UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM

CIÊNCIA DA COMPUTAÇÃO

Modelo de Controle de Acesso Adaptativo

Paulo Roberto Massa Cereda

São Carlos – SP

Maio/2008

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

C414mc

Cereda, Paulo Roberto Massa.
Modelo de Controle de Acesso Adaptativo / Paulo Roberto Massa Cereda. -- São Carlos : UFSCar, 2008.
119 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2008.

1. Teoria de autômatos. 2. Autômato adaptativo. 3. Privacidade e personalização. 4. Computadores - controle de acesso. I. Título.

CDD: 511.35 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Modelo de Controle de Acesso Adaptativo"

PAULO ROBERTO MASSA CEREDA

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação

Membros da Banca:

Prof. Dr. Sérgio Donizetti Zorzo
(Orientador - DC/UFSCar)

Prof. Dr. José de Oliveira Guimarães
(DC/UFSCar)

Profa. Dra. Sandra Abib
(DC/UFSCar)

Prof. Dr. João José Neto
(POLI/USP)

São Carlos
Maio/2008

"Maria de Nazaré, Maria me cativou. Fez mais forte a minha fé e por filho me adotou. Às vezes, eu paro e fico a pensar e, sem perceber, me vejo a rezar. E o meu coração se põe a cantar pra Virgem de Nazaré. Menina que Deus amou e escolheu pra Mãe de Jesus, o Filho de Deus. Maria que o povo inteiro elegeu Senhora e Mãe do Céu."

"Maria de Nazaré", PADRE ZEZINHO, SCJ.

Agradecimentos

Em primeiro lugar, à Virgem Maria, Mãe de Deus e nossa Mãe, que, sempre intercedendo por nós junto ao seu Filho, nos guia nos caminhos da vida.

Aos meus pais, Ederval e Lúcia, por todo o apoio, amor e carinho. Sempre tive o apoio de meus pais em todos os momentos, dos mais tristes aos mais felizes.

Ao meu orientador, prof. Sérgio Donizetti Zorzo, por todo o auxílio e cooperação. Saí das mãos de um grande orientador na graduação (prof. Durval Makoto Akamatu) e fui acolhido por um grande orientador no Mestrado. Muito obrigado por ter acreditado em mim.

Ao prof. José de Oliveira Guimarães, do Departamento de Computação da Universidade Federal de São Carlos (DC/UFSCar), por toda a ajuda na revisão e verificação das definições presentes nesta dissertação. Aprendi muito durante nossas reuniões. Obrigado pela participação na banca examinadora.

Ao prof. João José Neto, da Escola Politécnica da Universidade de São Paulo (Poli/USP), por todo o incentivo ao desenvolvimento deste trabalho e pela participação na banca examinadora. É um grande exemplo de pesquisador e uma pessoa sensacional, foi uma honra tê-lo na defesa desta dissertação.

À profa. Sandra Abib (DC/UFSCar) por toda a ajuda e pelos momentos de descontração. Foi uma alegria imensa tê-la na banca examinadora. Que Nossa Senhora sempre ilumine sua vida.

Aos professores do DC/UFSCar, por todos os ensinamentos, dicas e sugestões durante esse período. Em especial, agradeço aos profs. Hélio Crestana Guardia, Luís Carlos Trevelin, Antônio Francisco do Prado, Estevam Rafael Hrushka Júnior e Mauro Biajiz.

Ao prof. Sebastião Elias Kuri, do Departamento de Engenharia de Materiais da Universidade Federal de São Carlos (DEMa/UFSCar), por todo o apoio e incentivo para o meu ingresso no Programa de Pós-Graduação.

Ao prof. Ricardo Luís de Azevedo da Rocha (Poli/USP), coordenador da seção de ferramentas do Segundo Workshop de Tecnologias Adaptativas (WTA2008), pelas valiosas sugestões ao desenvolvimento deste trabalho. Paz e Bem!

Aos funcionários do DC/UFSCar, por todo o auxílio e amizade. Em especial, Evélton Cardoso de Marco, Dermeval de Jesus Ambrósio, Maria Cristina Carreira Trevelin, Mirian Cristina Thomé e Jorgina Vera de Moraes.

Aos meus grandes amigos Bruno Yuji Lino Kimura, Reginaldo Aparecido Gotardo, Ricardo Araújo Rios e Robson Eduardo de Grande, pelo companheirismo e amizade. Passamos grandes momentos juntos e superamos todos os obstáculos com muito bom humor. Muito obrigado, amigos.

À minha amiga Tatiane Marques Nogueira (DC/UFSCar), pela confecção de um modelo BibTeX da Biblioteca Comunitária da Universidade Federal de São Carlos. Valeu, Tati!

Ao prof. Sadao Massago, do Departamento de Matemática da Universidade Federal de São Carlos (DM/UFSCar), pela disponibilização do excelente material do curso de LaTeX. Utilizei vários trechos dos códigos presentes no material para a escrita desta dissertação.

À Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (Capes), pelo apoio financeiro para o desenvolvimento deste trabalho.

Por fim, agradeço a todos que, direta ou indiretamente, contribuiram para a realização deste trabalho. Muito obrigado por toda a ajuda.

Resumo

A privacidade tornou-se um aspecto importante na modelagem de sistemas computacionais que envolvem informações pessoais e ações que as manipulam. Para garantir a segurança das informações, são utilizados modelos de controle de acesso. Alguns destes modelos tratam dos requisitos de privacidade, mas possuem aplicação apenas para domínios específicos. Este trabalho propõe um modelo de controle de acesso simplificado e genérico o suficiente para contemplar os requisitos de segurança e privacidade de um sistema. O modelo proposto utiliza um autômato adaptativo para realizar o controle de acesso, e possui um conjunto de comandos de privacidade, codificados a partir de uma legislação ou política, que verificam se uma determinada ação no sistema pode ser caracterizada como violação de privacidade. Além disso, o modelo permite a utilização de mecanismos de auditoria para registrar as ações no sistema e garantir a privacidade.

Palavras-chave: Autômato Adaptativo, Privacidade, Controle de Acesso.

Abstract

Privacy has become an important aspect when modeling computational systems that have personal information and actions that deal with them. Control access models are used to provide security of such information. Some of these models, altough, are just applied to specific domains. It is proposed an access control model simple yet generic enough to comply with the security and privacy aspects of a system. The proposed model uses an adaptive automaton to perform the access control, and has a set of privacy commands, coded from a legislation or policy. These commands verify if a certain action in the systen may be characterized as a privacy violation. Moreover, the model may be used with auditing mechanisms to register actions in the system and guarantee the privacy.

Keywords: Adaptive Automaton, Privacy, Access Control.

Lista de Figuras

Lista de Tabelas

Lista de Códigos

Lista de Abreviaturas e Siglas

AACA	Autômato Adaptativo de Controle de Acesso
DAC	Controle de Acesso Discricionário
FTC	*Federal Trade Comission*
HRU	Modelo de Harrison, Ruzzo e Ullman
HTTP	Protocolo de Transferência de Hipertexto
MAC	Controle de Acesso Obrigatório
MACP	Modelo de Autorização Contextual baseado em Papéis
MASKS	*Managing Anonymity while Sharing Knowledge to Servers*
MCAA	Modelo de Controle de Acesso Adaptativo
MGD	Modelo de Graham-Denning
MPT	Modelo de Privacidade baseado em Tarefas
OECD	*Organization for Economic Co-operation and Development*
P3P	Plataforma para Preferências de Privacidade
Pricomlan	Linguagem para Comandos de Privacidade
PSA	Agente de Privacidade e Segurança
RBAC	Controle de Acesso Baseado em Papéis
SPA	Sistema de Privacidade Auditável
SPA^3	Sistema de Privacidade Auditável com Autômato Adaptativo

Sumário

CAPÍTULO

1

Introdução

"Tudo neste mundo tem uma resposta. O que leva é tempo para se formular as perguntas."

JOSÉ DE SOUSA SARAMAGO

Na sociedade de informação global, a privacidade está seriamente comprometida, pois o tráfego gerado em uma rede global transcende os limites territoriais, e não é gerenciado de modo centralizado. Com isso, surgem os riscos de invasão de privacidade, uma vez que os dados dos usuários podem ser interceptados ou monitorados. A padronização de ações consideradas intrusivas é caracterizada na forma de leis, que são distintas a uma determinada região ou país. Em situações que causem violação de tais leis, penas são aplicadas.

A privacidade tornou-se um aspecto importante na modelagem de sistemas computacionais que exigem o estabelecimento de relacionamentos 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.

Mecanismos de segurança são utilizados para garantir os requisitos de segurança de um determinado sistema. Entretanto, ao mesmo tempo que exercem um controle sobre os dados de usuários que trafegam no sistema, podem invadir a privacidade de tais usuários (FISCHER-HÜBNER et al., 1992). Assim, embora a segurança das informações em um sistema ser essencial, também é importante monitorar as possíveis ameaças à privacidade.

Modelos de controle de acesso estabelecem interações e relacionamentos entre indivíduos e objetos apropriados às regras definidas para esse 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 (LAMPSON, 1971).

Alguns modelos de controle de acesso tratam dos requisitos de privacidade. Entretanto, o conceito de privacidade é muito difícil de ser definido devido à sua subjetividade e amplitude. A solução adotada foi a utilização de especificações genéricas para privacidade como núcleo desses modelos. Uma deficiência desta abordagem reside no fato de que alguns detalhes e especificações de políticas de privacidade podem tornar-se ambígüos; a solução para esse problema foi a definição de um "nível de força" para cada detalhe a ser considerado, de modo que um detalhe mais forte seja escolhido quando comparado com um mais fraco. Além disso, a representação de um modelo de controle de acesso deve possibilitar mudanças no modelo, se necessário. Entretanto, alguns modelos na literatura têm aplicação apenas para cenários específicos (DAFA-ALLA et al., 2005) (JOSHI et al., 2001b) (JOSHI et al., 2001a) (OSBORN et al., 2000) (RAY et al., 2006) (SANDHU et al., 1996).

Diante desse panorama, este trabalho propõe um modelo de controle de acesso simples e genérico o suficiente para contemplar os requisitos de segurança e privacidade de um sistema, chamado *Modelo de Controle de Acesso Adaptativo* (MCAA). O MCAA utiliza um autômato adaptativo para realizar o controle de acesso, e possui *comandos de privacidade* para codificar um conjunto de regras obtidas a partir

de documentos de políticas de privacidade ou legislações, escritas em linguagem natural.

Os autômatos adaptativos constituem uma particular classe de dispositivos adaptativos (NETO, 2001). O termo "adaptativo", neste contexto, pode ser definido como a capacidade de um dispositivo em alterar seu próprio comportamento. Assim, um autômato adaptativo pode alterar sua topologia durante o processo de reconhecimento de uma cadeia de entrada.

De acordo com NETO (2007), a generalidade resultante do modelo, sua capacidade de aprendizado devida à característica de auto-modificação, bem como o seu poder de expressão faz dos autômatos adaptativos um dispositivo muito atraente para expressar fatos complexos e manipular situações difíceis que surgem durante uma busca por soluções computacionais para problemas complexos. Algumas das áreas de aplicação incluem robótica, aprendizado de máquina, processamento de linguagem natural, compiladores, controle de privacidade, entre outros.

A característica de auto-modificação, poder computacional e simplicidade do autômato adaptativo tornam-no adequado para a representação de um modelo de controle de acesso.

O modelo proposto neste trabalho oferece uma representação eficiente e compacta para o controle de acesso e permite a utilização de mecanismos de auditoria. Além disso, é possível expandir sua utilização além da privacidade, pois o modelo é genérico o suficiente para permitir outras classes de problemas que requerem tratamento das informações.

1.1 Objetivos

O objetivo deste trabalho é contribuir com a área de Privacidade, apresentando um modelo de controle de acesso que reforce os aspectos de segurança e privacidade de um sistema. O trabalho evidencia também a contribuição para a área de Tecnologias Adaptativas de um novo domínio de aplicabilidade.

1.2 Motivação

A motivação deste trabalho deve-se ao fato de que a privacidade de um indivíduo é um aspecto primordial a ser observado nos ambientes computacionais. Um modelo de controle de acesso com reforço em privacidade se faz adequado para tal propósito, de modo que possa servir como referência para outras abordagens e implementações.

1.3 Organização do trabalho

Este trabalho está organizado da seguinte forma:

No Capítulo 2, são apresentados os conceitos de privacidade, os mecanismos utilizados para proteção na Web e a legislação sobre privacidade. No Capítulo 3, são apresentadas as três classes principais de políticas de controle de acesso e quatro modelos de controle de acesso existentes na literatura. No Capítulo 4, é apresentada uma breve descrição sobre os autômatos finitos, autômatos de pilha e autômatos de pilha estruturados; a seguir, é apresentada a definição dos autômatos adaptativos. O modelo de controle de acesso baseado em autômatos adaptativos, chamado *Modelo de Controle de Acesso Adaptativo* (MCAA), proposto neste trabalho, é apresentado no Capítulo 5. No Capítulo 6, é apresentada uma avaliação do MCAA através de quatro análises realizadas. As conclusões são apresentadas no Capítulo 7.

CAPÍTULO

2

Privacidade

"We've become bored with watching actors give us phony emotions. We're tired of pyrotechnics and special effects. While the world he inhabits is, in some respects, conterfeit, there's nothing fake about Truman himself – no scripts, no cue cards. It isn't always Shakespeare, but it's genuine. It's a life."

CHRISTOF (Ed Harris), na cena inicial do filme *The Truman Show* (1998).

A privacidade sempre foi uma questão social e legal para advogados, cientistas políticos, filósofos e outros estudiosos do assunto. Com a chegada dos sistemas computacionais e, de modo especial, das redes de comunicação (como a Internet), a privacidade do indivíduo tornou-se mais exposta e, por conseqüinte, mais propensa

a ser violada (FISCHER-HÜBNER, 2001).

Neste Capítulo, são apresentados os conceitos de privacidade de uma forma geral, a legislação existente e os mecanismos disponíveis para proteção de privacidade nas redes de comunicação.

2.1 Conceitos iniciais

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.

O conceito de privacidade contempla três aspectos importantes definidos a seguir (ROSENBERG, 1992) (FISCHER-HÜBNER, 2001).

- **Privacidade territorial**: proteção das áreas físicas que rodeiam uma pessoa, por exemplo, locais domésticos ou de trabalho.
- **Privacidade da pessoa**: proteção de uma pessoa contra interferências excessivas ou ilegais, por exemplo, testes com drogas, buscas físicas ou violação de seu senso moral.
- **Privacidade da informação**: controle de como as informações pessoais de um indivíduo podem ser coletadas, armazenadas, processadas ou disseminadas.

No contexto das redes de comunicação, a privacidade pode ser caracterizada como a capacidade de um usuário de manter controle das informações pessoais (FRIEDMAN et al., 2000). FERNANDES (2003) 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 navegação do usuário por tais redes (como a Internet), essas informações devem ser divulgadas apenas com o consentimento do mesmo.

