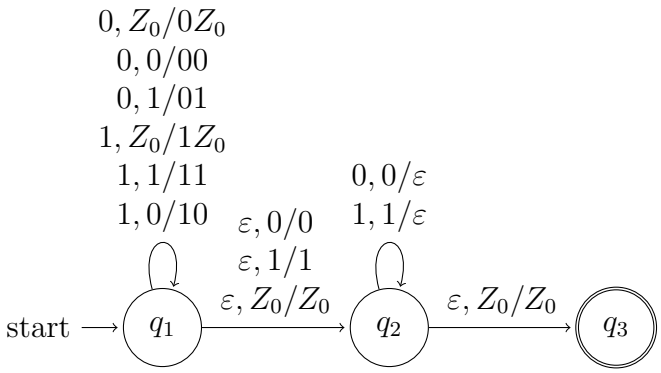
Capítulo 8

Autómatos de Pilha (PDA)

Os autómatos de pilha, ou PDAs (do inglês *Push-Down Automata*) são extensões dos ε-NFAs com uma pilha infinita que permite armazenar informação. Uma transição tem agora que ter em atenção não só o estado atual e o símbolo na cadeia de entrada, mas também o símbolo que se encontra no topo da pilha, substituíndo ao mesmo tempo o topo da pilha (fazendo *push* de um novo símbolo; fazendo *pop* do símbolo que lá estivesse; substituindo o símbolo existente por um novo; ou mantendo a pilha inalterada). De uma forma mais formal, um PDA é visto como um tuplo $P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$, em que Q é o conjunto de estados do PDA, Σ o alfabeto (símbolos de entrada), Γ o alfabeto da pilha, δ o conjunto de transições, $q_0 \in Q$ o estado inicial, $Z_0 \in \Gamma$ o símbolo inicial da pilha, e $F \subseteq Q$ o conjunto de estados finais. As transições são no formato $\delta(q, a, X) = \{(p_1, \gamma_1), (p_2, \gamma_2), ...\}$, em que $q, p_1, p_2, ... \in Q$ são estados do PDA, $a \in \Sigma$ é um símbolo de entrada, $X \in \Gamma$ é um símbolo da pilha e $\gamma_1, \gamma_2, ...$ são conjuntos de símbolos da pilha, pelos quais X é substituído (pode ser usado ε para fazer *pop* da pilha).

Vamos olhar novamente para o exemplo da linguagem L = $\{vv^R : v \in \{0,1\}^*\}$. Uma possível implementação desta linguagem com um PDA será usando um estado para ler os símbolos de v, fazendo *push* desses símbolos na pilha; outro estado para ler os símbolos de v^R, fazendo *pop* dos símbolos da pilha ao confirmar que v^R é efetivamente o reverso de v; e um estado final, para aceitar a cadeia caso esta cumpra efetivamente com o formato das cadeias da linguagem. Teríamos então um $PDA = (\{q_1, q_2, q_3\}, \{0,1\}, \{Z_0, 0, 1\}, \delta, q_1, Z_0, \{q_3\})$, cuja representação gráfica seria:



Repare-se que em q_1 existem transições que fazem *push* do símbolo lido (0 ou 1) para todos os possíveis símbolos da pilha (Z_0, 0 e 1); no estado q_2 é feito *pop* de um símbolo da pilha quando o símbolo correspondente aparece na entrada; finalmente, a transição para q_3 acontece apenas quando a pilha já está vazia (isto é, todos os símbolos que foram colocados na pilha foram já retirados). A aceitação num PDA como este, de aceitação por estado final, acontece quando o autómato está num estado final e já não existem mais símbolos na cadeia de entrada (independentemente do conteúdo da pilha).

Note-se que com este autómato, é necessário 'adivinhar' quando se atinge o meio da cadeia, de forma a fazer a passagem de q_1 para q_2 (o que, como vamos ver mais à frente, faz com que este seja um PDA não determinista).

