III. LINGUAGENS SENSÍVEIS AO CONTEXTO E AUTÔMATOS LIMITADOS LINEARMENTE

3.1. Gramáticas e Linguagens Sensíveis ao Contexto

Como uma *generalização* das regras livres de contexto, introduzimos agora regras sensíveis ao contexto que também especificam a substituição de um símbolo não terminal único, mas em geral requer um contexto para aplicação. Portanto a regra $bC \rightarrow bc$ é sensível ao contexto porque ela diz que um não terminal C pode ser substituído pelo símbolo terminal c apenas no contexto de um b precedente. Mais formalmente, temos a seguinte definição:

1. Definição. Uma gramática G é *sensível ao contexto* se cada produção ou é da forma

(1) $yAz \rightarrow ywz$, para A em V, y,z em $(V \cup \Sigma)^*$, w em $(V \cup \Sigma)^+$ (isto é, A pode ser substituído por w no contexto y - z); ou da forma

(2) $S \rightarrow \lambda$, dado que S não aparece no lado direito de nenhuma produção.

Uma linguagem é *sensível ao contexto* se pode ser gerada por uma gramática sensível ao contexto.

2. Fato. *Seja G uma gramática tal que toda produção de G é da forma $v \rightarrow w$, com $|v| \leq |w|$, exceto que pode existir uma única produção-λ $S \rightarrow \lambda$ se S não aparece do lado direito de nenhuma produção. Então, existe uma gramática sensível ao contexto G' que é equivalente a G: L(G) = L(G').*

Note que nem toda gramática livre de contexto é sensível ao contexto, porque as produções livres de contexto podem ser da forma $A \rightarrow \lambda$. Entretanto, vamos ver que toda linguagem livre de contexto é uma linguagem sensível ao contexto.

3. Lema. (O Lema da Cadeia Vazia). *Seja G uma gramática livre de contexto que envolve regras da forma $A \rightarrow \lambda$ para qualquer $A \in V$. Então existe uma gramática livre de contexto G' tal que*

(i) *L(G) = L(G')*;

(ii) Se $\lambda \notin L(G)$ então não existem produções da forma $A \rightarrow \lambda$ em G'; mas

(iii) Se $\lambda \in L(G)$, então existe uma única produção-λ em G', $S' \rightarrow \lambda$, onde S' é o símbolo inicial de G' e S' não aparece no lado direito de nenhuma produção em G'.

PROVA: Suponha que $\lambda \notin L(G)$. Para $C \in V$ considere todas as produções $C \rightarrow w$, onde w não é vazio. Agora suponha que w contém variáveis $A_1,..., A_k, B_1,..., B_j$ onde A_i, $1 \leq i < k$, são aquelas variáveis em w para as quais $A_i \Rightarrow^* \lambda$. Adicione a G a produção $C \rightarrow w'$, onde w' é obtida de w apagando zero ou mais ocorrências de uma variável dos A_i's. Então, por exemplo,

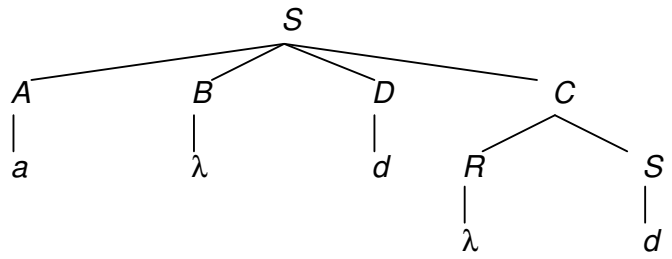
$$C \rightarrow aA_1BA_2$$

leva a quatro produções

$$C \rightarrow aA_1BA_2 \mid aBA_2 \mid aA_1B \mid aB$$

Quando este processo estiver completo, remova todas as produções-λ, incluindo qualquer produção-λ nova que tenha sido incluída no processo. A gramática resultante é *G*'. Então $L(G) = L(G')$ como se segue. Se uma derivação de alguma cadeia $w \in L(G)$ envolveu produções-λ, então a árvore de derivação para *w* terá nós rotulados com λ, como no seguinte exemplo.