Em uma pesquisa realizada por TELTZROW; KOBSA (2004), a preocupação dos usuários com relação à privacidade dos dados ficou evidente quando 64% dos

usuários relataram que deixaram de acessar algum site ou mesmo de fazer compras online por não terem conhecimento de como suas informações seriam utilizadas.

A coleta de dados nas redes de comunicação, em sua maioria, passa de modo não observado pelos usuários, acarretando riscos à privacidade. É comum a existência de grandes bancos de dados contendo todo o tipo de informação coletada acerca dos usuários; o comércio destes bancos de dados é praticado freqüentemente, devido à demanda crescente por esse tipo de informação (GOLDBERG et al., 1997).

Com o advento e consolidação do comércio eletrônico (*e-commerce*) na Internet, as questões relacionadas à privacidade tornaram-se mais importantes. É praticamente impossível completar uma transação sem revelar quaisquer dados de caráter pessoal – endereço de entrega, informações de pagamento, documentos ou mesmo preferências de produtos. Assim, os usuários podem não desejar informar esses dados caso julguem que sua privacidade está sendo invadida ou posta em risco (ACKERMAN; CRANOR, 1999), uma vez que a existência de um usuário no ambiente virtual é mantida única e exclusivamente por suas informações (GRANDE, 2006).

O conceito de privacidade pode divergir de modo significativo entre indivíduos, mesmo estes estando inseridos em ambientes sócio-culturais semelhantes. Por exemplo, um usuário pode sentir-se inseguro em relação à sua privacidade ao navegar em um determinado site, enquanto outro usuário, no mesmo site, pode não compartilhar das mesmas opiniões. As interpretações podem ser as mais diversas possíveis.

As ações do usuário, ao navegar pela Internet, podem ser monitoradas pelo site ao qual o usuário está visitando no momento, registradas e preservadas para um acesso futuro. Na maioria dos casos, os sites que comprometem a privacidade do usuário não têm interesse em avisá-lo sobre a coleta.

A interligação de sistemas computacionais também apresenta uma preocupação importante com a privacidade das informações que trafegam pelas redes. O

compartilhamento de informações pessoais dos usuários entre tais sistemas pode acarretar uma exposição não desejada e conseqüente violação de privacidade. Por exemplo, um sistema hospitalar pode enviar os dados de um determinado usuário à empresa de plano de saúde para efetuar a cobrança; o envio de dados adicionais que não dizem respeito à cobrança acabam por expôr o paciente.

2.2 Mecanismos de privacidade

Em geral, a privacidade pode ser protegida através de leis governamentais, regulamentações promovidas por organizações, tecnologias de melhoria da privacidade ou através uma conscientização da importância da privacidade aos indivíduos e administradores de sistema (FISCHER-HÜBNER, 2001).

Os sistemas computacionais possuem uma quantidade relevante de informações pessoais que trafegam por uma infra-estrutura de rede. A segurança dessas informações é usualmente tratada através de controles de acesso, que limitam o acesso a determinadas informações. Entretanto, as políticas definidas para segurança dos dados muitas vezes não contemplam os aspectos de privacidade necessários para evitar falhas de exposição de informações sigilosas. O Capítulo 3 apresenta alguns destes controles de acesso e suas características principais.

Em relação às redes de comunicação, como a Internet, 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 (SHUBINA; SMITH, 2003) (CHAUM, 1981) (GOLDSCHLAG et al., 1996) (REITER; RUBIN, 1997) (GOLDBERG et al., 1997) (ISHITANI, 2003). Outras, porém, baseiam-se em métodos de policiamento dos sites ou de aviso ao usuário sobre as políticas de privacidade adotadas (GRANDE, 2006) (COYLE, 1999) (ACKERMAN; CRANOR, 1999). Nesta seção, alguns destes mecanismos serão apresentados.

2.2.1 Anonimato

O *anonimato* pode ser caracterizado como a situação em que um indivíduo não é identificável entre um conjunto de indivíduos (PFITZMANN; KÖHNTOPP, 2001). Na Web, o anonimato permite ao usuário navegar por um determinado site sem que seja identificado. A privacidade é garantida devido à ocultação do número IP (*Internet Protocol*) de seu computador e de outras informações presentes nos cabeçalhos dos protocolos.

Temendo ter sua privacidade invadida, o usuário pode beneficiar-se de sites que oferecem serviços de navegação anônima, como *anonymizer.com*[1], *the-cloak.com*[2] ou *anonymouse.org*[3] (SHUBINA; SMITH, 2003).

O anonimato ocorre por intermédio de um servidor chamado *proxy*, que mascara todas as requisições do usuário como se fossem suas, através de alterações nas informações presentes nos cabeçalhos dos protocolos.

Apesar de esta solução ser eficaz na maioria dos casos, existem sistemas mais complexos que garantem maior segurança sobre as questões de privacidade do usuário.

Os mecanismos de anonimato podem ser divididos em duas vertentes: os *proxies* de anonimato de *um único nó* e os *de vários nós* (SHUBINA; SMITH, 2003) (CHAUM, 1981) (REITER; RUBIN, 1997).

Proxy de anonimato de um único nó

O funcionamento do *proxy de anonimato de um único nó* é simples: o usuário faz a requisição da URL ao *proxy*, que a processa e envia uma requisição ao site-alvo. O site-alvo, por sua vez, processa essa requisição, devolve ao *proxy*, e este por fim a encaminha ao usuário.

A maioria dos sites que oferecem esse tipo de serviço também disponibiliza re-

[1] *http://www.anonymizer.com*
[2] *http://www.the-cloak.com*
[3] *http://www.anonymouse.org*

cursos adicionais, como filtragem de *cookies*[4], reescrita de códigos *JavaScript*, bloqueio de pacotes HTTP, entre outros (GOLDBERG et al., 1997). Algumas opções, entretanto, são exclusivas a usuários registrados ou que tenham comprado o serviço.

A Figura 2.1 ilustra o funcionamento de um *proxy* de anonimato de um único nó.

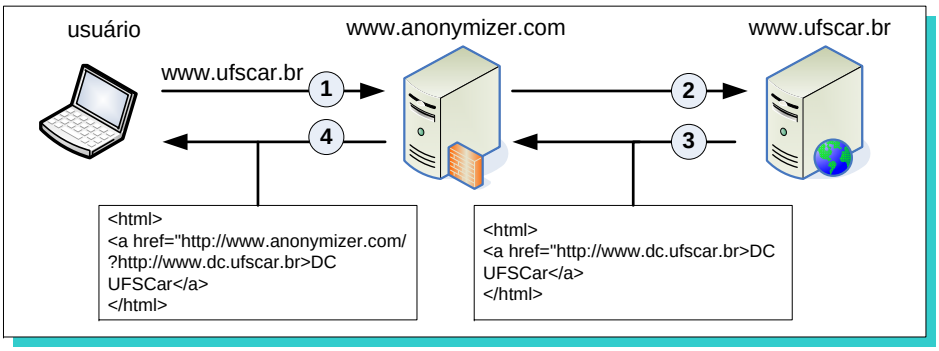

Figura 2.1: Exemplo de comunicação entre o usuário e o site da Web, através de um *proxy* de anonimato de um único nó (SHUBINA; SMITH, 2003).

De acordo com a Figura 2.1, o usuário faz uma requisição do site *www.ufscar.br*, utilizando o *proxy* de anonimato *anonymizer.com* (1); O *proxy* encaminha a requisição ao site-alvo, como se fosse sua própria requisição (2); O site *www.ufscar.br* processa a requisição e a retorna para o *proxy* (3); Os links da página retornada são reescritos, para que o usuário navegue pelo site anonimamente, e a página é, por fim, retornada (4).

Apesar de o usuário ser anônimo ao conectar-se com o site, ele não é anônimo para o servidor *proxy*. Dessa forma, o usuário precisa confiar no servidor *proxy*, além da presença de mecanismos de segurança neste último, para evitar acesso não autorizado às informações do usuário. Outra desvantagem diz respeito à ausência de personalização oferecida pelos sites, uma vez que o usuário é anônimo e não compartilha informações. Além disso, se o *proxy* falhar, não é possível conti-

[4] *Cookie* é um grupo de dados trocados entre o usuário e o servidor, armazenados em um arquivo texto criado no computador do usuário. As informações armazenadas nos *cookies* podem refletir algo sobre o perfil do usuário.

nuar a navegação.

Os *proxies* de anonimato de um único nó podem ter pessoas mal intencionadas observando o tráfego da rede, tentando estabelecer alguns padrões para definir perfis de acesso, comprometendo a privacidade dos usuários.

Proxy de anonimato de vários nós

Para aumentar a segurança da navegação, CHAUM (1981) definiu um conceito chamado *MixNets* (ou Redes *Mix*). *MixNets* são grupos de *proxies* que promovem anonimato, conduzindo o tráfego do usuário através dos vários nós da rede. Esses nós, também chamados *Mixes*, realizam operações para atrasar, reordenar, criptografar, adicionar dados e encaminhar o tráfego. Através dessas operações realizadas, o caminho traçado faz com que a origem da mensagem seja desconhecida.

Vários mecanismos utilizam-se deste conceito, dentre os quais os mais relevantes são *Onion Routing* e *Crowds*, apresentados a seguir.

- **Onion Routing**: O objetivo do *Onion Routing* é proteger a privacidade do emissor e receptor de uma mensagem, além de prover proteção a essa mensagem, enquanto trafega pela rede. Esse mecanismo é baseado na *MixNet*, na qual cada *Mix* (também chamado de *onion router*, ou roteador *onion*) é responsável por fazer o roteamento e ocultar o caminho anterior de uma mensagem através da rede. Uma estrutura de dados em camadas é criada, chamada *onion*, e os algoritmos e chaves de criptografia utilizados durante o envio dos dados são determinados. Seu funcionamento é ilustrado na Figura 2.2.

  O usuário faz uma requisição de um site ao *proxy*, que é encaminhada à *MixNet* (1); a mensagem é roteada entre os nós da rede (2); O site-alvo recebe a requisição (3).

  Antes de a informação ser transmitida, diversas camadas de proteção são encapsuladas a ela, gerando a mensagem. O iniciador (primeiro *proxy* da rede), ao receber a requisição, cria o *onion* e define a rota até o destino. Os dados

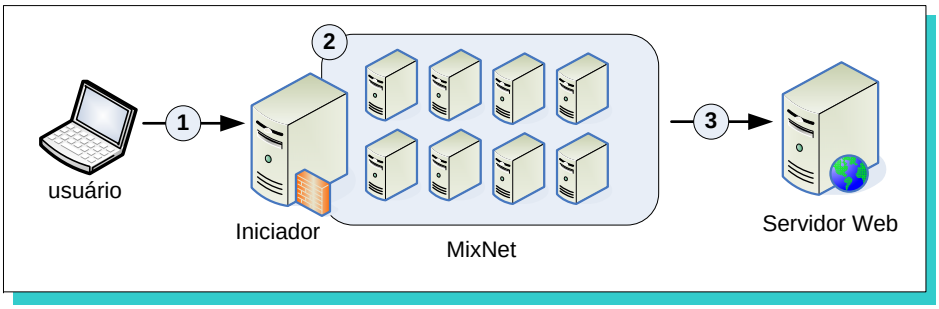

Figura 2.2: Formação de uma *MixNet* (GOLDSCHLAG et al., 1996).

são pré-criptografados repetidamente através de algoritmos de criptografia e das chaves especificadas no *onion*, indicando que, quando a mensagem chegar a um roteador da rede, este deve retirar uma camada do *onion*. A cada roteador *onion*, uma camada é retirada; dessa forma, ao chegar no destinatário, a mensagem está em seu estado original, contendo o número IP do último roteador do caminho (Figura 2.3).

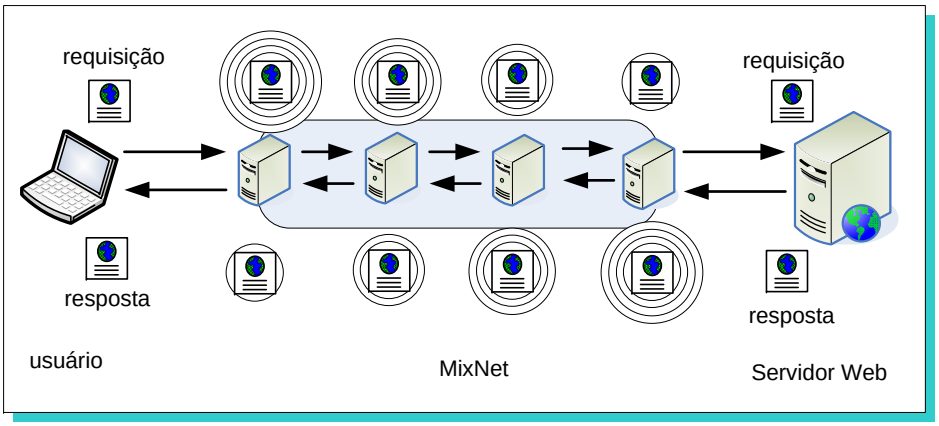

Figura 2.3: Exemplo de rota de comunicação em uma rede utilizando o mecanismo de *Onion Routing* (GOLDSCHLAG et al., 1996).

No caso de retorno da mensagem, o processo é o mesmo, mas utilizando diferentes algoritmos e chaves. Pessoas mal intencionadas não conseguirão estabelecer padrões, pois as mensagens terão chegadas aleatórias e estarão criptografadas.

- **Crowds**: O método *Crowds* (multidões), apresentado por REITER; RUBIN

(1997), também possibilita esconder a identidade do usuário enquanto ele navega pela Web. A idéia por trás deste método é fazer de cada usuário um colaborador para manter o anonimato do grupo. Para isso, o usuário deve pertencer a uma *crowd* (multidão), sendo representado por um processo no computador chamado *jondo*, uma palavra oriunda da expressão inglesa "John Doe", que significa uma pessoa que quer manter seu anonimato. Quando um usuário inicia o processo *jondo*, ele entra em contato com um servidor chamado *blender* (misturador) para requisitar sua admissão em uma *crowd*.

O *blender* exige que o usuário estabeleça uma conta com ele, criando um nome e senha. Essas informações são armazenadas no *blender* e utilizadas para autenticar a comunicação com os *jondos*. Após a autenticação e comunicação, o *blender* gera uma lista de chaves comuns e a envia aos *jondos*. Essas chaves poderão ser usadas para realizar a autenticação de outro membro da *crowd*. Uma vez aceito na *crowd*, o *blender* informa ao *jondo* a configuração atual da *crowd* e o anúncio para unir-se a ela, atualizando sua lista de membros.

A lista de membros é iniciada quando um *jondo* se junta à *crowd*, e é atualizada quando o *blender* o informar sobre membros novos ou excluídos. Um *jondo* pode remover um membro de sua lista, caso detecte sua falha.

De acordo com REITER; RUBIN (1997), o usuário escolhe um *jondo* para ser seu *proxy* Web, e especifica o nome do *host* e número de porta em seu navegador Web. Desta forma, toda requisição será enviada para esse jondo.

