Persistência em dispositivos adaptativos

P. R. M. Cereda e J. J. Neto

***Resumo*—Este artigo apresenta uma proposta inicial para a adição de um sistema de persistência em dispositivos adaptativos, tornando-os persistentes ao longo da própria execução ou de execuções subsequentes. Espera-se que tal funcionalidade possa contribuir para a extensão dos dispositivos adaptativos existentes, proporcionando-lhes a capacidade de preservar seus estados e de recuperá-los posteriormente.**

***Palavras-chave:*—Adaptatividade, persistência, dispositivos adaptativos**

I. Introdução

Persistência é o nome dado para o intervalo de disponibilidade do conteúdo de um determinado objeto [1], [2]. Tal característica de disponibilidade é primordial para a busca de soluções computacionais para os mais diversos problemas; informações coletadas e a própria evolução do modelo computacional ao longo do tempo necessitam de uma área de armazenamento para preservação do estado, sua atualização e eventual remoção [3].

A utilização da tecnologia adaptativa como alternativa para a resolução de problemas complexos tem apresentado resultados significativos [4]. Em [5], por exemplo, Cereda e José Neto introduzem uma proposta inicial do conceito de mineração adaptativa de dados como possível solução computacional aplicável aos problemas oriundos de grandes volumes de dados; entretanto, tais conceitos requerem, implicitamente, que existam subsídios para preservação dos modelos. A persistência é uma funcionalidade que pode contribuir para a extensão dos dispositivos adaptativos, conferindo-lhes a capacidade de preservar estados internos e recuperá-los conforme a necessidade.

Este artigo apresenta uma proposta inicial para a adição de persistência em dispositivos adaptativos, permitindo que estes tenham a capacidade de armazenar seu estado e recuperá-lo posteriormente. É importante mencionar que, no escopo deste artigo, a persistência é tratada sob um aspecto formal, sem, entretanto, entrar no mérito das técnicas de implementação. A área de armazenamento, neste caso, não é contemplada; admite-se que ela existe e que é transparente para o usuário.

Este artigo está organizado da seguinte forma: a Seção II introduz alguns conceitos iniciais para a formalização do conceito de persistência. A Seção III discute a persistência em dispositivos adaptativos propriamente dita, apresentando exemplos e discussões sobre a proposta. As considerações finais são apresentadas na Seção IV.

II. Conceitos iniciais

É necessário introduzir alguns conceitos iniciais para tratar da persistência em dispositivos adaptativos. Esta seção apresenta definições importantes para a compreensão, tratando de cada elemento componente do modelo proposto.

Os autores podem ser contatados através dos seguintes endereços de correio eletrônico: `paulo.cereda@usp.br` e `jjneto@usp.br`.

Definição 1 (Objeto)**.** Um *objeto* é qualquer elemento de armazenamento capaz de conter dados (*valores*) em um sistema. □

Um objeto é uma estrutura abstrata que contém dados. Esses dados, por sua vez, também podem ser outros objetos. Um objeto pode ter vários atributos, cada um deles armazenando um determinado dado. A estrutura

```
obj_pessoa {
   nome: Pedro
   idade: 30
}
```

é um exemplo de objeto contendo dois atributos, `nome` e `idade`, cada um deles contendo um valor associado. Cada sistema s_j possui um conjunto O_{s_j} contendo todos os objetos $o_1, \ldots, o_k$ componentes, $O_{s_j} = \{o_1, \ldots, o_k\}$, com $k \in \mathbb{N}$.

Definição 2 (Evento)**.** *Evento* é qualquer *ocorrência* que possa ser registrada ou identificada em um sistema. Por exemplo, são eventos a criação de um objeto, sua destruição, ou a alteração do seu conteúdo. □

De acordo com a Definição 2, um evento é sempre observável. Caso existam ocorrências que não interessam ao contexto do sistema, elas não serão registradas ou observadas e, portanto, não serão consideradas eventos. Seja E o conjunto de todos os eventos genéricos possíveis; existe uma relação de observação $\alpha\colon S \times E \mapsto \{0, 1\}$, no qual S é o conjunto de sistemas, retornando 0 caso o evento e_i não seja observável no sistema s_j, ou 1 caso contrário. Portanto, e_i é um evento de s_j se, e somente se, $\alpha(s_j, e_i) = 1$, e o conjunto de todos os eventos de s_j é definido como $E_{s_j} = \{e_i \mid \alpha(s_j, e_i) = 1\}$, $E_{s_j} \subseteq E$.