$S \to ABDC$
$C \to RS$
$B \to \lambda$
$A \to a$
$D \to d$
$R \to \lambda$
$S \to d$



Para obter uma derivação *G*' para *w* a partir de uma derivação *G* para *w*, comece no topo da árvore *G* e apague qualquer subárvore da raiz da árvore que deriva λ. O topo da árvore deve agora ilustrar uma produção a partir de *G*'. Então considere os sucessores imediatos do nó raiz que não derivam λ. Aplique o mesmo princípio a cada um deles: se alguma das suas subárvores deriva λ, apague a subárvore. Continue desta forma para baixo na árvore. A árvore final representa uma derivação *G*'. Além disto, esta árvore deriva *w*, porque uma subcadeia de *w* é removida pelo processo de apagamento se e somente se ela for λ. Por outro lado, se $w \in L(G')$ e a derivação de *w* envolve uma regra recém adicionada, então simule a derivação de *G*' usando a regra *G* original apropriada e regras-λ. Isto é, faça o processo descrito acima na ordem reversa, novamente de cima para baixo. No caso onde $\lambda \in L(G)$, primeiro adicione um novo símbolo inicial a *G*', por exemplo *S*', e adicione duas novas produções, $S' \to S$ e $S' \to \lambda$. Depois repita o processo descrito previamente em todas as produções exceto as produções *S*'. Então $L(G) = L(G')$. ◊

4. Teorema. *Uma linguagem L é sensível ao contexto se e somente se existe alguma gramática G tal que L = L(G) onde toda produção de G da forma* $u \to v$ *tem a propriedade que* $0 < |u| \leq |v|$ *com uma exceção: se* $\lambda \in L(G)$*, então a regra* $S \to \lambda$ *está também presente e neste caso S não pode aparecer no lado direito de nenhuma produção*.

5. Exemplo de Gramática Sensível ao Contexto

G = (Σ, V, S, P), Σ = { a, b }, V = { S, B, C } e P = {
$S \to aSBC$
$S \to aBC$
$CB \to BC$
$aB \to ab$
$bB \to bb$
$bC \to bc$
$cC \to cc$ }

A linguagem L(G) contém a palavra $a^nb^nc^n$ para cada $n \geq 1$, pois pode-se usar a produção (1) n-1 vezes para chegar a $S \Rightarrow^* a^{n-1} S (BC)^{n-1}$. Depois usa-se a (2) para $S \Rightarrow^* a^n (BC)^n$. A produção (3) permite arranjar os Bs e Cs tal que todo B preceda todos os Cs. Por exemplo, se n = 3:

$$aaaBCBCBC \Rightarrow aaaBBCCBC \Rightarrow aaaBBCBCC \Rightarrow aaaBBBCCC$$

Portanto, $S \Rightarrow^* a^nB^nC^n$. Depois usa-se (4) uma vez para chegar a $S \Rightarrow^* a^nbB^{n-1}C^n$. Aí usa-se a produção (5) n-1 vezes: $S \Rightarrow^* a^nb^nC^n$. Finalmente, usa-se a produção (6) uma vez e a produção (7) n-1 vezes para chegar a $S \Rightarrow^* a^nb^nc^n$. É necessário mostrar também que as cadeias $a^nb^nc^n$, $n \geq 1$, são as únicas cadeias terminais em L(G).

3.2. Máquinas de Turing

Nesta parte, vamos relacionar a teoria das linguagens formais com a teoria abstrata da computação. Como se pode caracterizar o conjunto de funções computadas por programas de computador? Esta questão - quais funções são realizáveis por algoritmos e quais não são? - tem suas raízes mais diretas no trabalho de Alan Turing nos anos 30. Usando, o que agora se chama de modelo de máquina de Turing, Turing mostrou que certos problemas naturais em computação não podem ser computados por nenhum algoritmo, real ou imaginado. Realmente, Turing mostrou apenas que estes problemas não são calculáveis especificamente por máquinas de Turing; mais tarde as investigações de outros pesquisadores levaram a crença geral que a computabilidade de Turing é sinônimo da computabilidade em qualquer outro sistema de especificação de algoritmo suficientemente poderoso.

Vamos começar esta parte introduzindo as máquinas de Turing e então apresentando os principais resultados de Turing, a universalidade das máquinas de Turing e a insolubilidade do problema do *halting*. Na sua forma mais geral, o último resultado diz que nenhum algoritmo pode corretamente decidir se um programa de computador arbitrário pára em uma entrada arbitrária. Vamos mostrar como codificar este resultado na teoria da linguagem. Usando esta codificação vamos provar vários resultados, incluindo a asserção de que nenhum algoritmo pode corretamente decidir se uma gramática livre de contexto arbitrária é ambígua.