REITER; RUBIN (1997) afirmam ainda que o *jondo*, ao receber sua primeira requisição, inicia a definição de uma rota aleatória de *jondos* que cuidará das requisições entre o usuário e o servidor Web. No instante em que o jondo recebe a requisição, ele gera um número aleatório que indicará a probabilidade de encaminhamento da requisição para outro *jondo* (incluindo ele mesmo). Dependendo do resultado, o *jondo* poderá encaminhar a requisição para ou-

tro, ou submeter a requisição diretamente ao servidor Web. A Figura 2.4 ilustra a comunicação em uma rede Crowds.

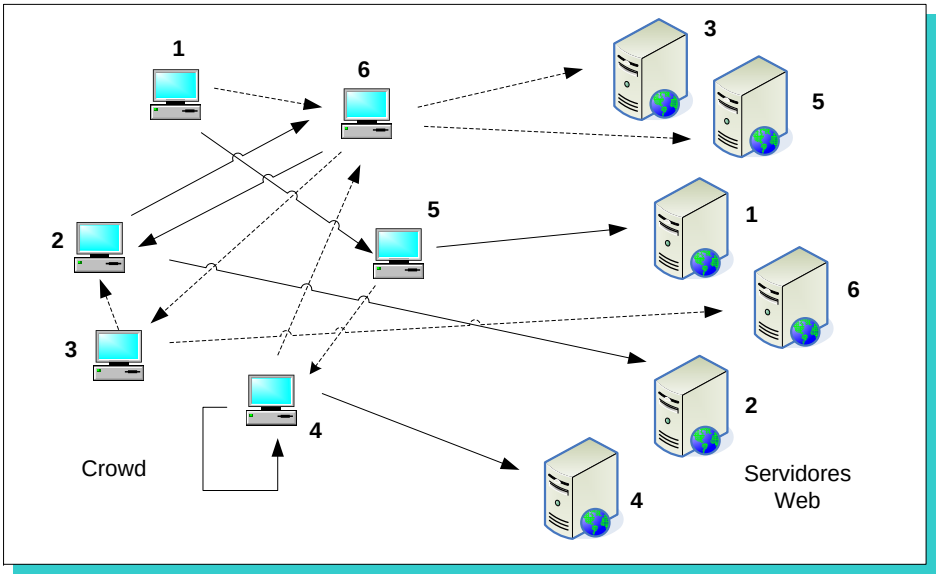

Figura 2.4: Comunicação em uma rede Crowds (REITER; RUBIN, 1997).

Alguns caminhos possíveis, de acordo com a Figura 2.4: 1, 5, servidor 1; 2. 6, 2, servidor 2; 5, 4, 6, servidor 3; 4, 4, servidor 4; e muitos outros.

Uma característica interessante do método *Crowds* é que em uma ausência do *blender*, apesar de não ser possível a admissão de novos *jondos*, a rede continuará funcionando normalmente, uma vez que o caminho das requisições é determinado pelos *jondos*.

Crowds diferencia-se do *Onion Routing* em relação às rotas; no primeiro, as rotas são definidas aleatoriamente, adaptando-se às mudanças da rede, enquanto no segundo, o *proxy* iniciador determina as rotas antes do envio das mensagens (REITER; RUBIN, 1997).

2.2.2 Pseudônimos

O método dos *pseudônimos*, como o próprio nome sugere, 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.

De acordo com GOLDBERG et al. (1997), o anonimato na Web possui dois aspectos: o anonimato de conteúdo de dados, no qual a identidade do usuário não é revelada através do conteúdo do fluxo de dados entre usuários e sites, e o anonimato de conexão, no qual o usuário não pode ser identificado por terceiros que estejam analisando a conexão.

O mecanismo a ser comentado chama-se *Janus Personalized Web Analyser* (GABBER et al., 1997), pelo fato de ser o primeiro sistema existente a oferecer aos usuários o anonimato com personalização. Atualmente, é conhecido por *Lucent Personalized Web Anonymizer* (GABBER et al., 1999).

O *Janus* atua como um servidor *proxy*, realizando a comunicação entre o usuário e o site, sendo também responsável pela geração dos apelidos (pseudônimos), o qual garante o anonimato de conteúdo de dados.

Para o usuário utilizar o sistema, é necessário iniciar uma sessão de navegação no *Janus*, fornecendo um endereço de e-mail e uma senha. De posse destes dados, o *Janus* gera todos os apelidos usados durante aquela sessão.

O *Janus* permite troca de e-mail anônimo entre usuário e site; os chamados "apelidos para endereços de e-mail" possuem o formato *apelido-email@hostname*, onde *hostname* é uma máquina intermediária e *apelido-email* é uma cadeia de caracteres que permite obter o endereço real do usuário (GABBER et al., 1999).

O funcionamento do sistema *Janus* pode ser visto na Figura 2.5.

No primeiro acesso utilizando o *proxy Janus*, será necessário informar os dados para se registrar no sistema. O próprio *Janus* se encarrega de fornecer uma tela de autenticação. No exemplo da Figura 2.5, *Alice* e *Bob* informaram seus dados, *[id1,S1]* e *[id2,S2]*, respectivamente, aos *proxies*. Uma vez identificados no sistema, os usuários podem navegar normalmente. *Janus* gera um pseudônimo de acordo com o nome de usuário, senha e o domínio do site, no caso do exemplo, o usuário *Bob* utilizará *[u1,p1]* para o site 2 e *[u2,p2]* para o site 1; o usuário *Alice* utilizará

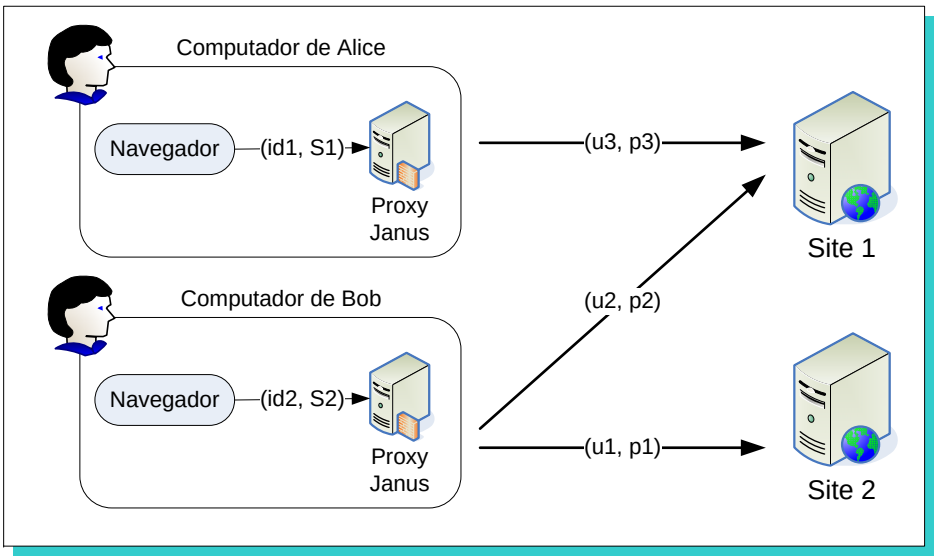

Figura 2.5: Exemplo de funcionamento do sistema *Janus*, com criação e autenticação de contas de usuário (GABBER et al., 1999).

[u3,p3] para o site 1.

O *proxy Janus* está situado no computador do usuário, logo, se a rede permitir rastreamento de conexões, o usuário poderá ser identificado, e *Janus* perderá sua utilidade. Desta forma, sugere-se que a comunicação se dê sobre uma rede anônima (GOLDBERG et al., 1997).

A Figura 2.6 ilustra uma nova configuração do *proxy*, desta vez situado fora do computador do usuário.

Os usuários estão inseridos em uma rede privada, onde as conexões passam pelo *proxy* e, em seguida, por um *firewall*. É necessário garantir a segurança da rede, pois uma intervenção ou observação alheia poderia comprometer a identidade de todos os usuários presentes na rede.

2.2.3 P3P

A *P3P*, Plataforma para Preferências de Privacidade (*Plataform for Privacy Preferences*) foi desenvolvida pelo Consórcio da *World Wide Web*[5] com o objetivo de permitir aos sites Web uma padronização da apresentação das práticas de coleta, de

[5] *http://www.w3.org*

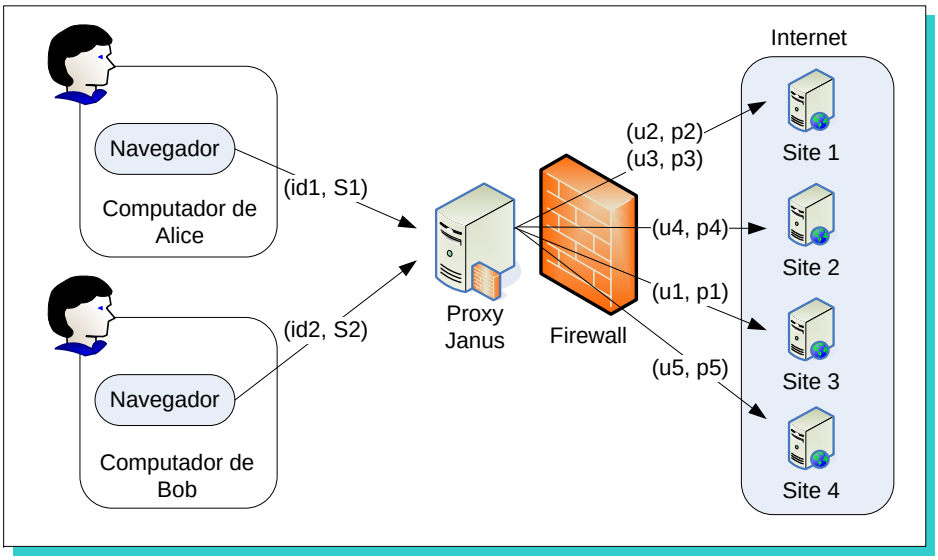

Figura 2.6: Sistema *Janus* como um *proxy* da Intranet (GABBER et al., 1999).

modo que o usuário tome conhecimento das informações coletadas. A P3P permite que site e usuário negociem quais informações serão coletadas, bem como sua utilização e mecanismo de coleta (COYLE, 1999).

A P3P introduz preferências de privacidade ao usuário, podendo este configurá-las como achar melhor. Desta forma, a análise das políticas de um site será feita baseada nessas preferências, minimizando o problema da subjetividade do conceito de privacidade.

Os sites podem utilizar-se do protocolo da P3P, em formato XML[6], para publicar sua(s) política(s) de privacidade. Através de um agente de usuário, durante a visita a um site, a política de privacidade, no formato definido pela P3P, é recuperada e comparada com as preferências de privacidade, pré-estabelecidas pelo usuário. Se a política está de acordo com as preferências, a navegação prossegue normalmente, caso contrário, o usuário será questionado (CRANOR et al., 2002). A Figura 2.7 ilustra a interação com o usuário, ao acessar o site *Yahoo!*[7].

Alguns sites já oferecem sua(s) política(s) de privacidade no padrão da P3P, oferecendo mais liberdade ao usuário para preocupar-se apenas com seus objetivos

[6] *Extensible Markup Language*. Disponível em *http://www.w3.org/XML*
[7] *http://www.yahoo.com*

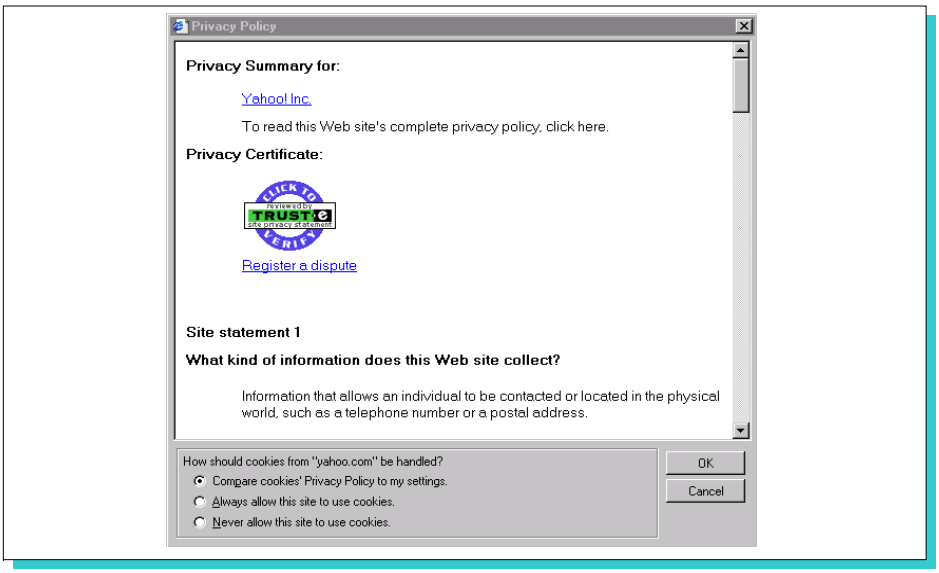

Figura 2.7: Exemplo de interação da P3P com o usuário, negociando a política de privacidade do site *Yahoo!*.

de visita, minimizando a necessidade de uma leitura intensa dessa(s) política(s). Além disso, muitos navegadores já oferecem mecanismos de leitura das políticas de privacidade que estejam no formato da P3P (CRANOR et al., 2002).

2.2.4 Agentes de privacidade

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 (por exemplo, uma busca por campos onde o usuário deveria inserir o número de algum documento) ou mesmo uma leitura da política de privacidade, desde que esteja em um formato pré-determinado, legível ao agente em questão.

Existe um agente proposto por ACKERMAN; CRANOR (1999) chamado *Privacy Critics*, que tem como objetivo auxiliar os usuários a resguardar suas informações, através de sugestões e *feedbacks*. Entretanto, não age sem o consentimento do usuário. Sua finalidade é alertar o usuário, para que este tenha um controle maior

sobre suas informações pessoais.

2.2.5 MASKS

O mecanismo *MASKS* (*Managing Anonymity while Sharing Knowledge to Servers*) visa garantir a privacidade do anonimato do usuário sem deixar de permitir a personalização. Essa arquitetura foi proposta por ISHITANI (2003) e fornece a privacidade aos usuários através de máscaras ou pseudônimos. Essas máscaras são identificações temporárias que um usuário pode adotar durante a interação com um determinado site, sem ser identificado.

O mecanismo MASKS possui as características descritas a seguir (ISHITANI, 2003).

- **Proteção de privacidade**: a utilização das máscaras permite o anonimato do usuário.
- **Compatibilidade parcial com o processo de personalização**: dados são disponibilizados aos sites para que possam ser utilizados em serviços de personalização, não sendo possível, entretanto, criar um perfil individualizado de cada usuário.
- **Segurança**: quanto maior a quantidade de informações, maior é a possibilidade de um ataque. Para minimizar o risco, o MASKS baseia seu processamento na última requisição de cada usuário.
- **Flexibilidade**: os serviços do MASKS adaptam-se dinamicamente a mudanças de comportamento do usuário.
- **Eficiência**: a latência percebida pelos usuários não deve ser maior que a existente sem o uso do MASKS.
- **Interoperabilidade e facilidade de implantação**: o MASKS utiliza os protocolos HTTP e TCP e trabalha com mecanismos usuais de identificação, como os *cookies*.

