Formalismo de Modelo de Controle de Acesso utilizando Autômato Adaptativo

P. R. M. Cereda e S. D. Zorzo

***Resumo*—A privacidade é um aspecto importante ao modelar sistemas computacionais que necessitam estabelecer relacionamentos entre usuários e suas informações. Para lidar com essa questão, muitos mecanismos de proteção de privacidade foram considerados. Mecanismos genéricos lidam com a segurança da informação de um determinado usuário restringindo o sistema de modo que existam apenas relacionamentos entre indivíduos e objetos que estão em conformidade com as regras deste sistema. Este artigo apresenta um formalismo de controle de acesso utilizando autômato adaptativo em um Sistema de Privacidade Auditável. O autômato adaptativo é adequado para este propósito, devido à capacidade de auto-modificação e reconhecimento de linguagens mais complexas. O modelo utilizando o formalismo proposto é uma nova abordagem para mecanismos de proteção de privacidade, e é genérico o suficiente para ser utilizado não apenas em privacidade, mas em outras classes de problemas.**

***Palavras-chave:*—Controle de acesso, Privacidade, Autômato Adaptativo.**

I. Introdução

Privacidade é um termo relativamente difícil de definir, devido à sua amplitude e subjetividade. De um modo genérico, a privacidade é o direito de um indivíduo em resguardar suas informações pessoais de outrem.

Fernandes [1] afirma que "para ter privacidade, uma pessoa precisa ter controle sobre as informações existentes sobre si mesma e exercer este controle de forma consistente com seus interesses e valores pessoais". Assim, durante a interação de um usuário em um determinado sistema, suas informações devem ser divulgadas apenas com o consentimento do mesmo.

O conceito de privacidade pode divergir de modo significativo entre indivíduos, mesmo estes estando inseridos em ambientes sócio-culturais semelhantes. A padronização de ações consideradas intrusivas são caracterizadas na forma de leis, que são distintas a uma determinada região ou país, e aplicando penas às situações que causem violação de tais leis.

A privacidade tornou-se um aspecto importante ao modelar sistemas computacionais que exigem o estabelecimento de relações entre usuários e seus dados; o usuário deve sentir-se seguro, de modo que suas informações não sejam utilizadas para fins diferentes dos estipulados nas regras desses sistemas.

No contexto da Web, existem várias propostas na literatura para tentar fornecer privacidade ao usuário. Algumas delas são baseadas em arquiteturas ou mecanismos que visam manter o anonimato do usuário. Outras, porém, baseiam-se em métodos de policiamento dos sites ou de aviso ao usuário sobre as políticas de privacidade adotadas [2]. Alguns destes mecanismos são apresentados a seguir.

O anonimato pode ser caracterizado como a situação de não-identificação de um determinado indivíduo, entre um conjunto [3]. Na Web, através do anonimato, o usuário pode navegar por um determinado site sem que seja possível identificá-lo. A privacidade é garantida pela ocultação do endereço do Protocolo de Internet do computador e de outras informações presentes nos cabeçalhos dos protocolos.

O método dos pseudônimos consiste em criar apelidos para o usuário, disfarçando assim sua verdadeira identidade. Como o usuário está navegando pela Internet utilizando-se de um apelido, pode usufruir dos serviços de personalização disponibilizados pelos sites.

O Projeto de Plataforma para Preferências de Privacidade (P3P), desenvolvido pelo Consórcio da *World Wide Web*[1], permite que os sites Web padronizem a apresentação das práticas de coleta, de modo que o usuário tome conhecimento das informações coletadas. O Projeto 3P permite que site e usuário negociem quais informações serão coletadas, bem como sua utilização e mecanismo de coleta [4].

Os agentes de privacidade são ferramentas que proporcionam ao usuário informações sobre o grau de exposição e os riscos referentes à invasão de privacidade que um determinado site pode conter. A análise é feita através de verificações no site ou mesmo uma leitura da política de privacidade.

O mecanismo MASKS (*Managing Anonymity while Sharing Knowledge to Servers*) visa garantir a privacidade de anonimato do usuário sem deixar de permitir a personalização [5]. Essa arquitetura fornece privacidade aos usuários através de máscaras ou pseudônimos. As máscaras são identificações temporárias que um usuário pode adotar durante a interação com um determinado site, sem ser identificado.

Existem mecanismos mais genéricos para lidar com a questão da segurança das informações de um determinado usuário. Modelos de controle de acesso permitem apenas interações e relações entre indivíduos e objetos que sejam pertinentes às regras definidas para um dado sistema. Como exemplo, pode-se citar um usuário que não pode modificar ou ler as informações de outro usuário sem uma permissão definida pelo primeiro [6]. A privacidade do usuário em um modelo de controle de acesso será o foco deste trabalho.

Este artigo descreve um modelo de controle de acesso para um Sistema de Privacidade Auditável, utilizando-se de autômatos adaptativos. O autômato adaptativo foi escolhido devido à sua simplicidade, capacidade de auto-modificação e de reconhecimento de linguagens mais complexas.

Paulo Roberto Massa Cereda e Sérgio Donizetti Zorzo.

Este trabalho foi financiado pela Capes.

Os autores são do Departamento de Computação, Universidade Federal de São Carlos, Rod. Washington Luís, Km 235, Caixa Postal 676, 13565-905, (e-mails: paulo_cereda@dc.ufscar.br e zorzo@dc.ufscar.br).

[1] http://www.w3.org

Na seção II, é apresentado o Sistema de Privacidade Auditável, utilizando um modelo de controle de acesso. Na seção III, é apresentada a proposta deste artigo, juntamente com o autômato adaptativo. Os resultados são apresentados na seção IV, e na seção V são apresentadas as conclusões.

II. Sistema de Privacidade Auditável

Um *Sistema de Privacidade Auditável* [7] é definido como um sistema que permite o gerenciamento de objetos entre domínios e a realização de auditorias em tempo de execução e *post-hoc*.

O gerenciamento de objetos entre domínios é suportado pelos seguintes eventos: transferência, ação, criação, estabelecimento de direitos, notificação e *logging*. O evento de transferência consiste em transferir um objeto, copiando-o e passando a cópia ao domínio do receptor. Na ação, usa-se um objeto privado para um determinado propósito, enquanto que no evento de criação permite-se realizar a adição de novos objetos ou indivíduos ao sistema e à matriz de acesso. No estabelecimento de direitos, um indivíduo ou dono de um dado objeto pode atribuir ou revogar permissões. O evento de notificação é uma obrigação de informar o indivíduo acerca de eventos feitos por outros indivíduos no sistema, e no *logging* o sistema tem obrigação de monitorar todos os eventos – os que foram bem-sucedidos e os que falharam (é usado para verificação e auditoria).