Definição 3 (Instante)**.** *Instante* é um número n arbitrário, $n \in \mathbb{N}$, ao qual se referem *eventos* ocorridos no sistema. □

A definição de instante permite que os eventos sejam ordenados de acordo com sua ocorrência.

Definição 4 (Caracterização de um evento)**.** Eventos caracterizam-se por sua *identificação*, pelo seu *tipo* e pelo *instante de ocorrência*. □

Cada evento ocorre em um instante, isto é, existe um número natural associado ao evento que determina uma relação de ordem. Um evento é caracterizado por um identificador unívoco (que só admite uma interpretação), pelo tipo de evento (por exemplo, uma alteração de um atributo de um objeto) e pelo instante de ocorrência. Assim, um evento $e_i \in E_{s_j}$ é uma tupla $e_i = (a, b, c)$, com a sendo um identificador, b o

tipo do evento, e $c \in \mathbb{N}$ o instante de ocorrência. Considere $\pi_n(x)$ como a projeção do n-ésimo elemento da sequência x. É importante observar que

$$\bigcap_{e_i \in E_{s_j}} \{\pi_1(e_i)\} = \emptyset$$

ou seja, os identificadores de todos os eventos do sistema s_j são unívocos – não podem existir dois ou mais eventos compartilhando do mesmo identificador.

Definição 5 (Eventos simultâneos)**.** Dois ou mais eventos que tenham o mesmo instante de ocorrência são chamados *eventos simultâneos*. □

Em um sistema s_j com seu conjunto de eventos E_{s_j}, se

$$\left| \bigcap_{e_i \in E_{s_j}} \{\pi_3(e_i)\} \right| > 0$$

existem pelo menos dois eventos que apresentam o mesmo instante de ocorrência. Portanto, o sistema apresenta eventos simultâneos.

Definição 6 (Intervalo)**.** *Intervalo* é um conjunto ordenado que compreende todos os instantes que ocorrem em um sistema, desde um instante *inicial* (*limite inferior*) até um instante *final* (*limite superior*). □

O intervalo do sistema s_j é denotado por $I_{s_j} = \{c_1, \ldots, c_l\}$, com $l \in \mathbb{N}$, sendo c_1 o limite inferior e c_l o limite superior. Existe uma relação de ordem que determine a posição de cada elemento.

Definição 7 (Caracterização de um objeto no instante)**.** O objeto se caracteriza em cada instante por sua *identificação* e pelo seu *conteúdo*. □

Um objeto $o_i \in O_{s_j}$ é representado no instante c como uma tupla $o_{i,c} = \{a, d\}$, com a sendo um identificador e d o conteúdo de o_i no instante c. Acerca da identificação de um objeto em um instante arbitrário,

$$\bigcap_{o_i \in O_{s_j}} \bigcap_{c \in I_{s_j}} \{\pi_1(o_{i,c})\} = \emptyset$$

ou seja, os identificadores de todos os objetos para cada instante no sistema s_j são unívocos – não podem existir objetos em instantes distintos compartilhando do mesmo identificador.

Definição 8 (Escopo de um objeto)**.** *Escopo* de um objeto é a parcela do programa que o contém na qual o conteúdo do objeto é de alguma forma visível (*acessível, ao menos, para consulta*). □

Seja um objeto $o_i \in O_{s_j}$, seu escopo Γ_{o_i} é definido como $\Gamma_{o_i} \colon I_{s_j} \mapsto \{0, 1\}$, com I_{s_j} sendo o intervalo do sistema s_j, retornando 0 caso o objeto não esteja disponível em um determinado instante, ou 1 caso contrário.

Definição 9 (Disponibilidade de conteúdo de um objeto)**.** O *conteúdo* de um objeto permanece *disponível* para uso desde o instante de seu armazenamento no objeto até o do armazenamento de um novo conteúdo (devido a *efeitos colaterais* ocorridos durante a operação do dispositivo), ou então até que o objeto *deixe de existir*. □

Um objeto $o_i \in O_{s_j}$ terá seu conteúdo disponível enquanto $\exists\ o_{i,c} \mid \pi_2(o_{i,c}) \neq$ vazio, tal que c é o limite superior para o_i, e $\Gamma_{o_i}(c) = 1$.