- **Facilidade de uso**: os usuários não necessitam fornecer informações prévias ao MASKS.

A arquitetura do MASKS possui dois componentes principais: o agente de privacidade e segurança (PSA – *Privacy and Security Agent*) e o servidor de máscaras (*Masks Server*).

O primeiro componente, o PSA, atua em conjunto com o navegador, sendo um intermediário entre os usuários e o *Masks Server*. Suas funcionalidades são: cifrar as requisições, manter o usuário informado sobre as máscaras atribuídas, permitir uma interação direta com o site (desligando o processo de mascaramento) e filtrar métodos conhecidos de invasão de privacidade.

O segundo componente é o *Masks Server*, um *proxy* de anonimato entre os usuários e o site. Ele é responsável pelo gerenciamento de máscaras e atribuição destas aos usuários. A atribuição de máscaras baseia-se no conceito de grupo, que representa um tópico de interesse. Cada requisição do usuário é associada a um grupo, de acordo com a semântica da requisição. Assim, por trás das requisições haverá grupos, e não mais indivíduos, permitindo a divulgação dos dados sobre o interesse comum dos usuários. Esses dados poderão ser utilizados para oferecer serviços personalizados e, ao mesmo tempo, a privacidade do usuário é preservada.

O *Masks Server* possui ainda dois componentes: O *Seletor*, que é responsável pela seleção do grupo de interesse de cada requisição do usuário, e o *Gerenciador de Máscaras*, que dado um grupo, deverá determinar a máscara correta para o usuário.

A arquitetura do MASKS, bem como o tratamento das requisições dos usuários, podem ser visualizadas na Figura 2.8.

Como pôde ser observado na Figura 2.8, um mesmo site pode receber requisições de grupos diferentes: as requisições A1 e A2, feitas por Alice, recebem grupos diferentes e, no entanto, acessam o mesmo site. Caso o usuário opte por não utilizar o anonimato, ele poderá desligá-lo (através do PSA), e então conectar-se

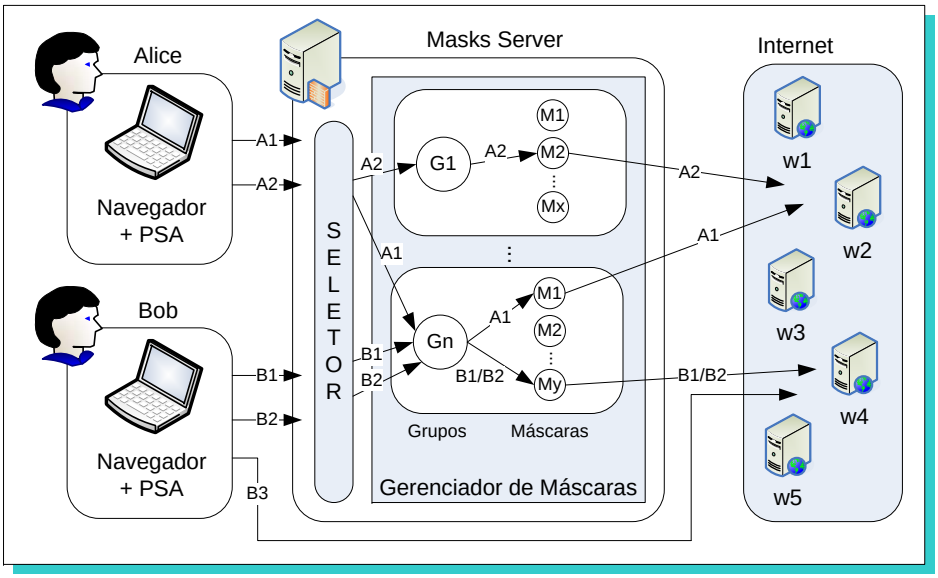

Figura 2.8: Arquitetura do MASKS e tratamento das requisições dos usuários (ISHITANI, 2003).

diretamente ao site. É o caso da requisição B3, feita por Bob.

As respostas dos sites seguem o caminho inverso, como pode ser visualizado na Figura 2.9

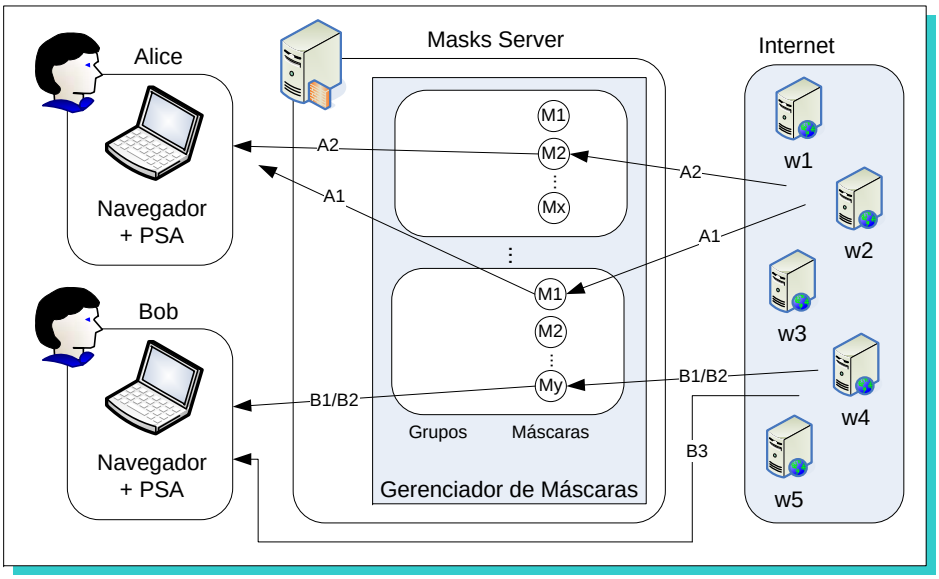

Figura 2.9: Tratamento das respostas dos sites no MASKS (ISHITANI, 2003).

A parte mais importante da arquitetura do MASKS é o algoritmo de seleção do

grupo, pois é através dele que um determinado grupo é selecionado, baseado na requisição do usuário.

Os *cookies* são aceitos pelo MASKS e utilizados no processo de mascaramento das requisições e, a partir disso, permitem serviços personalizados aos grupos.

O *Masks Server* representa o mecanismo principal do MASKS e, portanto, deve utilizar-se de métodos de segurança de comunicação para evitar riscos à privacidade dos usuários.

2.3 Camadas de proteção de privacidade

LOBATO; ZORZO (2006) apresentaram uma taxonomia para a proteção da privacidade, utilizando o conceito de *camadas de privacidade*. As camadas foram propostas de modo que haja um relacionamento entre elas, de modo que a existência de uma camada faz valer a existência de outra. As camadas representam funcionalidades próximas ao hardware até os benefícios oferecidos ao usuário, como pode ser visto na Figura 2.10.

Figura 2.10: Arquitetura para camadas de privacidade (LOBATO; ZORZO, 2006).

As descrições das camadas, partindo do usuário, são apresentadas a seguir.

- **Camada 1 – Leis de proteção de privacidade**: refere-se à legislação acerca da privacidade do usuário. Se existirem leis que regulem a privacidade, essa

camada será aplicada às demais, fornecendo maior segurança ao usuário.

- **Camada 2 – Políticas de privacidade**: a política de privacidade, também chamada de declaração de privacidade, é um documento que descreve as práticas de coleta que um site realiza e as questões relacionadas à segurança no armazenamento e o compartilhamento das informações coletadas. Essa camada é relacionada às políticas de privacidade que um site adota e à negociação das mesmas com o usuário. Como mecanismo desta camada pode-se mencionar a P3P.

- **Camada 3 – Certificação de privacidade**: essa camada diz respeito ao cumprimento das políticas de privacidade divulgadas por um determinado site. Essa verificação deve ser realizada por empresas de auditoria e grupos de privacidade idôneos. É necessário utilizar a camada 2 em conjunto, tendo em vista que a certificação é a garantia do cumprimento das práticas de coleta descritas na política de um site.

- **Camada 4 – Notificação**: essa camada tem por finalidade informar o usuário sobre riscos de invasão de privacidade. Por exemplo, o usuário poderia ser informado a respeito de um determinado *cookie* que um site deseja utilizar, inclusive as vantagens e desvantagens de aceitá-lo. Muitos navegadores informam o usuário sobre alguns riscos, por exemplo, o recebimento de *cookies* (Figura 2.11).

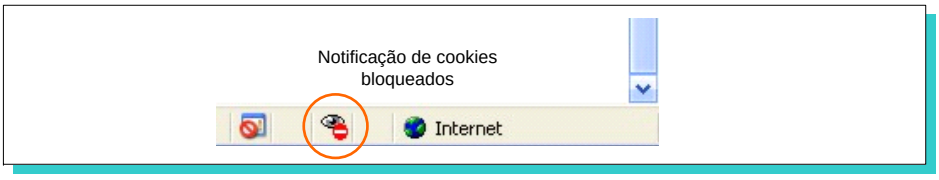

Figura 2.11: Notificação de riscos de navegação feita pelo navegador.

Entretanto, a maioria dos navegadores requer que o usuário informe explicitamente o que deseja ou não aceitar.

- **Camada 5 – Controle**: essa camada fornece mecanismos ao usuário, de modo que ele tenha controle sobre as informações, melhorando o nível de privacidade. Vários mecanismos usuais de coleta podem ser bloqueados, como *cookies* e *web bugs*[8]. Entretanto, é importante ter as ferramentas de notificação (citadas na camada anterior) atuando conjuntamente.

  A Figura 2.12 mostra um aviso de recebimento de um *cookie de terceiros*[9], ao tentar acessar o site *Yahoo! Brasil*[10], permitindo ao usuário aceitá-lo ou não. É possível notar a presença de uma notificação ao usuário, informando os possíveis riscos relacionados à aceitação.

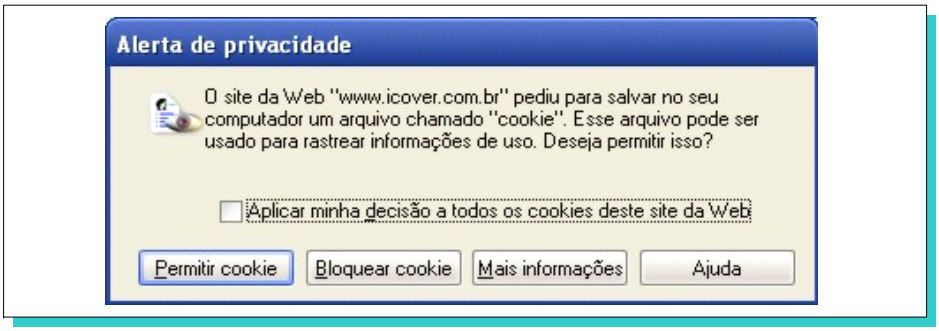

Figura 2.12: Aviso de recebimento de um *cookie de terceiros*.

  Os navegadores possuem muitas funcionalidades, como exclusão do histórico de páginas acessadas, bloqueio de *pop-ups*[11], bloqueio de *cookies* e muitos outros mecanismos, disponíveis através de *plugins*[12].

- **Camada 6 – Mecanismos para proteção de privacidade**: nessa camada estão incluídos os mecanismos e ferramentas disponíveis para proteção de privacidade. A diferença desta camada em relação à camada anterior está no

[8] *Web bugs* são mecanismos que tentam obter algum tipo de identificação de um usuário. Geralmente, estão inseridos em mensagens de e-mail ou páginas da Web.

[9] Um *cookie de terceiros* é um *cookie* associado a uma imagem, anúncio, *frame* ou qualquer outro conteúdo proveniente de outro domínio, que esteja incorporado ao site que o requisitou.

[10] *http://br.yahoo.com*

[11] *Pop-ups* são pequenas janelas que surgem em uma página visitada para informar algo ou exibir um anúncio.

[12] *Plugins* são programas auxiliares inseridos nos navegadores para desempenharem tarefas específicas.

local em que o mecanismo se encontra: na camada anterior, os mecanismos se encontram no computador do usuário, enquanto que nessa camada eles estão disponíveis em algum local da Web. A camada 6 pode ser utilizada em conjunto com as camadas 4 e 5 (Notificação e Controle, respectivamente).

Alguns desses mecanismos já foram citados anteriormente, como os *proxies* de anonimato (*Onion Routing*, *Crowds*, entre outros), Pseudônimos e o MASKS (Seção 2.2).

2.4 Legislação sobre privacidade

A Web não possui uma legislação específica. Dessa forma, cada país procura aplicar suas leis referentes à privacidade de seus cidadãos aos usuários da Internet. Em sistemas computacionais interligados, é necessário que exista uma política (ou um conjunto de políticas), dentro do domínio da aplicação, que descreva qual será o comportamento desses sistemas e seu impacto na privacidade dos usuários.

A regulamentação de proteção de privacidade sempre é colocada em pauta nas discussões de leis ao redor do mundo. A *Electronic Privacy Information Center* (EPIC), um centro americano de pesquisa de interesse público, juntamente com a *Privacy International*[13], uma organização não-governamental que trabalha com direitos humanos, elaboram conjuntamente um relatório anual sobre as legislações e os avanços sobre proteção de privacidade no mundo inteiro.

A legislação brasileira apresenta algumas determinações a respeito da privacidade. A Constituição, no artigo 5.º, incisos X e XII, cita (BRASIL, 1988):

> "X - são invioláveis a intimidade, a vida privada, a honra e a imagem das pessoas, assegurado o direito a indenização pelo dano material ou moral decorrente de sua violação; XII - é inviolável o sigilo da correspondência e das comunicações telegráficas, de dados e das comunicações telefônicas, salvo, no último caso, por ordem judicial, nas hipóteses e na forma que a lei estabelecer para fins de investigação criminal ou instrução processual penal;"
>
> (BRASIL, 1988)

[13] *http://www.privacyinternational.org*

A Constituição, ainda no artigo 5.º, inciso LXXII, alíneas *a* e *b*, também cita o *habeas data*, uma ação destinada à tutela dos direitos do cidadão a frente dos bancos de dados, a fim de permitir o fornecimento das informações, e uma eventual retificação, caso não corresponda à verdade:

> "LXXII - conceder-se-á 'habeas-data':
> a) para assegurar o conhecimento de informações relativas à pessoa do impetrante, constantes de registros ou bancos de dados de entidades governamentais ou de caráter público;
> b) para a retificação de dados, quando não se prefira fazê-lo por processo sigiloso, judicial ou administrativo;"
>
> (BRASIL, 1988)

O Código Civil brasileiro, no Capítulo II - "Dos direitos da personalidade", artigos 18.º e 21.º, também refere-se às questões de privacidade (BRASIL, 2002):

> "Art. 18. Sem autorização, não se pode usar o nome alheio em propaganda comercial.
> Art. 21. A vida privada da pessoa natural é inviolável, e o juiz, a requerimento do interessado, adotará as providências necessárias para impedir ou fazer cessar ato contrário a esta norma."
>
> (BRASIL, 2002)

Algumas propostas surgiram com o intuito de regularizar a proteção de privacidade, das quais duas merecem destaque: a *Organization for Economic Co-operation and Development* (OECD) e a *Federal Trade Comission* (FTC). Os princípios estabelecidos pela OECD fornecem diretrizes acerca da coleta de dados e como esses dados devem ser protegidos. Os 8 princípios são (OECD, 1980):

- **Limitação de coleta**: a coleta de dados pessoais deve ser limitada, e a aquisição dos mesmos deve ser feita de maneira justa e através de meios legais.
- **Qualidade dos dados**: Os dados devem ser relevantes aos propósitos de utilização.
- **Especificação de objetivo**: os propósitos da coleta devem ser especificados antes da coleta, e o uso deve corresponder ao objetivo.

