lambda} \cdots \xrightarrow{\lambda} \xrightarrow{a} \xrightarrow{\lambda} \cdots \xrightarrow{\lambda} q'.$$

Esta informação é coligida na forma de uma tabela. As linhas são indexadas pelos estados, as colunas pelos símbolos de entrada e na célula de linha q e coluna a colocam-se os estados q' a que se pode chegar a partir de q consumindo apenas a. Para o AFND dado obtém-se a seguinte tabela, dita auxiliar:

	a	b
1	$4, 6$	6
2	6	6
3		6
4	$1, 2, 3$	$2, 3$
5	$1, 2, 3, 6$	$4, 6$
6	3	$2, 3, 5$

O passo seguinte consiste em determinar o estado inicial do AFD equivalente. Os estados deste AFD são conjuntos de estados do AFND, e assim sucede com o seu estado inicial. Ele é constituído pelo estado inicial do AFND bem como por todos os estados a que se pode chegar a partir dele por transições $\xrightarrow{\lambda}$. No caso presente, o estado inicial do AFND é 1 e as cadeias de transições $\xrightarrow{\lambda}$ que nele se iniciam são $1 \xrightarrow{\lambda} 2 \xrightarrow{\lambda} 3$. Assim, o estado inicial do AFD é

$$\{1, 2, 3\}.$$

No terceiro e último passo constroem-se em simultâneo os restantes estados do AFD e a tabela da função de transição. A transição com a a partir do estado inicial $\{1, 2, 3\}$ calcula-se do seguinte modo: juntam-se todos os elementos que se encontram na coluna a e linhas $1, 2, 3$ da tabela auxiliar, resultando no estado $\{4, 6\}$. Procedendo de igual modo para a transição com b, obtinha-se o estado $\{6\}$. Repetindo o processo para os dois novos estados, calculavam-se as transições a partir deles, e depois a partir dos novos estados que se obtivessem, e assim sucessivamente, até não se criarem estados novos e a tabela ficar completamente preenchida. Mostra-se a seguir tabela completa, onde $\{\}$ representa o conjunto vazio:

	a	b
$> \{1, 2, 3\}$	$\{4, 6\}$	$\{6\}$
$\{4, 6\}$	$\{1, 2, 3\}$	$\{2, 3, 5\}$
$\{6\}$	$\{3\}$	$\{2, 3, 5\}$
$\bullet\{2, 3, 5\}$	$\{1, 2, 3, 6\}$	$\{4, 6\}$
$\{3\}$	$\{\}$	$\{6\}$
$\{1, 2, 3, 6\}$	$\{3, 4, 6\}$	$\{2, 3, 5, 6\}$
$\{\}$	$\{\}$	$\{\}$
$\{3, 4, 6\}$	$\{1, 2, 3\}$	$\{2, 3, 5, 6\}$
$\bullet\{2, 3, 5, 6\}$	$\{1, 2, 3, 6\}$	$\{2, 3, 4, 5, 6\}$
$\bullet\{2, 3, 4, 5, 6\}$	$\{1, 2, 3, 6\}$	$\{2, 3, 4, 5, 6\}$

Para completar a construção do AFD assinalaram-se os estados de aceitação, que são os conjuntos que contêm o estado de aceitação, 5, do AFND.

Exercício 2 Usando o algoritmo por aproximações sucessivas, construa o autómato reduzido do AFD $\mathcal{A} = (\{a, b\}, \{1, 2, 3, 4, 5, 6\}, 1, \delta, \{1, 4, 6\})$ em que a função de transição δ é dada pela seguinte tabela:

δ	a	b
1	1	4
2	4	5
3	6	5
4	2	6
5	1	3
6	3	6

(Exame Final de 2005/06)

Para construir o autómato reduzido tem de se determinar a relação $\equiv$ de equivalência entre estados. A determinação faz-se calculando sucessivas relações $\equiv_0$, $\equiv_1$, $\equiv_2$ e assim sucessivamente, até que $\equiv_{n+1}=\equiv_n$ para algum n. Quando isso suceder, o cálculo das relações $\equiv_i$ termina e tem-se $\equiv=\equiv_n$. Neste exercício obtem-se $\equiv_3=\equiv_2$.

A relação $\equiv_0$ agrupa os estados em duas classes: por um lado, a classe dos estados de aceitação, e por outro, a dos estados que não são de aceitação. Vamos chamar a essas classes A e B.[1] O passo seguinte consiste em preencher uma tabela baseada na tabela que define a função de transição do AFD dado, excepto que para cada estado se indica, não o estado para o qual ele transita, mas a classe do estado para o qual ele transita. Obtem-se a seguinte tabela:

$\equiv_0$	Q	a	b
A	1	A	A
	4	B	A
	6	B	A
B	2	A	B
	3	A	B
	5	A	B

No passo seguinte sub-dividem-se as classes, colocando em classes distintas estados que diferem nas classes para as quais transitam e agrupando numa mesma classe estados com iguais transições. Na classe A, os estados 4 e 6 transitam para as mesmas classes, mas o mesmo não se passa com o estado 1. A classe A subdivide-se em duas classes, uma contendo apenas o estado 1 e a outra os estados 4 e 6. A classe B não dá lugar a sub-divisões. Estas classes são nomeadas A, B e C, respectivamente, nomes que só são válidos para a próxima tabela e que não têm nada a ver com os nomes das

[1] Mais precisamente, $q \equiv_0 q'$ se, e só se, $q, q' \in A$ ou $q, q' \in B$. As relações $\equiv_i$ que vamos encontrar a seguir definem-se de um modo análogo a partir de um conjunto de classes.

classes do passo anterior. A tabela com as novas classes obtem-se como anteriormente, indicando para cada estado as novas classes dos estados para os quais ele transita:

$\equiv_1$	Q	a	b
A	1	A	B
B	4	C	B
	6	C	B
C	2	B	C
	3	B	C
	5	A	C

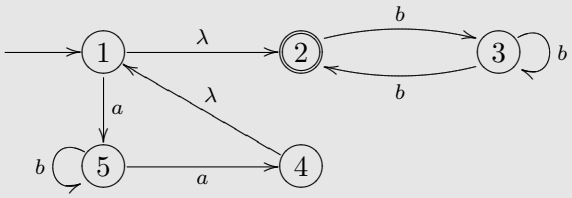
Neste passo é a classe C que se sub-divide em duas, uma com os estados 2 e 3 e a outra com o estado 5. A próxima tabela, obtida pelo mesmo processo anterior, é a seguinte:

$\equiv_2$	Q	a	b
A	1	A	B
B	4	C	B
	6	C	B
C	2	B	D
	3	B	D
D	5	A	C

Este passo não dá lugar a sub-divisão de classes, o que significa que a relação $\equiv_3$ seria igual a $\equiv_2$ e portanto a $\equiv$. O autómato reduzido tem as classes de $\equiv$ como estados, que são as classes usadas na última tabela. A sua tabela da função de transição resulta da última tabela eliminando linhas repetidas. O estado inicial é a classe A, que contém o estado inicial do autómato dado, e os estados de aceitação são as classes A e B, que contêm os estados de aceitação do autómato dado. O autómato reduzido é então o seguinte:

	a	b
$> \bullet A$	A	B
$\bullet B$	C	B
C	B	D
D	A	C

Exercício 3 Determine uma expressão regular para a linguagem reconhecida pelo seguinte AFND, resolvendo o sistema de equações lineares direitas que lhe está associado:



(Exame Final de 2005/06)

O sistema de equações associado a este AFND é o seguinte:

$$\begin{cases} X_1 = X_2 + aX_5 \\ X_2 = \lambda + bX_3 \\ X_3 = bX_2 + bX_3 \\ X_4 = X_1 \\ X_5 = aX_4 + bX_5 \ . \end{cases}$$

A resolução do sistema assenta nas seguintes observações:

- Só se pretende resolver o sistema até se encontrar a solução para X_1, que é a linguagem pedida no enunciado. Para isso precisamos de substituir X_2 e X_5 na equação de X_1 por expressões que envolvam no máximo a incógnita X_1.
- As incógnitas X_2 e X_3 só dependem uma da outra e não de qualquer outra incógnita. Este sub-sistema pode pois ser resolvido independentemente. Aqui só nos vai interessar a solução para X_2, visto que X_3 não ocorre em nenhuma outra equação.
- A expressão para X_5 em função de X_1 determina-se facilmente substituindo primeiro X_4 por X_1.

Podemos passar à resolução:

$$\begin{cases} \_\_\_\_\_ \\ \_\_\_\_\_ \\ X_3 = b^*bX_2 \\ \_\_\_\_\_ \\ X_5 = aX_1 + bX_5 \end{cases} \qquad \begin{cases} \_\_\_\_\_ \\ X_2 = \lambda + bb^*bX_2 \\ \_\_\_\_\_ \\ \_\_\_\_\_ \\ X_5 = b^*aX_1 \end{cases} \qquad \begin{cases} \_\_\_\_\_ \\ X_2 = (bb^*b)^* \\ \_\_\_\_\_ \\ \_\_\_\_\_ \\ X_5 = b^*aX_1 \end{cases}$$

$$\begin{cases} X_1 = (bb^*b)^* + ab^*aX_1 \\ \_\_\_\_\_ \\ \_\_\_\_\_ \\ \_\_\_\_\_ \\ \_\_\_\_\_ \end{cases} \qquad \begin{cases} X_1 = (ab^*a)^*(bb^*b)^* \\ \_\_\_\_\_ \\ \_\_\_\_\_ \\ \_\_\_\_\_ \\ \_\_\_\_\_ \end{cases}$$

Exercício 4 Seja $\mathcal{A} = (\{a, b\}, \{1, 2, 3, 4, 5, 6\}, 1, \delta, \{6\})$ um autómato finito não-determinista, em que $\delta : \{1, 2, 3, 4, 5, 6\} \times \{a, b, \lambda\} \to \wp(\{1, 2, 3, 4, 5, 6\})$ é definida pela tabela

δ	a	b	λ
1	$\{4\}$	$\{5\}$	$\{2\}$
2	$\{3\}$	$\emptyset$	$\emptyset$
3	$\{1\}$	$\{6\}$	$\{5\}$
4	$\emptyset$	$\{5\}$	$\emptyset$
5	$\emptyset$	$\{2\}$	$\emptyset$
6	$\{5\}$	$\emptyset$	$\emptyset$

1. Determine uma expressão regular para a linguagem reconhecida por $\mathcal{A}$.
2. Defina um autómato finito determinista $\mathcal{A}'$ equivalente a $\mathcal{A}$, aplicando o algoritmo dado.

(Exame de Recurso de 2006/07)

Será apresentada apenas a resolução da primeira alínea.
Sistema de equações:

$$\begin{cases} X_1 = X_2 + aX_4 + bX_5 \\ X_2 = aX_3 \\ X_3 = aX_1 + X_5 + bX_6 \\ X_4 = bX_5 \\ X_5 = bX_2 \\ X_6 = aX_5 + \lambda. \end{cases}$$

Passos da resolução do sistema:

- As equações mais simples são as de X_2, X_4 e X_5, que têm um termo cada no membro direito, e a de X_6, que tem dois. É fácil exprimir todas estas incógnitas em termos de X_3.
- A seguir exprimem-se, primeiro X_1 e depois X_3, em termos de X_3.
- Resolve-se então a equação de X_3 e substitui-se a solução na equação de X_1.