Definição 10 (Conjunto de todos os instantes)**.** O conjunto de todos os instantes do sistema é o conjunto dos números correspondentes a todas as ocasiões de ocorrências de eventos. □

Definição 11 (Atribuição de instantes)**.** Embora arbitrários, os instantes são *sempre atribuídos em ordem crescente*, exceto quando ocorrem eventos simultâneos. □

Definição 12 (Indicador de precedência das ocorrências dos eventos)**.** Não se trata do conceito de tempo, nem contínuo nem discreto, apenas de um *indicador de precedência* das ocorrências dos eventos no sistema. □

Definição 13 (Intervalo de usabilidade)**.** *Intervalo de usabilidade* de um objeto é o intervalo no qual existe algum escopo que inclua o objeto em questão, logo é o conjunto de instantes em que é possível consultar seu conteúdo. □

O intervalo de usabilidade do objeto o_i pode ser definido como $U_{o_i} = \{c \in I_{s_j} \mid \Gamma_{o_i}(c) = 1\}$, ou seja, é o conjunto formado por cada instante no intervalo em que o escopo do objeto i esteja disponível.

Definição 14 (Usabilidade do conteúdo dos objetos)**.** Durante a execução de um programa, a *usabilidade do conteúdo* dos objetos nele existentes corresponde à *união dos intervalos de atividade dos escopos* aos quais os objetos pertencem. □

A usabilidade de conteúdo V pode ser definida como

$$V = \bigcup_{o_i \in O_{s_j}} U_{o_i}$$

ou seja, é a disponibilização de todos os escopos de todos os objetos no sistema s_j.

Definição 15 (Objeto de conteúdo persistente)**.** *Objeto de conteúdo persistente* é aquele cujo conteúdo possa ser memorizado em qualquer instante desejado, para que possa posteriormente ser recuperado para uso a critério do programa que controla o uso do objeto. □

Definição 16 (Sistema de persistência)**.** Um *sistema de persistência* é um espaço de armazenamento que pode memorizar os dados contidos em objetos de conteúdo persistente, disponibilizando-o para que possam ser resgatados em qualquer instante pelo programa que controla o uso do objeto. □

Definição 17 (Conteúdo de um objeto persistente em um intervalo)**.** O conteúdo de um objeto é *persistente em um intervalo* se nesse intervalo for possível acessá-lo, ou seja, se tal conteúdo fizer parte de algum escopo ativo naquela ocasião. □

Em outras palavras, considerando um objeto $o_i \in O_{s_j}$, ele será persistente em um intervalo se seu conteúdo estiver ativo.

Definição 18 (Conteúdo ativo no intervalo de usabilidade)**.** Se uma parte do intervalo de usabilidade do conteúdo de um objeto *não pertencer ao intervalo de execução do programa* ao que ele pertence, diz-se que o conteúdo está ativo nesse intervalo. □

Sejam U_{o_i} o intervalo de usabilidade do objeto $o_i \in O_{s_j}$ e P um intervalo de execução do programa; se $|U_{o_i} \cap P| > 0$, o conteúdo de o_i está ativo no intervalo.

Definição 19 (Usabilidade de conteúdo para objetos não-persistentes)**.** Para programas que não tenham objetos com conteúdos persistentes, nenhum objeto apresentará usabilidade de conteúdo a não ser durante o intervalo de execução do programa. □

Analogamente, sejam U_{o_i} o intervalo de usabilidade do objeto $o_i \in O_{s_j}$ e P um intervalo de execução do programa; se $U_{o_i} \cup P \subseteq P$, o conteúdo de o_i só estará disponível durante o intervalo de execução do programa.

Duas operações básicas de persistência são necessárias para complementar as definições apresentadas até então: *salvar*, responsável por persistir o valor de um objeto $o_i \in O_{s_j}$ em um determinado instante c no sistema de persistência, e *restaurar*, que recupera um valor específico de um objeto $o_i \in O_{s_j}$, anteriormente persistido. As operações serão detalhadas a seguir.