- **Limitação de uso**: os dados não poderão ser utilizados além do que foi especificado (exceto por uma intervenção da lei).
- **Segurança**: os dados deverão ser protegidos, através de medidas de segurança, contra perda, acesso não-autorizado, destruição, entre outros.
- **Transparência**: deverá haver uma política que informe sobre as práticas e políticas acerca dos dados coletados.
- **Participação individual**: o usuário deverá ter o direito de acessar a todo tipo de dado pessoal coletado, podendo alterá-lo ou excluí-lo.
- **Responsabilidade**: o detentor dos dados deverá ser responsável por qualquer questão relacionada aos princípios anteriores.

Assim como a OECD, a FTC também busca a regulamentação das coletas de dados e a proteção da privacidade dos usuários. Alguns princípios também são apresentados por essa instituição (FTC, 2000):

- **Notificação**: o usuário deve ser notificado sobre as práticas de informação das entidades antes que a coleta de dados seja feita.
- **Escolha**: o usuário poderá escolher como suas informações serão utilizadas.
- **Acesso**: o usuário deverá ter controle sobre seus dados, podendo alterá-los ou excluí-los.
- **Segurança**: os dados devem ser precisos e estar armazenados com segurança.
- **Reforço**: mecanismos utilizados para efetivar os princípios anteriores.

Apesar dos mecanismos e legislação existentes, o usuário pode sempre reservar-se no direito de fornecer qualquer tipo de informação pessoal caso julgue que o sistema ou site esteja invadindo sua privacidade. Afinal, a privacidade constitui um limite natural ao direito da informação.

Resumo do Capítulo 2

Este Capítulo apresentou os conceitos gerais de privacidade, a legislação atual e alguns mecanismos para fornecer privacidade aos usuários na Internet.

Em sistemas computacionais, a segurança das informações é gerenciada através de modelos de controle de acesso. É necessário contemplar os aspectos de privacidade, de modo que uma ação no sistema não ofereça riscos à privacidade.

No próximo Capítulo, serão apresentadas as três classes principais de políticas de controle de acesso e quatro modelos de controle de acesso existentes na literatura.

CAPÍTULO

3

Controle de Acesso

> "É fácil ter-se um sistema de computação seguro. Você meramente tem que desconectar o seu sistema de qualquer rede externa, e permitir somente terminais ligados diretamente a ele. Pôr a máquina e seus terminais em uma sala fechada, e um guarda na porta."
>
> FRED GRAMPP e ROBERT MORRIS

O *controle de acesso* pode ser definido como uma limitação de acesso a qualquer tipo de recurso por usuários, programas, processos, outros sistemas ou outros tipos de entidades (DZIERZAWSKI, 1999). Geralmente, um controle de acesso reflete o conjunto de regras de um determinado sistema através de sua arquitetura, representada por uma matriz de controle, objetos, conjuntos, ou outras representações. Cada modelo explora as vantagens de uma dada representação e inclui

seu próprio conjunto de regras (políticas), juntamente com regras específicas para cada sistema; as regras do modelo definem quais tratamentos das informações são necessários para fornecer um bom nível de segurança para um sistema, sem falhas.

Neste Capítulo, são apresentadas as três principais classes de políticas para controle de acesso e alguns modelos de controle de acesso presentes na literatura.

3.1 Políticas

Políticas são requisitos que especificam como um determinado acesso é gerenciado e quem, sob quais circunstâncias, pode acessar tais informações (FERRAIOLO et al., 2003). Além disso, as políticas devem considerar quais são as possíveis ameaças à segurança e privacidade das informações e tratá-las de modo efetivo (SAMARATI; VIMERCATI, 2001).

Esta seção apresenta as três classes principais de políticas utilizadas para controle de acesso: o *Controle de Acesso Discricionário* (NCSC, 1985) (SANDHU; MUNAWER, 1998), o *Controle de Acesso Obrigatório* (NCSC, 1985) (OSBORN et al., 2000) e o *Controle de Acesso Baseado em Papéis* (SANDHU et al., 1996) (FERRAIOLO; KUHN, 1992).

3.1.1 Controle de Acesso Discricionário

O *Controle de Acesso Discricionário* (NCSC, 1985) (SANDHU; MUNAWER, 1998), mais conhecido como DAC (*Discretionary Access Control*), é uma política de acesso que apresenta meios para restringir o acesso a objetos baseados na identidade dos indivíduos ou dos grupos aos quais eles pertencem. De modo geral, o dono (*owner*) de um objeto decide quem terá direito de acesso a esse objeto e qual privilégio ele terá.

Em geral, uma política DAC inclui o conceito de *posse* (*ownership*) de um objeto, no qual o dono desse objeto tem direito de *controle* para conceder direito de *acesso*

ao objeto para outros indivíduos. Existem dois conceitos importantes: a) todo objeto deve ter um dono (com isso, a política é determinada pelo dono do objeto), e b) os direitos de acesso são concedidos pelo dono do objeto para um determinado indivíduo ou grupo.

A política DAC costuma ser muito flexível e amplamente utilizada em setores governamentais e comerciais (FERRAIOLO et al., 2003).

3.1.2 Controle de Acesso Obrigatório

A política de *Controle de Acesso Obrigatório* (NCSC, 1985) (OSBORN et al., 2000), mais conhecida como MAC (*Mandatory Access Control*), foi definida para minimizar algumas deficiências existentes na política DAC (FERRAIOLO et al., 2003). Na política MAC, é o sistema quem determina as políticas de acesso, e não o dono do objeto (como é feito na política DAC). Assim, todos os indivíduos e objetos recebem *rótulos de sensibilidade*.

Os *rótulos de sensibilidade* representam a sensibilidade das informações contidas nos objetos e a autorização necessária para indivíduos acessarem tais objetos (FERRAIOLO et al., 2003). Em outras palavras, o rótulo de sensibilidade de um indivíduo determina o seu nível de confiança, enquanto que o rótulo de sensibilidade de um objeto define o nível de confiança necessário para acessá-lo. Assim, para um indivíduo acessar um objeto, seu nível de confiança deve igual ou superior ao do objeto.

A política MAC permite centralizar a política de acesso de uma organização e é amplamente utilizada em setores cujos dados são altamente sensíveis, como a área militar (FERRAIOLO et al., 2003).

3.1.3 Controle de Acesso Baseado em Papéis

O *Controle de Acesso Baseado em Papéis* (FERRAIOLO; KUHN, 1992) (SANDHU et al., 1996), conhecido como RBAC (*Role Based Access Control*), limita o acesso

para recursos de um determinado sistema baseando-se no papel do indivíduo dentro de uma organização.

A política RBAC requer as três regras definidas a seguir (FERRAIOLO et al., 2003).

1. **Atribuição de papéis**: Um indivíduo pode executar uma transação somente se possuir um papel atribuído. As atividades de um sistema são conduzidas por meio de transações. Assim, todos os usuários ativos no sistema necessitam de papéis ativos.

2. **Autorização de papéis**: Uma regra deve ser autorizada para o indivíduo. Com a regra 1, esta regra garante que os usuários somente possuem papéis para os quais estão autorizados.

3. **Autorização de transações**: Um indivíduo pode executar uma transação se esta estiver autorizada para o papel do indivíduo. Com as regras 1 e 2, esta regra garante que um usuário pode executar somente as transações a que ele estiver autorizado.

Os direitos são atribuidos para papéis específicos, e não para indivíduos. Para atribuir direitos para um indivíduo, o procedimento consiste em atribuir os papéis apropriados para este indivíduo. Os papéis podem ser facilmente criados, alterados ou descontinuados sem a necessidade de atualização dos direitos de cada indivíduo. Um papel é basicamente uma coleção de direitos, e todos os indivíduos recebem direitos através dos papéis que lhes estão atribuídos, conforme é ilustrado na Figura 3.1 (FERRAIOLO; KUHN, 1992).

A Figura 3.1 indica os relacionamentos existentes na política RBAC, no qual os usuários relacionam-se com papéis, e estes relacionam-se com direitos. Em uma organização, os papéis são relativamente estáveis, enquanto que os indivíduos e direitos são numerosos e podem sofrer modificações ao longo do tempo. O controle de acesso efetuado através de papéis simplifica o gerenciamento e revisão dos

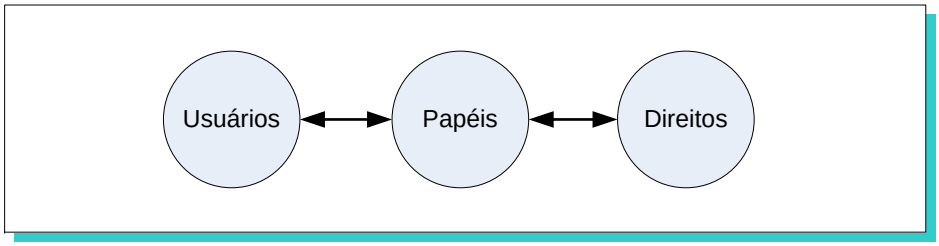

Figura 3.1: Relacionamentos da política RBAC (FERRAIOLO; KUHN, 1992).

direitos atribuídos a cada indivíduo (FERRAIOLO et al., 2003).

A política RBAC gerencia o controle de acesso em um nível que corresponde à estrutura de uma determinada organização (BARKLEY et al., 1997). Por exemplo, de acordo com JOSHI et al. (2001a), RBAC é uma solução atraente para fornecer aspectos de segurança em uma infra-estrutura governamental digital multidomínio, além de demonstrar grande relevância para tratar os requisitos complexos de segurança em aplicações baseadas na Web (JOSHI et al., 2001b).

3.2 Modelos de Controle de Acesso

Um *modelo de controle de acesso* fornece uma representação formal das políticas de controle de acesso e de seu funcionamento (SAMARATI; VIMERCATI, 2001). Nesta seção, são apresentados quatro modelos de controle de acesso entre os vários existentes na literatura (GRAHAM; DENNING, 1972) (HARRISON et al., 1976) (MOTTA, 2004) (FISCHER-HÜBNER, 2001) (BELL; LAPADULA, 1974) (LAMPSON, 1971) (HOFFMAN, 1997) (CLARK; WILSON, 1987) (BREWER; NASH, 1989) (BIBA, 1977) (BISKUP; LEINEWEBER, 2002).

3.2.1 Modelo HRU

O *modelo HRU*, proposto por Harrison, Ruzzo e Ullman (HARRISON et al., 1976) 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, apresentadas na Tabela 3.1.

Tabela 3.1: Conjunto de operações primitivas do modelo HRU.

Operação	**Significado**
enter r into (X_s, X_o)	insere r para o objeto o possuído pelo indivíduo s
delete r into (X_s, X_o)	remove r do objeto o possuído pelo indivíduo s
create subject X_s	cria o indivíduo s
create object X_o	cria o objeto o
destroy subject X_s	remove o indivíduo s
destroy object X_o	remove o objeto o

Cada operação (Tabela 3.1) atua sobre a matriz de controle, criando e removendo indivíduos, objetos e direitos. 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 definem. A política deste sistema resume-se ao conjunto de comandos que ele publica (GUNTER et al., 2004).

Uma configuração de um sistema de controle de acesso HRU é uma tripla (S, O, P), onde S é o conjunto dos indivíduos correntes, O é o conjunto de objetos correntes, $S \subseteq O$, e P é uma matriz de acesso, com uma linha para cada indivíduo $s_i \in S$ e uma coluna para cada objeto $o_j \in O$, R é o conjunto de direitos, e $P[s_i, o_j] \subseteq R$ contém os direitos do objeto o_j associado ao indivíduo s_i.

O Código 3.2.1 apresenta um comando do modelo HRU, responsável pela criação de um objeto associado a um indivíduo (HARRISON et al., 1976).

Comando do modelo HRU *create*(a, b)

```
command create(a, b)
    create object b
    enter own into (a, b)
end
```

Código 3.2.1: Comando *create*(a, b) do modelo HRU (HARRISON et al., 1976).

Exemplo 3.1. Supõe-se que a matriz de controle possui apenas um indivíduo, *alice*. O objeto *file1* será criado para *alice*, que terá direito de *posse* (own), através do comando *create(alice, file1)*. A Figura 3.2 ilustra as configurações da matriz de controle do HRU antes e após a execução do comando *create(alice, file1)*.

	alice
alice	∅

create(alice, file1)

	alice	*file1*
alice	∅	*own*

Figura 3.2: Configurações da matriz de controle do HRU antes e após a execução do comando *create(alice, file1)*.

□

O modelo HRU permite que apenas os aspectos de proteção do sistema sejam considerados, uma vez que não é necessário lidar com a semântica dos programas ou com modelos gerais de computação (HARRISON et al., 1976).

3.2.2 Modelo de Graham-Denning

O *modelo de Graham-Denning* (MGD), que leva o sobrenome de seus criadores (GRAHAM; DENNING, 1972), incorpora algumas constantes e conceitos relacionados a eventos. É muito semelhante ao modelo HRU (na realidade, o HRU é uma generalização do MGD), possuindo um conjunto S de indivíduos correntes, um conjunto O de objetos, um conjunto R de direitos e uma matriz de controle P.

Nesse modelo, existem dois direitos especiais: *own* (exercido sobre objetos) e *control* (exercido sobre indivíduos), criando-se a idéia de que um indivíduo é dono (*owner*) de um objeto, ou que um indivíduo tem controle (*control*) sobre outro indivíduo. Dessa forma, eventos podem ser proibidos se os devidos direitos não estiverem presentes. Todos os indivíduos também são tratados como objetos, para implementar o direito de controle (*control*).

As operações do MGD são apresentadas a seguir.

1. *create subject* s: cria o indivíduo s. O indivíduo que cria o novo indivíduo receberá o direito de controle (*control*) sobre esse novo indivíduo.

2. *remove subject* s: remove o indivíduo s. O indivíduo que remove deve possuir o direito de controle (*control*) sobre o indivíduo a ser removido.

3. *create object* o: cria o objeto o. O indivíduo que cria o novo objeto receberá o direito de posse (*own*) sobre esse novo objeto.