Um Sistema de Privacidade Auditável necessita de um mecanismo de controle de acesso para atender seus requisitos. O modelo HRU, proposto por Harrison, Ruzzo e Ullman [8] tem como objetivo fornecer uma representação formal para sistemas de controle de acesso. O modelo utiliza uma matriz de controle, que armazena o estado atual do sistema, e possui seis operações primitivas que atuam sobre essa matriz, atribuindo e removendo direitos, através das operações **enter** r **into** (X_s, X_o) e **delete** r **from** (X_s, X_o), e na criação e remoção de indivíduos e objetos, através das operações **create subject** X_s, **create object** X_o, **destroy subject** X_s e **destroy object** X_o. Os comandos são transacionais, contendo um conjunto de condições (opcionais) e um conjunto de operações primitivas. O modelo é representado por um sistema de controle de acesso, contendo os elementos formais que o define. A política deste sistema resume-se ao conjunto de comandos que ele publica [9].

O Sistema de Privacidade Auditável, proposto por May, Gunter e Lee [7], utiliza as seis operações primitivas presentes no modelo HRU, com a adição de duas novas operações que realizam a notificação (**inform** s **of** t) e *logging* (**log** t). Entretanto, essas alterações não seriam suficientes para implementar o sistema em sua totalidade. Dessa forma, o modelo HRU foi alterado para incluir os seguintes aspectos: (a) inclusão de um argumento implícito chamado *ator* em cada comando, pois não era possível identificar quem havia iniciado um comando; (b) definição do conceito de *grupos*, para oferecer papéis e regras; (c) inclusão de condições baseadas no ambiente e operações de notificação e *logging*, já que os comandos eram restritos às condições e operações primitivas; (d) criação de operações não-primitivas para atuarem como uma interface para o sistema, pois os comandos não tinham permissão para chamar outros comandos; e (e) inclusão de dois tipos de obrigação: as que o sistema computacional pode analisar, e as que ele não pode.

A utilização do modelo HRU estendido em um Sistema de Privacidade Auditável justifica-se por sua simplicidade de uso. Entretanto, outros modelos de controle de acesso poderiam ser utilizados [7], como o proposto neste trabalho, utilizando-se autômato adaptativo. A simplicidade de descrição do modelo HRU pode ser subjugada com a eficiência em tempo e espaço do modelo proposto neste trabalho.

III. Controle de Acesso em um Sistema de Privacidade Auditável

Os autômatos adaptativos, devido à capacidade de automodificação, têm sua aplicação em diversas áreas, por exemplo, na robótica [10], onde dois autômatos adaptativos são responsáveis pelo mapeamento e navegação de um robô em um ambiente desconhecido; em sistemas de decisão e aprendizado de máquina [11], onde é apresentado um sistema que visa possibilitar a construção de árvores de indução adaptativas; no processamento de linguagem natural [12], onde é proposto um etiquetador morfológico treinado de acordo com um determinado *corpus*[2]; na otimização de código em compiladores [13], onde é proposto um otimizador que faz uso de técnicas adaptativas para gerar um código objeto com tamanho reduzido.

Este artigo descreve um formalismo de um modelo de controle de acesso, utilizando autômato adaptativo, em um Sistema de Privacidade Auditável. O mecanismo de controle de acesso gerencia as informações contidas nas relações entre indivíduos e objetos pertencentes a esse sistema.

A. Autômato de Pilha Estruturado

O *Autômato de Pilha Estruturado* é um tipo de autômato de pilha formado por um conjunto de autômatos finitos mutuamente recursivos. Esses autômatos também são chamados de *sub-máquinas*. A pilha tem a finalidade exclusiva de armazenar os estados de retorno a cada chamada de uma sub-máquina. As chamadas e retornos consistem em transferir o controle entre uma sub-máquina e outra; assim, é efetuada uma operação chamada *lookahead*, que consiste em utilizar o símbolo de entrada apenas para a tomada de decisão do autômato, sendo consumido na próxima transição [14] [15].

Um *autômato de pilha estruturado* M pode ser definido como $M = (Q, A, \Sigma, \Gamma, P, q_0, Z_0, F)$, onde:

- Q é o conjunto finito de estados,
- A é o conjunto de sub-máquinas,
- Σ é o alfabeto de entrada,
- Γ é o alfabeto da pilha, $\Gamma = Q \cup \{Z_0\}$,
- P é o mapeamento,
- q_0 é o estado inicial,
- Z_0 é um símbolo especial, indicador de pilha vazia, que é o símbolo inicial da pilha,
- e F é o conjunto de estados finais.

[2]Palavra de origem latina que significa coleção de documentos e fontes.

O mapeamento P é definido como uma relação $P \subseteq (Q \times \Sigma \times \Gamma) \times (Q \times (\Sigma \cup \{\epsilon\}) \times (\Gamma \cup \{\epsilon\}))$. A relação de transição é definida por $(q, aw, \beta) \vdash (q', a'w, \beta')$, onde:

- q é estado atual, $q \in Q$,
- a é símbolo a ser consumido, $a \in \Sigma$,
- w é o restante da cadeia a ser consumida,
- β é o topo da pilha, $\beta \in \Gamma$,
- q' é o novo estado, $q' \in Q$,
- a' é o novo símbolo, $a' \in \Sigma \cup \{\epsilon\}$, $a' = a$ ou $a' = \epsilon$,
- e β' é o novo topo da pilha, $\beta' \in \Gamma \cup \{\epsilon\}$.

O conjunto de sub-máquinas do autômato de pilha estruturado é representado por A. Cada *sub-máquina* i é definida como $a_i = (Q_i, \Sigma_i, P_i, q_{i0}, F_i)$, onde:

- $Q_i \subseteq Q$ é o conjunto de estados da sub-máquina i,
- $\Sigma_i \subseteq \Sigma$ é o conjunto de símbolos de entrada da sub-máquina i,
- $P_i \subseteq P$ é o mapeamento da sub-máquina i,
- $q_{i0} \in Q_i$ é o estado de entrada da sub-máquina i,
- e $F_i \subseteq Q_i$ é o conjunto de estados finais da sub-máquina i.

Se $a_i, a_j \in A$, $i \neq j$, $a_i = (Q_i, \Sigma_i, P_i, q_{i0}, F_i)$, $a_j = (Q_j, \Sigma_j, P_j, q_{j0}, F_j)$, então $Q_i \cap Q_j = \varnothing$. Claramente, $P_i \cap P_j = \varnothing$.

Se o estado atual for q, a entrada for aw e houver um mapeamento $P(q, aw, \beta) = (q', aw, \beta')$, com $\beta' \neq \epsilon$, então haverá uma chamada à sub-máquina $a_i = (Q_i, \Sigma_i, P_i, q_{i0}, F_i)$, onde $q' \in Q_i$. Há uma única sub-máquina a_i onde $q' \in Q_i$.

Suponha que o estado atual seja q e a entrada seja aw, onde $q \in Q_i$ e $a_i = (Q_i, \Sigma_i, P_i, q_{i0}, F_i)$. Se não há transição $P(q, aw, \beta)$, $q \in F_i$ e há um elemento $q' \in Q$ no topo da pilha, $q' \neq Z_0$, então q' é retirado da pilha e o estado corrente passa a ser q'.

A linguagem aceita por um autômato de pilha estruturado M é dada por $L(M) = \{w \in \Sigma^* | (q_0, w, Z_0) \vdash^* (q_f, \epsilon, Z_0)$, onde $q_f \in F\}$.