Para determinar se uma cadeia é aceite pelo PDA, pode ver-se a sequência de descrições instantâneas do PDA, verificando se o estado final é atingível. A descrição instantânea do estado do PDA inclui o estado em que este se encontra, a cadeia na entrada que ainda não foi processada, e o conteúdo da pilha. Por exemplo, para a cadeia 10100101, a sequência de descrições que conduz à aceitação seria:

$(q_1, 10100101, Z_0) \vdash (q_1, 0100101, 1Z_0) \vdash (q_1, 100101, 01Z_0) \vdash (q_1, 00101, 101Z_0) \vdash (q_1, 0101, 0101Z_0) \vdash (q_2, 0101, 0101Z_0) \vdash (q_2, 101, 101Z_0) \vdash (q_2, 01, 01Z_0) \vdash (q_2, 1, 1Z_0) \vdash (q_2, \varepsilon, Z_0) \vdash (q_3, \varepsilon, Z_0)$

Note-se ainda que existe, em cada momento, uma outra alternativa, que seria passar para q_2 ainda nos primeiros símbolos, ou manter em q_1 mesmo após ter percorrido metade da cadeia. No entanto, essas transições levariam a estados de não aceitação, pelo que seriam 'abandonados' por não se revelarem úteis.

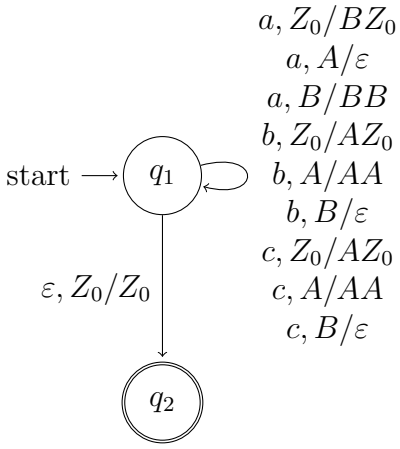
A aceitação das cadeias processadas por um PDAs pode, em alternativa ao uso de estados finais, ser determinado pelo facto de esvaziar por completo a pilha. Na representação formal, estes PDAs de aceitação por pilha vazia têm apenas os seis primeiros elementos, não incluíndo o conjunto de estados finais, pois estes deixam de existir. Estes PDAs podem ser facilmente obti-

dos a partir de um PDA de aceitação por estado final se, para cada estado final, for realizada uma transição para um novo estado, que faz *pop* de todos os símbolos da pilha, efetivamente esvaziando a pilha. Por outro lado, um PDA de aceitação por pilha vazia pode também ser convertido num PDA de aceitação por estado final se for adicionado um novo símbolo inicial na pilha antes do processamento, e em cada estado do PDA original for acrescentada uma transição para um estado final fazendo *pop* desse novo símbolo inicial.

Exercício 1

Obtenha um PDA para a linguagem das cadeias no alfabeto {a, b, c} em que o número de a's é igual à soma do número de b's e de c's.

Uma possível solução para este problema consiste em manter um contador que permita saber a relação entre o número de a's que já foi processado e o número de b's e c's. Como apenas possuímos uma pilha, podemos implementar este contador usando dois símbolos, um para representar valores positivos, e outro para representar valores negativos. Vamos ver essa solução:

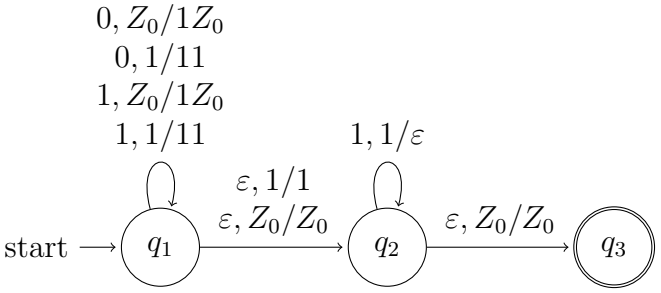


Podemos ver que no estado q_1, as transições associadas ao símbolo b são idênticas às associadas ao símbolo c, fazendo *pop* da pilha se nela existir um B, ou fazendo *push* de um A caso contrário. As transições associadas ao símbolo a fazem exatamente o contrário, isto é, é feito um *pop* da pilha, no caso de esta conter um A, ou *push* de um B caso contrário. A transição para q_2 permite que, quando a pilha esteja vazia (isto é, não há a's a mais nem a menos), o PDA transite para o estado de aceitação.

Exercício 2 (Desafio 2014/15)

Obtenha um PDA que reconheça a linguagem das cadeias binárias no formato $(0+1)^n 1^n : n \geq 0$. Apresente a sequência de descrições instantâneas para a cadeia 101111 e indique se esta é aceite pelo PDA.