Uma **máquina de Turing** *T* pode ser descrita (Figura 1) como um controle de estados finitos equipado com um dispositivo de armazenamento externo na forma de uma fita finita que pode ser estendida indefinidamente em ambas as direções.

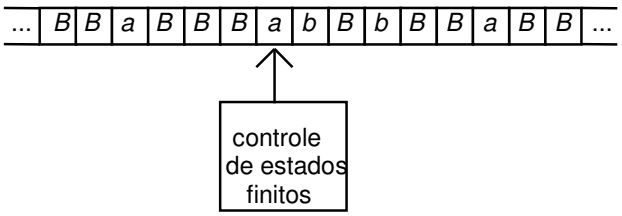


Figura 1. Uma máquina de Turing: A fita pode ser estendida indefinidamente pela adição de *B*'s (brancos) em qualquer lado.

Como mostrado na Figura 1, a fita é dividida em quadrados. Cada quadrado da fita pode estar em branco ou pode carregar qualquer símbolo de um alfabeto finito Σ de fita especificado. Por conveniência, um símbolo especial (aqui, a letra *B*) não em Σ é reservado para denotar um quadrado em branco na fita. Então, na Figura 1, todos os quadrados estão em branco exceto cinco, que possuem os símbolos *a* ou *b*. O controle de estados finitos é acoplado à fita através de uma cabeça de leitura/escrita. A qualquer instante dado, a cabeça estará percorrendo um quadrado da fita e o controle de estados finitos estará em um estado. Dependendo deste estado e do símbolo no quadrado percorrido pela cabeça da fita, a máquina irá, em um passo, fazer o seguinte:

(1) Entrar em um novo estado do controle de estados finitos;
(2) Sobrescrever um símbolo no quadrado percorrido (é possível sobrescrever o mesmo símbolo e portanto deixar o quadrado da fita sem mudar, ou sobrescrever um *B* e "apagar" o símbolo da fita);
(3) Deslocar a cabeça para esquerda ou para a direita um quadrado, ou não deslocar nada.

Como o alfabeto da fita é finito e o controle de estados finitos tem finitamente muitos estados, a operação de uma máquina de Turing pode ser completamente descrita por um conjunto finito de quíntuplas da forma (estado antigo, símbolo percorrido, estado novo, símbolo escrito, direção de movimento). No exemplo seguinte, usa-se nomes mnemônicos para os estados.

1. Exemplo. A seguinte máquina de Turing aceita o conjunto $\{a^{2n} \mid n > 0\}$ de cadeias de comprimento par de *a*'s: quando começa no estado "par" no *a* mais a esquerda, com a configuração de fita inicial *Baaa ... aaaB* ela irá terminar no estado "aceitação" apenas no caso do número de *a*'s ser par. Os indicadores direcionais *R* e *N* especificam "mova a direita um quadrado" e "não mova," respectivamente. (Um *L* especifica um movimento a esquerda.) A qualquer momento esta máquina está no estado "par" se ela viu uma cadeia de comprimento par de *a*'s. Aqui está a máquina:

(par *a* ímpar *B R*)
(ímpar *a* par *B R*)
(par *B* aceitação *B N*)

Quando ela alcança primeiro um *B* ela muda para o estado "aceitação" se ela viu um número par (incluindo 0) de *a*'s mas de outra forma ela irá parar porque não há quíntupla para guiar seu próximo movimento.

Por outro lado, a máquina com quíntuplas:

(par *a* ímpar *B R*)
(ímpar *a* par *B R*)
(par *B* aceitação *B N*)
(ímpar *B* ímpar *B R*)

tem a propriedade de que em cadeias de comprimento par de *a*'s ela pára no estado de aceitação, mas em cadeias de comprimento ímpar ela "roda" para sempre. Isto ilustra como as máquinas de Turing são realmente diferentes dos Autômatos Finitos (AFDs). Em particular, as computações da máquina de Turing podem fornecer resultados indefinidos - computações que nunca terminam. Tal comportamento não é possível com AEFs.