B. Autômato Adaptativo

O *Autômato Adaptativo*, proposto por Neto [14], é uma extensão do modelo do Autômato de Pilha Estruturado que permite o reconhecimento de linguagens mais complexas, dos tipos 1 e 0 segundo a Hierarquia de Chomsky. O termo *adaptativo*, neste contexto, pode ser definido como a capacidade de um dispositivo em alterar seu comportamento de forma espontânea. Logo, um autômato adaptativo tem como característica a possibilidade de sofrer alterações em sua topologia durante o processo de reconhecimento de uma dada cadeia [15].

Essa capacidade de alteração do autômato faz-se possível através da inclusão de ações adaptativas, que podem ser executadas antes e/ou depois de uma transição. A cada execução de uma ação adaptativa, o autômato tem sua topologia alterada, obtendo-se uma nova configuração. O objetivo de uma ação adaptativa é lidar com situações esperadas, mas ainda não consideradas, detectadas na cadeia submetida para reconhecimento pelo autômato [16]. Uma transição pode ter ações adaptativas associadas, que permitam a inclusão ou eliminação de estados e transições.

Ao executar uma transição que contém uma ação adaptativa associada, o autômato sofre mudanças, obtendo-se então uma nova máquina de estados. Para a aceitação de uma determinada cadeia, o autômato percorrerá um caminho em um espaço de máquinas de estados; em outras palavras, haverá uma máquina de estados inicial E_0, que iniciará o reconhecimento de uma determinada cadeia; máquinas de estados intermediárias E_i, que serão criadas ao longo do reconhecimento; e uma máquina de estados final E_n, que corresponde ao final do reconhecimento da cadeia. Seja a cadeia $w = \alpha_0\alpha_1 \ldots \alpha_n$; então o autômato M descreverá um caminho de máquinas de estados $< E_0, \alpha_0 > \rightarrow < E_1, \alpha_1 > \rightarrow \ \ldots \ \rightarrow < E_n, \alpha_n >$, onde E_i representa um autômato correspondente à aceitação da sub-cadeia α_i.

Um *autômato adaptativo* M é definido por $M = (Q, S, \Sigma, \Gamma, P, q_0, Z_0, F)$, tal que:

- Q é o conjunto finito de estados, $Q \subset Q^A$, Q^A é o conjunto de todos os estados possíveis, Q^A é enumerável,
- S é o conjunto de sub-máquinas,
- Σ é o alfabeto finito de entrada, $\Sigma \subset \Sigma^A$, Σ^A é o alfabeto enumerável de todos os símbolos possíveis,
- Γ é o alfabeto finito da pilha, $\Gamma \subset \Gamma^A$, $\Gamma = Q \cup \{Z_0\}$, Γ^A é o alfabeto enumerável de todos os símbolos possíveis da pilha, $\Gamma^A = Q^A \cup Z_0$,
- P é uma função $P : Q^A \times \Sigma^A \times \Gamma^A \rightarrow Q^A \times (\Sigma^A \cup \{\epsilon\}) \times (\Gamma^A \cup \{\epsilon\}) \times H^0 \times H^0\}$, H^0 definido a seguir,
- $q_0 \in Q$ é o estado inicial, $Z_0 \in \Gamma$ é o símbolo inicial da pilha,
- e $F \subset Q$ é o conjunto de estados finais.

H^0 é definido como $H^0 = \{f \mid f : E \rightarrow E\}$. Foi utilizado $g : X \rightarrow Y$ para uma relação g tal que $g \subseteq X \times Y$ e se $g(x, y)$ e $g(x, z)$, $x \in X$, $y, z \in Y$, então $y = z$. Como abreviação, usa-se $g(x) = y$ para $g(x, y)$.

Define-se $E = \{N : N$ é um autômato adaptativo $N = (Q', \Sigma', \Gamma', P', q_0, Z_0, F)$, onde $Q' \subset Q^A$, $\Sigma' \subset \Sigma^A$, $\Gamma' \subset \Gamma^A$, $P' : Q^A \times \Sigma^A \times \Gamma^A \rightarrow Q^A \times (\Sigma^A \cup \{\epsilon\}) \times (\Gamma^A \cup \{\epsilon\}) \times H^0 \times H^0\}$. Observe que q_0, Z_0 e F são os mesmos em qualquer $N \in E$.

O conjunto de todas as sub-máquinas do autômato adaptativo é representado por S. Cada *sub-máquina* i é definida como $s_i = (Q_i, \Sigma_i, P_i, q_{i0}, F_i)$, onde:

- $Q_i \subseteq Q$ é o conjunto de estados da sub-máquina i,
- $\Sigma_i \subseteq \Sigma$ é o conjunto de símbolos de entrada da sub-máquina i,
- $P_i \subseteq P$ é o mapeamento da sub-máquina i,
- $q_{i0} \in Q_i$ é o estado de entrada da sub-máquina i,
- e $F_i \subseteq Q_i$ é o conjunto de estados finais da sub-máquina i.

Uma transição é da forma $(q, a, \beta) \vdash (q', a', \beta', A, B)$ para $P(q, a, \beta) = (q', a', \beta', A, B)$, A e B são funções adaptativas, definidas a seguir, $A \in H^0$, $B \in H^0$. Se $\tilde{q}$, $\tilde{a}$ ou $\tilde{\beta}$ não pertencem à máquina atual, então $P(\tilde{q}, \tilde{a}, \tilde{\beta}) = (\tilde{q}, \tilde{a}, \tilde{\beta}, I, I)$, onde I é a função identidade em E.

As funções adaptativas de uma transição são funções de E em E. Contudo, foram definidas a seguir funções adaptativas como funções de $E \times G_1 \times G_2 \times \ldots \times G_k$ em E, $G_i = Q^A \cup \Sigma^A \cup \Gamma^A$, $k \in \mathbb{N}$. Isto é necessário porque uma

função adaptativa pode ser utilizada em mais de uma transição, com parâmetros reais correspondentes a G_i diferentes. Ao definir uma transição $(q, a, \beta) = (q', a', \beta', A, B)$, é sempre necessário fornecer os parâmetros reais correspondentes aos conjuntos G_i. Isto torna uma função adaptativa $A' : E \times G_1 \times G_2 \times \ldots \times G_k \rightarrow E$ em uma função $A : E \rightarrow E$. Em resumo, $A(e) = A'(e, r_1, r_2 \ldots r_k)$, onde r_i, $1 \leq i \leq k$, são os parâmetros reais, $r_i \in G$. Estes parâmetros reais podem variar de transição a transição; assim, uma função adaptativa $A' : E \times G_1 \times G_2 \times \ldots \times G_k \rightarrow E$ pode ser utilizada em diversas transições.

Uma *função adaptativa* é qualquer função $A \in H$, onde $H = \bigcup_{k \geq 0} \{f \mid f : E \times G_1 \times \ldots \times G_k \rightarrow E\}$, onde $G_i = G$ para todo $i \in \mathbb{N}$ e $G = Q^A \cup \Sigma^A \cup \Gamma^A$. Em outras palavras, H é o conjunto de todas as funções que tomam um autômato adaptativo mais um conjunto de parâmetros em G (qualquer número), e retorna um autômato adaptativo.