Para obter um PDA para esta linguagem, necessitamos de um estado para reconhecer a primeira parte da cadeia, fazendo *push* de um símbolo para a pilha, e outro para reconhecer os 1's da segunda parte da cadeia, fazendo *pop* dos símbolos da pilha. Será ainda necessário um estado final para indicar aceitação.



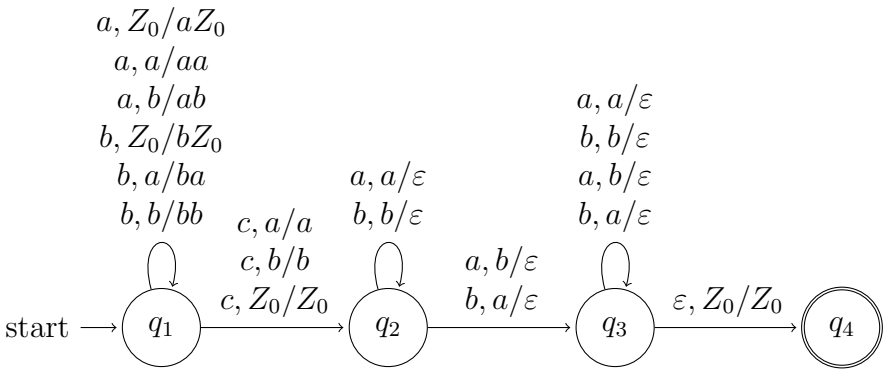
Podemos ver que esta solução é muito semelhante à do exemplo apresentado acima (as linguagens são também muito semelhantes). Para a cadeia pretendida, temos então a seguinte sequência de descrições intantâneas que conduz à aceitação da cadeia:

$(q_1, 101111, Z_0) \vdash (q_1, 01111, 1Z_0) \vdash (q_1, 1111, 11Z_0) \vdash (q_1, 111, 111Z_0)$
$\vdash (q_2, 111, 111Z_0) \vdash (q_2, 11, 11Z_0) \vdash (q_2, 1, 1Z_0) \vdash (q_2, \varepsilon, Z_0) \vdash (q_3, \varepsilon, Z_0)$

Exercício 3

Obtenha um PDA para a linguagem $\mathrm{L} = \{w_1 c w_2 : w_1, w_2 \in \{a,b\}^+, |w_1| = |w_2|, w_1 \neq w_2^R\}$. Apresente ainda a sequência de descrições instantâneas para as cadeias *abacaba* e *abacaab*, indicando se são ou não aceites pelo PDA.

Para obter uma solução para este problema, será necessário usar a pilha para guardar a primeira sequência de símbolos (w_1), e usar dois estados para validar que a segunda sequência (w_2) não é o reverso de w_1. Uma possível solução seria então:



No estado q_1 estamos então a preencher a pilha com os símbolos da primeira sequência (w_1), fazendo *push* do símbolo lido da entrada qualquer que seja o símbolo já na pilha. A passagem de q_1 para q_2 dá-se quando é lido o c que marca o meio da cadeia (podendo ter qualquer símbolo na pilha). Em q_2 e q_3 é feito o pop do símbolo no topo da pilha por cada símbolo na cadeia de entrada; em q_2 estamos ainda numa situação em que a segunda sequência está a corresponder ao reverso da primeira, pois cada símbolo lido é igual ao símbolo retirado da pilha; a passagem para q_3 acontece quando é lido um símbolo diferente do que está no topo da pilha; a partir deste momento, e como já temos a garantia de que $w_1 \neq w_2^R$, interessa-nos apenas contar o número de símbolos lidos, de forma a garanir que $|w_1| = |w_2|$. Quando a pilha é esvaziada, é realizada a transição para q_4.

Vejamos então as sequências para as cadeias indicadas:

$(q_1,\ abacaba,\ Z_0) \vdash (q_1,\ bacaba,\ aZ_0) \vdash (q_1,\ acaba,\ baZ_0) \vdash (q_1,\ caba,\ abaZ_0) \vdash (q_2,\ aba,\ abaZ_0) \vdash (q_2,\ ba,\ baZ_0) \vdash (q_2,\ a,\ aZ_0) \vdash (q_2,\ \varepsilon,\ Z_0)$