Resolução:

$$\left\{\begin{array}{l} \\ X_2 = aX_3 \\ \\ X_4 = bX_5 = bbaX_3 \\ X_5 = bX_2 = baX_3 \\ X_6 = aX_5 + \lambda = abaX_3 + \lambda \end{array}\right.$$

$$\left\{\begin{array}{l} X_1 = X_2 + aX_4 + bX_5 \\ \quad = (a + abba + bba)X_3 \\ \\ X_3 = aX_1 + X_5 + bX_6 \\ \quad = (aa + aabba + abba + ba + baba)X_3 + b \\ \quad = (aa + aabba + abba + ba + baba)^*b \end{array}\right.$$

$$\left\{\begin{array}{l} X_1 = (a + abba + bba)(aa + aabba + abba + ba + baba)^*b \\ \\ \end{array}\right.$$

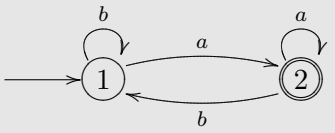
Exercício 5 Dado um alfabeto T, $a \in T$ e $L \subseteq T^*$, define-se a linguagem

$$\text{Impar}_a(L) = \{x \in L \mid \text{as posições ímpares de } x \text{ contêm } a\}.$$

Por exemplo,

$$\text{Impar}_a(\{\lambda, a, b, aa, ab, ba, bb, aaa, aab\}) = \{\lambda, a, aa, ab, aaa\}.$$

1. A partir do seguinte autómato finito determinista, que reconhece uma linguagem M que não importa precisar, crie um novo autómato finito determinista que reconheça $\text{Impar}_a(M)$:



2. Seja $\mathcal{A} = (T, Q, q_I, \delta, F)$ um autómato finito determinista e $\mathcal{L}(\mathcal{A})$ a linguagem por ele reconhecida. Caracterize um autómato finito determinista $\mathcal{A}'$ tal que $\mathcal{L}(\mathcal{A}') = \text{Impar}_a(\mathcal{L}(\mathcal{A}))$.

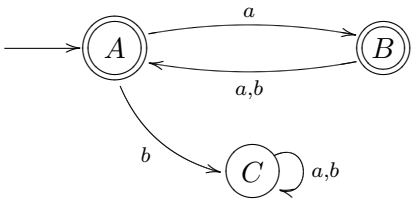
(Exame da 1ª Chamada de 2001/02)

A resolução de ambas as alíneas baseia-se na observação de que

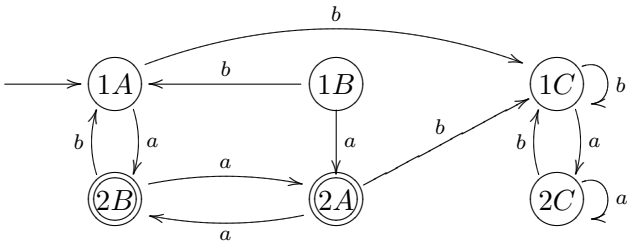
$$\text{Impar}_a(L) = \text{Impar}_a(T^*) \cap L.$$

A afirmação $x \in \text{Impar}_a(T^*)$ diz apenas que as posições ímpares de x contêm a. Quando esta afirmação se conjuga com $x \in L$, fica equivalente a $x \in \text{Impar}_a(L)$. Assim, para construir um autómato que reconheça $\text{Impar}_a(L)$ dado um autómato que reconhece L, basta definir um autómato que reconhece $\text{Impar}_a(T^*)$ e compor os dois autómatos para se obter o autómato que reconhece a intersecção das respectivas linguagens.

O AFD $\mathcal{A}_0 = (T, Q_0, q_{0I}, \delta_0, F_0)$ que reconhece $\text{Impar}_a(T^*)$ é o seguinte:



1. Nesta alínea temos de compor o autómato dado com o autómato $\mathcal{A}_0$ atrás definido. Os seus estados são os pares formados por um estado do primeiro autómato e um estado do segundo, aqui representados na notação simplificada $1A, 1B, 1C, 2A, 2B, 2C$. O estado inicial é aquele em que as duas componentes são os estados iniciais dos respectivos autómatos: $1A$. Os estados de aceitação têm nas duas componentes estados de aceitação dos respectivos autómatos: $2A, 2B$. A função de transição calcula-se componente a componente. Obtém-se o seguinte autómato:



   Observação: A resolução desta alínea termina aqui. No entanto, convém observar que o autómato pode ser simplificado. Por um lado, o estado $1B$ pode ser omitido, por ser inacessível. Por outro lado, os estados $1C$ e $2C$ são equivalentes — a partir de qualquer deles não se reconhece nenhuma palavra — e por isso podem ser fundidos num só.

2. O autómato $\mathcal{A}' = (T, Q', q'_I, \delta', F')$ tem a seguinte definição:

   - $Q' = Q \times \{A, B, C\} = \{(q, X) : q \in Q, X \in \{A, B, C\}\}$.
   - $q'_I = (q_I, A)$.
   - $F' = F \times \{A, B\} = \{(q, X) : q \in F, X \in \{A, B\}\}$.
   - Para todo o $q \in Q$, $X \in \{A, B, C\}$ e $a \in T$, pondo $\delta(q, a) = q'$ em $\mathcal{A}$ e $\delta_0(X, a) = X'$ em $\mathcal{A}_0$, define-se

     $$\delta'((q, X), a) = (q', X').$$

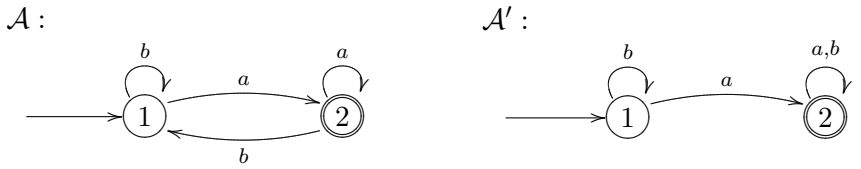
Exercício 6 Seja $\mathcal{A} = (T, Q, q_I, \delta, F)$ um autómato finito determinista. Sendo $\mathcal{L}(\mathcal{A})$ a linguagem reconhecida por $\mathcal{A}$, seja

$$L = \{xy : x \in \mathcal{L}(\mathcal{A}) \text{ e } y \in T^*\} = \mathcal{L}(\mathcal{A})T^*.$$

Caracterize um autómato finito determinista $\mathcal{A}' = (T, Q', q_I', \delta', F')$ que reconheça L, isto é, tal que $\mathcal{L}(\mathcal{A}') = \mathcal{L}(\mathcal{A})T^*$.

(Exame de Recurso de 2006/07)

Antes de resolver o exerício, vejamos um exemplo. Mostram-se a seguir um autómato $\mathcal{A}$ e um autómato $\mathcal{A}'$ que reconhece a linguagem $\mathcal{L}(\mathcal{A})T^*$.



O que caracteriza a linguagem $\mathcal{L}(\mathcal{A})T^*$ é que ela contém todas as extensões das palavras de $\mathcal{L}(\mathcal{A})$. Em termos do funcionamento dos autómatos, podemos dizer que mal uma palavra force $\mathcal{A}$ a entrar num estado de aceitação, $\mathcal{A}'$ aceita tudo o que vier a seguir. A maneira mais simples de garantir isso é fazer com que $\mathcal{A}'$ nunca saia de um estado de aceitação assim que entre nele. Foi essa a solução adoptada no exemplo anterior e a que será usada na resolução do exercício.

Resolução:

Pela discussão precedente, podemos definir $\mathcal{A}'$ alterando apenas as transições a partir de estados de aceitação para que terminem neles próprios. A caracterização de $\mathcal{A}'$ é então a seguinte:

- $Q' = Q$, $q_I' = q_I$ e $F' = F$.
- Para todo o $q \in Q$ e todo o $a \in T$,

$$\delta'(q, a) = \begin{cases} \delta(q, a) & \text{se } q \notin F, \\ q & \text{se } q \in F. \end{cases}$$

Exercício 7 Considere gramática

$$G = (\{;, \text{if}, \text{then}, a_1, \ldots, a_n\}, \{C, A\}, C, \mathcal{R}),$$

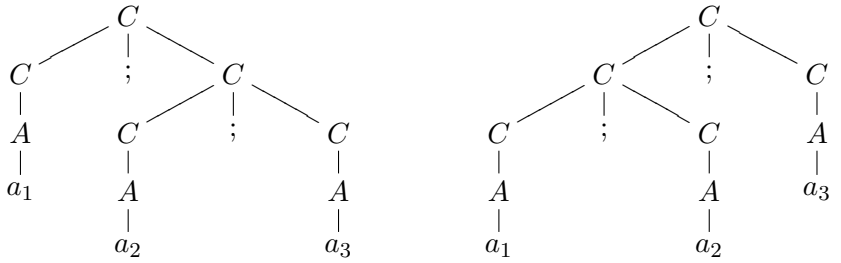
com $\mathcal{R}$ o seguinte conjunto de produções:

$$\begin{array}{l} C \longrightarrow A \mid C;C \mid \text{if } A \text{ then } C \\ A \longrightarrow a_1 \mid \cdots \mid a_n \end{array}$$

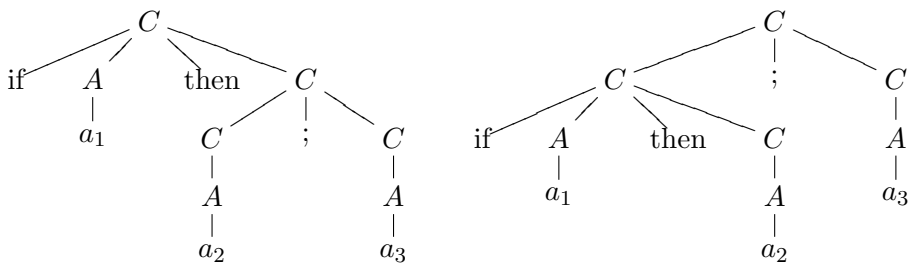
Mostre que a gramática é ambígua e defina uma gramática não-ambígua equivalente.

(Exame de Recurso de 2006/07)

Existem duas fontes de ambiguidade nas produções de C: a produção de corpo $C;C$, por si só, e a de corpo if A then C, em conjugação com a anterior. No primeiro caso, tomemos a palavra $a_1;a_2;a_3$. A produção $C \to C;C$ tem de ser usada duas vezes, uma por cada ';' que ocorre na palavra. A ambiguidade resulta de que a primeira ocorrência de ';' pode ser gerada quer pela primeira quer pela segunda aplicações da produção. Obtêm-se assim duas árvores de derivação distintas:



Isto já mostra que a gramática é ambígua. No entanto, com vista a remover completamente a ambiguidade, convém analisar também o segundo caso. Consideremos agora a palavra if a_1 then $a_2;a_3$. Neste caso as produções $C \to C;C$ e $C \to$ if A then C são usadas uma vez cada. O que se observa é que o "if" pode afectar tanto $a_2;a_3$, se a produção do 'if' for usada em primeiro lugar, como apenas a_2, se for usada em segundo. Eis as correspondentes árvores de derivação:



A ambiguidade remove-se escolhendo uma ordem de associação das várias ocorrências de ';' e estabelecendo uma ordem de prioridades entre ';' e 'if'.

É costume assumir que ';' associa à direita, de forma que $a_1; a_2; a_3$ representa implicitamente $a_1; (a_2; a_3)$. Quanto às prioridades, também é habitual assumir que o 'if' tem prioridade superior a ';', de forma que $\text{if}\, a_1 \,\text{then}\, a_2; a_3$ significa de facto $(\text{if}\, a_1 \,\text{then}\, a_2); a_3$.

Começando pelas prioridades, cria-se um novo símbolo não-terminal B que gera todas as expressões sem ocorrências de ';' e redefine-se C para gerar todas as sequências de B separadas por ';'. Na definição de C tem-se o cuidado de assegurar que o ';' gerado em primeiro lugar é o primeiro (o mais à esquerda) da palavra. A gramática modificada tem as produções:

$$\begin{array}{l} C \longrightarrow B \mid B; C \\ B \longrightarrow A \mid \text{if } A \text{ then } B \\ A \longrightarrow a_1 \mid \cdots \mid a_n \end{array}$$

Observação: O tipo de situação descrita nesta gramática é vulgar quando existe um operador binário e um unário, possivelmente entre outros. Por exemplo, numa gramática de expressões aritméticas pode ter-se $E \to A \mid E + E \mid -E \mid \cdots$, noutra de fórmulas lógicas, $F \to A \mid F \vee F \mid \forall V F \mid \cdots$, etc. O tratamento a dar a estes casos é semelhante.