As funções adaptativas utilizam-se de variáveis e geradores para executar as ações de edição no autômato. As variáveis são preenchidas uma única vez na execução da função adaptativa. Os geradores são tipos especiais de variáveis, usados para definir nomes a estados recém-criados; os valores definidos são únicos e identificados pelo símbolo $*$, por exemplo, g_1^*, g_2^*.

Os autômatos adaptativos possuem três tipos de ações adaptativas elementares utilizadas para edição: ação adaptativa elementar de consulta (realiza uma busca no autômato por produções cujos componentes sejam correspondentes aos valores definidos), ação adaptativa elementar de remoção (remove uma produção de acordo com os valores definidos) e ação adaptativa elementar de inclusão (inclui uma produção de acordo com os valores definidos).

A execução do Autômato Adaptativo é igual à do Autômato de Pilha Estruturado, com a adição das funções adaptativas.

A linguagem aceita por um autômato adaptativo M é dada por $L(M) = \{w \in \Sigma^* | (q_0, w, Z_0) \vdash^* (q_f, \epsilon, Z_0)$, onde $q_f \in F\}$. O autômato adaptativo, graças à sua capacidade de sofrer auto-modificação, possui o poder de representação de uma Máquina de Turing [17].

C. Modelo de Controle de Acesso

O modelo de controle de acesso é definido por uma extensão do autômato adaptativo, chamado de *autômato adaptativo de controle de acesso*, que permite que um símbolo da cadeia de entrada seja passado como parâmetro a uma dada função adaptativa.

Um *autômato adaptativo de controle de acesso* M é definido por $M = (Q, S, \Sigma, \Gamma, P, q_0, Z_0, F)$, tal que:

- Q é o conjunto finito de estados,
- $Q \subset Q^A$, Q^A é o conjunto de todos os estados possíveis, Q^A é enumerável,
- S é o conjunto de sub-máquinas,
- Σ é o alfabeto finito de entrada, $\Sigma \subset \Sigma^A$, Σ^A é o alfabeto enumerável de todos os símbolos possíveis,
- Γ é o alfabeto finito da pilha, $\Gamma \subset \Gamma^A$, $\Gamma = Q \cup \{Z_0\}$, Γ^A é o alfabeto enumerável de todos os símbolos possíveis da pilha, $\Gamma^A = Q^A \cup \{Z_0\}$,
- P é uma função $P : Q^A \times (\Sigma^A \cup \{\epsilon\})^2 \times \Gamma^A \rightarrow Q^A \times (\Sigma^A \cup \{\epsilon\}) \times (\Gamma^A \cup \{\epsilon\}) \times H^0 \times H^0\}$, H^0 definido a seguir,
- $q_0 \in Q$ é o estado inicial,
- $Z_0 \in \Gamma$ é o símbolo inicial da pilha, e
- $F \subset Q$ é o conjunto de estados finais.

H^0 é definido como $H^0 = \{f \mid f : E \rightarrow E\}$. Foi utilizado $g : X \rightarrow Y$ para uma relação g tal que $g \subseteq X \times Y$ e se $g(x, y)$ e $g(x, z)$, $x \in X$, $y, z \in Y$, então $y = z$. Como abreviação, usa-se $g(x) = y$ para $g(x, y)$.

Define-se $E = \{N : N$ é um autômato adaptativo de controle de acesso $N = (Q', \Sigma', \Gamma', P', q_0, Z_0, F)$, onde $Q' \subset Q^A$, $\Sigma' \subset \Sigma^A$, $\Gamma' \subset \Gamma^A$, $P' : Q^A \times \Sigma^A \times (\Sigma^A \cup \{\epsilon\}) \times \Gamma^A \rightarrow Q^A \times (\Sigma^A \cup \{\epsilon\}) \times (\Gamma^A \cup \{\epsilon\}) \times H^0 \times H^0\}$. Observe que q_0, Z_0 e F são os mesmos em qualquer $N \in E$.

Uma transição pode ser da forma $(q, a, \beta) \vdash (q', a', \beta', A, B)$ para $P(q, a, \beta) = (q', a', \beta', A, B)$, ou $(q, ab, \beta) \vdash (q', a', \beta', A, B)$ para $P(q, ab, \beta) = (q', a', \beta', A, B)$, A e B são funções adaptativas, $A \in H^0$, $B \in H^0$. Se $\tilde{q}$, $\tilde{a}$, $\tilde{b}$ ou $\tilde{\beta}$ não pertencem à máquina atual, então $P(\tilde{q}, \tilde{a}, \tilde{\beta}) = (\tilde{q}, \tilde{a}, \tilde{\beta}, I, I)$ ou $P(\tilde{q}, \tilde{a}\tilde{b}, \tilde{\beta}) = (\tilde{q}, \tilde{a}\tilde{b}, \tilde{\beta}, I, I)$, onde I é a função identidade em E.

A linguagem aceita por um autômato adaptativo de controle de acesso M é dada por $L(M) = \{w \in \Sigma^* | (q_0, w, Z_0) \vdash^* (q_f, \epsilon, Z_0)$, onde $q_f \in F\}$.

A especificação do modelo de controle de acesso é descrita por um autômato adaptativo de controle de acesso M. O alfabeto do autômato adaptativo de controle de acesso é representado por Σ, $\Sigma \subset \Sigma^A$, $\Sigma^A = \{w \mid w$ é um nome de indivíduo válido $\} \cup \{x \mid x$ é um nome de objeto válido $\} \cup \{\lambda_x \mid x$ é um nome de objeto válido $\} \cup \{y \mid y$ é um direito válido $\} \cup \{\lambda_{NS}, \lambda_{NO}, \lambda_{RS}, \lambda_{RO}, \lambda_{RAO}, G, T\}$. Assim, o modelo de controle de acesso contém um conjunto finito de indivíduos atuais $S = \{s_1 \ldots s_{n_1}\}$, $S \subset \Sigma$; um conjunto finito de objetos atuais $O = \{o_1 \ldots o_{n_2}\}$, $O \subset \Sigma$; um conjunto finito de direitos $R = \{r_1 \ldots r_{n_3}\}$, $R \subset \Sigma$; um conjunto λ de símbolos que representam as ações sobre os objetos, $\lambda = \{\lambda_{o_1} \ldots \lambda_{o_{n_2}}\}$, $\lambda \subset \Sigma$; e um conjunto de símbolos que representam as alterações a serem realizadas na configuração do autômato, $\{\lambda_{NS}, \lambda_{NO}, \lambda_{RS}, \lambda_{RO}, \lambda_{RAO}, G, T\}$.

O autômato adaptativo de controle de acesso M reconhece as cadeias apresentadas a seguir.