Definição 20 (Operação de salvamento)**.** Seja $P_\downarrow$ uma operação básica de persistência, definida como $P_\downarrow \colon O_{s_j} \times I_{s_j} \mapsto \mathbb{N}$; $P_\downarrow$ toma um objeto $o_i \in O_{s_j}$ e um instante $c \in I_{s_j}$, ambos pertencentes ao sistema s_j, persistindo o valor de o_i no instante c no sistema de persistência, e retornando um valor inteiro que representa um identificador unívoco. □

É importante observar que a operação de salvamento gera um evento $e_k = (u, v, w)$, onde $u = P_\downarrow(o_i, c)$ (identificador), $v \equiv P_\downarrow(o_i, c)$ (tipo de evento, neste caso, salvamento de o_i no instante c), e $w \in I_{s_j}$ (instante do evento, pode ser diferente de c).

O salvamento de o_i no instante c é armazenado pelo sistema de persistência em uma estrutura de lista ligada, disponível para cada objeto no sistema (Figura 1). Cada elemento da lista de o_i contém uma operação de persistência sobre esse objeto, consistindo no objeto, no instante e no identificador unívoco gerado. Os elementos da lista estão encadeados através de referências de endereço, no qual cada elemento aponta para aquele inserido na sequência (no exemplo da Figura 1, as ligações são representadas por setas), ou nulo se este é o último elemento (na Figura 1, representado por •). O identificador gerado a partir da operação de salvamento será utilizado como rótulo de seu elemento correspondente.

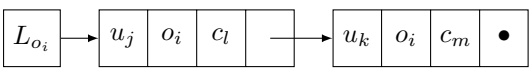

Figura 1. Estrutura de lista ligada para o objeto o_i. No exemplo, foram realizadas duas operações de salvamento, gerando os identificadores u_j e u_k.

A cada operação de salvamento do objeto o_i, os elementos anteriores na lista não são removidos, criando-se, portanto, um histórico de persistência para o_i, desde a primeira operação até a última. Todos os objetos são contemplados no sistema de persistência, e $\forall\ o_i \in O_{s_j}, \exists\ L_{o_i}$, ou seja, existe sempre uma lista associada a cada objeto.

Definição 21 (Operação de restauração)**.** Seja $P_\uparrow$ uma operação básica de persistência, definida como $P_\uparrow \colon O_{s_j} \times I_{s_j} \mapsto O_{s_j} \cup \{\epsilon\}$; $P_\uparrow$ toma um objeto $o_i \in O_{s_j}$ e um instante $c \in I_{s_j}$, ambos pertencentes ao sistema s_j, realiza a busca de o_i no instante c no sistema de persistência, e retornando o objeto o_i persistido no instante c, ou ϵ caso não exista ocorrência de persistência com tais índices. □

É importante observar que a operação de restauração gera um evento $e_k = (u, v, w)$, onde u é um identificador unívoco, $v \equiv P_\uparrow(o_i, c)$ (tipo de evento, neste caso, restauração de o_i no instante c), e $w \in I_{s_j}$ (instante do evento, pode ser diferente de c).

Além disso, a cada operação de restauração, o objeto o_i é novamente persistido, de modo que uma cópia de o_i no novo instante $c' \in I_{s_j}$ é colocada na cauda da lista L_{o_i}. Em outras palavras, $P_\uparrow(o_i, c) = o_i \wedge P_\downarrow(o_i, c')$.

É possível também definir $P_\uparrow$ como $P_\uparrow \colon \mathbb{N} \mapsto O_{s_j} \cup \{\epsilon\}$, tomando um número natural representando o identificador unívoco previamente gerado pela operação de salvamento; a nova operação realiza a busca de o_i cujo indicador é um número n na lista L_{o_i} do sistema de persistência, e retorna um objeto o_i do elemento com rótulo n, ou ϵ caso não exista ocorrência de persistência com tal índice.

Do mesmo modo, a cada operação de restauração, o objeto o_i é novamente persistido, de modo que uma cópia de o_i no novo instante $c' \in I_{s_j}$ é colocada na cauda da lista L_{o_i}.