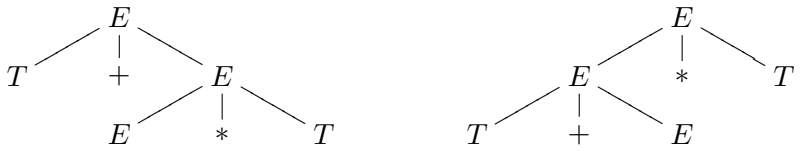
Exercício 8 Considere a gramática $\mathcal{G} = (\{+, *, (,), a\}, \{E, T\}, E, \mathcal{R})$, com $\mathcal{R}$ o seguinte conjunto de produções:

$$\begin{array}{l} E \longrightarrow T + E \mid E * T \mid T \\ T \longrightarrow (E) \mid a \end{array}$$

Mostre que $\mathcal{G}$ é ambígua e defina uma gramática não ambígua equivalente a $\mathcal{G}$.

(Teste 1-A de 2005/06)

A ambiguidade resulta de que as duas primeiras produções de E podem ser aplicadas por qualquer ordem. Por exemplo, se aplicadas uma vez cada, obtem-se sempre $T + E * T$. As árvores de derivação para este caso são:



Isto mostra que a gramática é ambígua. Remove-se a ambiguidade obrigando a que todas as aplicações de uma produção precedam as aplicações da outra. Por exemplo, se se pretendesse que a primeira produção fosse sempre aplicada antes da segunda, criava-se um novo símbolo não-terminal

A e alteravam-se as produções para:

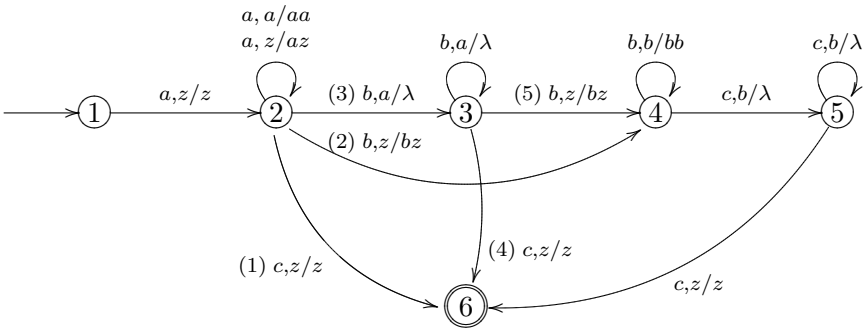
$$E \longrightarrow T + E \mid A$$
$$A \longrightarrow A * T \mid T$$
$$T \longrightarrow (E) \mid a$$

Exercício 9 Sejam $T = \{a, b, c\}$ e $L = \{a^{m+1}b^{m+n}c^{n+1} : m \geq 0, n \geq 0\}$.

1. Defina um autómato de pilha que reconheça a linguagem L.
2. Construa a árvore da linguagem L com suficiente pormenor para que se possa concluir que L não é uma linguagem regular, e indique explicitamente uma família infinita de sub-árvores distintas.

(Exame de Recurso de 2006/07)

1. No diagrama seguinte, o "eixo" horizontal corresponde ao reconhecimento de uma palavra $a^{m+1}b^{m+n}c^{n+1}$ com $m > 0$ e $n > 0$. O primeiro a lido não tem qualquer efeito sobre a pilha (supõe-se que o símbolo inicial da pilha é z). Cada um dos seguintes a's, em número de m, é lido e empilhado. Quando começam a ser lidos b's, os primeiros m desempilham o mesmo número de a's e os seguintes n empilham um b cada (a operação de empilhar b's começa quando está z no topo da pilha). Cada c lido desempilha então um b, enquanto isso for possível. Quando a pilha tiver z no topo, ainda é preciso ler o c final e aceitar. As restantes combinações possíveis de valores para m e n originam diferentes percursos no autómato. Para facilitar a compreensão, o início de cada um desses percursos encontra-se assinalado e é comentado mais abaixo.



As transições numeradas correspondem aos seguintes casos:

(1) $m = 0 \wedge n = 0$ (3) $m > 0$ (4) $m > 0 \wedge n = 0$
(2) $m = 0 \wedge n > 0$ (5) $m > 0 \wedge n > 0$

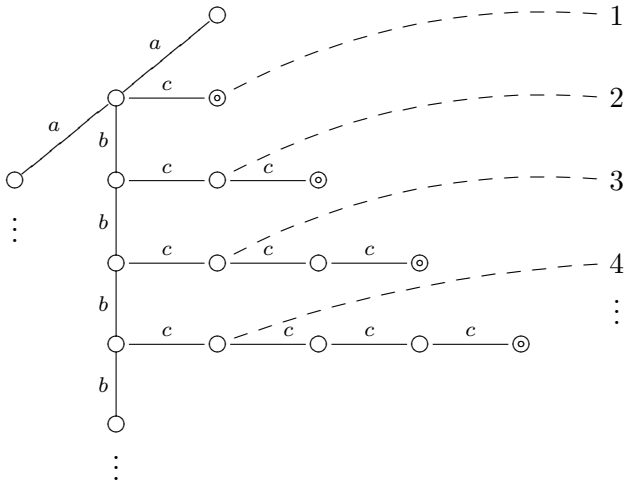
Observações:

- Nas transições (1) e (2) sabe-se que $m = 0$ porque, estando z no topo da pilha, nenhum a foi empilhado.
- Na transição (1) tem-se $n = 0$ porque c chega antes de vir algum b. Na transição (2) tem-se $n > 0$ porque vem um b e $m = 0$.
- Em (3) tem-se $m > 0$ porque foi empilhado pelo menos um a. Como esta transição é necessária para se efectuarem as transições (4) e (5), estas correspondem ambas ao caso em que $m > 0$.
- Em (4) tem-se $n = 0$ porque não chegou a ser empilhado nenhum b. Em (5) tem-se $n > 0$ porque se está a empilhar o primeiro b depois de se terem desempilhado os a's.

2. Para $m = 0$, fazendo variar $n = 0, 1, 2, 3, \ldots$ obtêm-se as palavras

$$\begin{array}{ll} ac & (= a^{0+1}b^{0+0}c^{0+1}) \\ abcc & (= a^{0+1}b^{0+1}c^{1+1}) \\ abbccc & (= a^{0+1}b^{0+2}c^{2+1}) \\ abbbcccc & (= a^{0+1}b^{0+3}c^{3+1}) \\ & \vdots \end{array}$$

Representando apenas esta parte da árvore, já se encontra um número infinito de sub-árvores não-isomorfas. Uma possível família de tais sub-árvores está indicada na figura que segue:



Exercício 10 Considere a seguinte linguagem sobre o alfabeto $\{a, b\}$:

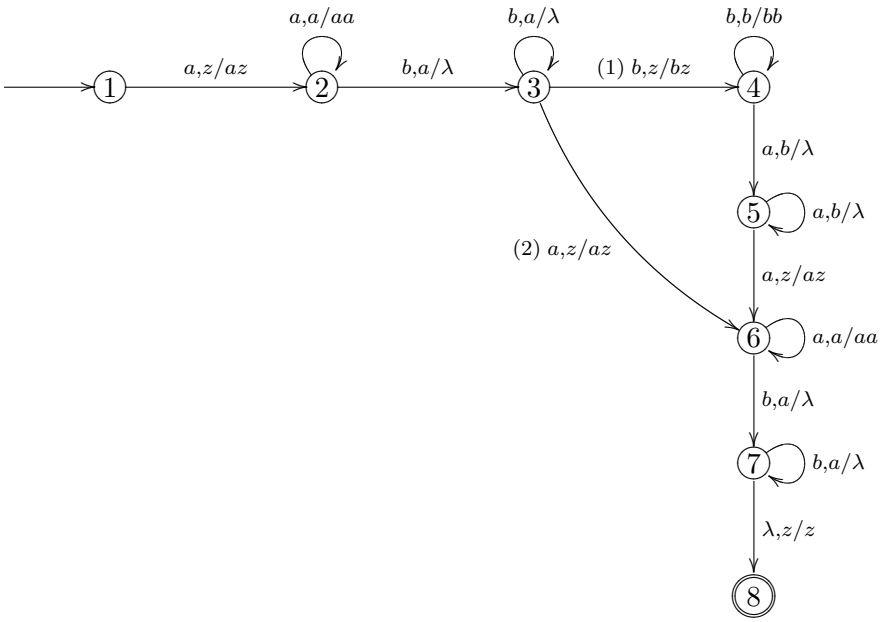
$$L = \{a^i b^j a^m b^n : i, j, m, n \geq 1, i + m = j + n, i \leq j, n \leq m\}.^a$$

1. Defina um autómato de pilha que reconheça a linguagem L.
2. Construa a árvore da linguagem L com suficiente pormenor para que se possa concluir que L não é uma linguagem regular, e indique explicitamente uma família infinita de sub-árvores distintas.

(Exame Final de 2005/06)

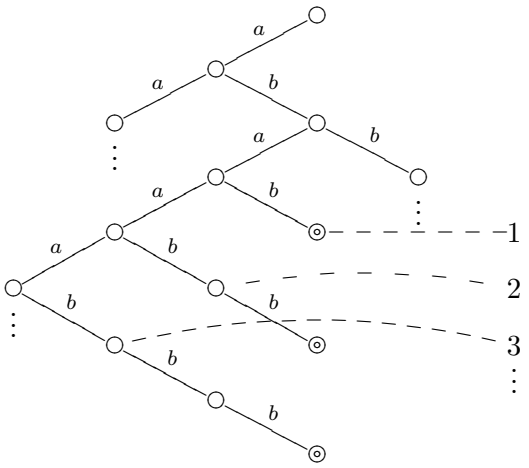
[a] A condição $n \leq m$ é redundante: $i \leq j \Longrightarrow i + n \leq j + n = i + m \Longrightarrow n \leq m$.

1. A pilha contabiliza a diferença entre os números de ocorrências de a e de b contendo as ocorrências em excesso. No início e no fim esses números devem ser iguais, pelo que a pilha conterá apenas o símbolo inicial z. A igualdade dos números de ocorrências verifica-se também durante o reconhecimento, por uma ou duas vezes consoante $j = i$ ou $j > i$, respectivamente. Estes dois casos encontram-se assinalados nas transições apropriadas do autómato:



A transição (1) corresponde a $j > i$ visto que, estando z no topo da pilha, os números de ocorrências de a e b até esse ponto foram iguais e está a ser lido um novo b. Na transição (2), como se está a ler a imediatamente a seguir a os números de ocorrências terem sido iguais, tem-se $j = i$.

2. Fixando $i = 1$ e $j = 1$, a condição $i + m = j + n$ implica $m = n$. A parte da árvore que contém as palavras $aba^n b^n$ para todo o $n \geq 0$ já apresenta um número infinito de sub-árvores não-isomorfas.