Assim como com os AEFs, pode-se dar uma representação equivalente das quíntuplas no Exemplo 1 usando uma tabela de estado, como se segue:

estado antigo	símbolo percorrido	
	a	*B*
par	ímpar *B R*	aceitação *B N*
ímpar	par *B R*	

A fim de tornar o comportamento da máquina determinístico, necessita-se que quando duas quíntuplas têm a mesma combinação (estado antigo, símbolo de estado percorrido), então as duas quíntuplas são idênticas.

Vamos agora dar a definição formal de uma máquina de Turing e a linguagem que ela aceita.

2. Definição. Uma máquina de Turing M é uma quíntupla $M = (Q, \Sigma, q_0, q_a, \delta)$, onde

Q é um conjunto finito de estados;
Σ é um alfabeto finito da fita. Escreve-se $\Sigma' = \Sigma \cup \{B\}$ para Σ aumentado por um símbolo de fita distinto B, o símbolo branco;
q_0 é o estado inicial distinto;
q_a é o estado de aceitação distinto;
δ é a função (parcial) de transição de estado,

$$\delta: Q \times \Sigma' \rightarrow Q \times \Sigma' \times \{L, N, R\}$$

sujeita a condição que $\delta(q_a, x)$ não é definida para nenhum x em Σ'. Pode-se representar δ por uma lista de quíntuplas, com $(q\ x\ q'\ x'\ D)$ na lista apenas no caso em que $\delta(q, x) = (q', x', D)$.

Na Figura 1, pode-se descrever a configuração da máquina de Turing M especificando

(i) a cadeia w_1 impressa na fita a esquerda da cabeça de leitura/escrita;
(ii) o estado corrente q; e
(iii) a cadeia w_2 na fita, começando na cabeça de leitura/escrita e movendo para a direita.

Pode-se resumir isto na cadeia

$$w_1 q w_2$$

que é chamada de *descrição instantânea* (DI) de *M*. Note que w_1 e w_2 *não* são únicos - a cadeia $w_1 w_2$ deve conter todos os quadrados não brancos da fita de *M*, mas ela pode ser estendida adicionando brancos em qualquer lado: $1q_0 1B11$ e $BB1q_0 1B11BBB$ são duas formas de escrever a mesma DI.

As máquinas de Turing têm fitas, cabeças de leitura/escrita, etc., e não podem obviamente serem descritas usando a maquinaria da teoria da linguagem. As descrições instantâneas, por outro lado, são realmente cadeias de símbolos e como tais podem ser manipuladas por uma gramática formal. Vamos ligar a teoria da linguagem e a noção de computabilidade da máquina de Turing mais tarde nesta parte, mostrando como uma gramática rescrevendo uma descrição instantânea pode confiavelmente seguir a trilha das transições de uma computação da máquina de Turing.

3. Definição. Seja *M* uma máquina de Turing com conjunto de estados *Q* e alfabeto de fita aumentado Σ'. Então uma descrição instantânea (DI) de *M* é uma cadeia $w_1 q w_2$ do conjunto $(\Sigma')^* \cdot Q \cdot (\Sigma')^*$. Cada DI $w_1 q w_2$ é equivalente às DIs $Bw_1 q w_2$ e $w_1 q w_2 B$.

Dizemos que *M pára em* $w_1 q w_2$ se nem $w_1 q w_2$ nem $w_1 q w_2 B$ for da forma $w_1 qxw$ para um par (q, x) para o qual $\delta(q, x)$ é definido. Se *M* não pára para $w_1 q w_2$, definimos como as DIs são transformadas pela ação de *M*. Primeiro, escreve-se $w_1 q w_2$ como $\neg w_1 yqx \neg w_2$ onde $\neg w_1 y = w_1$ se $w_1 \neq \lambda$, e $\neg w_1 = \lambda$, $y = B$ caso contrário; e $x \neg w_2 = w_2$ se $w_2 \neq \lambda$, e $x = B$, $\neg w_2 = \lambda$ caso contrário. Define-se então a *relação próxima DI*

$$w_1 q w_2 \Rightarrow w_1' q' w_2'$$

como se segue:

(i) se $\delta(q, x) = (q', x', L)$ então

$$w_1' q' w_2' = \neg w_1 q' y x' \neg w_2$$

(ii) se $\delta(q, x) = (q', x', N)$ então

$$w_1' q' w_2' = \neg w_1 \, y q' x' \neg w_2$$

(iii) se $\delta(q, x) = (q', x', R)$ então

$$w_1' q' w_2' = \neg w_1 \, y x' q' \neg w_2.$$

Como de costume, estende-se $\Rightarrow$ para formar seu fechamento reflexivo e transitivo $\Rightarrow^*$ tal que

$$\alpha \Rightarrow^* \alpha'$$

significa que *M* pode transformar a DI α na DI α' através de 0 ou mais transições de DI sucessoras como descrito em (i)-(iii) anteriormente.