As operações básicas do sistema de persistência são atômicas sobre um mesmo objeto o_i, ou seja, se dois ou mais eventos simultâneos sobre o_i têm como tipo de evento $P_\downarrow$ ou $P_\uparrow$, apenas *uma operação* será realizada, descartando as demais. Caso todas as operações sejam de restauração, apenas uma delas (não é possível predizer qual delas será) será efetivamente considerada; caso exista ao menos uma operação de salvamento, ela sempre terá prioridade sobre a operação de restauração mas, do mesmo modo, apenas uma delas será considerada.

III. Persistência em dispositivos adaptativos

Após a definição dos conceitos iniciais, já é possível tratar da persistência em dispositivos propriamente dita. O valor a ser persistido, neste caso, é a configuração corrente do dispositivo e o estímulo de entrada. Uma possível interpretação do sistema de persistência é através do conceito de camada, no qual o dispositivo está contido em uma camada de persistência, podendo fazer uso dela ou não (Figura 2). Mesmo no caso do dispositivo não utilizar persistência, o objeto continua ativo (Definição 19).

Uma vez que a característica de persistência é agregada a um determinado dispositivo, conferindo a característica de ser persistente, é necessário definir a interface entre eles. Para que a persistência ocorra de modo transparente, espera-se que o dispositivo, na aplicação de alguma regra, execute uma ação

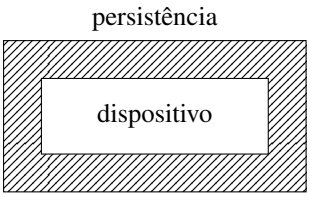

Figura 2. Interpretação de persistência como uma camada envolvendo o dispositivo. A camada em questão representa o sistema de persistência propriamente dito (Definição 16).

semântica que indique à camada de persistência qual operação realizar (Definição 22).

Definição 22 (Ação semântica de persistência)**.** Uma *ação semântica de persistência* é uma chamada a operação básica presente no sistema de persistência, de acordo com a definição explícita de um determinado dispositivo d que a dispara. □

Como exemplo, considere um autômato finito determinístico $M = (K, \Sigma, \delta, s, F)$, onde K é um conjunto finito de estados, Σ é um alfabeto, δ é uma função de transição, $\delta\colon K \times \Sigma \mapsto K$, $s \in K$ é o estado inicial e $F \subseteq K$ é o conjunto de estados finais [6]. A Tabela I contém o mapeamento δ e apresenta uma ação semântica da operação de salvamento $P_{\downarrow}$ que indica ao sistema de persistência para salvar a configuração do dispositivo e a cadeia w sendo consumida, no instante corrente. Admita a função $\rho(M, w)$ que codifica a configuração corrente de M e a cadeia w consumida até então em um valor para o objeto correspondente ao autômato.

Tabela I
EXEMPLO DE AÇÃO SEMÂNTICA ASSOCIADA AO MAPEAMENTO.

$q \in K$	$\sigma \in \Sigma$	$\delta(q, \sigma)$	Ação semântica
q_0	a	q_0	–
q_0	b	q_1	$P_{\downarrow}(\rho(M, w), 1)$
q_1	a	q_1	–
q_1	b	q_0	$P_{\downarrow}(\rho(M, w), 2)$

Na Tabela I, é possível verificar que o sistema de persistência é acionado para salvar a configuração de M e a cadeia w, consumida até então, a cada ocorrência do símbolo $b \in \Sigma$, em um determinado instante. A ação semântica $P_{\downarrow}$ (correspondente à operação de salvamento do sistema de persistência) não faz parte da definição formal do dispositivo, mas é associada às suas regras.

O exemplo apresentado apenas faz a persistência da configuração corrente de M e a cadeia de entrada consumida até então, mas não realiza a operação de restauração. Observe que o sistema de persistência possui uma lista L_M que armazenará o conteúdo persistido. É possível utilizar essa lista como forma de *backtracking*, caso o dispositivo necessite restaurar uma configuração anterior. Como novo exemplo, considere o autômato finito da Figura 3.