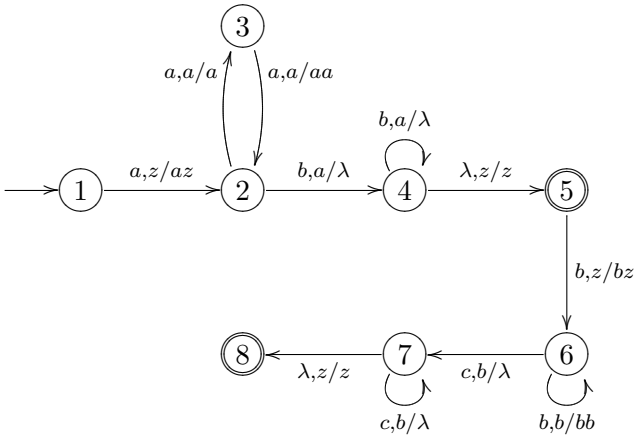
Exercício 11 Sejam $T = \{a, b, c\}$ e $L = \{a^{2n+1} b^{n+k+1} c^k : n \geq 0, k \geq 0\}$.

1. Defina um autómato de pilha que reconheça a linguagem L.
2. Construa a árvore da linguagem L com suficiente pormenor para que se possa concluir que L não é uma linguagem regular, e indique explicitamente uma família infinita de sub-árvores distintas.

(Exame de Recurso de 2007/08)

(Só se apresenta a resolução referente ao autómato de pilha.)

Escrevendo as palavras da linguagem na forma $a^{1+2n} b^{1+n} b^k c^k$, podemos pensar que o autómato começa por reconhecer as palavras $a^{1+2n} b^{1+n}$ e depois as palavras $b^k c^k$. Assim, empilha o a inicial seguido de um a por cada dois a's lidos e depois, por cada b lido, desempilha um a enquanto houver a's na pilha. Para o caso de ser $k = 0$, esta fase termina num estado de aceitação. Caso seja $k > 0$, empilham-se então os restantes b's e por cada a lido desempilha-se um b.



Exercício 12 Uma estrutura de apontadores E consiste num simples apontador P, que se considera a apontar para o vazio, ou num apontador P a apontar para uma célula C, o que se representa por $P : C$. Uma célula C tem um primeiro campo onde está guardado um valor V, seguido de uma lista F de um ou mais campos, cada um deles contendo uma estrutura de apontadores E. Uma célula é representada por VF; com um ponto-e-vírgula terminador. Uma gramática para estas estruturas de apontadores é

$$\mathcal{G} = (\{\$, :, ;, p_1, \ldots, p_n, v_1, \ldots, v_k\}, \{S, E, C, F, P, V\}, S, \mathcal{R})$$

com $\mathcal{R}$ o seguinte conjunto de produções:

$S \longrightarrow E\$$	% Símbolo inicial
$E \longrightarrow P \mid P : C$	% Estrutura de apontadores
$C \longrightarrow VF;$	% Célula
$F \longrightarrow E \mid FE$	% Campos
$P \longrightarrow p_1 \mid \cdots \mid p_n$	% Apontadores
$V \longrightarrow v_1 \mid \cdots \mid v_k$	% Valores

Defina uma gramática LL(1) equivalente a $\mathcal{G}$ e construa a respectiva tabela de análise sintáctica, utilizando os algoritmos de grafos para a determinação dos primeiros e seguintes.

(Exame Final de 2005/06)

Factorização: A factorização de P nas produções

$$E \longrightarrow P \mid P : C$$

resulta nas produções

$$E \longrightarrow PX$$
$$X \longrightarrow \lambda \mid : C$$

onde X é um novo símbolo não-terminal.

Eliminação da recursividade à esquerda: Nas produções

$$F \longrightarrow E \mid FE$$

o símbolo F é recursivo à esquerda. Eliminando a recursividade à esquerda obtêm-se as produções

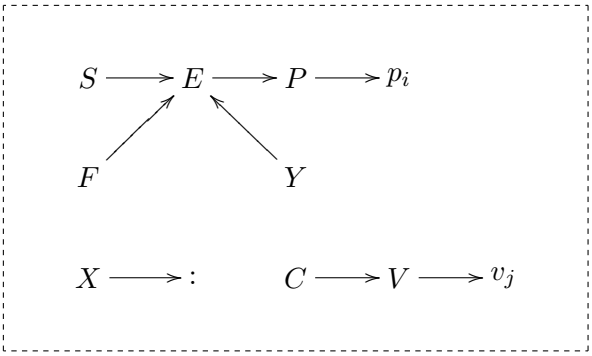
$$F \longrightarrow EY$$
$$Y \longrightarrow \lambda \mid EY$$

onde Y é um novo símbolo não-terminal.

Gramática modificada: As produções originais são substituídas pelas que resultaram da factorização e da eliminação da recursividade à esquerda:

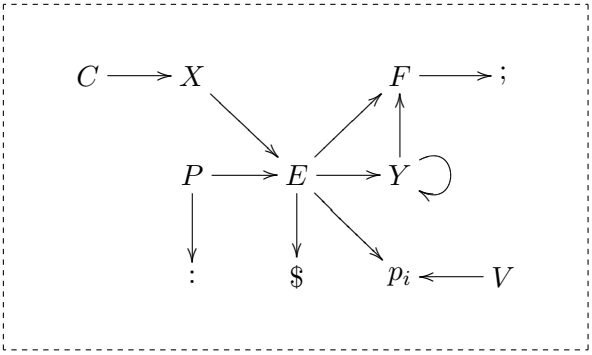
$$S \longrightarrow E\$$$
$$E \longrightarrow PX$$
$$X \longrightarrow \lambda \mid : C$$
$$C \longrightarrow VF;$$
$$F \longrightarrow EY$$
$$Y \longrightarrow \lambda \mid EY$$
$$P \longrightarrow p_1 \mid \cdots \mid p_n$$
$$V \longrightarrow v_1 \mid \cdots \mid v_k$$

Grafo dos primeiros:



Omitiram-se os nós '$' e ';' por estarem isolados, isto é, sem arcos incidentes, já que não têm qualquer repercussão na determinação dos primeiros. Por simplicidade representaram-se apenas elementos genéricos p_i e v_j. No que se segue, adopta-se o mesmo procedimento.

Grafo dos seguintes:



Símbolos directores:

Produções	Simb. directores
$S \longrightarrow E\$$	$\{p_1, \ldots, p_n\}$
$E \longrightarrow PX$	$\{p_1, \ldots, p_n\}$
$X \longrightarrow \lambda$	$\{\$, p_1, \ldots, p_n, ;\}$
$X \longrightarrow : C$	$\{:\}$
$C \longrightarrow VF;$	$\{v_1, \ldots, v_k\}$
$F \longrightarrow EY$	$\{p_1, \ldots, p_n\}$
$Y \longrightarrow \lambda$	$\{;\}$
$Y \longrightarrow EY$	$\{p_1, \ldots, p_n\}$
$P \longrightarrow p_i$	$\{p_i\}$
$V \longrightarrow v_j$	$\{v_j\}$

Nos únicos casos em que há duas produções com a mesma cabeça, os de X e Y, os respectivos conjuntos de símbolos directores são disjuntos, logo a gramática modificada é LL(1).

Tabela de análise sintáctica:

	$\$$	:	;	p_i	v_j
S				$E\$$	
E				PX	
X	λ	$: C$	λ	λ	
C					$VF;$
F				EY	
Y			λ	EY	
P				p_i	
V					v_j

Embora não seja pedido, ilustra-se o funcionamento do autómato de pilha reconhecedor com o reconhecimento da palavra $p_1 : v_1 p_2 ; \$$.

E	P	FL
q_L	S	$p_1 : v_1 p_2 ; \$$
q_{p_1}	S	$: v_1 p_2 ; \$$
q_{p_1}	$E\$$	$: v_1 p_2 ; \$$
q_{p_1}	$PX\$$	$: v_1 p_2 ; \$$
q_{p_1}	$p_1 X\$$	$: v_1 p_2 ; \$$
q_L	$X\$$	$: v_1 p_2 ; \$$
$q_:$	$X\$$	$v_1 p_2 ; \$$
$q_:$	$: C\$$	$v_1 p_2 ; \$$

E	P	FL
q_L	$C\$$	$v_1 p_2 ; \$$
q_{v_1}	$C\$$	$p_2 ; \$$
q_{v_1}	$VF ; \$$	$p_2 ; \$$
q_{v_1}	$v_1 F ; \$$	$p_2 ; \$$
q_L	$F ; \$$	$p_2 ; \$$
q_{p_2}	$F ; \$$	$; \$$
q_{p_2}	$EY ; \$$	$; \$$
q_{p_2}	$PXY ; \$$	$; \$$

E	P	FL
q_{p_2}	$p_2 XY ; \$$	$; \$$
q_L	$XY ; \$$	$; \$$
$q_;$	$XY ; \$$	$\$$
$q_;$	$Y ; \$$	$\$$
$q_;$	$; \$$	$\$$
q_L	$\$$	$\$$
$q_\$$	$\$$	λ
q_L	λ	λ

Exercício 13 Considere a gramática $\mathcal{G} = (\{e, p, ;\}, \{S, E, F\}, S, \mathcal{R})$, com $\mathcal{R}$ o seguinte conjunto de produções:

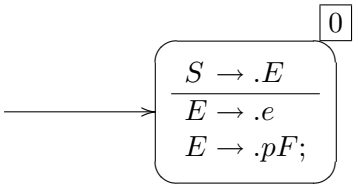
$$\begin{aligned} S &\longrightarrow E \\ E &\longrightarrow e \mid pF; \\ F &\longrightarrow E \mid EF \end{aligned}$$

Apresente o autómato finito determinista dos itens válidos LR(0) e justifique se a gramática é LR(0).

(Exame Final de 2005/06 — adaptado)

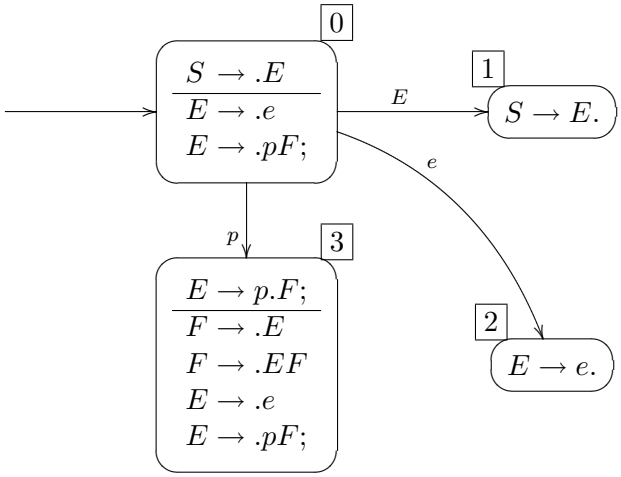
A construção do autómato dos itens LR(0) começa pelo seu estado inicial. Recorde-se que os estados do autómato são conjuntos de itens da gramática. O estado inicial obtém-se partindo dos itens iniciais do símbolo inicial da gramática e aplicando-lhes a operação de fecho.