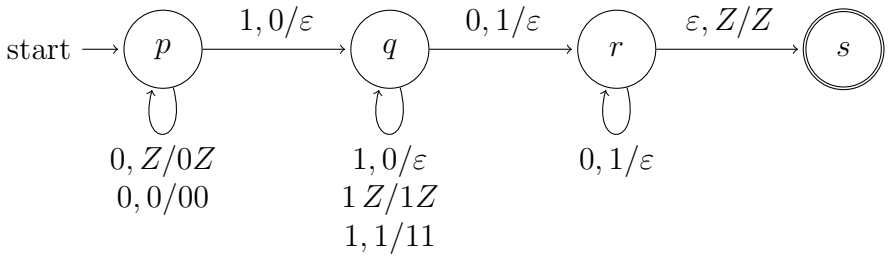
$(q_1,\ abacaab,\ Z_0) \vdash (q_1,\ bacaab,\ aZ_0) \vdash (q_1,\ acaab,\ baZ_0) \vdash (q_1,\ caab,\ abaZ_0) \vdash (q_2,\ aab,\ abaZ_0) \vdash (q_2,\ ab,\ baZ_0) \vdash (q_3,\ b,\ aZ_0) \vdash (q_3,\ \varepsilon,\ Z_0) \vdash (q_4,\ \varepsilon,\ Z_0)$

A primeira cadeia não é aceite pelo PDA (permanece no estado q_2, um estão de não aceitação), enquanto que a segunda cadeia já é aceite pelo PDA (termina em q_4, um estado de aceitação).

Exercício 4 (Exercício 2015/16)

Apresente um PDA para a linguagem $L = \{0^p 1^{p+q} 0^q : p, q > 0\}$, e a sequência de descrições instantâneas para a cadeia 011100.

Para garantir tanto a ordem dos símbolos na cadeia como a relação entre o número de zeros e de uns, torna-se necessário guardar a quantidade de 0's lidos inicialmente, assim como a quantidade de 1's que ultrapassa os zeros iniciais (e que terá de corresponder aos zeros finais). Uma possível solução, considerando um PDA com aceitação por estado final, e com símbolo inicial da pilha Z, será então:



Olhando para esta solução, podemos ver que no estado p estão a ser lidos os zeros iniciais da cadeia, colocando esses zeros na pilha, de forma a manter a sua contagem. Na transição para o estado q e no estado q, são lidos os uns, retirando os zeros da pilha enquanto ainda existem zeros, e colocando uns na pilha quando é ultrapassado o número de zeros inicial. Na passagem para o estado r e no estado r são lidos os zeros finais, sendo retirados da pilha os uns correspondentes. Finalmente, a passagem ao estado s garante a aceitação quando termina a leitura.

Para a cadeia 011100, temos então a seguinte sequência de descrições instantâneas:

$(p,\ 011100,\ Z) \vdash (p,\ 11100,\ 0Z) \vdash (q,\ 1100,\ Z) \vdash (q,\ 100,\ 1Z) \vdash (q,\ 00,\ 11Z) \vdash (r,\ 0,\ 1Z) \vdash (r,\ \varepsilon,\ Z) \vdash (s,\ \varepsilon,\ Z)$

Como no final da computação a cadeia de entrada foi completamente processada e o autómato está no estado s, significa que a aceia é aceite.

Equivalência entre CFGs e PDAs

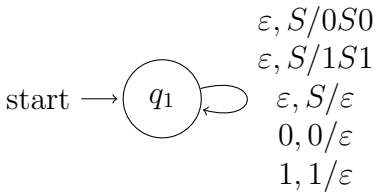
Sendo os PDAs e as CFGs duas representações de CFLs, podem ser convertidos entre si. A conversão de uma CFG num PDA de aceitação por pilha vazia é simples, bastando a existência de um estado, com várias transições para si próprio: para cada variável, uma transição que substitui essa variável pelo corpo de cada uma das suas produções na pilha, sem consumir qualquer símbolo na entrada; para cada símbolo, uma transição que consome o símbolo

e faz *pop* desse mesmo símbolo do topo da pilha. Inicialmente, a pilha contém apenas o símbolo correspondente à variável de arranque da gramática.