4. *remove object* o: remove o objeto o. O indivíduo que remove deve possuir o direito de posse (*own*) sobre o objeto a ser removido.

5. *read access right* r *of* s *on* o: lê o direito r do indivíduo s sobre o objeto o. O indivíduo deve controlar s ou ser dono de o.

6. *grant right* r *to* s *on* o: concede direito r do indivíduo s sobre o objeto o. O indivíduo deve ser dono de o.

7. *remove right* r *of* s *on* o: remove direito r do indivíduo s sobre o objeto o. O indivíduo deve ser dono de o ou controlar s.

8. *transfer right* r *to* s *on* o: transfere o direito r do indivíduo s sobre o objeto o. O indivíduo deve ter o direito r^* (que é uma versão transferível de r) sobre o.

O MGD reforça a matriz de controle de acesso adicionando relacionamentos entre indivíduos para controlar a delegação de direitos e a criação e remoção de indivíduos e objetos. Essas características tornam o modelo útil para sistemas multiusuário (GRAHAM; DENNING, 1972).

3.2.3 Modelo de Autorização Contextual baseado em Papéis

O *Modelo de Autorização Contextual baseado em Papéis* (MACP) foi proposto por MOTTA (2004) e apresenta-se como um modelo de limitação de recursos de acesso ao prontuário eletrônico do paciente em ambientes de saúde abertos e distribuídos. O acesso aos prontuários é feito de acordo com os papéis dos usuários no sistema, isto é, as funções que cada usuário exerce (MOTTA; FURUIE, 2003). Uma autorização contextual incorpora uma expressão lógica, chamada de *regra de autorização*, que é avaliada no momento em que o usuário tenta acessar um recurso. A autorização é positiva quando o resultado da regra de autorização é verdadeiro,

e negativa no caso contrário. O Código 3.2.2 apresenta um exemplo de regra de autorização para acesso ao laudo de um paciente (MOTTA, 2004).

Regra de autorização *Acesso ao laudo de um paciente*

```
< exp-abs(umCodPac) {
    umCodPac in pacCtx.pacientes_internados
}, verLaudo, prontuário_do_paciente, forte >.
```

Código 3.2.2: Exemplo de regra de autorização do MACP (MOTTA, 2004).

A regra de autorização apresentada no Código 3.2.2 especifica que o acesso aos laudos do prontuário do paciente identificado pelo parâmetro *umCodPac* somente é permitido se ele está internado (MOTTA, 2004). A permissão dependerá da avaliação da regra de autorização, representada pela expressão lógica *exp-abs(umCodPac)*.

De acordo com MOTTA (2004), "o tipo de autorização indica se ela é forte ou fraca. Autorizações fortes estabelecem políticas absolutas, que não podem ser revogadas, enquanto as fracas são usadas para definir políticas mais permissivas". No exemplo apresentado no Código 3.2.2, a autorização para acesso ao laudo de um paciente é forte.

A autorização contextual confere ao modelo flexibilidade e poder expressivo para o estabelecimento de políticas de acesso aos prontuários eletrônicos e políticas administrativas que se adequam ao ambiente e cultura das organizações de saúde.

3.2.4 Modelo de Privacidade baseado em Tarefas

Para conseguir um modelo formal de segurança que utilize aspectos legais de privacidade, FISCHER-HÜBNER (2001) propôs um *Modelo de Privacidade baseado em Tarefas* (MPT) (*Task-based Privacy Model*). O modelo reforça dois requisitos importantes de privacidade, o de *purpose binding*, no qual dados obtidos para um determinado propósito não podem ser utilizados para outro propósito sem haver um consentimento explícito, e o de *necessidade de processamento de dados*, no qual os dados serão processados apenas se forem necessários para a realização de

uma tarefa. A política de privacidade do MPT é informalmente descrita a seguir.

> Um indivíduo somente pode ter acesso às informações pessoais se esse acesso for necessário para realizar sua tarefa atual (o indivíduo também necessita de autorização para realizar essa tarefa). O indivíduo pode apenas acessar as informações de uma maneira controlada através de um procedimento de transformação, para qual a tarefa atual deve ser autorizada. Além disso, o propósito da tarefa atual deve corresponder aos propósitos das informações pessoais (definidos quando foram obtidas), ou ter um consentimento dos donos das informações.
>
> (FISCHER-HÜBNER, 2001)

Os elementos presentes no MPT são apresentados a seguir (FISCHER-HÜBNER, 2001).

1. **Indivíduos**: são as entidades ativas do sistema (por exemplo, usuários).
2. **Objetos**: são as entidades passivas do sistema (por exemplo, arquivos e registros).
3. **Objetos pessoais**: são objetos que possuem informações pessoais.
4. **Classes de objetos**: um objeto de informações pessoais pode ser classificado por uma classe, como, por exemplo, um prontuário de um paciente em um ambiente hospitar.
5. **Tarefas**: um indivíduo pode acessar um objeto de informações pessoais somente ao realizar uma tarefa (por exemplo, realizar diagnóstico de um paciente).
6. **Tarefa corrente**: é a tarefa que um determinado indivíduo está realizando no momento.
7. **Tarefas autorizadas**: são as tarefas autorizadas para um indivíduo.
8. **Papéis de usuário**: são os papéis associados para cada indivíduo.
9. **Usuários responsáveis**: cada tarefa possui um conjunto de indivíduos responsáveis pela execução.

10. **Propósitos**: cada tarefa possui algum propósito. Além disso, as informações pessoais também são coletadas para algum propósito. Assim, para cada ação, é necessário determinar qual é o propósito das tarefas e do acesso aos objetos.

11. **Procedimentos de transformação**: são procedimentos que acessam os objetos de uma maneira controlada.

12. **Procedimento de transformação corrente**: é o procedimento de transformação realizado por um determinado indivíduo no momento.

13. **Procedimentos de transformação autorizados**: Ao realizar uma tarefa, um indivíduo pode executar apenas os procedimentos de transformação autorizados.

14. **Direitos de acesso**: são os direitos que um indivíduo pode ter em relação a um objeto de informações pessoais (por exemplo, direito de leitura).

15. **Direitos necessários**: conjunto de direitos necessários para realizar uma determinada tarefa.

16. **Acessos correntes**: conjunto de acessos correntes de um determinado indivíduo.

17. **Consentimento**: consentimento para acessar alguns tipos de objetos de informações pessoais.

O modelo de privacidade contém as *propriedades de privacidade*, que são variáveis de estado, invariantes e restrições, e as *regras*, que são transições de estado, podendo ser utilizado para implementar um controle de acesso com reforço em privacidade.

O MPT é um modelo que somente reforça os princípios de privacidade e, portanto, necessita de uma combinação de outros modelos para realizar o controle de acesso (FISCHER-HÜBNER, 2001).

Resumo do Capítulo 3

Este Capítulo apresentou as três classes principais de políticas de controle de acesso (*Controle de Acesso Discricionário*, *Controle de Acesso Obrigatório* e *Controle de Acesso Baseado em Papéis*), que especificam como o gerenciamento de um determinado acesso é realizado. Foram apresentados quatro modelos de controle de acesso presentes na literatura (HRU, Graham-Denning, MACP e MPT), ressaltando suas características principais.

Em sistemas computacionais, a segurança das informações dos indivíduos é gerenciada através de controles de acesso. Entretanto, os modelos existentes contemplam superficialmente os aspectos de privacidade ou necessitam de controles adicionais para suprir tais lacunas. Além disso, os sistemas computacionais são dinâmicos e alteram suas regras ou estados periodicamente. A capacidade de adaptatividade de um modelo de controle de acesso permite que tais sistemas possam atender os requisitos de privacidade e segurança de forma flexível.

No próximo Capítulo, será apresentada uma breve descrição sobre os autômatos finitos, autômatos de pilha, autômatos de pilha estruturados e a definição dos autômatos adaptativos.

CAPÍTULO

4

Autômatos Adaptativos

"Depois de mais de vinte anos trabalhando nisso [na área de Tecnologias Adaptativas], tenho o mesmo entusiasmo que tinha no primeiro dia. [...] Este assunto é algo que me cativa muito. E o fato de as pessoas se animarem também com isso é que me mantém entusiasmado."

JOÃO JOSÉ NETO

Os autômatos adaptativos (NETO, 1993) (NETO, 1994) constituem um mecanismo formal para a descrição de linguagens recursivamente enumeráveis; sua simplicidade e facilidade de entendimento em relação ao modelo clássico de reconhecimento destas linguagens, a Máquina de Turing[1] (TURING, 1936), fazem com que sua utilização seja muito ampla (NETO, 2007). Neste Capítulo, são apresentados

[1] Dispositivo teórico proposto pelo matemático Alan Turing, também denominada de máquina universal.

os conceitos básicos da teoria de autômatos, uma breve descrição dos autômatos finitos, autômatos de pilha e, por fim, dos autômatos adaptativos.

4.1 Conceitos iniciais

Algumas definições são importantes para tratar da teoria de autômatos e, portanto, serão apresentadas a seguir (NETO, 1993) (HOPCROFT et al., 2002) (LEWIS; PAPADIMITRIOU, 1998).

Um *símbolo* é um elemento individual, abstrato e indivisível. As letras e dígitos são os exemplos mais comuns de símbolos.

Um *alfabeto* é um conjunto finito e não vazio de símbolos. Geralmente, o símbolo Σ é utilizado para representar um alfabeto. Alguns exemplos são:

- $\Sigma = \{a, b, c, \ldots, z\}$, conjunto de todas as letras minúsculas.
- $\Sigma = \{A, B, C, \ldots, Z\}$, conjunto de todas as letras maiúsculas.
- $\Sigma = \{0, 1, \ldots, 9, A, B, \ldots, F\}$, alfabeto dos dígitos do sistema numérico hexadecimal.

Uma *cadeia* é uma seqüência finita de símbolos (com possíveis repetições) do alfabeto, justapostos. Quando uma cadeia pertence a uma linguagem, pode ser também ser chamada de *palavra* ou *sentença*. Alguns exemplos de cadeias, assumindo o alfabeto $\Sigma = \{a, b\}$: ab, aa, bb, aab, $aabb$.

A *cadeia vazia* é uma cadeia que não possui ocorrências de símbolos. Essa cadeia é denotada por ϵ.

O *comprimento de uma cadeia* corresponde ao número de símbolos presentes nessa cadeia. A notação utilizada para representar o comprimento de uma cadeia w é $|w|$. Por exemplo, o comprimento da cadeia aab é expresso na forma $|aab| = 3$. O comprimento de ϵ, dado por $|\epsilon| = 0$.

O conjunto de todas as cadeias de comprimento i de um alfabeto é definido como Σ^i, chamado de *potência de um alfabeto*. Por exemplo, dado o alfabeto $\Sigma = \{a, b\}$,

$\Sigma^1 = \{a, b\}$, $\Sigma^2 = \{aa, ab, ba, bb\}$, e assim por diante. O conjunto de todas as cadeias de um alfabeto é denotado por Σ^*, ou seja, $\Sigma^* = \Sigma^0 \cup \Sigma^1 \cup \Sigma^2 \cup \ldots$.

Uma *linguagem formal* L, ou simplesmente *linguagem*, é um conjunto arbitrário de cadeias específicas sobre um alfabeto Σ, de modo que $L \subseteq \Sigma^*$. Alguns exemplos de linguagens, dado o alfabeto $\Sigma = \{a, b\}$:

- A linguagem de todas as cadeias que possuem um número n de a's seguidos por n b's, para $n \geq 0$: $\{\epsilon, ab, aabb, aaabbb, aaaabbbb, \ldots\}$
- A linguagem de todas as cadeias que possuem número de a's igual ao número de b's: $\{\epsilon, ab, ba, aabb, abab, baba, abba, bbaa, \ldots\}$

Um *problema*, na teoria de autômatos, é a decisão sobre pertinência de uma cadeia a uma linguagem (HOPCROFT et al., 2002). Em outras palavras, se Σ é um alfabeto e L é uma linguagem sobre Σ, o problema será o seguinte: dado uma cadeia w em Σ^*, definir se w pertence ou não a L.

Uma *gramática* é uma descrição formal de uma linguagem, e é descrita por um modelo matemático contendo um conjunto finito de regras de reescrita, um alfabeto terminal, um alfabeto não-terminal e a identificação de um símbolo não-terminal inicial. Formalmente, uma gramática G é descrita por uma quádrupla $G = (V, T, P, S)$, onde V é o conjunto de símbolos não-terminais, T é o conjunto de símbolos terminais, P é o conjunto de regras (também chamadas *produções*), $P \subseteq (V \cup T)^* V (V \cup T)^* \times (V \cup T)^*$, e S é o símbolo inicial, $S \in V$.

O processo de geração de uma cadeia por uma gramática é chamado *derivação*.

O símbolo $\Rightarrow$ denota uma etapa de derivação. A linguagem gerada por uma gramática G, denotada como $L(G)$, é composta por todas as cadeias, formadas somente de símbolos terminais, deriváveis a partir do símbolo inicial S. Em outras palavras, $L(G) = \{w \in T^* \mid S \Rightarrow^+ w\}$, onde $\Rightarrow^+$ representa uma ou mais etapas de derivação.

As linguagens podem ser representadas através de suas gramáticas ou de dispositivos que reconhecem cadeias pertencentes a elas. As gramáticas são geradoras

de sentenças pertencentes a uma linguagem, enquanto os reconhecedores aceitam ou rejeitam uma determinada cadeia, de acordo com a pertinência destas à linguagem. Os autômatos são dispositivos reconhecedores.

Existe uma classificação para linguagens formais, por ordem de complexidade, feita por Chomsky (CHOMSKY, 1956) (CHOMSKY, 1957), no final da década de 50. Essa classificação leva o nome de *Hierarquia de Chomsky* (Figura 4.1). É importante observar que uma linguagem pode ser representada por infinitas gramáticas.