No caso presente, $S \to .E$ é o único item inicial de S. A operação de fecho acrescenta-lhe os itens $E \to .e$ e $E \to .pF$; visto que à direita do ponto se encontra o símbolo não-terminal E. Estes itens têm símbolos terminais à direita do ponto, pelo que a operação de fecho fica concluída. O estado inicial do autómato, ao qual se atribuiu o número 0, é então o seguinte:



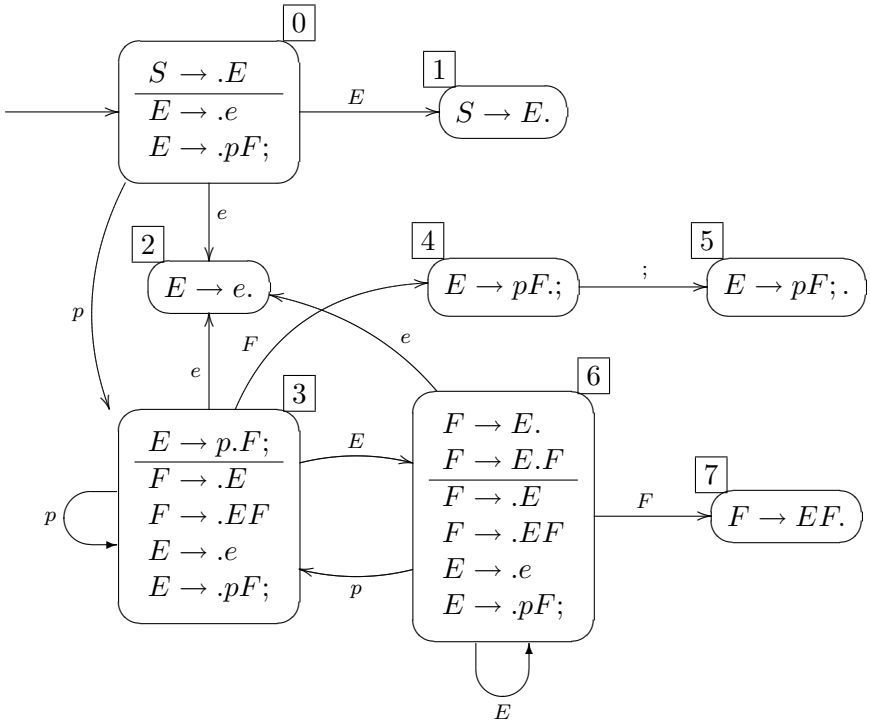
(A linha horizontal tem o efeito meramente visual de separar o item que iniciou a construção do estado dos que resultaram pela operação de fecho; nos outros estados utilizar-se-á o mesmo procedimento.)

Este estado tem transições etiquetadas por E, e e p, que são os símbolos que ocorrem à direita do ponto nos seus itens. Para cada transição, passa-se o ponto para a direita do símbolo em questão e aplica-se a operação de fecho, se necessário. Eis o estado inicial com as suas transições:



(Para os restantes símbolos da gramática, terminais ou não-terminais, existem transições para o estado formado pelo conjunto vazio de itens; tal estado será deixado implícito.)

Dos novos estados, apenas o 3 tem transições (para um conjunto não vazio de itens). Determinando as transições como anteriormente e iterando o processo obtem-se o autómato:



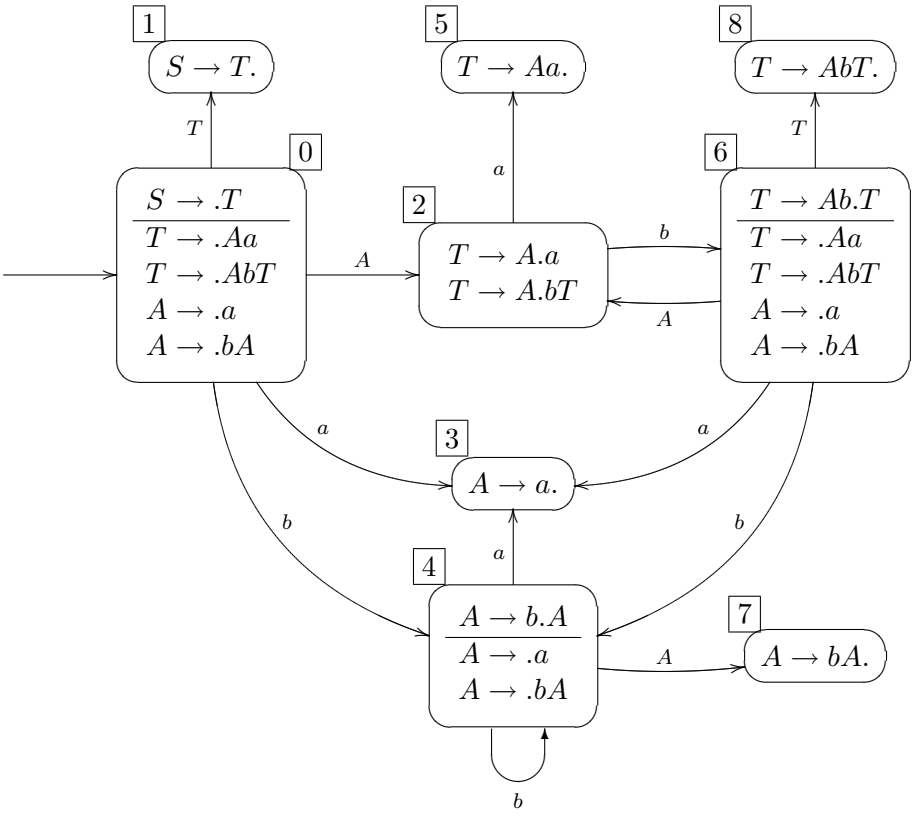
Vê-se que a gramática não é LR(0), porque no estado 6 existe o item completo $F \rightarrow E.$ juntamente com itens que têm um símbolo terminal à direita do ponto, nomeadamente, os itens $E \rightarrow .e$ e $E \rightarrow .pF$; (conflito redução/transferência).

Exercício 14 Considere a gramática $\mathcal{G} = (\{a, b\}, \{S, T, A\}, S, \mathcal{R})$, com $\mathcal{R}$ o seguinte conjunto de produções:

$$\begin{array}{l} S \longrightarrow T \\ T \longrightarrow Aa \mid AbT \\ A \longrightarrow a \mid bA \end{array}$$

Mostre que a gramática é LR(0), apresente a respectiva tabela das acções e ilustre o funcionmento do autómato de pilha reconhecedor com a palavra *babaa* .

Autómato dos itens válidos:



Como é habitual, omitiu-se o estado relativo ao conjunto vazio de itens.

Verificação das condições LR(0):

Nos estados em que aparecem itens completos (1,5,8,3,7), não há outros itens, logo as condições LR(0) são automaticamente verificadas. A gramática é portanto LR(0).

Tabela das acções:

	S	T	A	a	b	ACÇÕES
0		1	2	3	4	Transferência
1						Aceitação
2				5	6	Transferência
3						Redução por $A \to a$
4			7	3	4	Transferência
5						Redução por $T \to Aa$
6		8	2	3	4	Transferência
7						Redução por $A \to bA$
8						Redução por $T \to AbT$
9						Erro

Acrescentou-se o estado 9 relativo ao conjunto vazio de itens, ao qual está associada a acção de erro. As casas em branco na tabela são transições para o estado 9; não foram preenchidas para maior clareza.

Reconhecimento de *babaa*:

Pilha	Fita	Acção
0	*babaa*	Transferência
0*b*4	*abaa*	Transferência
0*b*4*a*3	*baa*	Red. $A \to a$
0*b*4*A*7	*baa*	Red. $A \to bA$
0*A*2	*baa*	Transferência
0*A*2*b*6	*aa*	Transferência

Pilha	Fita	Acção
0*A*2*b*6*a*3	*a*	Red. $A \to a$
0*A*2*b*6*A*2	*a*	Transferência
0*A*2*b*6*A*2*a*5	λ	Red. $T \to Aa$
0*A*2*b*6*T*8	λ	Red. $T \to AbT$
0*T*1	λ	Aceitação

Exercício 15 Dada a gramática de produções

$$\begin{aligned} S &\to A \\ A &\to Bb \mid Aa \\ B &\to c \mid AE \\ E &\to dA \mid \lambda \end{aligned}$$

determine o fecho LR(1) do conjunto formado pelo item $S \to .A, \{\$\}$.

O cálculo do fecho é feito numa sequência de passos.

1. Parte-se do item inicial:
$$S \to .A, \{\$\}$$

2. Aplicando a operação de fecho a este item, obtem-se:
$$\begin{array}{l} S \to .A \ , \ \{\underline{\$}\} \\ A \to .Bb, \ \{\$\} \\ A \to .Aa, \ \{\$\} \end{array}$$
No primeiro item sublinhou-se o símbolo de avanço \$ para indicar que se aplicou a operação de fecho ao item em questão quando os seus símbolos de avanço eram os que estão sublinhados (no caso, só o \$).

3. Aplica-se a operação de fecho ao primeiro item com símbolos de avanço não sublinhados, neste caso o $A \to .Bb, \{\$\}$. Obtém-se:
$$\begin{array}{l} S \to .A \ \ , \ \{\underline{\$}\} \\ A \to .Bb \ , \ \{\underline{\$}\} \\ A \to .Aa \ , \ \{\$\} \\ B \to .c \ \ \ , \ \{b\} \\ B \to .AE, \ \{b\} \end{array}$$
De novo se sublinhou o símbolo de avanço no item tratado neste passo.

4. Passa-se ao próximo item com símbolos de avanço não sublinhados, o $A \to .Aa, \{\$\}$, vindo:
$$\begin{array}{l} S \to .A \ \ , \ \{ \ \ \underline{\$}\} \\ A \to .Bb \ , \ \{\underline{\$}, a\} \\ A \to .Aa \ , \ \{\underline{\$}, a\} \\ B \to .c \ \ \ , \ \{ \ \ b\} \\ B \to .AE, \ \{ \ \ b\} \end{array}$$
Neste passo sublinhou-se o \$ como nos passos anteriores, mas acrescentou-se um a não sublinhado.

5. Usando a mesma regra de selecção (primeiro item com símbolos não sublinhados), escolhe-se $A \to .Bb, \{\underline{\$}, a\}$. Este item não acrescenta nada de novo, por isso limitamo-nos a sublinhar o a:
$$\begin{array}{l} S \to .A \ \ , \ \{ \ \ \underline{\$}\} \\ A \to .Bb \ , \ \{\underline{\$}, \underline{a}\} \\ A \to .Aa \ , \ \{\underline{\$}, a\} \\ B \to .c \ \ \ , \ \{ \ \ b\} \\ B \to .AE, \ \{ \ \ b\} \end{array}$$

6. O próximo item, $A \to .Aa, \{\underline{\$}, a\}$, também não acrescenta nada de novo:

$$\begin{array}{l} S \to .A \ \ , \{ \ \ \underline{\$}\} \\ A \to .Bb \ , \{\underline{\$}, \underline{a}\} \\ A \to .Aa \ , \{\underline{\$}, \underline{a}\} \\ B \to .c \ \ \ , \{ \ \ b\} \\ B \to .AE, \{ \ \ b\} \end{array}$$

7. O item $B \to .c, \{b\}$ tem um símbolo terminal à direita do ponto, portanto não produz novos itens:

$$\begin{array}{l} S \to .A \ \ , \{ \ \ \underline{\$}\} \\ A \to .Bb \ , \{\underline{\$}, \underline{a}\} \\ A \to .Aa \ , \{\underline{\$}, \underline{a}\} \\ B \to .c \ \ \ , \{ \ \ \underline{b}\} \\ B \to .AE, \{ \ \ b\} \end{array}$$

8. Para o item que falta, $B \to .AE, \{b\}$, há que ter em conta que E gera λ. Assim, os símbolos de avanço para os itens de A são os de $\text{prim}(E) \cup \{b\} = \{d, b\}$, que terão de se juntar aos já existentes:

$$\begin{array}{l} S \to .A \ \ , \{ \ \ \ \ \underline{\$}\} \\ A \to .Bb \ , \{\underline{\$}, \underline{a}, d, b\} \\ A \to .Aa \ , \{\underline{\$}, \underline{a}, d, b\} \\ B \to .c \ \ \ , \{ \ \ \ \ \underline{b}\} \\ B \to .AE, \{ \ \ \ \ \underline{b}\} \end{array}$$

9. É preciso reconsiderar os dois itens para A visto que contêm símbolos não sublinhados. No entanto, tal como aconteceu anteriormente, estes dois itens não originam nada de novo. Juntando os dois passos num só, chega-se a:

$$\begin{array}{l} S \to .A \ \ , \{ \ \ \ \ \underline{\$}\} \\ A \to .Bb \ , \{\underline{\$}, \underline{a}, \underline{d}, \underline{b}\} \\ A \to .Aa \ , \{\underline{\$}, \underline{a}, \underline{d}, \underline{b}\} \\ B \to .c \ \ \ , \{ \ \ \ \ \underline{b}\} \\ B \to .AE, \{ \ \ \ \ \underline{b}\} \end{array}$$

