Determinismo em Autômatos de Estados Finitos Adaptativos

A. A. de Castro Jr., J. J. Neto, H. Pistori

***Resumo*— O determinismo é uma das características que devem ser observadas durante o projeto de dispositivos formais, quando se busca bom desempenho. No caso dos formalismos adaptativos, existe uma séria dificuldade relacionada com a manutenção do determinismo, pois, devido ao seu comportamento automodificável, é muito difícil garantir que o determinismo será mantido durante toda a operação do dispositivo. Este trabalho apresenta, formalmente, o conceito de determinismo para uma classe de dispositivos adaptativos que é particularmente importante devido à sua relativa simplicidade, a dos autômatos de estados finitos adaptativos. Partindo do estudo deste formalismo específico, além da definição de uma subclasse de dispositivos adaptativos determinísticos, foi relacionado um conjunto de restrições que devem ser obedecidas para a obtenção de autômatos de estados finitos adaptativos determinísticos.**

Palavras-chave*— *Autômatos de estados finitos adaptativos, Não-determinismo, Dispositivos adaptativos.

I. Introdução

ALGUNS trabalhos recentes discutem a aplicação dos formalismos adaptativos em problemas complexos da teoria da computação [1], construção de compiladores [2], inteligência artificial [3], [4], processamento de linguagem natural [5], robótica [6], entre outros. Além disso, os autômatos adaptativos têm sofrido mudanças com o objetivo de simplificar sua formulação original e torná-lo um dispositivo mais prático e eficiente [7].

O determinismo, por exemplo, possui relação direta com o bom desempenho das aplicações que fazem uso dos dispositivos formais projetados. Em [8], foi realizado um estudo comparativo sobre a complexidade de autômatos de estados finitos determinísticos e não-determinísticos. Em [9], foi destacada a importância da compreensão do relacionamento existente entre os tempos de computação determinística e não-determinística, uma vez que os algoritmos desenvolvidos para muitos problemas práticos, apesar de serem inerentemente não-determinísticos, são executados em máquinas que atuam de forma determinística. Allauzen e Mohri [10], [11] desenvolveram um algoritmo eficiente para preé-determinização de transdutores que podem ser aplicados em reconhecimento de voz e processamento de imagens. Apesar disso, a maior parte das soluções encontradas para o processamento de linguagens naturais sempre se apresenta com características de não-determinismo ou de ambigüidade, as quais são resolvidas através da simulação seqüencial do paralelismo inerente aos processos em questão, em tempo exponencial [1].

Quando lidamos com formalismos convencionais, o único requisito para que haja determinismo é que uma e somente uma regra de transição possa ser aplicada a cada passo de operação, seja qual for a configuração corrente do dispositivo. No caso dos formalismos adaptativos, existe uma séria dificuldade relacionada com os seus aspectos de determinismo: sua natureza dinâmica não nos permite determinar a *priori*, com facilidade, se o dispositivo é ou não determinístico.

Para simplificar tal problema, neste trabalho, são discutidos alguns aspectos relacionados com a manutenção do determinismo, usando-se uma instância particularmente importante dos dispositivos adaptativos: os autômatos de estados finitos adaptativos, definidos em [7]. Além disso, é formalizado o conceito de determinismo sobre dispositivos adaptativos e, com base nele, definida a subclasse dos dispositivos adaptativos determinísticos.

II. Autômato de Estados Finitos Adaptativo

A característica mais importante dos dispositivos adaptativos é que a cada uma de suas regras podem ser associadas ações adaptativas. Isso permite a execução de operações de edição que modificam dinamicamente o conjunto de regras responsável por determinar o comportamento do dispositivo. Portanto, pela aplicação de alguma dessas regras adaptativas, o comportamento do dispositivo como um todo pode ser modificado, algumas vezes dramaticamente. Assim, caso a execução de ações arbitrárias for permitida, não se poderá garantir o determinismo do dispositivo resultante, agora definido pelo conjunto de regras resultantes da aplicação das ações adaptativas.

Os autores agradecem o apoio financeiro da Fundação de Apoio ao Desenvolvimento do Ensino, Ciência e Tecnologia do Estado de Mato Grosso do Sul (FUNDECT) e da Financiadora de Estudos e Projetos (FINEP). Os autores também agradecem a Universidade Católica Dom Bosco (UCDB) pelo apoio indireto à realização deste trabalho e ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela bolsa de produtividade concedida ao Prof. Dr. Hemerson Pistori.

A. A. Castro Jr. atua no Departamento de Ciências Exatas (DEX) do Campus de Coxim (CPCX) da Universidade Federal de Mato Grosso do Sul (UFMS), Av. Márcio de Lima Nantes, S/N – Vila da Barra – CEP: 79400-000 - Coxim/MS, Brasil (endereço eletrônico: amaury@cpcx.ufms.br).