Vamos ver um exemplo tendo por base a gramática para a linguagem L $= \{vv^R : v \in \{0,1\}^* \}$, vista anteriormente:

$S \to 0S0 \mid 1S1 \mid \varepsilon$

O PDA equivalente terá então apenas um estado, com duas transições para os símbolos da linguagem (0 e 1) e três transições que representam as três produções da gramática. Temos assim o seguinte PDA:



Vejamos o exemplo de processamento da cadeia 101101:

(q_1, 101101, S) $\vdash$ (q_1, 101101, $1S1$) $\vdash$ (q_1, 01101, $S1$) $\vdash$ (q_1, 01101, $0S01$) $\vdash$ (q_1, 1101, $S01$) $\vdash$ (q_1, 1101, $1S101$) $\vdash$ (q_1, 101, $S101$) $\vdash$ (q_1, 101, 101) $\vdash$ (q_1, 01, 01) $\vdash$ (q_1, 1, 1) $\vdash$ (q_1, ε, ε)

Estando a pilha completamente vazia no final do processamento da cadeia de entrada, podemos afirmar que a cadeia é aceite pelo PDA.

Exercício 5

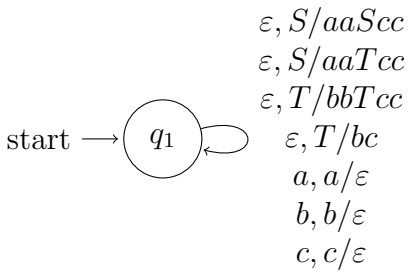
Considere as gramáticas do Exercícios 2, 3 e 4 do capítulo anterior. Converta cada uma delas para um PDA de aceitação por pilha vazia.

Começando pelo **Exercício 2**, e recordando a gramática obtida:

$S \to aaScc \mid aaTcc$

$T \to bbTcc \mid bc$

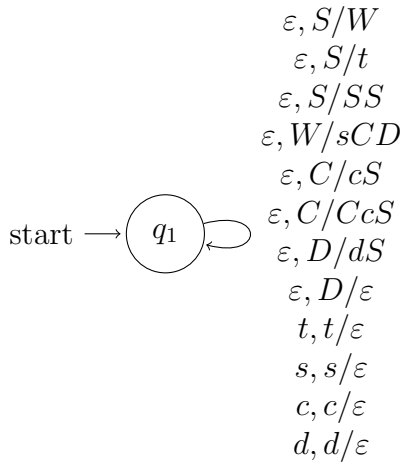
Podemos então converter esta gramática num PDA com apenas um estado usando o método visto acima:



Temos novamente apenas um estado, as quatro transições correspondentes às quatro produções da gramática, e as três transições correspondentes aos três símbolos do alfabeto da linguagem.

No caso do **Exercício 3**, temos a seguinte gramática:
$S \to W|t|SS$
$W \to sCD$
$C \to cS|CcS$
$D \to dS|\varepsilon$

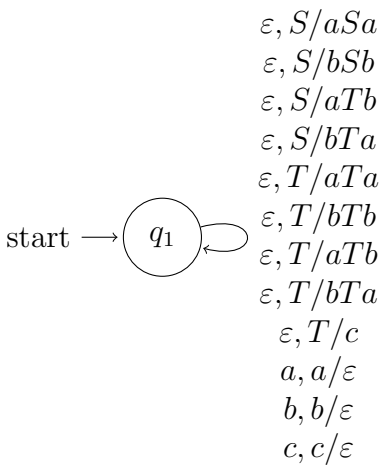
Fazendo igualmente a conversão num PDA:



Neste PDA existem bastante mais transições, dado que a gramática tem também mais produções.

Finalmente, recordado a gramática do **Exercício 4**:
$S \to aSa \mid bSb \mid aTb \mid bTa$
$T \to aTa \mid aTb \mid bTa \mid bTb \mid c$

Podemos ter então o seguinte PDA:

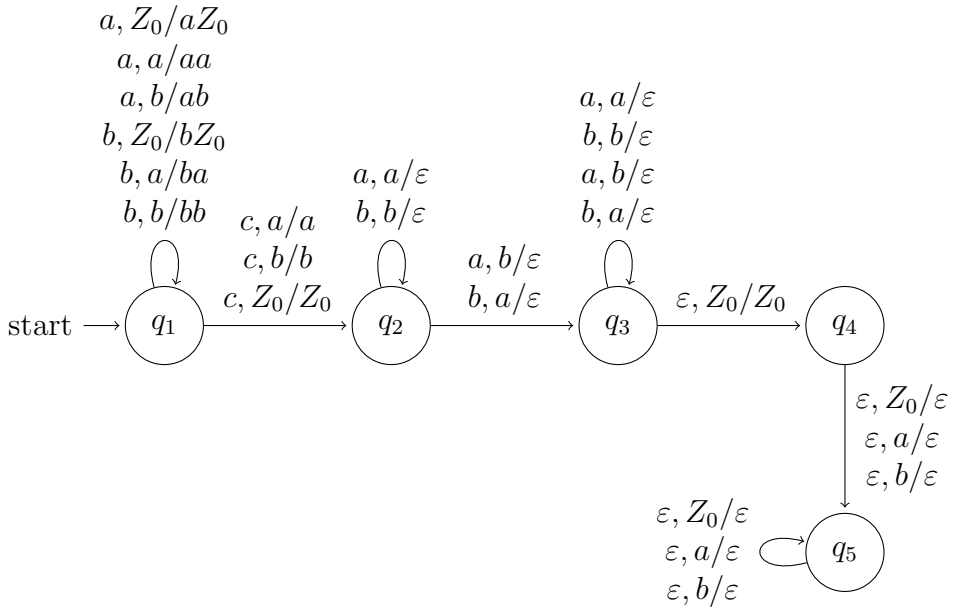


Repare-se neste último exemplo, que a linguagem pedida é a mesma que foi pedida acima, no **Exercício 3**, e para o qual foi obtido um PDA com quatro estados (sendo um deles usado apenas para garantir a aceitação das cadeias).

Exercício 6

Converta o PDA obtido no **Exercício 3** para um PDA de aceitação por pilha vazia, e o PDA obtido no **Exercício 5** por conversão da Gramática obtida no **Exercício 4** do capitulo anterior para um PDA de aceitação por estado final. Apresente as definições formais de cada um dos PDAs resultantes.

Este exercício envolve novamente os dois PDAs para a mesma linguagem. Começando pelo PDA obtido no **Exercício 3** acima, a conversão para um PDA de aceitação por pilha vazia é feita inserindo um novo estado no PDA, e acrescentando transições de cada um dos estados finais do PDA original para este novo estado em que é feito um *pop* de qualquer símbolo do topo da pilha sem consumir nada na entrada. No novo estado, deve também existir uma transição para o próprio estado, sendo também feito *pop* de qualquer símbolo da pilha. Vamos ver como fica então o PDA modificado (todas as modificações foram inseridas na metade de baixo do PDA, para ser mais simples de as identificar):



O novo estado q_5 será então o estado que tratará de esvaziar a pilha, garantindo que a pilha fica vazia no final do processamento. Repare-se que o estado q_4 deixou de ser estado final, uma vez que num PDA de aceitação por pilha vazia estes deixam de ser úteis.

Note-se ainda que, neste caso, o estado q_5 seria desnecessário, e que teria bastado modificar a transição de q_3 para q_4 de $\varepsilon, Z_0/Z_0$ para $\varepsilon, Z_0/\varepsilon$, esvaziando assim a pilha no único ponto onde esta poderia ser esvaziada.

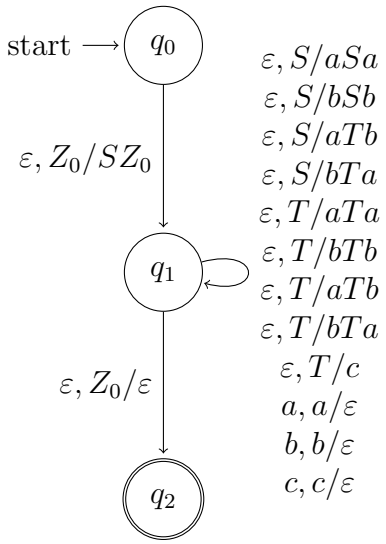
Em relação à definição formal, temos então a seguinte definição:

PDA V = $(\{q_1, q_2, q_3, q_4, q_5\}, \{a, b, c\}, \{a, b, Z_0\}, \delta, q_1, Z_0)$