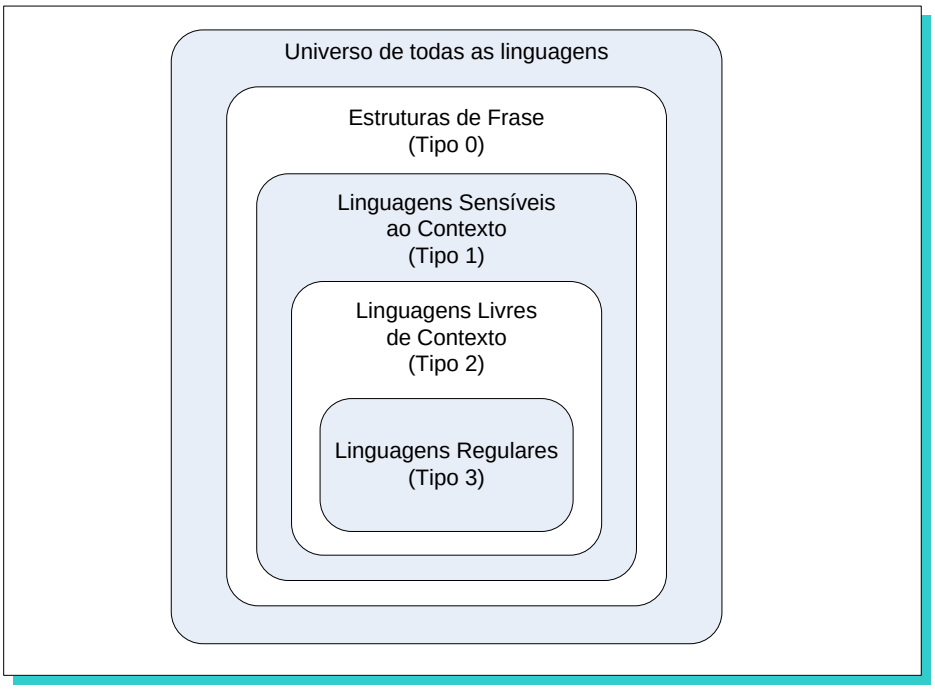

Figura 4.1: Hierarquia de Chomsky

De acordo com a Hierarquia de Chomsky, apresentada na Figura 4.1, as linguagens podem ser:

- **Estrutura de frase (Tipo 0)**: essas linguagens não possuem nenhuma limitação. Para gerar cadeias, é utilizada uma gramática que não possui restrições sobre a forma das produções; essa gramática é chamada Irrestrita. Para o reconhecimento de cadeias pertencentes a essas linguagens, utiliza-se uma

Máquina de Turing com fita infinita.

- **Linguagens sensíveis ao contexto (Tipo 1)**: as linguagens sensíveis ao contexto possuem a única restrição de que nenhuma substituição possa reduzir o comprimento da forma sentencial. Uma *forma sentencial* de uma gramática é qualquer cadeia que pode ser derivada a partir do símbolo inicial. O reconhecimento deste tipo de linguagens pode ser feito utilizando-se uma Máquina de Turing com fita limitada.

- **Linguagens livres de contexto (Tipo 2)**: as linguagens do tipo 2 são definidas por gramáticas cujas regras de produção baseiam-se em substituições simples e incondicionais de símbolos não-terminais, além de permitir construções aninhadas. Os reconhecedores dessas linguagens são os Autômatos de Pilha.

- **Linguagens regulares (Tipo 3)**: as linguagens do tipo 3 possuem regras de formação baseadas apenas nas operações de concatenação, união e *fecho de Kleene*[2]. Constituem o tipo mais simples de linguagem, e seu reconhecimento pode ser feito através de um Autômato Finito.

A Tabela 4.1 mostra os tipos de linguagens associadas com suas gramáticas e reconhecedores (HOPCROFT et al., 2002).

Tabela 4.1: Linguagens, gramáticas e reconhecedores (HOPCROFT et al., 2002).

Hierarquia de Chomsky	**Gramática**	**Linguagem**	**Reconhecedor**
Tipo 0	Estrutura de frase	Recursivamente enumerável	Máquina de Turing
Tipo 1	Sensível ao contexto	Recursiva	Máquina de Turing com fita limitada
Tipo 2	Livre de contexto	Livre de contexto	Autômato de Pilha
Tipo 3	Regular	Regular	Autômato Finito

[2]Representa a repetição arbitrária ilimitada de símbolos

A seguir, serão apresentados os conceitos dos Autômatos Finitos, Autômatos de Pilha e os Autômatos Adaptativos.

4.2 Autômato Finito

O *Autômato Finito*, também chamado *Máquina de Estados Finitos*, é o dispositivo reconhecedor das linguagens regulares (tipo 3). É baseado em estados e transições, e não utiliza memória auxiliar durante o processo de reconhecimento.

Definição 4.1 (Autômato Finito)**.** Um *autômato finito* M é definido por uma quíntupla $M = (Q, \Sigma, P, q_0, F)$, onde Q é o conjunto finito não vazio de estados, Σ é o alfabeto, P é o mapeamento que indica as transições possíveis em cada configuração do autômato, q_0 é o estado inicial, $q_0 \in Q$, e F é o conjunto de estados finais ou de aceitação, $F \subseteq Q$. □

Se, para cada elemento do conjunto $Q \times \Sigma$, o mapeamento $P : Q \times \Sigma \rightarrow Q$ possui um único estado q' $\in Q$, o autômato é chamado *determinístico*.

Uma configuração é definida pelo par (q, w), onde $q \in Q$ e $w \in \Sigma^*$. Define-se $\vdash$ como uma relação entre configurações e, dado o mapeamento $P(q, a) \rightarrow q$', $(q, aw) \vdash (q', w)$, onde q é o estado antes do consumo do símbolo a, q' é o novo estado e w é o resto da cadeia. A relação $\vdash^*$ denota o fecho da relação $\vdash$, ou seja, zero ou mais ocorrências da relação $\vdash$.

Dada uma cadeia $w \in \Sigma^*$ e o estado inicial $q_0 \in Q$, a configuração inicial (q_0, w) poderá ser relacionada com a configuração (q_f, ϵ), ou seja, $(q_0, w) \vdash (q_f, \epsilon)$. Se o estado corrente q_f pertencer ao conjunto F, a cadeia w é aceita pelo autômato finito.

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

Exemplo 4.2. Considere um autômato finito determinístico $M = (\{q_0, q_1\}, \{a, b\}, P, q_0, \{q_1\})$, responsável pelo reconhecimento de cadeias que pertençam à linguagem

$L_1 = \{w \in \{a,b\}^* | w$ é da forma $a^*b\}$, ou seja, w é uma cadeia que contém zero ou mais ocorrências do símbolo a e termina com um b. O mapeamento P é definido na Tabela 4.2.

Tabela 4.2: Mapeamento P do autômato finito determinístico do exemplo 4.2.

Mapeamento P
$P(q_0, a) \rightarrow q_0 \quad P(q_0, b) \rightarrow q_1$

O autômato finito determinístico M correspondente é ilustrado na Figura 4.2.

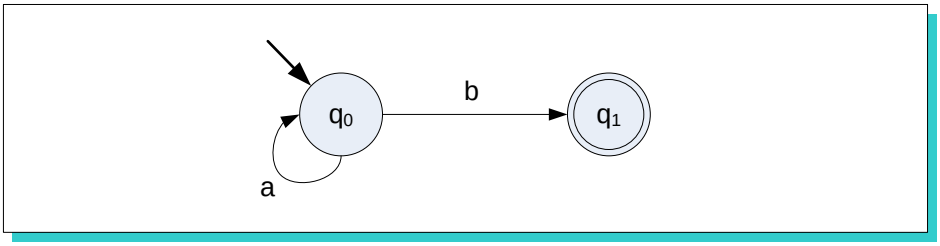

Figura 4.2: Exemplo de um autômato finito determinístico.

Supondo a cadeia $aaab$, deseja-se saber se esta pertence à linguagem L_1. A Tabela 4.3 mostra a seqüência de passos para o reconhecimento dessa cadeia. Como $q_1 \in F$, a cadeia é aceita e, portanto, pertence à linguagem L_1. Caso o estado não pertencesse ao conjunto F, ou não existissem transições compatíveis com um determinado símbolo, a cadeia seria rejeitada.

Tabela 4.3: Seqüência de passos para o reconhecimento da cadeia $aaab$.

Origem	**Transição**	**Destino**
q_0	$(q_0, aaab) \vdash (q_0, aab)$	q_0
q_0	$(q_0, aab) \vdash (q_0, ab)$	q_0
q_0	$(q_0, ab) \vdash (q_0, b)$	q_0
q_0	$(q_0, b) \vdash (q_1, \epsilon)$	q_1

□

Se existissem duas ou mais transições consumindo o mesmo símbolo, partindo de um mesmo estado, e atingindo estados diferentes, o autômato seria *não-determinístico* (HOPCROFT et al., 2002). Sua definição formal é parecida com a

do autômato determinístico, com uma única mudança: o mapeamento, ao invés de retornar um único estado de destino, retornará um subconjunto de Q. Ou seja, $P : Q \times \Sigma \rightarrow 2^Q$, onde 2^Q são partes de Q. A linguagem descrita por um autômato finito não-determinístico também pode ser descrita por um autômato determinístico.

4.3 Autômato de Pilha

O *Autômato de Pilha* é um tipo de autômato que permite o reconhecimento de linguagens livres de contexto (tipo 2), que apresentam as mesmas características das linguagens regulares, com o acréscimo de permitir construções aninhadas (HOPCROFT et al., 2002). Basicamente, é um autômato finito que permite a existência de construções sintáticas aninhadas e utiliza-se de uma memória auxiliar, a pilha (Figura 4.3).

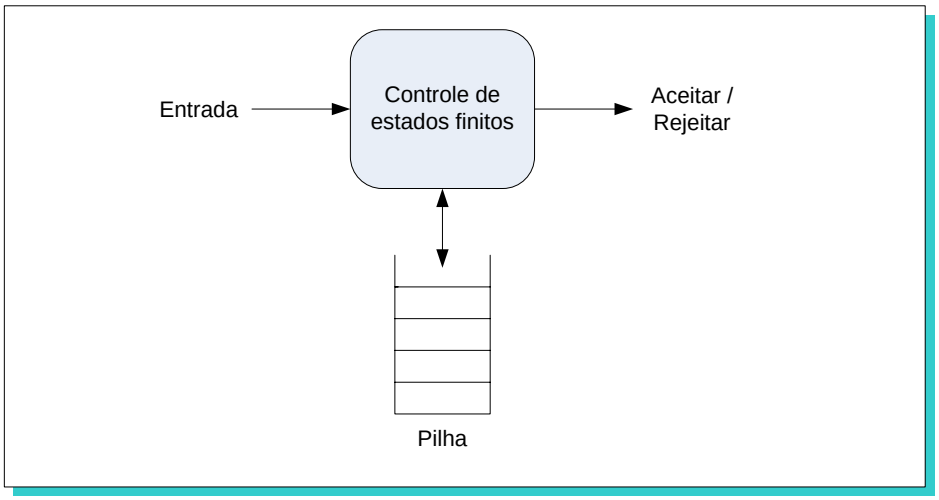

Figura 4.3: O autômato de pilha é um autômato finito com uma memória auxiliar com estrutura de pilha (HOPCROFT et al., 2002).

Essa pilha não depende da cadeia de símbolos a ser analisada, e não possui limitação de tamanho. Sua política de acesso é *LIFO* (*Last In, First Out*), ou seja, o último dado armazenado será o primeiro a ser lido. A base da pilha é fixa, enquanto o topo é variável e indica a posição do último símbolo inserido.

Definição 4.3 (Autômato de Pilha)**.** Um *autômato de pilha* M é descrito por $M = (Q, \Sigma, \Gamma, P, q_0, Z_0, F)$, onde Q é o conjunto finito de estados, Σ é o alfabeto, Γ é o alfabeto da pilha representando o conjunto de símbolos que têm permissão para serem inseridos na pilha, P é o mapeamento, q_0 é o estado inicial, $q_0 \in Q$, Z_0 é um símbolo especial indicador de pilha vazia (no início, a pilha contém este elemento no topo), e F é o conjunto de estados finais, $F \subseteq Q$. □

O mapeamento do autômato de pilha é dado por $P : Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \rightarrow Q \times (\Gamma \cup \{\epsilon\})^*$. Existem dois tipos de mapeamento:

- $P(q, a, \beta) \rightarrow (q', \beta')$, significando que o autômato de pilha, estando no estado q, consumindo o símbolo a e com β no topo da pilha, deverá ir para um novo estado q', substituindo β por β'.
- $P(q, \epsilon, \beta) \rightarrow (q', \beta')$, significando que o autômato de pilha, estando no estado q, independente do símbolo de entrada e com β no topo da pilha, deverá ir para um novo estado q', substituindo β por β'.

β' representa o novo topo da pilha. De acordo com seu comprimento, ele poderá realizar as seguintes ações:

- $|\beta'| > 1$: troca-se o símbolo do topo da pilha e outros símbolos são empilhados.
- $|\beta'| = 1$: troca-se o símbolo do topo da pilha.
- $|\beta'| = 0$: o símbolo do topo da pilha é desempilhado ($\beta' = \epsilon$).

Assim, $(q, aw, \beta) \vdash (q', w, \beta')$, onde q é o estado antes do consumo do símbolo a (que pode ser ϵ), w é o resto da cadeia, β é o símbolo no topo da pilha, q' é o novo estado, e β' é o novo topo da pilha.

O autômato de pilha pode reconhecer uma cadeia através de duas formas: *estados finais* ou por *pilha vazia*. Ambos são equivalentes, logo, é possível ter um autômato que reconheça cadeias de uma linguagem utilizando ou estados finais ou por pilha vazia (HOPCROFT et al., 2002).

A linguagem aceita por um autômato de pilha M é dada por:

- **Reconhecimento por estado final**: $L(M) = \{w \in \Sigma^* | (q_0, w, Z_0) \vdash^* (q_f, \epsilon, \alpha)$ onde $q_f \in F$ e α representa qualquer conteúdo na pilha $\}$.
- **Reconhecimento por pilha vazia**: $N(M) = \{w \in \Sigma^* | (q_0, w, Z_0) \vdash^* (q_f, \epsilon, \epsilon)$ onde $q_f \in F\}$.

Exemplo 4.4. Considere um autômato de pilha $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{Z_0, X\}, P, q_0, Z_0, \{q_2\})$, responsável pelo reconhecimento de cadeias que pertençam à linguagem $L_2 = \{w \in a, b^* | w$ é da forma $a^n b^n$, $n \geq 1\}$, ou seja, w é uma cadeia que contém um certo número de a's seguido pelo mesmo número de b's. O autômato em questão reconhece a cadeia por pilha vazia. O mapeamento P é definido na Tabela 4.4.

Tabela 4.4: Mapeamento P do autômato de pilha do exemplo 4.4.

Mapeamento P		
$P(q_0, a, Z_0) \rightarrow (q_0, XZ_0)$	$P(q_0, a, X) \rightarrow (q_0, XX)$	$P(q_0, b, X) \rightarrow (q_1, \epsilon)$
$P(q_1, b, X) \rightarrow (q_1, \epsilon)$	$P(q_1, \epsilon, Z_0) \rightarrow (q_2, \epsilon)$	

O autômato de pilha M correspondente é ilustrado na Figura 4.4.

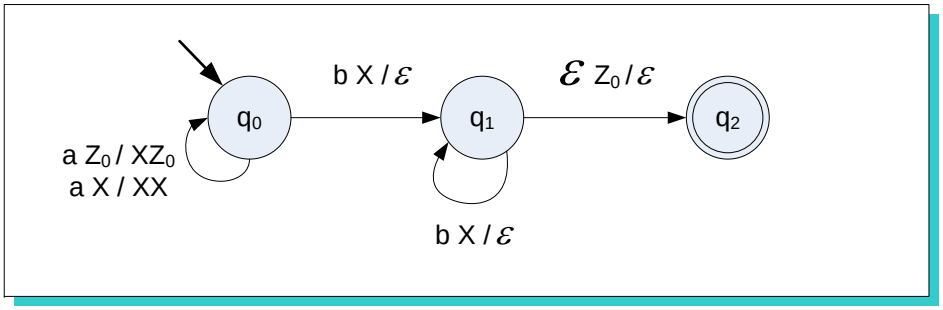

Figura 4.4: Exemplo de um autômato de pilha.