J. J. Neto atua no Departamento de Engenharia de Computação e Sistemas Digitais (PCS) da Escola Politécnica (POLI) da Universidade de São Paulo (USP), Av. Prof. Luciano Gualberto, Trav. 3, N. 158 - CEP: 05508-900 - São Paulo/SP, Brasil (endereço eletrônico: joao.jose@poli.usp.br).

H. Pistori atua no Grupo de Pesquisa em Engenharia e Computação (GPEC) da Universidade Católica Dom Bosco (UCDB), Av. Tamandaré, 6000 - Jardim Seminário - CEP: 79117-900 - Campo Grande/MS, Brasil (endereço eletrônico: pistori@ucdb.br).

Em sua formulação geral, os dispositivos adaptativos são caracterizados através de duas camadas conceituais: o dispositivo subjacente e o mecanismo adaptativo [12]. O dispositivo subjacente é representado por um dispositivo guiado por regras (formalismo convencional, não-adaptativo) e opera movendo-se de uma configuração para outra, sucessivamente, em resposta a estímulos recebidos em sua entrada. Tais movimentos são definidos por um conjunto de regras que mapeiam cada possível configuração em uma configuração seguinte correspondente. O mecanismo adaptativo responde pelas mudanças incrementais no comportamento do dispositivo subjacente. As propriedades de automodificação são representadas pelas funções e ações adaptativas, capazes de alterar o conjunto de regras do dispositivo subjacente, determinando, conseqüentemente, seu comportamento a partir de então.

Os autômatos adaptativos podem ser enquadrados entre os principais representantes dessa classe de dispositivos formais. Tal formalismo utiliza, como dispositivo subjacente, o autômato de pilha estruturado [13]. Pistori [7], ao propor o autômato de estados finitos como dispositivo subjacente, elimina a "pilha" existente no autômato adaptativo inicialmente definido, simplificando o estudo do comportamento de dispositivos com tais características, em particular dos aspectos de determinismo.

Um autômato de estados finitos adaptativo pode, portanto, ser formalmente definido como um par ordenado $A = (ND_0, MA)$, onde ND_0 corresponde ao autômato de estados finitos subjacente inicial e *MA*, ao mecanismo adaptativo. As Seções II-A e II-B descrevem, em detalhes, essas componentes dos autômatos de estados finitos adaptativos.

A. Dispositivo Subjacente (autômato de estados finitos não-adaptativo)

Nos autômatos de estados finitos adaptativos, o dispositivo subjacente é o autômato de estados finitos (não-adaptativo). Instanciando para este caso a formulação proposta em Neto [12], resulta o seguinte:

Definição 1: Seja $D = (\Sigma, Q, \delta, q_0, F)$ a 5-upla que representa um autômato de estados finitos. Esse formalismo pode ser definido como o dispositivo subjacente de um autômato de estados finitos adaptativo pela 6-upla $ND = (C, NR, \Sigma, c_0, A, NA)$, onde:

- NR é o conjunto de regras que definem ND através da relação $NR \subseteq C \times S \times C \times NA$. As regras $r \in NR$ têm a forma $r = (c_i, s, c_j, z)$, significando que, como resposta a algum estímulo de entrada $s \in \Sigma$, r move-se da configuração corrente c_i para c_j, consumindo o estímulo s e produzindo o símbolo de contexto $z \in NA$;
- C é o conjunto de todas as possíveis configurações de *ND*;
- Uma configuração $c_i \in C$ é representada por um par $c_i = (q_j, \sigma\alpha)$, onde $q_j \in Q$ é o estado corrente; $\alpha \in \Sigma^*$ simboliza a parte da cadeia ainda não consumida e $\sigma \in \Sigma \cup \{\varepsilon\}$ é o próximo símbolo da cadeia de entrada a ser tratado;
- $c_0 = (q_0, \omega) \in C$ ($\omega \in \Sigma^*$) é a sua configuração inicial;
- $A \subseteq C$ é o subconjunto de suas configurações de aceitação. Uma configuração de aceitação $c_a \in C$ é representada por uma 2-upla $c_a = (q_f, \varepsilon)$, onde $q_f \in F$ é o estado corrente e ε indica que toda a cadeia foi consumida, não restando mais símbolos a serem tratados.

A componente *NA* é útil em operações de transdução, podendo, por exemplo, ser utilizada na obtenção de analisadores sintáticos [13]. Entretanto, o conjunto $NA \subset \Sigma$, que contém os símbolos de contexto que podem ser inseridos na cadeia de entrada e mapeados em chamadas de procedimentos [12], não foi utilizado por Pistori [7] na formalização original dos autômatos de estados finitos adaptativos. Dessa forma, para tratar questões relacionadas com o determinismo, será considerado, por conveniência, que $NA = \{\varepsilon\}$.