Como todos os símbolos aparecem sublinhados, o processo termina. Removendo os sublinhados, o fecho do item inicial é então o conjunto formado pelos seguintes itens:

$$\begin{array}{l} S \to .A \ \ , \{ \ \ \ \ \$\} \\ A \to .Bb \ , \{\$, a, d, b\} \\ A \to .Aa \ , \{\$, a, d, b\} \\ B \to .c \ \ \ , \{ \ \ \ \ b\} \\ B \to .AE, \{ \ \ \ \ b\} \end{array}$$

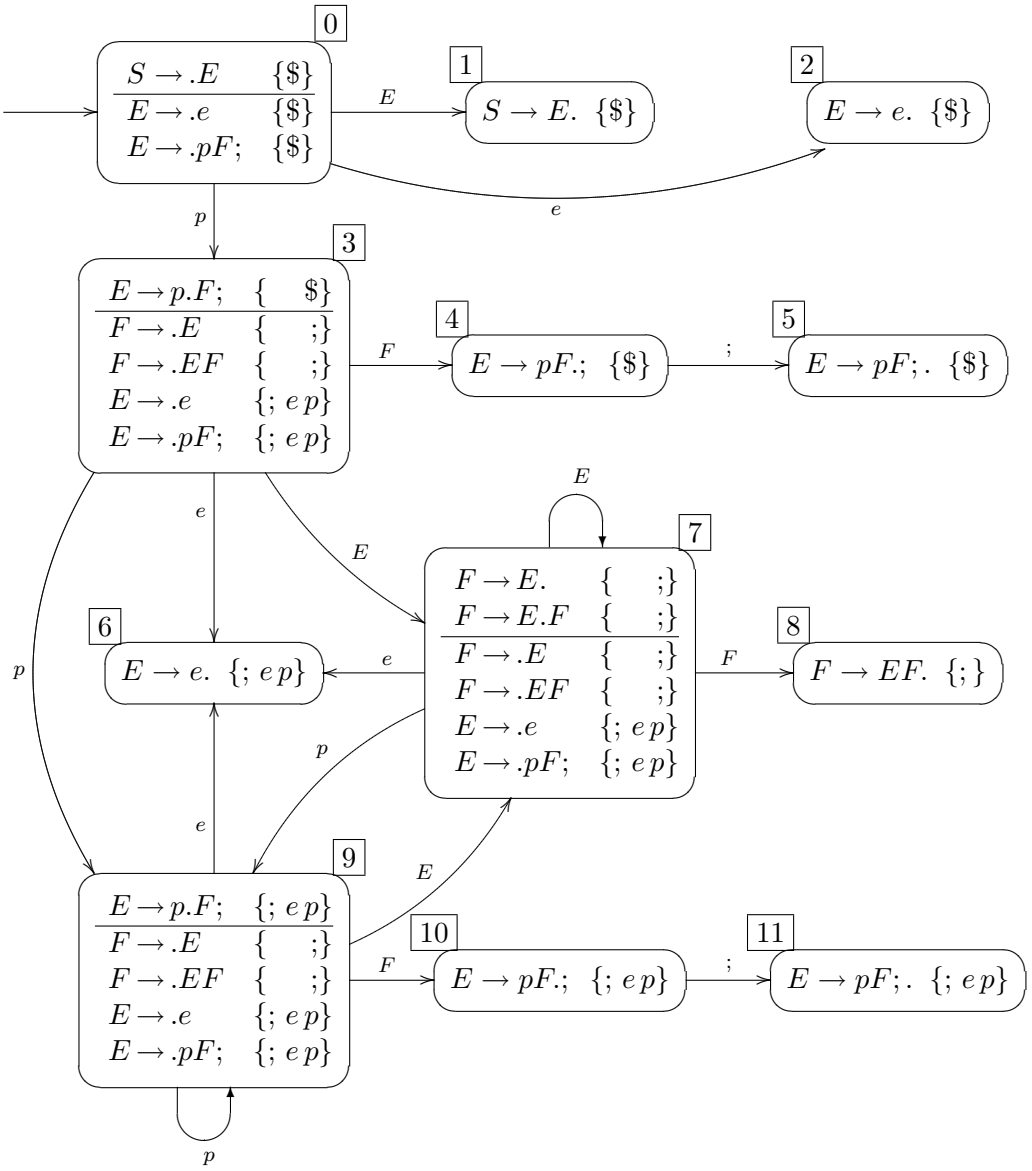
Exercício 16 Considere a gramática $\mathcal{G} = (\{e, p, ;\}, \{S, E, F\}, S, \mathcal{R})$, com $\mathcal{R}$ o seguinte conjunto de produções:

$$\begin{array}{l} S \longrightarrow E \\ E \longrightarrow e \mid pF; \\ F \longrightarrow E \mid EF \end{array}$$

Apresente o autómato finito determinista dos itens válidos LR(1) e justifique se a gramática é LR(0), LR(1) e/ou LALR(1).

(Exame Final de 2005/06)

Esta gramática foi utilizada num exercício anterior para a construção do autómato dos itens válidos LR(0). Aqui vamos construir o autómato dos itens válidos LR(1). No autómato apresentado a seguir, para facilitar a leitura representámos o conjunto $\{;, e, p\}$ apenas por $\{; e\, p\}$, omitindo a vírgula como separador dos seus elementos. Igualmente, omitimos a vírgula a separar as duas componentes de um item LR(1). O autómato é o seguinte:



A gramática dada não é LR(0) porque no estado 7 existe um item completo ($F \to E.\ \{;\}$) e pelo menos outro item que tem um símbolo terminal à direita do ponto (por exemplo, $E \to .e\ \{;\, e\, p\}$).

A gramática é LR(1) porque (i) nos estados 1, 2, 5, 6, 8 e 11 os itens completos não estão acompanhados de outros itens; (ii) no estado 7 existe um só item completo ($F \to E.\ \{;\}$) e os símbolos terminais que figuram à direita do ponto em outros itens (e e p) não são símbolos de avanço do item completo; (iii) não existem outros estados com itens completos.

Os estados equivalentes são o 2 e o 6, o 3 e o 9, o 4 e o 10, e o 5 e o

11. Para ver se a gramática é LALR(1) temos de criar um novo autómato onde estados equivalentes são fundidos num só. Representemos os novos estados por 2-6, 3-9, 4-10 e 5-11. Estes novos estados continuam a satisfazer as condições LR(1), por conseguinte a gramática é LALR(1).

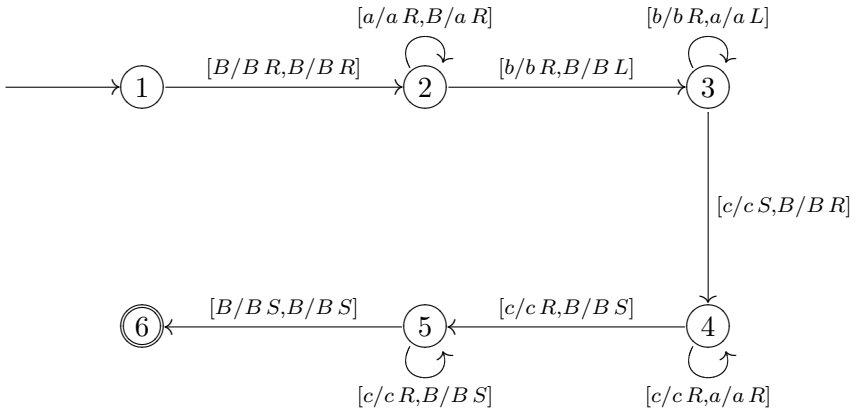
Exercício 17 Considere a seguinte linguagem sobre o alfabeto $\{a, b, c\}$:

$$L = \{a^n b^{n+1} c^k : 0 \leq n < k\}.$$

Defina uma máquina de Turing (modelo básico ou modelo multi-fita, à sua escolha) que reconheça L pelo critério dos estados de aceitação.

(Exame de Recurso de 2005/06)

A seguinte MT de duas fitas resolve o problema. A explicação do seu funcionamento é apresentada mais abaixo.



A MT executa as seguintes operações:

1. Posiciona as cabeças de ambas as fitas na segunda casa deixando a primeira casa em branco (transição do estado 1 para o estado 2).

2. Copia os a's da primeira fita para a segunda, avançando sempre para a direita (laço no estado 2).

3. Quando na primeira fita surge o primeiro b, move a primeira cabeça uma casa para a direita e move a segunda cabeça uma casa para a esquerda (transição do estado 2 para o estado 3).

4. As cabeças ficam posicionadas respectivamente no segundo b e no último a. A primeira cabeça tem n b's à sua direita e a segunda cabeça tem o mesmo número de a's à sua esquerda. Então, avança a primeira cabeça pelos b's enquanto recua a segunda pelos a's (laço no estado 3).

5. Um c é encontrado na primeira fita quando na segunda se chega à casa inicial em branco. Enquanto deixa a primeira cabeça parada, posiciona a segunda no primeiro a movendo uma casa para a direita (transição de 3 para 4).

6. As duas cabeças avançam agora em conjunto, a primeira lendo c's e a segunda lendo a's (laço no estado 4).

7. Como $n < k$, os a's esgotam-se antes dos c's. Tem de haver pelo menos um c em excesso, que é lido enquanto a segunda cabeça fica parada (transição de 4 para 5).

8. A finalizar, lê os últimos c's (laço no estado 5) e quando não houver mais, transita para um estado de aceitação (transição de 5 para 6).

Exercício 18 Seja $T = \{a, b\}$. Considere a linguagem L formada por todas as palavras $a_1a_2a_3a_4 \cdots a_{2n-1}a_{2n}$ sobre T de comprimento par ($n \geq 0$) tal que $a_1a_3 \cdots a_{2n-1} = a_{2n} \cdots a_4a_2$, isto é, a palavra formada pelos símbolos que ocupam as posições ímpares é igual ao inverso da palavra formada pelos símbolos que ocupam as posições pares. Defina uma máquina de Turing (modelo básico ou modelo multi-fita, à sua escolha) que reconheça a linguagem L.

(Exame Final de 2007/08)

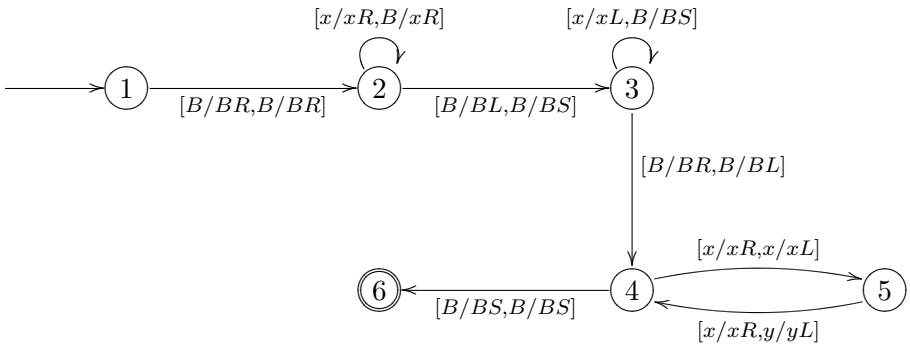
Vamos apresentar várias soluções. Todas são interessantes e foram seleccionadas de entre as apresentadas pelos alunos no referido exame.

Em qualquer das soluções, quando se usa x ou y numa transição significa que tanto pode ser a como b.