A linguagem reconhecida pelo autômato M da Figura 3 é $L(M) = \{w \in \{a, b\}^* \mid w = aa(ba)^*\}$. A primeira transição, $(q_0, a) \rightarrow q_1$, consome a de w e realiza a ação semântica chamando a operação de salvamento; a segunda transição, $(q_1, a) \rightarrow q_2$ consome o segundo a e chega ao estado final q_2; existe um laço em q_2 que consome b e

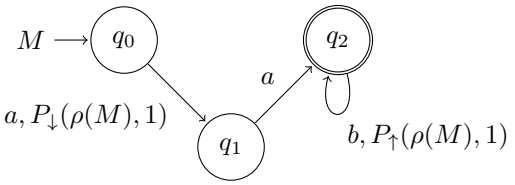

Figura 3. Autômato finito M utilizando a camada de persistência.

restaura o conteúdo persistido no instante 1; observe que, deliberadamente, a cadeia w, consumida até então, não foi persistida. Ao realizar a operação de restauração, o autômato M é restaurado para a configuração do instante 1, isto é, M retorna ao estado q_1 esperando, então, mais um a para alcançar o estado final. A Tabela II ilustra o consumo da cadeia $aababa$ por M.

Tabela II
CONSUMO DA CADEIA $aababa$ POR M.

Estado corrente	Símbolo corrente	Estado de destino	Ação semântica	Restante da cadeia
q_0	a	q_1	$P_{\downarrow}(\rho(M), 1)$	$ababa$
q_1	a	q_2	–	$baba$
q_2	b	q_2	$P_{\uparrow}(\rho(M), 1)$	aba
q_1	a	q_2	–	ba
q_2	b	q_2	$P_{\uparrow}(\rho(M), 1)$	a
q_1	a	q_2	–	–

Observe que o exemplo da Figura 3 não apresenta benefícios em relação ao mapeamento tradicional, sem a utilização de ações semânticas de persistência – a remoção do laço em q_2 e a adição de uma transição consumindo b de $q_2 \rightarrow q_1$ seriam suficientes. Entretanto, é interessante notar a possibilidade de restaurar configurações e estímulos de entrada conforme a necessidade; no caso da utilização de dispositivos adaptativos, pode partir-se de configurações já existentes, oriundas de execuções anteriores, além da possibilidade de obter um *snapshot* da configuração corrente.

Considere agora o exemplo do autômato adaptativo M', introduzido em [7], com a adição do sistema de persistência. A linguagem aceita por M' foi adicionada de marcadores de início e fim para facilitar as operações de salvamento e restauração; assim, $L(M) = \{w \in \{a, b, c, \$\} \mid w = \$\$\$^* a^n b^n c^n \$\}$, ou seja, uma cadeia com um número n de a's, seguido pelo mesmo número de b's, seguido pelo mesmo número de c's. O autômato adaptativo M' é apresentado na Figura 4. A função adaptativa f é apresentada na Figura 5.

De modo simplificado, o autômato M' foi remodelado para reconhecer cadeias na forma $a^n b^n c^n$, com n, no mínimo, igual à sua maior ocorrência. Isto é, se M' reconheceu $aaabbbccc$, com $n = 3$, ele não será mais capaz de reconhecer cadeias com $n < 3$, apenas com $n \geq 3$.

Como exemplo, considere a submissão da cadeia $w = \$\$aabbcc\$$ para reconhecimento pelo autômato adaptativo M'. Observe que, de acordo com a Figura 4, deliberadamente, a cadeia w, consumida até então, não é persistida; apenas a configuração do dispositivo é utilizada. A Tabela III ilustra o consumo da cadeia $\$\$aabbcc\$$ por M'.

O autômato M' resultante após o reconhecimento de $\$\$aabbcc\$$ foi salvo após o consumo do marcador de final.

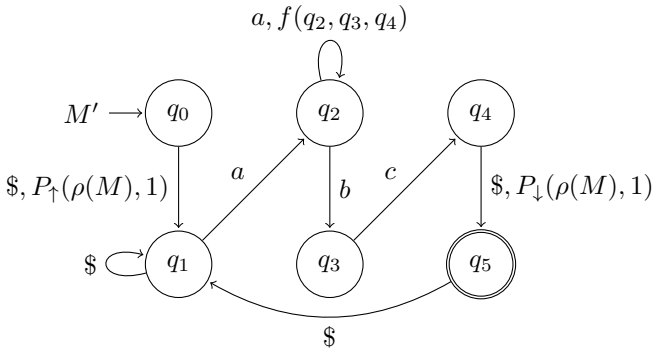

Figura 4. Autômato adaptativo M'.