Com isso, um passo de operação do dispositivo subjacente é executado da mesma forma que um passo de operação de um autômato de estados finitos, que corresponde à aplicação de uma das regras definidas em *NR*, tendo como estímulos de entrada os símbolos pertencentes a Σ. A execução de um passo de operação leva o dispositivo subjacente da configuração c_i para a configuração c_j (denotado por $c_i \Rightarrow c_j$).

Nos dispositivos adaptativos, define-se um passo de operação adaptativo k como um superconjunto de passos de operação do dispositivo subjacente no instante k. Portanto, no autômato de estados finitos adaptativo, o passo de operação muda juntamente com as modificações no conjunto de regras *NR* do autômato finito subjacente. Durante um passo de operação adaptativo $k \geq 0$, o autômato de estados finitos adaptativo obedecerá ao comportamento definido por NR_k até a execução de alguma ação adaptativa não-vazia que iniciará o passo de operação $k + 1$, evoluindo seu conjunto de regras para NR_{k+1}. Tais mudanças são executadas por uma ação adaptativa, que faz parte do mecanismo adaptativo definido a seguir, na Seção II-B.

B. Mecanismo Adaptativo

Seja $AD = (ND_0, MA)$ um autômato de estados finitos adaptativo e seja $ND_k = (C_k, NR_k, \Sigma, c_k, A, NA)$ o dispositivo subjacente correspondente ao passo de operação adaptativo $k \geq 0$ do autômato, para uma dada cadeia de entrada $w \in \Sigma^*$. A execução de alguma ação adaptativa não-vazia evolui ND_k para ND_{k+1}. Com isso, *AD* inicia a sua operação no passo $k + 1$, criando o conjunto NR_{k+1} como versão modificada de NR_k. *AD* seguirá o comportamento definido por ND_{k+1}, até que uma futura execução de ação adaptativa não-vazia provoque um novo passo de operação adaptativo.

Para tanto, o dispositivo adaptativo correspondente incorpora à definição geral do mecanismo subjacente os conjuntos *BA* e *AA*, das ações adaptativas, respectivamente, anteriores e posteriores. As ações adaptativas *anteriores* se aplicam ao conjunto de regras do dispositivo *antes* da mudança de configuração. As *posteriores* são aplicadas *após* a mudança de configuração. No caso específico dos autômatos de estados finitos adaptativos, por definição, $AA = \varnothing$ [7].

A cada passo de operação adaptativo, um novo conjunto de regras $AR = BA \times NR \times AA$ é definido. Esse novo conjunto de regras associa a cada regra não-adaptativa um par de ações adaptativas (cada uma delas possivelmente vazia). Tais ações são aplicadas em qualquer passo de operação k a cada regra em $NR_k \subseteq NR$ e são implementadas através de *funções adaptativas*.

As funções adaptativas especificam as modificações a serem realizadas no dispositivo subjacente, quando a ação adaptativa correspondente é executada. O funcionamento das funções adaptativas lembra o das funções paramétricas em um programa de computador. O corpo de uma função adaptativa consiste de uma lista de *ações adaptativas elementares* (ou, simplesmente, ações elementares) e de duas ações adaptativas, inicial e final (opcionais), que são executadas, respectivamente, antes e após a execução do conjunto de ações elementares especificado. Ações elementares podem ser de três tipos: as *de consulta* possibilitam a busca de padrões na estrutura definida pelas regras da camada subjacente; as *de inserção* e de *remoção* determinam, respectivamente, as regras a serem inseridas ou removidas do conjunto corrente de regras do dispositivo subjacente:

Definição 2: Uma função adaptativa é uma 9-upla $AF = (F, P, V, G, C, D, I, S, E)$, na qual:

- F é o nome da função adaptativa.
- P é a lista $(r_0,r_1,...,r_m)$ de parâmetros formais.
- V é a lista $(v_1,v_2,...,v_n)$ de identificadores de variáveis.
- G é a lista $(g_1*,g_2*,...,g_p*)$ de identificadores de geradores.
- C é a lista de ações elementares de consulta.
- D é a lista de ações elementares de remoção.
- I é a lista de ações elementares de inserção
- S é uma *ação adaptativa inicial*, opcional, a ser executada antes de F.
- E é uma *ação adaptativa final*, opcional, a ser executada depois de F.

Todas as ações adaptativas são da forma (f, p), na qual:

- f é o nome de uma função adaptativa.
- p é a lista $(p_0,p_1,...,p_k)$ de argumentos a serem passados para f.