Para uma máquina de Turing *M* sobre o alfabeto Σ, define-se $T(M)$, o conjunto aceitação de *M*, como o conjunto das cadeias *w* sobre Σ tal que se *w* é escrita em uma fita em branco e *M* começa processando no estado q_0 no símbolo mais a esquerda de *w*, então *M* eventualmente pára no estado de aceitação. Formalmente:

4. Definição. O *conjunto de aceitação* $T(M)$ de uma máquina de Turing *M* sobre o alfabeto Σ é o conjunto

$$T(M) = \{ w \in \Sigma^* \mid \text{existe } w_1, w_2 \in (\Sigma')^* \text{ tal que } q_0 w \Rightarrow^* w_1 q_a w_2 \}.$$

5. Exemplo. A seguinte máquina de Turing aceita a linguagem $\{ a^n b^n c^n \mid n > 0 \}$. Outra vez vamos usar nomes de estado mnemônicos na descrição da máquina.

(i) Quando a máquina lê um *a*, ela o substitui por #, então se move para a direita para achar um *b*:

(inicia *a* apaga-b # *R*)
(apaga-b *a* apaga-b *a R*)
(apaga-b # apaga-b # *R*)

Note que ela parará se encontrar um *c* enquanto estiver no estado apaga-b.

(ii) Quando a máquina, no estado apaga-b, lê um *b*, ela o substitui por #, então se move para a direita para achar um *c*:

(apaga-b *b* apaga-c # *R*)
(apaga-c *b* apaga-c *b R*)
(apaga-c # apaga-c # *R*)

(iii) Se ela acha um *c* (não tendo parado por um *a* no caminho), ela então o substitui por um # e se move para a esquerda para achar um *a*, a partir de onde ela repete o ciclo inteiro, partindo de (i).

(apaga-c *c* busca-a # *L*)
(busca-a # busca-a # *L*)
(busca-a *b* busca-a *b L*)
(busca-a *a* apaga-b # *R*)

(iv) Entretanto, se nenhum *a* permanece, deve-se então checar que todos os *b*'s e *c*'s foram apagados. Apenas neste caso ela irá aceitar a cadeia inicial:

(busca-a *B* cheque *B R*)
(cheque # cheque # *R*)
(cheque *B* aceita *B N*)

Note que se uma cadeia não está na forma $a^n b^n c^n$, então o estado de aceitação nunca é alcançado, e de fato a máquina descansará em alguns outros estados. Por exemplo, na entrada b^{10}, a máquina nunca deixa o estado inicial.

3.3. Autômatos Limitados Linearmente

Vamos considerar agora os *autômatos limitados linearmente*. Estes dispositivos são máquinas de Turing não determinísticas com a restrição de que a cabeça de leitura/escrita não tem a permissão de deixar os quadrados da fita que possuem a entrada original. Mostraremos que estas máquinas são precisamente os aceitadores para linguagens sensíveis ao contexto.

1. Definição. Um *autômato limitado linearmente* (*ALL*) é uma máquina de Turing não determinística que deve operar dentro do espaço definido pela colocação original da entrada na fita.

2. O Lema do Alfabeto. *Seja L uma linguagem tal que para qualquer* $w \in \Sigma^*$, *a pertinência de w em L pode ser decidida por uma máquina de Turing M usando* $k \cdot |w|$ *quadrados da fita, onde k é um inteiro positivo independente de w. Então L pode ser reconhecida por alguma outra máquina de Turing M' usando apenas* $|w|$ *quadrados da fita*.

Então o *lema do alfabeto* estabelece que se alguma máquina de Turing pode reconhecer uma linguagem *L* no espaço que é uma função linear do comprimento de uma cadeia de entrada, então alguma outra máquina de Turing pode de fato reconhecer *L* dentro do espaço exatamente igual do comprimento da entrada. O relacionamento linear identificado no lema justifica o nome *autômatos limitados linearmente*.