Função adaptativa

```
f(p1 , p2, p3) = {
  ?x, ?y, ?z, g1*, g2*, g3* :
  ?[(?x, a) -> (p1)]
  -[(?x, a) -> (p1)]
  ?[(?y, b) -> (p2)]
  -[(?y, b) -> (p2)]
  ?[(?z, c) -> (p3)]
  -[(?z, c) -> (p3)]
  -[(q2, a) -> (q2), f(p1, p2, p3)]
  +[(?x, a) -> (g1*)]
  +[(g1*, a) -> (p1)]
  +[(?y, b) -> (g2*)]
  +[(g2*, b) -> (p2)]
  +[(?z, c) -> (g3*)]
  +[(g3*, c) -> (p3)]
  +[(q2, a) -> (q2), f(g1 , g2, g3)]
}
```

Figura 5. Função adaptativa f do autômato M'.

É importante mencionar que a restauração de M' na transição de $q_0 \rightarrow q_1$ falha na primeira vez em que M' é utilizado, pois a lista L_M está vazia, e $P_\uparrow(\rho(M), 1) = \epsilon$; entretanto, o erro na restauração não impede o que o reconhecimento de w prossiga. É possível tratar o erro adicionando ações semânticas complementares para a avaliação das operações básicas, além de estender o próprio sistema de persistência.

Apesar da sequência de consumo apresentada na Tabela III ilustrar a criação de novos estados e alteração e inclusão de transições para reconhecer b's e c's adicionais, existe uma ação adaptativa adicional no corpo da função adaptativa f (Figura 5): como o autômato adaptativo M' está sendo remodelado para oferecer a possibilidade de reuso, é necessário adicionar um estado adicional e rearranjar transições para o reconhecimento de a's adicionais, tal qual acontece com b's e c's. Observe que a adição de persistência em um dispositivo pode requerer uma análise mais profunda sobre a manifestação do fenômeno da adaptatividade.

Tabela III
CONSUMO DA CADEIA $\$\$aabbcc\$$ POR M'.

Estado corrente	Símbolo corrente	Estado de destino	Ação semântica	Restante da cadeia
q_0	$\$$	q_1	$P_\uparrow(\rho(M), 1)$	$\$aabbcc\$$
q_1	$\$$	q_1	–	$aabbcc\$$
q_1	a	q_2	–	$abbcc\$$
q_2	a	$q_2, f(\ldots)$	–	$bbcc\$$
q_2	b	q_7	–	$bcc\$$
q_7	b	q_3	–	$cc\$$
q_3	c	q_8	–	$c\$$
q_8	c	q_4	–	$\$$
q_4	$\$$	q_5	$P_\downarrow(\rho(M), 1)$	–

Ao submeter uma nova cadeia w' ao autômato M', ele não a reconhecerá caso esta tenha $n < 2$, dado que a operação de salvamento foi executada quando M' estava preparado para reconhecer cadeias com $n = 2$. A função adaptativa f não foi alterada e está disponível, portanto M' ainda é capaz de reconhecer cadeias com $n \geq 2$. Considere agora a submissão de uma nova cadeia $w' = \$\$aaabbbccc\$$ para reconhecimento pelo autômato adaptativo M'. A Tabela IV ilustra o consumo de w'.

Tabela IV
CONSUMO DA CADEIA $\$\$aaabbbccc\$$ POR M'.

Estado corrente	Símbolo corrente	Estado de destino	Ação semântica	Restante da cadeia
q_0	$\$$	q_1	$P_\uparrow(\rho(M), 1)$	$\$aaabbbccc\$$
q_5	$\$$	q_1	–	$aaabbbccc\$$
q_1	a	q_6	–	$aabbbccc\$$
q_6	a	q_2	–	$abbbccc\$$
q_2	a	$q_2, f(\ldots)$	–	$bbbccc\$$
q_2	b	q_{10}	–	$bbccc\$$
q_{10}	b	q_7	–	$bccc\$$
q_7	b	q_3	–	$ccc\$$
q_3	c	q_{11}	–	$cc\$$
q_{11}	c	q_8	–	$c\$$
q_8	c	q_4	–	$\$$
q_4	$\$$	q_5	$P_\downarrow(\rho(M), 1)$	–

Ao consumir o primeiro símbolo, o marcador de início, M' é restaurado para o instante 1, e a execução passa a assumir q_5 como estado corrente, de acordo com a Tabela IV; desse ponto em diante, o autômato segue seu fluxo normal de execução até o consumo do marcador final, sendo persistido mais uma vez. Observe que, após o reconhecimento de w', M' passa a reconhecer $a^n b^n c^n$ tão somente com $n \geq 3$.