As ações elementares possuem o seguinte formato:

(? | − | +) [*padrão*]

onde *padrão* corresponde à representação de uma regra do dispositivo subjacente, contendo, possivelmente, nomes de variáveis e de geradores nas posições correspondentes aos elementos da regra. Além disso, os prefixos ?, – e + denotam as ações elementares de consulta, remoção e inserção, respectivamente. Um gerador opera como uma espécie de constante, cujo valor é gerado no início da execução da ação adaptativa, e se mantém inalterado enquanto durar sua execução. O valor de um gerador é único e diferente de qualquer símbolo em uso no dispositivo subjacente. Já as variáveis têm os seus valores atribuídos durante a execução das ações adaptativas elementares de consulta e remoção (estas executam uma consulta implícita). A atribuição de valor à variável se dá através de um mecanismo de busca de padrões que é responsável por essa funçãoatribuir os valores, tendo como base o conjunto de regras do dispositivo subjacente. Uma vez atribuídos, os valores de cada variável e de cada gerador não se alteram durante a execução da função adaptativa. Quando o mecanismo de busca de padrões não encontra qualquer regra na camada subjacente que satisfaça o formato determinado pela ação elementar de consulta e remoção, as variáveis são marcadas como indefinidas e todas as ações elementares que delas se utilizam são desconsideradas. Para as ações elementares de inserção, que são sempre executadas depois das ações de consulta e remoção, todas as variáveis referenciadas devem ter sido previamente instanciadas ou marcadas como indefinidas.

Definição 3: Um autômato de estados finitos adaptativo é uma 7-upla $AD = (C_k, AR_k, \Sigma, c_k, A, NA, BA)$ $(k \geq 0)$, onde:

- C_k é o conjunto de todas as possíveis configurações para o dispositivo subjacente ND no passo k. Para $k = 0$, tem-se em C_0 o conjunto inicial de todas as configurações válidas;
- Uma configuração $c_i \in C$ é definida da mesma forma que para o autômato de estados finitos subjacente;
- $c_k \in C_k$ é a sua configuração inicial no passo k. Para $k = 0$, tem-se em $c_0 \in C_0$ a configuração inicial para o autômato de estados finitos adaptativo correspondente;
- ε ("vazio") denota a ausência de qualquer elemento válido do conjunto correspondente;
- BA é o conjunto das ações adaptativas anteriores, contendo a ação nula ($\varepsilon \in BA$);
- AR é o conjunto de regras que definem AD através da relação $AR \subseteq BA \times C \times S \times C \times NA$. As regras $r_A \in AR$ têm a forma $r_A = (\psi, c_i, s, c_j, z)$, significando que, como resposta a algum símbolo de entrada $s \in \Sigma$, r_A inicialmente executa a ação adaptativa $\psi \in BA$. A execução de r_A é descontinuada após, eventualmente, a ação ψ ter eliminado r_A de AR. Caso contrário, aplica-se a regra subjacente (não-adaptativa) $r = (c_i, s, c_j, z) \in NR$, como descrito anteriormente, senão, nova regra adaptativa é selecionada para ser aplicada à configuração corrente;
- $A \subseteq C$ é o subconjunto de suas configurações de aceitação, o mesmo definido para o autômato de estados finitos subjacente.

Segundo Pistori [7], a utilização exclusiva de ações adaptativas anteriores não prejudica a capacidade expressiva do modelo, pois transições vazias, executadas *antes* de uma transição convencional, podem ser empregadas para simular, através de sua própria ação adaptativa posterior, uma ação adaptativa anterior que fosse eventualmente necessária na transição convencional mencionada.

C. Modo de Operação

Um autômato de estados finitos adaptativo opera, portanto,

através da execução de passos de operação adaptativos, a cada chamada de uma função adaptativa, e de passos de operação do dispositivo subjacente, similares aos passos de operação dos autômatos de estados finitos convencional (não-adaptativo). Esses passos de operação são definidos através do conjunto de regras de transição do autômato:

Definição 4: Toda *regra* de *transição* de um A-AEF possui o seguinte formato:

$$[(q, \sigma\alpha), \beta \vdash (q', \sigma'\alpha')]$$

onde:

- q e q' são, respectivamente, os estados de origem e destino da transição. Eventualmente, $q = q'$.
- $\sigma\alpha$ denota a parte da cadeia de entrada ainda não consumida antes da transição para o estado q'.
- $\sigma'\alpha'$ denota a parte da cadeia de entrada ainda não consumida após a transição para o estado q'.
- σ e σ' são átomos do alfabeto, correspondentes ao primeiro símbolo da cadeia ainda não consumida, explicitamente manipulados na transição especificada.
- $\beta \in BA$ é uma função adaptativa anterior. Se $\beta = \varepsilon$, então a transição é executada como em um AEF, e o conjunto de regras não é modificado.

Quando $\beta \neq \varepsilon$, executa-se um passo de operação adaptativo, o qual pode acarretar a introdução de não-determinismos no dispositivo subjacente, e essas situações podem depender da cadeia de entrada. No entanto, é possível definir um conjunto de restrições a serem impostas ao conjunto de regras que o definem, as quais sejam capazes de tornar o determinismo uma característica inerente ao autômato, para qualquer instância da cadeia de entrada. Essas questões estão discutidas em detalhes na Seção IV.