- $\boldsymbol{\lambda_{NS} a}$: criação de um indivíduo a
- $\boldsymbol{s_i \lambda_{NO} b}$: criação de um objeto b associado a um indivíduo s_i
- $\boldsymbol{s_i \lambda_{RS}}$: remove o indivíduo s_i
- $\boldsymbol{s_i \lambda_b \lambda_{RO}}$: remove o objeto b associado ao indivíduo s_i
- $\boldsymbol{\lambda_{RAO} b}$: remove o objeto b do sistema
- $\boldsymbol{s_i \lambda_b G r_k}$: insere um direito r_k no objeto b do indivíduo s_i
- $\boldsymbol{s_i \lambda_b T r_k}$: remove um direito r_k do objeto b associado ao indivíduo s_i
- $\boldsymbol{s_i o_j}$: verifica se o indivíduo s_i tem relação com o objeto o_j
- $\boldsymbol{s_i o_j r_k}$: verifica se o indivíduo s_i tem direito r_k sobre o objeto o_j
- $\boldsymbol{s_i o_j r_1 \ldots r_k}$: verifica se o indivíduo s_i tem direitos $r_1 \ldots r_k$ sobre o objeto o_j

No modelo proposto, relações nulas entre indivíduos e objetos não serão representadas; dessa forma, um objeto estará sempre relacionado a i indivíduos, $1 \leq i \leq n_1$. Isso permite a redução do espaço computacional em comparação com o modelo HRU, onde todos os objetos possuem relação com todos os indivíduos, $O \times S$, mesmo que essa relação seja nula ($\varnothing$).

O modelo de controle de acesso possui um conjunto C de *comandos de privacidade*, $C = \{c_1 \dots c_i\}$, $i \in \mathbb{N}$. Esses comandos são codificações de regras-alvo obtidas a partir de uma legislação ou política de privacidade escrita em linguagem natural, e verificam se uma determinada situação pode ser caracterizada como violação de privacidade. A gramática da linguagem de um comando de privacidade é descrita a seguir.

Program ::= Id "(" ParamList ")" "início"
StatementList "fim"
ParamList ::= Id | Id "," ParamList
StatementList ::= | Statement StatementList
Statement ::= CondStat | AtribStat | ReturnStat
CondStat ::= "se" CondExpr "então" "{" StatementList "}"
["senão" "{" StatementList "}"]
CondExpr ::= CondCom | CondCom "e" CondExpr
CondCom ::= EvalCom | EquivCom
EvalCom ::= "eval" "(" (Arg)+ "," Id ")"
EquivCom ::= "equiv" "(" Id "," Id ")"
Arg ::= "<" (Id | "[" Id "]" | Id "[" Id "]") ">"
AtribStat ::= Id "=" (Id | ExecCom) ";"
ExecCom ::= "exec" "(" (Arg)+ "," Id ")"
ReturnStat ::= "retornar" Valor
Valor ::= "true" | "false"

Um comando de privacidade $c_i \in C$ é uma função que recebe como parâmetro um conjunto não-vazio de argumentos e retorna "*true*" se as condições presentes no corpo da função foram satisfeitas e se os comandos foram executados com sucesso, ou "*false*" caso contrário.

É definida uma variável de escopo global "e", que representa o autômato adaptativo de controle de acesso do modelo. Todos os comandos de privacidade podem referenciar-se ao autômato através da variável "e".

A função *eval*(u, e) recebe uma cadeia u e um autômato e, e retorna "*true*" se essa cadeia pertencer à linguagem descrita pelo autômato, $u \in L(e)$, ou "*false*" caso contrário.

A função *exec*(u, e) recebe uma cadeia u e um autômato e, e retorna um autômato adaptativo. A cadeia u deve possuir algum dos símbolos que representam as alterações a serem realizadas na configuração do autômato. Ao executar a função *exec*(u, e), espera-se que a cadeia altere a configuração do autômato; assim, o retorno da função será o autômato alterado, em caso de sucesso, ou o autômato original, caso a cadeia não pertença à linguagem descrita por e, $u \notin L(e)$, ou que u seja apenas uma cadeia de consulta.

A função *equiv*(e, f) toma como argumentos os autômatos adaptativos e e f e verifica se são equivalentes pela inspeção de isomorfismo. Se e e f são isomorfos, a função *equiv*(e, f) retorna '*true*", ou '*false*" caso contrário.

As cadeias submetidas às funções são definidas através da concatenação de símbolos pré-definidos pertencentes ao alfabeto Σ e os argumentos do comando de privacidade c_i, $i \in \mathbb{N}$. De acordo com a gramática da linguagem do comando de privacidade, cada símbolo pré-definido $v_j \in \Sigma$ é representado na forma $<v_i>$, e um argumento α é representado na forma $<[\alpha]>$. A seguir, um exemplo do comando *create*, que cria um objeto b e informa que a é o dono deste (direito o, de *owner*).

create(a, b)
início
auto = *exec*($<[a]><\lambda_{NO}><[b]>$, *auto*);
auto = *exec*($<[a]><\lambda_{[b]}><G><o>$, *auto*);
retornar *true*;
fim

A criação do autômato adaptativo de controle de acesso é feita de maneira *offline*, definido-se qual será a configuração inicial do modelo de controle de acesso. A configuração do autômato e as funções adaptativas que este possui podem ser alteradas de acordo com a necessidade de cada sistema. Por exemplo, cada indivíduo tem um objeto associado a ele, com as devidas permissões; é possível, no entanto, modelar o sistema para que uma determinada combinação de permissões seja aplicada a um grupo de indivíduos, criando o conceito de *classes de indivíduos*; assim, ao alterar a permissão de um indivíduo s_i para com um determinado objeto o_j, todos os indivíduos que acessam o_j terão suas permissões alteradas.

De um modo geral, o autômato adaptativo de controle de acesso $M = (Q, S, \Sigma, \Gamma, P, q_0, Z_0, F)$ pode ser modelado com as considerações definidas a seguir.

Para cada indivíduo $s_j \in S$, existe uma transição consumindo s_j, partindo do estado inicial q_0, $q_0 \in Q$, que leva até o estado q_{s_j}, $(q_0, s_j) \vdash (q_{s_j}, \epsilon)$, e para cada indivíduo $s_j \in S$ existem i objetos associados a ele, $0 \leq i \leq n_2$.

Há de se observar que o autômato possuirá $(\sum_{i=1}^{n_3} C_{n,i}) + 1$ estados que representam todas as combinações possíveis de direitos, $q_{r_1} \dots q_{r_n}, q_{r_1 r_2} \dots q_{r_1 r_n} \dots q_{r_1 \dots r_n}$, juntamente com um estado q_{nulo} que representa um estado inicial de direitos, sem nenhuma permissão determinada. $C_{n,i}$ é uma combinação simples, cuja fórmula é representada por:

$$C_{n,i} = \frac{n!}{i!(n-i)!} \tag{1}$$

$C_{n,i}$ (Equação 1) denota o número total de combinações de n elementos tomados i a i, onde i é o número de elementos presentes em cada combinação.