É interessante notar que, ao executar a ação semântica $P_\downarrow(\rho(M), 1)$, o sistema de persistência sobrescreve o valor antigo ao invés de criar um novo elemento na lista L_M; dessa forma, M' não tem histórico de persistência, apenas a última referência. É possível alterar M' de tal modo que as referências em instantes anteriores sejam mantidas, ora através de ações adaptativas alterando ações semânticas no mapeamento, ou por meio de extensões no sistema de persistência, criando, por exemplo, uma operação de salvamento estendida $P'_\downarrow$ que salva o objeto sempre no instante corrente, e uma operação de restauração estendida $P'_\uparrow$ que retorna o objeto com o identificador de maior valor (portanto, a referência mais recente).

IV. CONSIDERAÇÕES FINAIS

Este artigo apresentou uma proposta para a adição de um sistema de persistência em dispositivos adaptativos, tornando-os persistentes ao longo da própria execução ou de execuções subsequentes.

A característica de persistência para dispositivos adaptativos permite salvar e restaurar configurações através de ações semânticas; a interpretação do sistema de persistência através do conceito de camada confere uma independência de modelo ao dispositivo, ou seja, não é necessário alterar explicitamente a definição do dispositivo para torná-lo persistente, bastando apenas envolvê-lo na camada de persistência e utilizar ações semânticas, que não fazem parte da definição formal do dispositivo, mas podem ser associadas às suas regras, para realizar a interface entre ambos.

REFERÊNCIAS

[1] M. P. Atkinson, P. J. Bailey, K. J. Chisholm, P. W. Cockshott, and R. Morrison, "An approach to persistent programming," in *Readings in object-oriented database systems*, S. B. Zdonik and D. Maier, Eds. Morgan Kaufmann Publishers Inc., 1990, pp. 141–146.

[2] M. Atkinson and R. Morrison, "Orthogonally persistent object systems," *The VLDB Journal*, vol. 4, no. 3, pp. 319–402, 1995.

[3] H. Kaplan, "Persistent data structures," in *Handbook on data structures and applications*, D. Mehta and S. Sahni, Eds., 1995, pp. 241–246.

[4] J. José Neto, "Um levantamento da evolução da adaptatividade e da tecnologia adaptativa," *IEEE Latin America Transactions*, vol. 5, pp. 496–505, 2007.

[5] P. R. M. Cereda and J. José Neto, "Mineração adaptativa de dados: Aplicação à identificação de indivíduos," in *Memórias do Sexto Workshop de Tecnologia Adaptativa*, São Paulo, 2012.

[6] H. R. Lewis and C. H. Papadimitriou, *Elements of the theory of computation*. Prentice Hall, 1997.

[7] J. José Neto, "Contribuições à metodologia de construção de compiladores," Tese de livre-docência, Escola Politécnica, Universidade de São Paulo, 1993.

Paulo Roberto Massa Cereda é graduado em Ciência da Computação pelo Centro Universitário Central Paulista (2005) e mestre em Ciência da Computação pela Universidade Federal de São Carlos (2008). Atualmente, é doutorando do Programa de Pós-Graduação em Engenharia de Computação do Departamento de Engenharia de Computação e Sistemas Digitais da Escola Politécnica da Universidade de São Paulo, atuando como aluno pesquisador no Laboratório de Linguagens e Técnicas Adaptativas do PCS. Tem experiência na área de Ciência da Computação, com ênfase em Teoria da Computação, atuando principalmente nos seguintes temas: tecnologia adaptativa, autômatos adaptativos, dispositivos adaptativos, linguagens de programação e construção de compiladores.

João José Neto é graduado em Engenharia de Eletricidade (1971), mestre em Engenharia Elétrica (1975), doutor em Engenharia Elétrica (1980) e livre-docente (1993) pela Escola Politécnica da Universidade de S ão Paulo. Atualmente, é professor associado da Escola Politécnica da Universidade de São Paulo e coordena o LTA – Laboratório de Linguagens e Técnicas Adaptativas 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.