Quando $\beta = \varepsilon$, as transições são executadas como em um autômato de estados finitos.

III. Aspectos de Determinismo e Não-determinismo em Autômatos de Estados Finitos Adaptativos

Pode-se dizer que um autômato de estados finitos adaptativo é determinístico se e somente se, a cada passo de sua operação, o autômato de estados finitos subjacente permanece determinístico. Considerando a definição mais geral para a classe de dispositivos guiados por regras [12], tem-se:

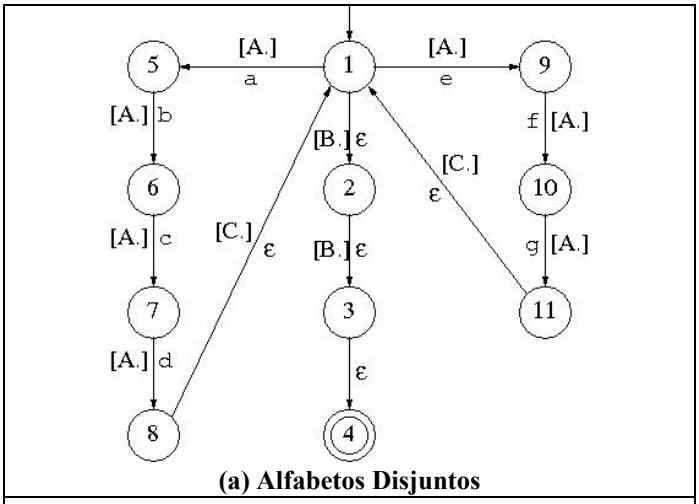
Definição 5: Um dispositivo adaptativo é dito *determinístico* se e somente se, a cada passo de sua operação, o mecanismo subjacente for determinístico.

Apesar de parecer simples e direto, no caso dos autômatos de estados finitos adaptativos, existem situações nas quais, dependendo da cadeia de entrada apresentada ao autômato, os não-determinismos nunca se manifestam durante a operação do dispositivo, embora estejam presentes.

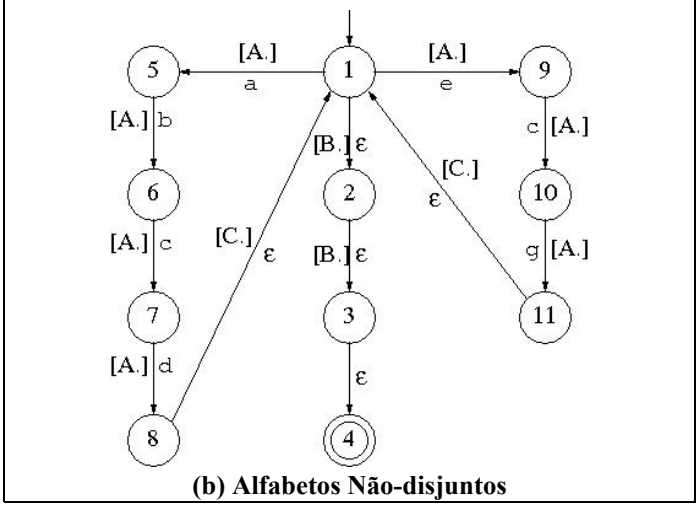
Um exemplo que ilustra esse aspecto do determinismo em tais dispositivos é o da solução compacta e eficiente, baseada em autômatos adaptativos, proposta em [1], para reconhecer uma linguagem formada a partir de um número finito de cadeias de alfabetos disjuntos, pela intercalação arbitrária (*merge*) dos símbolos das cadeias participantes do "merge".

As Figuras 1(a) e 1(b) mostram, respectivamente, um autômato de estados finitos adaptativo que reconhece o *merging* de duas seqüências curtas com alfabetos disjuntos [1] e um autômato de estados finitos adaptativo que faz o mesmo, mas usando seqüências com alfabetos não-disjuntos. Ambos operam da seguinte forma: todos os símbolos da cadeia de entrada são consumidos a partir do estado inicial. Cada transição que consome um símbolo de uma das seqüências está associada à ação adaptativa *A*, que, quando executada, exclui a transição associada e conecta o estado inicial à próxima transição correspondente àquela seqüência. Quando todos os símbolos de uma das seqüências tiverem sido consumidos, a ação adaptativa *C* será executada, removendo do autômato uma transição contendo uma chamada da função adaptativa *B*. Portanto, se os símbolos de todas as seqüências forem consumidos, todas as transições associadas à correspondente chamada da função *B* terão sido removidas e, dessa forma, o autômato alcançará o estado final, reconhecendo assim a cadeia de entrada. No entanto, se a ação adaptativa *B* chegar a ser executada durante a operação, a transição que levaria o autômato ao estado final será removida e, portanto, a cadeia de entrada, rejeitada. Vale ressaltar que, de acordo com o mecanismo de operação dos autômatos adaptativos [13], a ação *B* só será executada se não houver símbolo algum (de nenhuma das cadeias envolvidas na operação de *merging*) que possa ser consumido a partir do estado inicial do autômato nessa ocasião.