A partir do estado q_{s_j}, existe uma transição que consome λ_{NO} que leva ao estado final q_f, $q_f \in F$, e passando o parâmetro presente na cadeia à função responsável pela criação do novo objeto o_m, $(q_{s_j}, \lambda_{NO}[o_m]) \vdash (q_f, \epsilon)$. Se o indivíduo s_j possui um objeto o_i associado a ele, então a partir do estado q_{s_j}, existe uma transição consumindo o_i que leva ao estado $q_{r_i \dots r_j}$, que representa uma das possíveis combinações de direitos, $(q_{s_j}, o_i) \vdash (q_{r_i \dots r_j}, \epsilon)$. A partir do estado $q_{s_j o_i}$, haverá uma transição consumindo λ_{RO} que leva ao estado final qf, $q_f \in F$, removendo o objeto o_i do indivíduo s_j, $(q_{s_j o_i}, \lambda_{RO}) \vdash (g_f, \epsilon)$.

Um objeto recém-criado o_k, associado a um indivíduo s_j, não terá permissões definidas *a priori*, $(q_{s_j}, o_k) \vdash (q_{nulo}, \epsilon)$.

Para cada objeto o_i, que está associado a um indivíduo s_j, haverá uma transição consumindo λ_{o_i}, partindo do estado q_{s_j} até o estado $q_{s_j o_i}$, $(q_{s_j}, \lambda_{o_i}) \vdash (q_{s_j o_i}, \epsilon)$, onde o estado $q_{s_j o_i}$ representa as operações aplicáveis ao objeto o_i do indivíduo s_j.

Cada estado $q_{r_i \ldots r_j}$, que representa uma das possíveis combinações de direitos, terá uma sub-máquina N_i associada que fará o reconhecimento do símbolo α, $(q_{r_i \ldots r_j}, \alpha) \vdash (q_f, \epsilon)$, $q_f \in F$.

A partir do estado $q_{s_j o_i}$, existe uma transição consumindo G que leva ao estado $q_{s_j o_i G}$, responsável pela concessão de direitos ao objeto o_i do indivíduo s_j, $(q_{s_j o_i}, G) \vdash (q_{s_j o_i G}, \epsilon)$, e a partir do estado $q_{s_j o_i}$, existe uma transição consumindo T que leva ao estado $q_{s_j o_i T}$, responsável pela remoção de direitos ao objeto o_i do indivíduo s_j, $(q_{s_j o_i}, T) \vdash (q_{s_j o_i T}, \epsilon)$.

A partir do estado inicial q_0, existe uma transição consumindo λ_{NS} que leva ao estado final q_f, e passando o parâmetro presente na cadeia à função responsável pela criação do novo indivíduo s_m, $(q_0, \lambda_{NS}[s_m]) \vdash (q_f, \epsilon)$, enquanto que a partir do estado inicial q_0, existe uma transição consumindo λ_{RS} que leva ao estado final q_f, e passando o parâmetro presente na cadeia à função responsável pela remoção do indivíduo s_i, $(q_0, \lambda_{RS}[s_i]) \vdash (q_f, \epsilon)$.

A partir do estado inicial q_0, existe uma transição consumindo λ_{RAO} que leva ao estado final q_f, e passando o parâmetro presente na cadeia à função responsável pela remoção do objeto o_j, $(q_0, \lambda_{RAO}[o_j]) \vdash (q_f, \epsilon)$.

O autômato possuirá $\sum_{i=0}^{n_3-1} C_{n,i}(n-i)$ funções adaptativas de inclusão de direitos, e $\sum_{i=0}^{n_3-1} C_{n,i}(n-i)$ funções adaptativas de remoção de direitos. Ao criar um objeto, estarão disponíveis apenas as funções de inclusão de direitos, já que o objeto não possui nenhum direito; a medida que os direitos são incluídos ou removidos, as funções atuais são substituídas por outras. $C_{n,i}$ é uma combinação simples, representada na fórmula 1.

A partir do estado $q_{s_j o_i G}$, existirão $k = n_3 - l$ transições consumindo $r_1 \ldots r_k$, com funções adaptativas de inclusão de direitos $F_{inc,1} \ldots F_{inc,k}$ que levam até o estado final q_f, e a partir do estado $q_{s_j o_i T}$, existirão $l = n_3 - k$ transições consumindo $r_1 \ldots r_l$ funções adaptativas de remoção de direitos $F_{rem,1} \ldots F_{rem,l}$ que levam até o estado final q_f.

A Figura 1 ilustra a configuração de uma sub-máquina genérica N_i, responsável pelo reconhecimento dos direitos permitidos de um determinado objeto associado a um indivíduo. Ela é chamada recursivamente para todo direito a ser verificado na cadeia e, se todos estiverem presentes, ela retornará cada chamada, até chegar ao estado final q_f.

A Figura 2 apresenta um exemplo de configuração do autômato adaptativo M do modelo de controle de acesso $W = (S, O, R, \lambda, M)$, $S = \{Ana\}$, $O = \{arq1\}$, $R = \{r, w\}$ e $\lambda = \{\lambda_{NO}, \lambda_{RO}, \lambda_{NS}, \lambda_{RS}, \lambda_{RAO}, \lambda_{arq1}\}$. A configuração do autômato M indica que *Ana* é um indivíduo do sistema e possui um objeto *arq1* associado a ela, com permissões de *leitura* (r) e *escrita* (w).

As funções adaptativas do autômato adaptativo M da Figura 2 estão descritas a seguir; os parâmetros das mesmas foram omitidos.

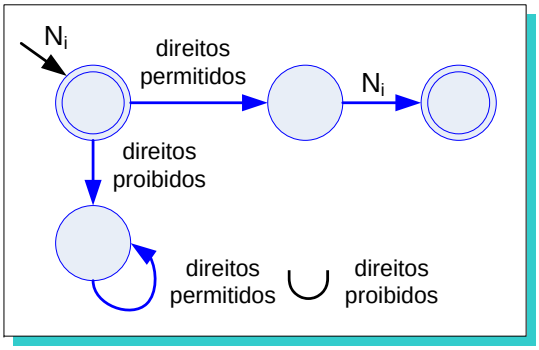

Figura 1. Sub-máquina genérica N_i.

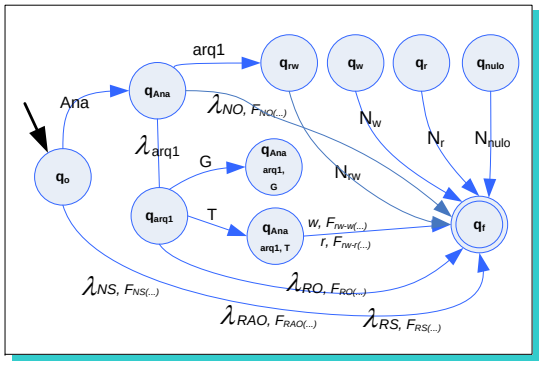

Figura 2. Exemplo de configuração do autômato adaptativo M.