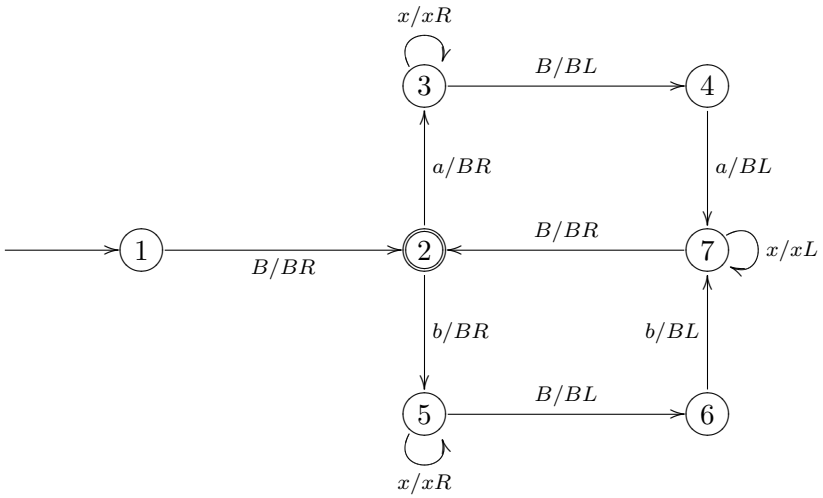
Máquina com três fitas

Máquina com duas fitas

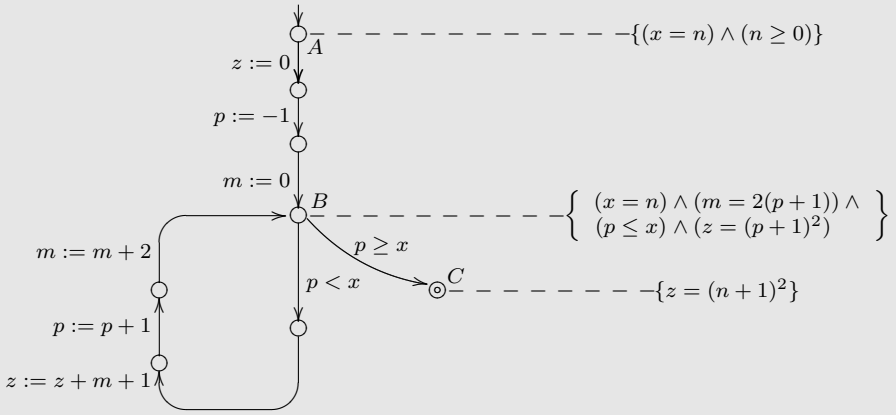
Outra máquina com duas fitas



Máquina com uma fita



Exercício 19 Considere o seguinte programa, em que a variável lógica n e as variáveis do programa x, z, p e m são de tipo inteiro:



Considere um percurso do ciclo $B - B$. Suponha que σ é o estado no início do ciclo em B e que σ' o estado no fim do ciclo de novo em B.

1. Com as convenções usuais para representar os valores das variáveis nos estados σ e σ', escreva a condição no início do ciclo $p_B(\sigma)$, a condição de percurso $p_{B-B}(\sigma)$, a condição de transformação de estado $\sigma' = t_{B-B}(\sigma)$, a condição no fim do ciclo $p_B(\sigma')$ e verifique a correcção local do ciclo $B - B$.

2. Verifique que o ciclo $B - B$ termina, definindo uma função apropriada f_B no ponto B e justificando.

(Exame Final de 2007/08)

1. Condição no início do ciclo, $p_B(\sigma)$:

$$(x = n) \wedge (m = 2(p + 1)) \wedge (p \leq x) \wedge (z = (p + 1)^2)$$

Condição de percurso, $p_{B-B}(\sigma)$:

$$p < x$$

Condição de transformação de estado, $\sigma' = t_{B-B}(\sigma)$:[2]

$$(x' = x) \wedge (z' = z + m + 1) \wedge (p' = p + 1) \wedge (m' = m + 2)$$

Condição no fim do ciclo, $p_B(\sigma')$:

$$(x' = n) \wedge (m' = 2(p' + 1)) \wedge (p' \leq x') \wedge (z' = (p' + 1)^2)$$

Condição de verificação, $[p_B(\sigma) \wedge p_{B-B}(\sigma) \wedge (\sigma' = t_{B-B}(\sigma))] \Longrightarrow p_B(\sigma')$:[3]

$$\underbrace{(x = n)}_{1} \wedge \underbrace{(m = 2(p + 1))}_{2} \wedge \underbrace{(p \leq x)}_{3} \wedge \underbrace{(z = (p + 1)^2)}_{4}$$
$$\wedge$$
$$\underbrace{p < x}_{5}$$
$$\wedge$$
$$\underbrace{(x' = x)}_{6} \wedge \underbrace{(z' = z + m + 1)}_{7} \wedge \underbrace{(p' = p + 1)}_{8} \wedge \underbrace{(m' = m + 2)}_{9}$$
$$\Longrightarrow$$
$$\underbrace{(x' = n)}_{10} \wedge \underbrace{(m' = 2(p' + 1))}_{11} \wedge \underbrace{(p' \leq x')}_{12} \wedge \underbrace{(z' = (p' + 1)^2)}_{13}$$

Verificação das conclusões:[4]

(10) $x' \overset{6}{=} x \overset{1}{=} n$
(11) $m' \overset{9}{=} m + 2 \overset{2}{=} 2(p + 1) + 2 \overset{8}{=} 2p' + 2 \overset{M}{=} 2(p' + 1)$
(12) $p' \overset{8}{=} p + 1 \overset{5,M}{\leq} x \overset{6}{=} x'$
(13) $z' \overset{7}{=} z + m + 1 \overset{2,4}{=} (p + 1)^2 + 2(p + 1) + 1 \overset{8}{=} p'^2 + 2p' + 1 \overset{M}{=} (p' + 1)^2$

[2]Para as variáveis do programa, não para a variável lógica.

[3]Em testes ou exames não é necessário numerar as hipóteses e as conclusões; isso foi feito aqui apenas com fins didácticos, para mais facilmente justificar todos os passos do raciocínio.

[4]Uma etiqueta i sobre $=$ ou $\leq$ indica que foi usada a hipótese i para estabelecer essa relação. A etiqueta M indica uma justificação de índole matemática.

2. *Explicação:*[5]

   É preciso definir uma função $f_B(n, x, z, p, m)$ de todas as variáveis e com valores inteiros (positivos, negativos ou nulo) que satisfaça duas condições: (i) sempre que o ciclo $B - B$ é percorrido (isto é, quando a sua condição de percurso for verdadeira), no início do percurso tem-se $f_B(n, x, z, p, m) > 0$; (ii) quando o ciclo acabar de ser percorrido, para os novos valores das variáveis tem-se $f_B(n, x', z', p', m') < f_B(n, x, z, p, m)$.

   Porquê estas condições? Suponhamos que $f_B(n, x, z, p, m) = k$ na primeira passagem por B. Se $k \leq 0$, o ciclo não é percorrido, pela condição (i). Se $k > 0$, como em cada iteração a função diminui estritamente de valor pela condição (ii), ao fim de no máximo k iterações o seu valor é ≤ 0 e o ciclo termina, de novo pela condição (i).

   Resolução:

   Define-se $f_B(n, x, z, p, m) = x - p$.

   Verificação das condições de f_B para um percurso de $B - B$:

   Início do percurso:

   Como a condição de percurso $p < x$ é verdadeira no início,

$$p < x \quad \Longrightarrow \quad x - p > 0 \quad \Longrightarrow \quad f_B(n, x, z, p, m) > 0.$$

   Fim do percurso:

$$f_B(n, x', z', p', m') = x' - p' = x - (p + 1) < x - p = f_B(n, x, z, p, m).$$

Exercício 20 Sem usar a operação de exponenciação, escreva um programa para calcular uma potência a^k, em que a é um número real $\neq 0$ e k um inteiro ≥ 0. Utilize um algoritmo de complexidade logarítmica em função do expoente. Forneça as asserções de entrada e de saída e demonstre a correcção parcial e a terminação do programa.

Por complexidade logarítmica em k entende-se que o número de passos do ciclo principal do algoritmo é $\lceil \log_2 k \rceil$, em que $\lceil x \rceil$ é o menor inteiro $\geq x$, isto é, $\lceil x \rceil$ é inteiro e $\lceil x \rceil - 1 < x \leq \lceil x \rceil$. Em cada passo é executado um número limitado de operações aritméticas (duas nuns casos e três noutros), logo o número de operações aritméticas é da ordem de $\lceil \log_2 k \rceil$.

A estrutura do algoritmo é exemplificada com o seguinte cálculo de a^k com $k = 13$ (ver abaixo a explicação dos cálculos efectuados):

[5] Esta explicação não faz parte da resolução e não é necessária em testes ou exames.

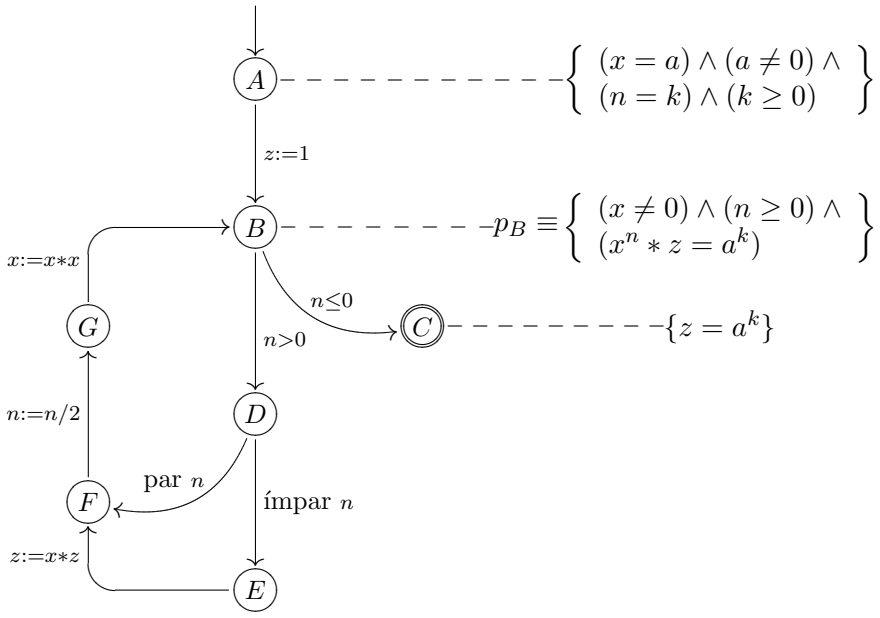
$$
\begin{aligned}
a^{\overbrace{13}^{k}} &= \underbrace{a}_{x}{}^{\overbrace{13}^{n}} \times \underbrace{1}_{z} \\
&= \underbrace{(a^2)}_{x}{}^{\overbrace{6}^{n}} \times \underbrace{a \times 1}_{z} \\
&= \underbrace{((a^2)^2)}_{x}{}^{\overbrace{3}^{n}} * \underbrace{a \times 1}_{z} \\
&= \underbrace{(((a^2)^2)^2)}_{x}{}^{\overbrace{1}^{n}} \times \underbrace{(a^2)^2 \times a \times 1}_{z} \\
&= \underbrace{((((a^2)^2)^2)^2)}_{x}{}^{\overbrace{0}^{n}} \times \underbrace{((a^2)^2)^2 \times (a^2)^2 \times a \times 1}_{z}
\end{aligned}
$$

A primeira igualdade inicializa o processo reescrevendo a^k na forma $x^n * z$, com $x = a$, $n = k$ e $z = 1$. Cada igualdade é obtida dividindo n por 2 (divisão inteira), elevando x ao quadrado e mantendo o mesmo valor de z ou multiplicando-o por x consoante n é par ou ímpar, respectivamente. Em todos os casos continua a verificar-se que $a^k = x^n * z$. Quando n atinge o valor 0, o termo x^n vale 1 e tem-se $z = a^k$. (Note-se que o número de passos após o inicial é $\lceil \log_2 13 \rceil = \lceil 3{,}6... \rceil = 4$.)