Na Figura 1(a), a operação do autômato adaptativo exemplificado é totalmente determinística, já que os alfabetos são disjuntos, impossibilitando que surja algum não-determinismo devido à operação de "merge" implementado por esse método. Na Figura 1(b), o não-determinismo do autômato só se manifesta quando, para casos particulares de cadeias de entrada, houver um conflito entre dois símbolos iguais, provenientes de cadeias diferentes. Nesses casos, o autômato não consegue determinar, diretamente, se o símbolo conflitante proveio de uma das cadeias ou da outra.



(a) Alfabetos Disjuntos



(b) Alfabetos Não-disjuntos

Fig. 1. Dois autômatos de estados finitos adaptativos que aceitam o *merging* de duas cadeias curtas. Em (a), o autômato adaptativo sempre opera deterministicamente, seja qual for a cadeia de entrada. Em (b), algum não-determinismo pode, eventualmente, se manifestar. Por exemplo, para a cadeia `aecbgcd`, a operação do autômato será determinística, o que não ocorre com a cadeia `aebcgcd`.

IV. Autômatos de Estados Finitos Adaptativos Permanentemente Determinísticos

O exemplo utilizado na seção anterior demonstra uma situação na qual, apesar de estar presente, o não-determinismo pode não se manifestar, para uma dada cadeia de entrada. Desse modo, o autômato pode operar de forma determinística, mesmo sendo potencialmente não-determinístico.

Seja A um autômato finito adaptativo determinístico que executa uma ação adaptativa durante a sua operação. Nesse caso, para garantir que o determinismo seja mantido, algumas restrições devem ser observadas durante a definição e a operação das ações elementares de consulta, remoção e inserção.

Definição 6: Um autômato de estados finitos adaptativos é dito **fortemente determinístico** quando seu autômato de estados finitos subjacente permanece determinístico durante qualquer ação adaptativa executada para uma dada cadeia de entrada $w \in \Sigma^*$.

Observe que as ações adaptativas elementares de consulta e remoção[1] não introduzem não-determinismos de forma direta. No entanto, os efeitos das consultas realizadas podem, ocasionalmente, introduzir não-determinismos através de ações de remoção e inserção que façam uso de variáveis utilizadas nas consultas. As seções seguintes estudam, separadamente, as situações em que as ações elementares, presentes nos autômatos de estados finitos adaptativos, podem influenciar o determinismo do autômato.

A. Ações Elementares de Consulta

Em um autômato de estados finitos adaptativo, seja AF uma função adaptativa descrita conforme a definição apresentada em [7]. Uma ação elementar de consulta $\varphi \in \rho$ tem a seguinte forma geral:

$$?[(q, s\alpha),\ \mathtt{B} \vdash (q', \alpha)]$$

Tais ações elementares executam consultas buscando, no atual conjunto de regras, um padrão similar ao que foi especificado na forma geral. O padrão encontrado será utilizado para preencher os valores das variáveis correspondentes. Dessa forma, não-determinismos podem ser introduzidos quando essas variáveis, devidamente preenchidas, são utilizadas em ações elementares de remoção e inserção. Em tais casos, para manter o determinismo, as seguintes condições devem ser observadas:

- Se $q \in \upsilon$: Quando o estado de origem q não é conhecido, uma forma de evitar a introdução de potenciais não-determinismos (quando a mesma é usada em ações elementares de inserção) é garantir que a variável q não seja preenchida com múltiplos valores. Embora seja muito restritiva, uma forma simples e direta é fazer com que s e q' sejam conhecidos. Caso isso não seja possível, deve-se eliminar o uso dessas variáveis em ações elementares de inserção na função adaptativa correspondente.
- Se $s \in \upsilon$: Quando s não é conhecido, a única restrição sobre as ações elementares de inserção é que o valor de q' seja único. Com isso, mesmo que a variável s seja preenchida com múltiplos valores, garante-se que não haverá risco da introdução de potenciais não-determinismos.
- Se $q' \in \upsilon$: Neste caso, a presença de um não-determinismo é imediatamente constatada quando $\exists[q, s]$, tal que à variável q' são atribuídos múltiplos valores. Caso contrário, q' terá sempre um valor único.

Se todas as restrições acima forem satisfeitas, pode-se garantir que os efeitos da execução das ações elementares de consulta não introduzirão não-determinismos. Na seção seguinte, são analisadas as situações ocasionadas exclusivamente pela execução das ações elementares de inserção.

[1] No caso das ações elementares de remoção, por definição, podem ser realizadas consultas implícitas [7].