Sendo δ constituído pelas seguintes transições: $\delta(q_1, a, Z_0) = \{(q_1, aZ_0)\}$, $\delta(q_1, a, a) = \{(q_1, aa)\}$, $\delta(q_1, a, b) = \{(q_1, ab)\}$, $\delta(q_1, b, Z_0) = \{(q_1, bZ_0)\}$, $\delta(q_1, b, a) = \{(q_1, ba)\}$, $\delta(q_1, b, b) = \{(q_1, bb)\}$, $\delta(q_1, c, Z_0) = \{(q_2, Z_0)\}$, $\delta(q_1, c, a) = \{(q_2, a)\}$, $\delta(q_1, c, b) = \{(q_2, b)\}$, $\delta(q_2, a, a) = \{(q_2, \varepsilon)\}$, $\delta(q_2, b, b) = \{(q_2, \varepsilon)\}$, $\delta(q_2, a, b) = \{(q_3, \varepsilon)\}$, $\delta(q_2, b, a) = \{(q_3, \varepsilon)\}$, $\delta(q_3, a, a) = \{(q_3, \varepsilon)\}$, $\delta(q_3, a, b) = \{(q_3, \varepsilon)\}$, $\delta(q_3, b, a) = \{(q_3, \varepsilon)\}$, $\delta(q_3, b, b) = \{(q_3, \varepsilon)\}$, $\delta(q_3, \varepsilon, Z_0) = \{(q_4, Z_0)\}$, $\delta(q_4, \varepsilon, Z_0) = \{(q_5, \varepsilon)\}$, $\delta(q_4, \varepsilon, a) = \{(q_5, \varepsilon)\}$, $\delta(q_4, \varepsilon, b) = \{(q_5, \varepsilon)\}$, $\delta(q_5, \varepsilon, Z_0) = \{(q_5, \varepsilon)\}$, $\delta(q_5, \varepsilon, a) = \{(q_5, \varepsilon)\}$, $\delta(q_5, \varepsilon, b) = \{(q_5, \varepsilon)\}$

Vendo agora o processo inverso, para o PDA obtido no **Exercício 5** acima, será necessário introduzir dois novos estados: um novo estado inicial (e um novo símbolo inicial para a pilha) que introduza o símbolo inicial original (variável de arranque da gramática); e um novo estado final, para o qual devem ser inseridas transições a partir de cada um dos estados do PDA

sem consumir nenhum símbolo da cadeia de entrada e tendo na pilha o novo símbolo. Vejamos então como fica o PDA modificado:



O novo estado inicial tem então uma transição para o estado inicial do PDA original, em que é feito *push* do símbolo inicial da pilha original (S), sendo agora usado Z_0 como símbolo inicial da pilha. Existe também uma transição de cada um dos estados do PDA original (neste caso só havia um estado, q_1) para o novo estado final (q_2) quando é detetado o novo símbolo inicial da pilha (o que siginifica que a pilha teria ficado completamente vazia no PDA original).

Em relação à definição formal, temos então a seguinte definição:

PDA A = $(\{q_0, q_1, q_2\}, \{a, b, c\}, \{a, b, c, S, T, Z_0\}, \delta, q_0, Z_0, \{q_2\})$

Sendo δ constituído pelas seguintes transições: $\delta(q_0, \varepsilon, Z_0) = \{(q_1, SZ_0)\}$,
$\delta(q_1, \varepsilon, S) = \{(q_1, aSa), (q_1, bSb), (q_1, aTb), (q_1, bTa)\}$,
$\delta(q_1, \varepsilon, T) = \{(q_1, aTa), (q_1, bTb), (q_1, aTb), (q_1, bTa), (q_1, c)\}$,
$\delta(q_1, a, a) = \{(q_1, \varepsilon)\},\ \delta(q_1, b, b) = \{(q_1, \varepsilon)\},\ \delta(q_1, c, c) = \{(q_1, \varepsilon)\}$,
$\delta(q_1, \varepsilon, Z_0) = \{(q_2, \varepsilon)\}$

Determinismo dos PDAs

Os PDAs vistos até aqui são maioritariamente não determinísticos, isto é, em cada momento, poderia ser realizada mais do que uma transição, dando efetivamente origem não a uma única sequência de descrições instantâneas,

mas sim a uma *árvore*[1]. No entanto, os PDAs podem ser deterministas (chamando-se normalmente de DPDA, do inglês *Deterministic PDA*), o que pode facilitar a sua execução, e análise. Um exemplo de DPDA foi obtido no **Exercício 3**, em que a cada momento apenas uma transição pode ser executada.