$\boldsymbol{F_{NS}(\ldots)}$: função de criação de um indivíduo s_i
$\boldsymbol{F_{NO}(\ldots)}$: função criação de um objeto b associado a um indivíduo s_i
$\boldsymbol{F_{RS}(\ldots)}$: função de remoção de um indivíduo s_i
$\boldsymbol{F_{RO}(\ldots)}$: função de remoção de um objeto b associado a um indivíduo s_i
$\boldsymbol{F_{RAO}(\ldots)}$: função de remoção de um objeto b do sistema
$\boldsymbol{F_{rw-r}(\ldots)}$: função que remove a permissão de leitura (r) de um objeto o_j, quando este possui as permissões rw.
$\boldsymbol{F_{rw-w}(\ldots)}$: função que remove a permissão de escrita (w) de um objeto o_j, quando este possui as permissões rw.
$\boldsymbol{F_{r-r}(\ldots)}$: função que remove a permissão de leitura (r) em um objeto o_j, quando este possui permissão r.
$\boldsymbol{F_{w-w}(\ldots)}$: função que remove a permissão de escrita (w) em um objeto o_j, quando este possui permissão w.
$\boldsymbol{F_{r+w}(\ldots)}$: função que insere a permissão de escrita (w) em um objeto o_j, quando este possui permissão r.
$\boldsymbol{F_{w+r}(\ldots)}$: função que insere a permissão de leitura (r) em um objeto o_j, quando este possui permissão w.
$\boldsymbol{F_{+r}(\ldots)}$: função que insere a permissão de leitura (r) em um objeto o_j, quando este não possui permissões definidas.
$\boldsymbol{F_{+w}(\ldots)}$: função que insere a permissão de escrita (w) em um objeto o_j, quando este não possui permissões definidas.

As funções de inclusão de direitos F_{r+w}, F_{w+r}, F_{+r} e F_{+w} não estão presentes no autômato adaptativo M, pois uma vez que o objeto *arq1*, associado ao indivíduo *Ana*, possui todas

as permissões possíveis, leitura (r) e escrita (w), não há mais permissões a serem inseridas.

A Figura 3 ilustra a sub-máquina N_{rw} do autômato adaptativo M. As definições das demais sub-máquinas N_{nulo}, N_r e N_w foram omitidas.

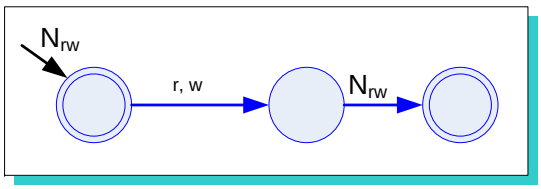

Figura 3. Definição da sub-máquina N_{rw}.

O modelo permite a consulta dos direitos de um determinado objeto, associado a um indivíduo. Assim, as seguintes consultas <*Ana*><*arq1*>, <*Ana*><*arq1*><*r*>, <*Ana*><*arq1*><*w*>, <*Ana*><*arq1*><*r*><*w*> e <*Ana*><*arq1*><*w*><*r*> seriam aceitas pelo autômato e, portanto, estariam em conformidade com as permissões definidas na modelagem do sistema.

Através de um Sistema de Privacidade Auditável com autômato adaptativo, uma determinada legislação sobre privacidade pode ser formalizada em um conjunto de comandos robustos para verificar a conformidade de um modelo de controle de acesso com essa legislação.

IV. Resultados

O Sistema de Privacidade Auditável que utiliza autômato adaptativo como mecanismo de controle de acesso é genérico o suficiente e permite incorporar as alterações dinâmicas do autômato adaptativo no mecanismo de controle de acesso, permitindo customizar para a área de aplicação.

A eficiência em tempo e espaço pode ser comparada com outros mecanismos existentes, e é apresentado a seguir uma comparação com o modelo HRU que é considerado simples e suficiente para implementar os requisitos do Sistema de Privacidade Auditável, sendo escolhido "[...] parar iniciar o modelo desde o começo, adicionando apenas os elementos que forem realmente necessários" [7].

A criação e remoção de indivíduos são semelhantes nos dois modelos.

No modelo proposto, relações nulas não serão representadas; assim, um objeto estará sempre relacionado a i indivíduos, $1 \leq i \leq n_1$, enquanto que no modelo HRU, o objeto é criado e relacionado com todos os indivíduos do sistema, mesmo que estes indivíduos possuam relação nula com o objeto.

Como exemplo, suponha um conjunto S com k indivíduos, $S = \{s_1, s_2 \dots s_k\}$, e um conjunto O com k objetos, $O = \{o_1, o_2 \dots o_k\}$, onde cada indivíduo s_i possui relação apenas com o objeto o_i. A Tabela I mostra a representação das relações entre indivíduos e objetos do exemplo no modelo HRU; o símbolo • indica as relações válidas, enquanto que o símbolo ∅ representa as relações nulas.

No modelo HRU, todos os indivíduos possuem relação com todos os objetos, $S \times O$. Assim, se $k = 30$, existirão 30 indivíduos relacionando-se com 30 objetos, obtendo-se um

Tabela I
Representação das relações entre indivíduos e objetos no modelo HRU.

	o_1	o_2	$\cdots$	o_k
s_1	•	∅	$\cdots$	∅
s_2	∅	•	$\cdots$	∅
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
s_k	∅	∅	$\cdots$	•

total de 900 relações. No modelo proposto, apenas as relações válidas são consideradas; dessa forma, existirão apenas 30 relações no sistema.

Seja T_{AA} o total de relações do modelo proposto, e T_{HRU} o total de relações do modelo HRU. No pior caso, onde $S \times O$, ou seja, todos os indivíduos têm relação com todos os objetos, $T_{AA} = T_{HRU}$; em todas as outras situações, $T_{AA} < T_{HRU}$.

Ao remover um objeto, o modelo proposto permite que todas as relações de um determinado objeto sejam removidas, como também acontece no modelo HRU, ou simplesmente remover uma determinada relação entre um indivíduo e este objeto.

Os comandos para inserção e remoção de direitos são semelhantes nos dois modelos.

De forma geral, o modelo proposto permite que consultas sobre permissões de um determinado objeto sejam feitas em uma única cadeia, na forma $s_j o_k r_1 \dots r_n$, onde s_j é o indivíduo, o_k é o objeto, e $r_1 \dots r_n$ são as permissões a serem verificadas, $n \geq 1$. Cada combinação i de permissões está associada a uma sub-máquina N_i, que faz o reconhecimento de cadeias que possuem tais elementos. No modelo HRU, é necessário um conjunto de comandos para realizar uma consulta.

Como exemplo, suponha que deseja-se saber se o indivíduo s_i possui n direitos, $r_1 \dots r_n$, sobre o objeto o_j. A Figura 4 mostra as consultas necessárias para a verificação do exemplo através do modelo HRU.

$$\left.\begin{array}{l} \textbf{if } r_1 \textbf{ in } (X_{s_i}, X_{o_j}) \textbf{ and} \\ \quad r_2 \textbf{ in } (X_{s_i}, X_{o_j}) \textbf{ and} \\ \qquad \vdots \\ \quad r_n \textbf{ in } (X_{s_i}, X_{o_j}) \textbf{ then} \end{array}\right\} n \text{ consultas}$$

Figura 4. Conjunto de consultas no modelo HRU.