B. *Ações Elementares de Inserção*

Ações elementares de inserção possuem a seguinte forma geral:

$$+[(q, s\alpha), \mathtt{B} \vdash (q', \alpha)]$$

no momento de sua aplicação, para cada transição na forma

$$[(q'', s''\alpha), \mathtt{B}' \vdash (q''', \alpha)]$$

as seguintes situações devem ser verificadas:

1) $q \notin Q$: Neste caso, q está representado por um gerador. Com isso, não há conflitos e o determinismo será mantido.
2) $q \notin Q$: Neste caso, poderá haver conflitos entre as regras de transição correntes e a regra incluída, resultando na introdução de não-determinismos nas seguintes situações: (1) q' não é um conjunto unitário (efeito de uma ação de consulta); (2) q' possui valor unitário, mas, no momento da aplicação da ação elementar de inserção, para algum q'', o autômato possui regras na forma (ocasionalmente, $\mathtt{B}' = \varepsilon$):

$$[(q, s\alpha), \mathtt{B}' \vdash (q'', \alpha)]$$

3) $q \in Q$ e $s \neq s''$: Se $!\exists[q,s,q'']\forall q''$, então o autômato transita com um símbolo s que não é consumido pelas regras existentes. Portanto, o determinismo será mantido, pois estão sendo incluídas transições que especificam valores distintos daqueles utilizados em transições existentes.

4) $q \in Q$ e $s = s''$: Nesta situação, pode haver a inclusão de não-determinismo caso o autômato possua transições que utilizem o mesmo símbolo s que é consumido por alguma regra já existente, uma vez que, na configuração corrente do autômato, uma das transições existentes satisfaz o lado esquerdo daquela que está sendo inserida. Nesse caso, as condições devem ser satisfeitas:

   a) $\mathtt{B} = \mathtt{B}'$: Para uma dada configuração, se as ações adaptativas de duas transições igualmente aplicáveis não forem idênticas, então haverá a introdução de um não-determinismo, caso o autômato venha a conter ambas as transições. Isso se deve ao perigo potencial de que as duas ações adaptativas gerem efeitos incompatíveis.
   b) $q' = q'''$: Nesta situação, se $q' \neq q'''$, estaremos inserindo uma transição que consome o mesmo símbolo e alcança um estado distinto do especificado em transição já existente no autômato. Isso caracteriza um não-determinismo, exceto quando ambas as transições (a corrente e a que está sendo inserida) são as mesmas (portanto, não-determinismos não serão incluídos).

Exceto para os casos (1) e (3), que inserem novas regras com segurança, se cumpridas as demais condições, além da impossibilidade de introdução de não-determinismos por parte das ações adaptativas elementares de inclusão, a regra a ser incluída já estará fazendo parte do conjunto de transições do autômato e, portanto, não afetará o determinismo do autômato.

V. CONCLUSÕES

As recentes pesquisas realizadas sobre os autômatos adaptativos [13] vêm provocando mudanças diversas em seu modo de operação, simplificando a sua formulação original e tornando-os um dispositivo formal mais prático e eficiente. Uma das mudanças mais significativas resultou na definição dos autômatos de estados finitos adaptativos [7]. Dessa forma, tal formalismo foi utilizado para ilustrar, definir e apresentar alguns aspectos relacionados com o determinismo em dispositivos adaptativos.

A importância dos aspectos práticos do determinismo tem recebido atenção especial, visto que o desempenho está diretamente relacionado com tais aspectos [11], [10], [9], [8]. Este trabalho registra e ilustra os recentes avanços obtidos nessa área, visando a definição de uma metodologia de projeto para usuários dos dispositivos adaptativos. A inserção dinâmica de não-determinismos através do mecanismo adaptativo do dispositivo foi discutida e algumas regras para o projeto de funções adaptativas foram propostas, visando garantir a permanente execução determinística do autômato de estados finitos adaptativo. Por sua vez, os resultados obtidos podem ser facilmente estendidos e generalizados para a superclasse dos dispositivos adaptativos.

Os autores esperam que os resultados deste trabalho possam motivar estudos futuros sobre a análise de complexidade e sobre os aspectos de determinismo em outras instâncias de dispositivos adaptativos. Além disso, sugerem-se o desenvolvimento e a implementação de algoritmos e ferramentas de software para detecção de determinismo e determinização de dispositivos adaptativos.

REFERÊNCIAS

[1] J. J. Neto, "Solving complex problems efficiently with adaptive automata", in *Conference on the Implementation and Application of Automata - CIAA 2000*, Ontario, Canada, July 2000.

[2] J. J. Neto e M. K. Iwai, "Adaptive automata for syntax learning", in *Anais da XXIV Conferencia Latinoamericana de Informática – CLEI 98*, Quito, Equador, pp. 135-149, 1998.

[3] H. Pistori e J. J. Neto, "Decision tree induction using adaptive FSA", *CLEI Electronic Journal*, vol. 6, no. 1, 2003.