Estas observações estão consagradas no algoritmo descrito pelo seguinte diagrama de transições:

{ a, x, z reais; k, n inteiros }
{ $n/2$ = divisão inteira de n por 2: $2 * (n/2) \leq n < 2 * (n/2 + 1)$ }
{ par $n \equiv n = 2 * (n/2)$ }

{ ímpar $n \equiv n = 2 * (n/2) + 1$ }



A **asserção de entrada** na localização inicial A indica que as **variáveis do programa** x e n contêm os dados do problema representados pelas **variáveis lógicas** a e k e apresenta as condições que os dados devem satisfazer. A **asserção de saída** na localização final C indica que o valor final da variável do programa z é a^k.

Para raciocinar sobre a **correcção parcial** do programa (se o estado inicial satisfizer a condição de entrada e o programa terminar, o estado final satisfaz a condição de saída), na localização B do ciclo que precede o teste de saída e corresponde ao início e ao fim de cada iteração, colocou-se uma asserção que é uma conjunção de três outras asserções. A mais importante é a terceira, $x^n * z = a^k$, que como vimos é válida inicialmentre e continua válida após cada iteração; por essa razão se chama um **invariante** do ciclo. As outras duas condições, $x \neq 0$ e $n \geq 0$, têm um papel auxiliar. A condição $n \geq 0$ permite estabelecer que $n = 0$ quando a iteração termina, pois isso sucede quando $n \leq 0$ (condição do percurso $B - C$). A condição $x \neq 0$ implica que $x^n = 1$ quando $n = 0$, o que é essencial para se concluir que $z = a^k$.

Para verificar a correcção parcial consideram-se todos os caminhos entre dois pontos de corte (os pontos onde estão localizadas asserções) sem outros pontos de corte pelo meio. Neste exemplo existem os quatro caminhos $A-B$, $B-C$, $B-D-E-F-G-B$ e $B-D-F-G-B$. Para cada caminho verifica-se que se a asserção no início for verdadeira para os valores correntes das variáveis, a asserção no fim é verdadeira para os novos valores das variáveis nesse ponto. Recorde-se que o valor de qualquer variável v no início de um

percurso se representa pela mesma letra v, e o valor de v no fim do percurso se representa por v'. Para cada percurso, verifica-se a condição no seu fim (com as variáveis com pelicas) usando como hipóteses (1) a condição no início de percurso (com as variáveis sem pelicas), (2) a condição do percurso e (3) as relações dos novos (com pelicas) com os antigos (sem pelicas) valores das variáveis. A título de exemplo consideremos o percurso $B - D - F - G - B$.

Condição no início:

$$(x \neq 0) \wedge (n \geq 0) \wedge (x^n * z = a^k).$$

Condição no fim:

$$(x' \neq 0) \wedge (n' \geq 0) \wedge (x'^{n'} * z' = a^k).$$

Condição de percurso:

$$(n \geq 0) \wedge (\text{par } n).$$

Relações entre os antigos e os novos valores das variáveis:

$$(x' = x * x) \wedge (n' = n/2) \wedge (z' = z).$$

Na tabela seguinte efectuam-se as verificações para todos os percursos. Por simplicidade as hipóteses de cada percurso não foram explicitadas, embora sejam usadas quando necessário.

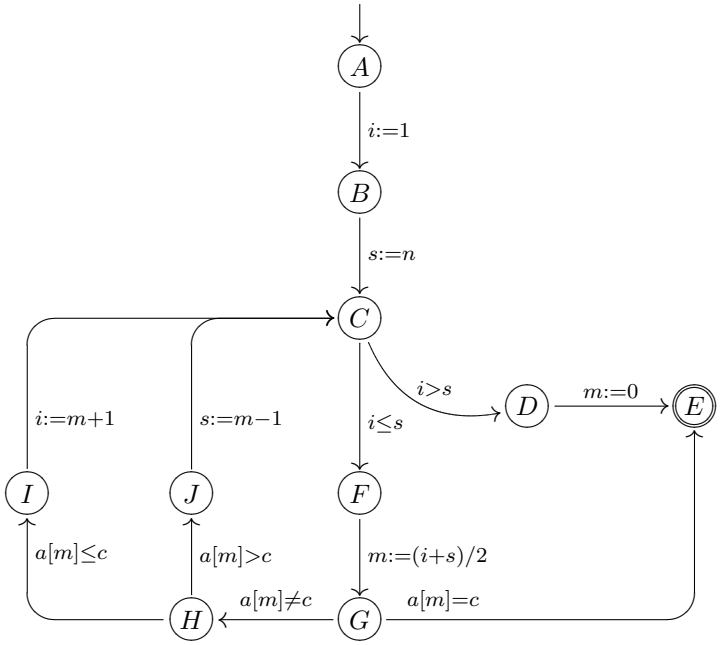
$A - B$	Condição final: $(x' \neq 0) \wedge (n' \geq 0) \wedge (x'^{n'} * z' = a^k)$ Verificação: $x' = x = a \neq 0$ $n' = n = k \geq 0$ $x'^{n'} * z' = a^k * 1 = a^k$
$B - C$	Condição final: $z' = a^k$ Verificação: $n \geq 0 \wedge n \leq 0 \Longrightarrow n = 0$ $x \neq 0 \wedge n = 0 \Longrightarrow x^n = 1$ $\Longrightarrow z' = z = x^n * z = a^k$
$B - D - E - F - G - B$	Condição final: $(x' \neq 0) \wedge (n' \geq 0) \wedge (x'^{n'} * z' = a^k)$ Verificação: $x \neq 0 \wedge x' = x * x \Longrightarrow x' \neq 0$ $n \geq 0 \wedge n' = n/2 \Longrightarrow n' \geq 0$ ímpar $n \Longrightarrow n = 2 * (n/2) + 1$ $\Longrightarrow x'^{n'} * z' = (x^2)^{n/2} * x * z$ $= x^{2*(n/2)+1} * z = x^n * z = a^k$
$B - D - F - G - B$	Condição final: $(x' \neq 0) \wedge (n' \geq 0) \wedge (x'^{n'} * z' = a^k)$ Verificação: $x \neq 0 \wedge x' = x * x \Longrightarrow x' \neq 0$ $n \geq 0 \wedge n' = n/2 \Longrightarrow n' \geq 0$ par $n \Longrightarrow n = 2 * (n/2)$ $\Longrightarrow x'^{n'} * z' = (x^2)^{n/2} * z$ $= x^{2*(n/2)} * z = x^n * z = a^k$

Para verificar que o programa termina é preciso mostrar que o seu (único) ciclo termina. A técnica consiste em definir uma função $f_B(x, n, z)$ dos valores das variáveis do programa no ponto B satisfazendo duas condições: (1) sempre que o ciclo é percorrido, $f_B(x, n, z) > 0$; (2) no fim do percurso, $f_B(x', n', z') < f_B(x, n, z)$ para os novos valores das variáveis. Se existir uma tal função, fica garantido que o ciclo termina. Neste exemplo a função $f_B(x, n, z) = n$ satisfaz as duas condições anteriores.

Exercício 21 Dado um vector ordenado $a[1..n]$ de números reais e um número real c, escreva um programa que implemente a pesquisa dicotómica de c em a. Especifique o programa e prove a sua correcção parcial e a sua terminação.

O algoritmo de pesquisa dicotómica é bem conhecido. Sem mais comentários, segue-se o diagrama de transições que implementa o algoritmo e a verificação da sua correcção parcial.

{ n inteiro, c real, $a[1..n]$ vector ordenado de reais }
{ Os valores de n, c, e a manter-se-ão fixos }
{ i = índice inferior, s = índice superior }
{ m = índice intermédio = possível posição de c em a }
{ Se c não se encontrar em a, no final $m = 0$ }



$$p_A \equiv \forall j : 1 \leq j < n : a[j] \leq a[j+1]$$

$$p_C \equiv p_A \wedge [1 \leq i] \wedge [s \leq n] \wedge \forall j : [1 \leq j < i] \vee [s < j \leq n] : a[j] \neq c$$

$$p_E \equiv [(m = 0) \wedge \forall j : 1 \leq j \leq n : a[j] \neq c] \vee [(1 \leq m \leq n) \wedge a[m] = c]$$

$$p_G \equiv p_C \wedge [i \leq m \leq s]$$

$$p_I \equiv p_G \wedge [a[m] < c]$$

$$p_J \equiv p_G \wedge [a[m] > c]$$

$A - C$	$p_A \equiv T$ (porque i, s não ocorrem em p_A) $i' = 1 \geq 1$ $s' = n \leq n$ $\forall j : [1 \leq j < i'] \vee [s' < j \leq n] : a[j] \neq c$ $\equiv \forall j : [1 \leq j < 1] \vee [n < j \leq n] : a[j] \neq c \equiv T$
$C - E$	$m' = 0$ $i > s \Longrightarrow (1 \leq j \leq n \Longrightarrow [1 \leq j < i] \vee [s < j \leq n])$ $\forall j : [1 \leq j < i] \vee [s < j \leq n] : a[j] \neq c$ $\Longrightarrow \forall j : 1 \leq j \leq n : a[j] \neq c$
$C - G$	$p_C \equiv T$ (porque m não ocorre em p_C) $i' \leq m' \leq s'$ porque $m' = (i + s)/2$ e $i \leq s \Longrightarrow i' = i = (i + i)/2 \leq (i + s)/2 \leq (s + s)/2 = s = s'$
$G - E$	$m' = m \geq i \geq 1$ $m' = m \leq s \leq n$ $a[m'] = a[m] = c$
$G - I$	$\surd$
$G - J$	$\surd$
$I - C$	$p_A \equiv T$ (porque i não ocorre em p_A) $i' = m + 1 \geq i + 1 \geq 1 + 1 \geq 1$ $s' = s \leq n$ $1 \leq i \leq m \wedge a[m] < c \Longrightarrow \forall j : 1 \leq j < m + 1 : a[j] \leq a[m] < c$ $\Longrightarrow \forall j : [1 \leq j < i'] \vee [s' < j \leq n] : a[j] \neq c$
$J - C$	$p_A \equiv T$ (porque s não ocorre em p_A) $i' = i \geq 1$ $s' = m - 1 \leq n - 1 \leq n$ $m \leq s \leq n \wedge a[m] > c \Longrightarrow \forall j : m - 1 < j \leq n : a[j] \geq a[m] > c$ $\Longrightarrow \forall j : [1 \leq j < i'] \vee [s' < j \leq n] : a[j] \neq c$

Para verificar a terminação considera-se em C a função $f_C(i, s, \ldots) = s - i + 1$ (os argumentos em falta no lugar de '...' são as restantes variáveis, as lógicas e as do programa). Como de cada vez que o ciclo é percorrido se tem $i \leq s$ em C, então $s - i \geq 0$, donde $s - i + 1 > 0$ e portanto $f_C(i, s, \ldots) > 0$. Outra consequência de $i \leq s$ é $i + i \leq i + s \leq s + s$, donde $i \leq (i + s)/2 \leq s$, ou seja, $i \leq m' \leq s$. Depois de percorrido o ciclo (caso ele não termine com a condição $a[m'] = c$), no regresso a C há dois casos: 1) $a[m'] > c$, vindo $i' = i$ e $s' = m' - 1$, ou 2) $a[m'] \leq c$, e então $i' = m' + 1$ e $s' = s$. No primeiro caso, $s' - i' + 1 = m' - i \leq s - i < s - i + 1$. No segundo caso, $s' - i' + 1 = s - m' \leq s - i < s - i + 1$. Em ambos os casos, $f_C(i', s', \ldots) < f_C(i, s, \ldots)$. Conclui-se que o ciclo termina.