De acordo com a Figura 4, seriam necessárias n consultas para verificar se o indivíduo s_i possui n direitos de permissão sobre o objeto o_j. No modelo proposto, a verificação pode ser realizada em apenas uma consulta, conforme observado na Figura 5.

$$s_i o_j \underbrace{r_1 r_2 \dots r_n}_{n \text{ direitos}}$$

Figura 5. Consulta de direitos no modelo proposto.

As consultas no modelo proposto são realizadas em apenas uma cadeia a ser submetida ao autômato, enquanto que no

modelo HRU é necessária uma consulta individual para cada direito a ser verificado. Assim, se $n = 10$, serão necessárias 10 consultas no modelo HRU, e apenas 1 no modelo proposto.

O Sistema de Privacidade Auditável com autômato adaptativo, apresentado neste artigo, é definido como uma abordagem diferente dos demais mecanismos utilizados para proteção de privacidade, citados na seção I. Partindo-se de um documento escrito em linguagem natural, é possivel reduzí-lo a um conjunto de regras-alvo para serem codificadas em comandos, sendo então fornecidos ao sistema; esses comandos verificam se uma situação pode ser caracterizada como violação de privacidade, de acordo com a legislação sobre a qual os comandos foram definidos.

Como exemplo, suponha o seguinte parágrafo a ser analisado:

"Um indivíduo pode copiar um objeto para outro indivíduo, se este primeiro for o dono do objeto a ser copiado. O objeto copiado deverá ser apenas para leitura."

O primeiro passo é reduzir o parágrafo anterior a conjunto de regras-alvo:

sejam
A, B: indivíduos
C: objeto

se A é dono de C **então**
A copia C para B
Atribuir leitura ao objeto copiado C

A partir destas regras, o comando *COPY* pode ser codificado:

copy(a, b, c)
início
se *eval*(<[a]><[c]><o>, *auto*) **então** {
auto = *exec*(<[b]><λ_{NO}><[c]>, *auto*);
auto = *exec*(<[b]><$\lambda_{[c]}$><G><r>, *auto*);
retornar *true*;
}
fim

O comando recém-criado *COPY* está em conformidade com o texto no qual foi codificado.

V. Conclusões

O Sistema de Privacidade Auditável com autômato adaptativo pode comportar-se de maneira mais eficiente que o Sistema de Privacidade Auditável, reduzindo o espaço computacional requerido, pois o modelo HRU apresenta uma relação $S \times O$, onde S é o conjunto de indivíduos e O é o conjunto de objetos, demandando espaço para representar relações, inclusive as nulas ($\varnothing$). Além disso, as consultas são otimizadas, reduzindo o tempo computacional para executar um determinado comando.

O trabalho apresentado evidencia a contribuição para a área de Tecnologias Adaptativas de um novo domínio de aplicabilidade.

A partir da definição do Sistema de Privacidade Auditável com autômato adaptativo, é possível expandir sua utilização além da privacidade, pois o modelo é genérico o suficiente para permitir outras classes de problemas, sem sofrer alterações.

Referências

[1] C. H. Fernandes, *A Privacidade na Sociedade da Informação*, 2003.

[2] R. E. Grande, *Sistema de Integração de Técnicas de Proteção de Privacidade que permitem Personalização*, Master's Thesis – Programa de Pós-Graduação em Ciência da Computação, Departamento de Computação. Universidade Federal de São Carlos, São Carlos, 2006.

[3] A. Pfitzmann and M. Köhntopp, "Anonymity, unobservability, and pseudonymity – a proposal for terminology," *Designing Privacy Enhancing Technologies: Proceedings of the International Workshop on the Design Issues in Anonymity and Observability*, vol. 2009, no. 2, pp. 1–9, July 2000.

[4] K. Coyle, *A social analysis of the platform for privacy preferences (P3P)*, Platform for Privacy Preferences (P3P) Project, 1999.

[5] L. Ishitani, *Uma Arquitetura para Controle de Privacidade na Web*, Doctoral Dissertation – Departamento de Ciência da Computação. Universidade Federal de Minas Gerais, 2003.

[6] B. Lampson, "Protection," in *Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems*, Princeton University, 1971, pp. 437–443. [Online]. Available: citeseer.ist.psu.edu/287804.html

[7] M. J. May, C. A. Gunter, and I. Lee, "Privacy apis: Access control techniques to analyze and verify legal privacy policies," in *CSFW '06: Proceedings of the 19th IEEE workshop on Computer Security Foundations*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 85–97.

[8] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman, "Protection in operating systems," *Commun. ACM*, vol. 19, no. 8, pp. 461–471, 1976.

[9] C. A. Gunter, M. J. May, and S. G. Stubblebine, "A formal privacy system and its application to location based services." in *Privacy Enhancing Technologies*, 2004, pp. 256–282.

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

[11] H. Pistori and J. J. Neto, "Adaptree – proposta de um algoritmo para indução de Árvores de decisão baseado em técnicas adaptativas," *Anais Conferência Latino Americana de Informática – CLEI 2002*, November 2002.

[12] C. Menezes and J. J. Neto, "Um método para a construção de analisadores morfológicos, aplicado à língua portuguesa, baseado em autômatos adaptativos," *V PROPOR, Encontro para o Processamento Computacional de Português Escrito e Falado*, November 2000.

[13] J. Luz and J. J. Neto, "Tecnologia adaptativa aplicada à otimização de código em compiladores," *IX Congreso Argentino de Ciencias de la Computación*, October 2003.

[14] J. J. Neto, *Contribuições à Metodologia de Construção de Compiladores*, Thesis (Livre Docência) – Escola Politécnica da Universidade de São Paulo, São Paulo, 1993.

[15] ——, "Adaptive automata for context-dependent languages," *ACM SIGPLAN Notices*, vol. 29, no. 9, 1994.

[16] ——, "Solving complex problems efficiently with adaptive automata," *Lecture Notes in Computer Science – Implementation and Application of Automata 5th International Conference*, vol. 2088, 2000.

[17] R. L. A. Rocha and J. J. Neto, "Autômato adaptativo, limites e complexidade em comparação com máquina de Turing," *Proceedings of the second Congress of Logic Applied to Technology - LAPTEC'2000*, pp. 33–48, 2001.

Paulo Roberto Massa Cereda obteve a graduação em Computação: Sistemas de Informação pelo Centro Universitário Central Paulista em 2005. Atualmente é aluno do Programa de Pós-Graduação em Ciência da Computação, do Departamento de Computação da Universidade Federal de São Carlos. É aluno pesquisador do Grupo de Sistemas Distribuídos e Redes (GSDR), atuando nas áreas de Privacidade e Personalização de Serviços e de Tecnologias Adaptativas.

Sérgio Donizetti Zorzo obteve o doutorado em Engenharia Elétrica pela Universidade de São Paulo em 1996. Atualmente é professor associado do Departamento de Computação da Universidade Federal de São Carlos. Tem experiência na área de Ciência da Computação, com ênfase em Teleinformática, atuando principalmente em Qualidade, Privacidade e Personalização de Serviços, Multimídia, Computação em Ambientes sem Fio e Tecnologias Adaptativas.