[4] H. Pistori, P. S. Martins, e A. A. de Castro Jr., "Adaptive finite state automata and genetic algorithms: Merging individual application and populations evolution", in *Proceedings of the International Conference on Adaptive and Natural Computing Algorithms – ICANNGA*, Coimbra, Portugal, 2005.

[5] J. J. Neto e M. Moraes, "Using adaptive formalisms to describe context-dependencies in natural language", Lecture Notes in Artificial Intelligence. N.J. Mamede, J. Baptista, I. Trancoso, M. das Graças, V. Nunes (Eds.): *Computational Processing of the Portuguese Language 6th International Workshop, PROPOR 2003*, vol. 2721, pp. 94-97, June 2003.

[6] M. A. A. Sousa e A. H. Hirakawa, "Robotic mapping and navigation in unknown environments using adaptive automata", in *Proceedings of International Conference on Adaptive and Natural Computing Algorithms – ICANNGA 2005*, Coimbra, Portugal, 2005.

[7] H. Pistori, J. J. Neto, "Tecnologia adaptativa em engenhara de computação: Estado da arte e aplicações", Tese de Doutorado, Universidade de São Paulo, São Paulo, Brasil, 2003 [In Portuguese].

[8] J. Hromkovic, J. Karhumaki, H. Klauck, G. Schnitger, e S. Seibert, "Measures of nondeterminism in finite automata", in *Proc. ICALP 2000, Lecture Notes in Computer Science*, vol. 1853, 2000, pp. 199-210.
[9] R. E. Stearns, "Deterministic versus nondeterministic time and lower bound problems", *Journal of the ACM*, vol. 50, no. 1, pp. 91-95, 2003.
[10] C. Allauzen e M. Mohri, "The determinizability of weighted automata and transducers", in *Workshop of Weighted Automata: Theory and Applications (WATA)*, Dresden, Germany, March 2002.
[11] ——, "An efficient pre-determinization algorithm", in *CIAA – 8th International Conference on Implementation and Application of Automata, ser. Lecture Notes in Computer Science*, vol. 2759. Springer, 2003, pp. 83-95.
[12] J. J. Neto, "Adaptative rule-driven devices – general formulation and a case study", in *CIAA'2001 Sixth International Conference on Implementation and Application of Automata*, Pretoria, South Africa, July 2001, pp. 234-250.
[13] ——, "Adaptive automata for context-sensitive languages", *SIGPLAN NOTICES*, vol. 29, no. 9, pp. 115-124, September 1994.

Amaury Antônio de Castro Junior é graduado em Ciência da Computação pela Universidade Federal de Mato Grosso do Sul (1997) e mestre em Ciência da Computação pela mesma universidade (2003). Atualmente, é aluno de doutorado da Escola Politécnica da USP, sob a orientação do Prof. João José Neto. É membro do Laboratório de Linguagens e Técnicas Adaptativas e do Grupo de Pesquisa em Engenharia e Computação da UCDB e professor assistente da Universidade Federal de Mato Grosso do Sul (UFMS), Campus de Coxim (CPCX). Tem experiência na área de Ciência da Computação, com ênfase em Teoria da Computação, atuando nos seguintes temas: Tecnologias Adaptativas, Autômatos Adaptativos, Projeto de Linguagens de Programação e Modelos de Computação.

João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975) e doutor em Engenharia Elétrica (1980). É livre-docente (1993) pela Escola Politécnica da Universidade de São Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo e coordena o LTA – Laboratório de Linguagens e Tecnologia Adaptativa do PCS – Departamento de Engenharia de Computação e Sistemas Digitais da EPUSP. Tem experiência na área de Ciência da Computação, com ênfase nos Fundamentos da Engenharia da Computação, atuando principalmente nos seguintes temas: dispositivos adaptativos, tecnologia adaptativa, autômatos adaptativos e em suas aplicações à Engenharia de Computação, particularmente em sistemas de tomada de decisão adaptativa, análise e processamento de linguagens naturais, construção de compiladores, robótica, ensino assistido por computador, modelagem de sistemas inteligentes, processos de aprendizagem automática e inferências baseadas em tecnologia adaptativa.

Hemerson Pistori concluiu o doutorado pela Universidade de São Paulo, USP, em 2003, e o mestrado pela Universidade Estadual de Campinas, UNICAMP. Atualmente, é bolsista de produtividade em desenvolvimento tecnológico e extensão inovadora do CNPq, professor-pesquisador da Universidade Católica Dom Bosco, UCDB, e coordenador do Grupo de Pesquisa em Engenharia e Computação, GPEC. É membro do conselho editorial do periódico científico Colabora e do *International Journal for Computer Vision and Biomechanics*. Seus interesses em pesquisa concentram-se na intersecção das áreas de Visão Computacional, Aprendizagem Automática e Fundamentos de Computação, em especial Tecnologias Adaptativas.