Assumimos, quando conveniente, que a entrada da máquina de Turing é limitada a esquerda por um símbolo \$ e a direita por um ¢. Assumimos que estes símbolos são não apagáveis e que a cabeça não pode mover sobre eles.

3. Teorema. *Se* $L = T(M)$ *para algum ALL M, então L é gerável por alguma gramática sensível ao contexto*.

PROVA: Vamos assumir que $\lambda \notin L$. Para $\lambda \in L$ usamos o teorema para $L - \{\lambda\}$ e com cuidado adicionamos uma produção $S \rightarrow \lambda$ depois do fato. Para estabelecer o resultado assumindo $\lambda \notin L$, primeiro escrevemos uma GSC para a linguagem

$$\{w \$ q_0 w ¢ \mid w \in \Sigma^*\}$$

A cadeia a direita do símbolo \$ é de comprimento $|w| + 2$. Já que a máquina neste caso é um ALL, as DIs nunca serão maiores que isto. Entretanto, no término da simulação de gramática de *M*, não podemos apagar todos os símbolos a direita de \$, porque as regras de apagamento não são sensíveis ao contexto. Ao invés disto, se q_a aparece, fazemos o seguinte:

(1) Primeiro, convertemos todos os símbolos a direita de \$ para um novo símbolo, digamos "/", tal que ficamos com a seguinte cadeia: $w \$ /^n q_a ¢$

(2) Se $w = x_1x_2...x_n$, usamos regras sensíveis ao contexto para construir a seguinte cadeia: r / x_1 / x_2 / ... / x_n # q (onde r e q são novos símbolos terminais);
(3) Apagamos todas as ocorrências de r, /, # e q. ◊

Nosso próximo resultado remete ao estudo de homomorfismos da parte 2 para refinar nosso entendimento de linguagens sensíveis ao contexto. Freqüentemente, na escrita de gramáticas sensíveis ao contexto a abordagem mais natural é usar um sistema de símbolos marcadores especiais para guiar o processamento gramatical. E uma gramática é escrita que é quase certa, mas deixa símbolos marcadores desnecessários nas cadeias da linguagem, marcadores que aparentemente não podem ser removidos exceto por uma regra lambda óbvia mas ilegal. O próximo teorema estabelece que regras de apagamento limitadas que eliminam tais marcadores são legais, tal que os marcadores permanecem "esparsos" nas cadeias de uma linguagem.

4. Definição. Um homomorfismo $h: \Sigma \rightarrow Y$ é *k-limitado em* $L \subset \Sigma^*$ se para toda w em L, $h(w)$ apaga (isto é, envia a λ) no máximo k símbolos consecutivos de w. Então o mapa $h(a) = a$, $h(b) = \lambda$ é 2-limitado na linguagem $(ab)^*$ mas não é k-limitado para nenhum k na linguagem $\{a^nb^n \mid n > 0\}$.

Enquanto as linguagens sensíveis ao contexto não são fechadas sob homomorfismos arbitrários, elas são fechadas sob homomorfismos k-limitados.

5. Teorema. *As linguagens sensíveis ao contexto são fechadas sob homomorfismos k-limitados*.

6. Exemplo. A seguinte gramática gera todas as cadeias da forma $\{wmwr \mid w \in \{0,1\}^+\}$. Os símbolos m e r devem ser considerados como símbolos marcadores terminais que identificam o centro e final a direita da cadeia. O homomorfismo h que os apaga e deixa todos os outros terminais fixos é claramente k-limitado para $k = 1$. Portanto, a linguagem $\{ww \mid w \in \{0,1\}^*\}$ é sensível ao contexto. Aqui está a gramática.

$$S \Rightarrow Amr$$
$$Am \Rightarrow 0mT_0 | 1mT_1$$
$$T_00 \Rightarrow 0T_0$$
$$T_01 \Rightarrow 1T_0$$
$$T_11 \Rightarrow 1T_1$$
$$T_10 \Rightarrow 0T_1$$
$$T_0r \Rightarrow 0r | 0Dr$$
$$T_1r \Rightarrow 1r | 1Dr$$
$$1D \Rightarrow D1$$
$$0D \Rightarrow D0$$
$$mD \Rightarrow Am$$