Para ser determinista, um PDA só pode ter uma escolha em cada momento. Para isso, e em cada estado, só pode existir no máximo uma transição possível para cada par (símbolo de entrada, símbolo na pilha). Isto implica também especial cuidado com transições espontâneas, em que não é consumido nenhum símbolo da entrada: no caso de existir uma transição destas para um determinado símbolo na pilha, não poderá existir mais nenhuma transição com esse símbolo na pilha (uma vez que essa transição-ε significa que pode ser feita com qualquer símbolo na entrada).

Exercício 7

Classifique os PDAs obtidos nos exercícios anteriores quanto ao seu determinismo.

Começando pelo **Exercício 1**, o PDA obtido não é determinista, dado que em q_1 existe uma transição para q_2 com Z_0 no topo da pilha e qualquer símbolo presente na entrada, e existem ainda transições de q_1 para si próprio também com Z_0 no topo da pilha.

No caso do **Exercício 2**, o PDA volta a ser não determinista, novamente devido a transições ε: em q_1 existem transições para si próprio com Z_0 e com 1 no topo da pilha (com 0 e 1 como símbolos de entrada), mas existem também transições-ε para q_2 com Z_0 e 1 no topo da pilha, o que faz com que o PDA seja não determinista.

No **Exercício 3**, já temos um DPDA: em cada estado existe apenas uma possibilidade para cada par (símbolo de entrada, símbolo no topo da pilha), e a transição-ε de q_3 para q_4 é feita com Z_0 no topo da pilha, e como não existe nenhuma transição em q_3 com Z_0 no topo da pilha, o determinismo não é invalidado.

No **Exercício 4**, temos também um PDA determinista: em cada estado temos apenas uma única possibilidade de transição para cada par (símbolo de entrada, símbolo no topo da pilha). Como a transição ε do estado r para

[1]Por questões de simplificação, temos estado a apresentar apenas uma sequência de descrições instantâneas que conduza à aceitação da cadeia, podendo, como já foi referido, existir possíveis caminhos alternativos (que poderiam ou não conduzir a aceitação).

s é feita com Z no topo da pilha e não há mais nenhuma transição em r considerando Z no topo da pilha, mantém-se o determinismo.

No **Exercício 5** temos vários PDAs, sendo que nenhum deles é determinista, uma vez que em cada um deles existe mais do que uma transição-ε para um determinado símbolo no topo da pilha. Por exemplo, e em qualquer um dos três PDAs apresentados, para o símbolo S existem sempre pelo menos duas possibilidades de substituição (as quais advêm da existência de mais do que uma produção para aquele símbolo (variável) na gramática que deu origem aos PDAs), o que faz com que estes PDAs não sejam deterministas. Aliás, como se pode ver, para quem um PDA obtido por este método de conversão de CFG para PDA seja determinista, a gramática original não pode ter mais do que uma produção por cada variável.

Finalmente, no **Exercício 6**, temos novamente dois PDAs. O primeiro pode ser considerado determinístico, uma vez que em cada estado existe apenas uma possibilidade de transição para cada par (símbolo de entrada, símbolo no topo da pilha). Se repararmos, todas as transições-ε são feitas em estados para os quais não existem mais nenhuma transição para cada um dos símbolos da pilha usados nessas transições. Por exemplo, em q_5 existem quatro transições-ε para o próprio estado, mas todas elas são feitas com símbolos diferentes no topo da pilha, e não existe mais nenhuma transição possível para nenhum desse símbolos. Já o segundo PDA não pode ser considerado determinístico, uma vez que apresenta várias possibilidades de transição em q_1 (sendo novamente um PDA obtido por transformação de uma CFG com várias produções para cada variável, são apresentadas várias opções de transição para os símbolos S e T no topo da pilha).

Exercício Proposto 1 Obtenha um PDA (aceitação por estado final) para a linguagem L das cadeias binárias em que o reverso dos primeiros símbolos da cadeia pode ser encontrado no resto da cadeia ($L = \{w_1 w_2 : w_1^R \subseteq w_2\}$), e converta-o para um PDA de aceitação por pilha vazia. Obtenha também uma CFG para esta mesma linguagem, converta-a para um PDA (aceitação por pilha vazia), e finalmente converta esse PDA para um PDA de aceitação por estado final. Classifique cada um dos PDAs obtidos em relação ao seu determinismo. Apresente ainda uma sequência de descrições instantâneas para a cadeia 011010. Exemplos: 01010 pertence a L (por exemplo, se $w_1 = 01$, $w_2 = 010$, que contém w_1^R (10))