Supondo a cadeia $aaabbb$, deseja-se saber se esta pertence à linguagem L_2. A Tabela 4.5 mostra a seqüência de transições para o reconhecimento dessa cadeia. Nas colunas referentes ao conteúdo da pilha (2 e 4), considere o símbolo mais à esquerda como sendo o topo da pilha.

Tabela 4.5: Seqüência de transições para o reconhecimento da cadeia $aaabbb$.

Origem	Pilha (antes)	Transição	Pilha (depois)	Destino
q_0	Z_0	$(q_0, aaabbb, Z_0) \vdash (q_0, aabbb, XZ_0)$	XZ_0	q_0
q_0	XZ_0	$(q_0, aabbb, X) \vdash (q_0, abbb, XX)$	XXZ_0	q_0
q_0	XXZ_0	$(q_0, abbb, X) \vdash (q_0, bbb, XX)$	$XXXZ_0$	q_0
q_0	$XXXZ_0$	$(q_0, bbb, X) \vdash (q_1, bb, \epsilon)$	XXZ_0	q_1
q_1	XXZ_0	$(q_1, bb, X) \vdash (q_1, b, \epsilon)$	XZ_0	q_1
q_1	XZ_0	$(q_1, b, X) \vdash (q_1, \epsilon, \epsilon)$	Z_0	q_1
q_1	Z_0	$(q_1, \epsilon, Z_0) \vdash (q_2, \epsilon, \epsilon)$	$-$	q_2

O estado q_2 pertence ao conjunto de estados de aceitação, $q_2 \in F$, logo, a cadeia é aceita e pertence à linguagem L_2. □

4.3.1 Autômato de Pilha Estruturado

O *Autômato de Pilha Estruturado* (NETO, 1993) é um tipo de autômato de pilha formado por um conjunto de autômatos finitos mutuamente recursivos. Esses autômatos também são chamados *sub-máquinas*. A pilha tem, neste caso, a finalidade exclusiva de armazenar 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; essa transição consiste em utilizar o símbolo de entrada apenas para a tomada de decisão do autômato em relação a qual transição executar, sendo o tal símbolo consumido na próxima transição (NETO, 1993) (NETO, 1994).

Definição 4.5 (Autômato de Pilha Estruturado)**.** Um *autômato de pilha estruturado* M é descrito por $M = (Q, A, \Sigma, \Gamma, P, q_0, Z_0, F)$, onde Q é o conjunto finito não vazio 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 o símbolo inicial da pilha, e F é o conjunto de estados finais. □

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 $\vdash$ 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' $\in \{a, \epsilon\}$, e β' é o novo topo da pilha, β' $\in \Gamma \cup \{\epsilon\}$.

Definição 4.6 (Sub-máquina do Autômato de Pilha Estruturado)**.** O conjunto de sub-máquinas do autômato de pilha estruturado é representado por A. Cada *sub-máquina* a_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 a_i, $\Sigma_i \subseteq \Sigma$ é o conjunto de símbolos de entrada da sub-máquina a_i, $P_i \subseteq P$ é o mapeamento da sub-máquina a_i, $q_{i0} \in Q_i$ é o estado de entrada da sub-máquina a_i, e $F_i \subseteq Q_i$ é o conjunto de estados finais da sub-máquina a_i. □

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

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

Suponha que o estado atual seja q e a cadeia 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 estado $q' \in Q$ no topo da pilha, $\beta \neq Z_0$, então q' é retirado da pilha e o estado corrente passa a ser q'.

Graficamente, uma chamada de sub-máquina pode ser representada como na Figura 4.5, na qual a partir do estado q_j, a sub-máquina a_i é chamada, e o estado de retorno q_k é inserido no topo da pilha. Entretanto, é necessário reforçar que a transição $q_j \rightarrow q_k$ não está em P, mas denota um mapeamento existente em P. No exemplo, o estado atual passa a ser o estado inicial da sub-máquina a_i, que é q_{i0}.

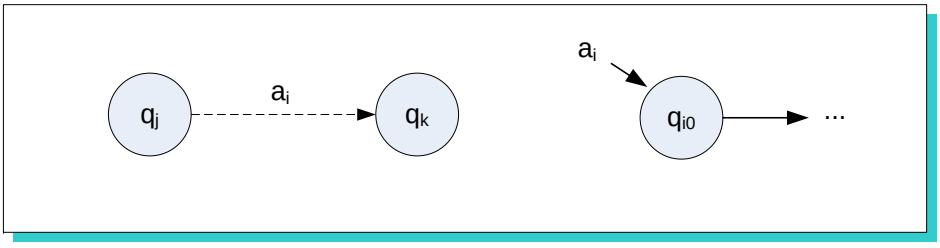

Figura 4.5: Exemplo de chamada da sub-máquina a_i.

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 $q_f \in F\}$.

Exemplo 4.7. Considere um autômato de pilha estruturado $M = (\{q_0, q_1, q_2, q_3\}, A, \{a, b\}, \{Z_0, q_0, q_1, q_2, q_3\}, P, q_0, Z_0, \{q_0, q_3\})$ responsável pelo reconhecimento de cadeias que pertençam à linguagem $L_3 = \{w \in \{a, b\}^* | w$ é da forma $a^n b^n$, $n \geq 0\}$, ou seja, w é uma cadeia que contém um número n de a's seguido de um número n de b's. O conjunto de sub-máquinas de M é definido por $A = \{t_1\}$, onde $t_1 = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, P_{t_1}, q_0, \{q_0, q_3\})$, $P_{t_1} = P$. O mapeamento P é definido na Tabela 4.6.

Tabela 4.6: Mapeamento P do autômato de pilha estruturado do exemplo 4.7.

Mapeamento P		
$P(q_0, a, Z_0) \rightarrow (q_1, \epsilon, Z_0)$	$P(q_1, a, Z_0) \rightarrow (q_0, a, q_2 Z_0)$	$P(q_1, b, Z_0) \rightarrow (q_0, b, q_2 Z_0)$
$P(q_0, a, q_2) \rightarrow (q_1, \epsilon, q_2)$	$P(q_1, a, q_2) \rightarrow (q_0, a, q_2 q_2)$	$P(q_1, b, q_2) \rightarrow (q_0, b, q_2 q_2)$
$P(q_0, b, q_2) \rightarrow (q_2, b, \epsilon)$	$P(q_2, b, Z_0) \rightarrow (q_3, \epsilon, Z_0)$	$P(q_2, b, q_2) \rightarrow (q_3, \epsilon, q_2)$
$P(q_3, b, q_2) \rightarrow (q_2, b, \epsilon)$		

O autômato de pilha estruturado M correspondente é ilustrado na Figura 4.6.

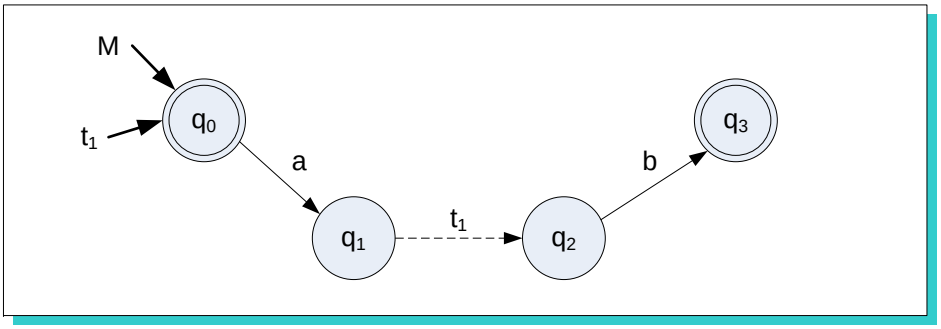

Figura 4.6: Exemplo de um autômato de pilha estruturado.

Supondo a cadeia $aabb$, deseja-se saber se esta pertence à linguagem L_3. A Tabela 4.7 mostra a seqüência de transições para o reconhecimento dessa cadeia.

O estado q_3 pertence ao conjunto de estados de aceitação e a pilha está vazia (através do indicador de pilha vazia Z_0), logo, a cadeia é aceita e pertence à linguagem L_3.

Tabela 4.7: Seqüência de transições para o reconhecimento da cadeia $aabb$.

Origem	**Pilha (antes)**	**Transição**	**Pilha (depois)**	**Destino**
q_0	Z_0	$(q_0, aabb, Z_0) \vdash (q_1, abb, Z_0)$	Z_0	q_1
q_1	Z_0	$(q_1, abb, Z_0) \vdash (q_0, abb, q_2 Z_0)$	$q_2 Z_0$	q_0
O estado anterior era q_1. Chamou-se a sub-máquina M, cujo estado inicial é q_0. Foi feita uma operação de *lookahead*, onde não houve o consumo do símbolo a. O estado de retorno q_2 é inserido na pilha.				
q_0	$q_2 Z_0$	$(q_0, abb, q_2) \vdash (q_1, bb, q_2)$	$q_2 Z_0$	q_1
q_1	$q_2 Z_0$	$(q_1, bb, q_2) \vdash (q_0, bb, q_2 q_2)$	$q_2 q_2 Z_0$	q_0
q_0	$q_2 q_2 Z_0$	$(q_0, bb, q_2) \vdash (q_2, bb, \epsilon)$	$q_2 Z_0$	q_2
q_2	$q_2 Z_0$	$(q_2, bb, q_2) \vdash (q_3, b, q_2)$	$q_2 Z_0$	q_1
q_3	$q_2 Z_0$	$(q_3, b, q_2) \vdash (q_2, b, \epsilon)$	Z_0	q_2
q_2	Z_0	$(q_2, b, Z_0) \vdash (q_3, \epsilon, Z_0)$	Z_0	q_3

O autômato de pilha estruturado possui o mesmo poder de reconhecimento do autômato de pilha tradicional, apresentado anteriormente.

4.4 Autômato Adaptativo

O *Autômato Adaptativo*, proposto por NETO (1993), é uma extensão do formalismo do Autômato de Pilha Estruturado que permite o reconhecimento de linguagens do tipo 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 provocar alterações em sua própria topologia durante o processo de reconhecimento de uma dada cadeia (NETO, 1994).

Essa capacidade de alteração do autômato faz-se possível através da utilização 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 (NETO, 2001). 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 configuração do autômato. Para a aceitação de uma determinada cadeia, o autômato percorrerá um caminho em um espaço de autômatos; em outras palavras, haverá um autômato E_0, que iniciará o reconhecimento de uma determinada cadeia; autômatos intermediários E_i, que serão criadas ao longo do reconhecimento; e um autômato 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 autômatos $< E_0, \alpha_0 > \rightarrow < E_1, \alpha_1 > \rightarrow \ldots \rightarrow < E_n, \alpha_n >$, onde E_i representa um autômato correspondente à aceitação da sub-cadeia α_i.

Definição 4.8 (Autômato Adaptativo)**.** 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 finito de sub-máquinas, Σ é o alfabeto de entrada, $\Sigma \subset \Sigma^A$, Σ^A é o conjunto enumerável de todos os símbolos possíveis, Γ é o alfabeto da pilha, $\Gamma \subset \Gamma^A$, $\Gamma = Q \cup \{Z_0\}$, Γ^A é o conjunto enumerável de todos os símbolos possíveis da pilha, $\Gamma^A = Q^A \cup \{Z_0\}$, P é um mapeamento $P : Q^A \times \Sigma^A \times \Gamma^A \rightarrow Q^A \times (\Sigma^A \cup \{\epsilon\}) \times (\Gamma^A \cup \{\epsilon\}) \times H^0 \times H^0\}$, H^0 definido a seguir, $q_0 \in Q$ é o estado inicial, $Z_0 \in \Gamma$ é o símbolo inicial da pilha, $F \subset Q$ é o conjunto de estados finais. Os conjuntos "^{A}" ("*All*" – para todos) são convenientes porque as *funções adaptativas* podem a) inserir novos estados q, $q \notin Q$ mas $q \in Q^A$, e b) usar novos símbolos de pilha $\gamma \notin \Gamma$ mas $\gamma \in \Gamma^A$. Em resumo, as *funções adaptativas* podem modificar o autômato, mas os novos símbolos que elas introduzem estão todos nos conjuntos "^{A}".

H^0 é o conjunto de todas as funções adaptativas no autômato adaptativo M. Define-se $H^0 = \{f \mid f : E \times G_1 \times G_2 \times \ldots \times G_k \rightarrow E\}$, onde f é uma função, $k \in \mathbb{N}$ é o número de argumentos em f, e $G_i = Q^A \cup \Sigma^A \cup \Gamma^A$.

E é o conjunto de todos os autômatos adaptativos que têm o estado inicial

q_0, o símbolo inicial de pilha Z_0 e o conjunto de estados finais F iguais aos do autômato adaptativo M. Define-se $E = \{N \mid N$ 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 $N \in E$. □

Definição 4.9 (Sub-máquina do Autômato Adaptativo)**.** O conjunto de todas as sub-máquinas do autômato adaptativo M é representado por S. Cada *sub-máquina* s_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 s_i, $\Sigma_i \subseteq \Sigma$ é o alfabeto de entrada da sub-máquina s_i, $P_i \subseteq P$ é o mapeamento da sub-máquina s_i, $q_{i0} \in Q_i$ é o estado inicial da sub-máquina s_i, e $F_i \subseteq Q_i$ é o conjunto de estados finais da sub-máquina s_i. □

Uma transição do Autômato Adaptativo é uma relação da forma $(q, a, \beta) \vdash (q', a', \beta')$ para $P(q, a, \beta) \rightarrow (q', a', \beta', A, B)$, 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 ao autômato corrente, então $P(\tilde{q}, \tilde{a}, \tilde{\beta}) \rightarrow (\tilde{q}, \tilde{a}, \tilde{\beta}, I, I)$, onde I é a função identidade em E.

As chamadas de funções adaptativas associadas a uma transição são funções de $E \times G_1 \times G_2 \times \ldots \times G_k$ em E. Isto é conveniente porque uma função adaptativa pode ser utilizada em mais de uma transição, com argumentos correspondentes a G_i diferentes. Ao definir uma transição $P(q, a, \beta) \rightarrow (q', a', \beta', A, B)$, é necessário fornecer os argumentos correspondentes aos conjuntos G_i. Estes argumentos 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.

Definição 4.10 (Função Adaptativa)**.** 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 $G_i = Q^A \cup \Sigma^A \cup \Gamma^A$, $i, k \in \mathbb{N}$. 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 podem ser definidas a partir de ações adaptativas elementares de consulta, remoção e inclusão, definidas informalmente a seguir.

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 associar nomes únicos 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 de seu conjunto de regras (NETO, 1993). O trecho de autômato da Figura 4.7 será utilizado para exemplificar as ações adaptativas elementares.

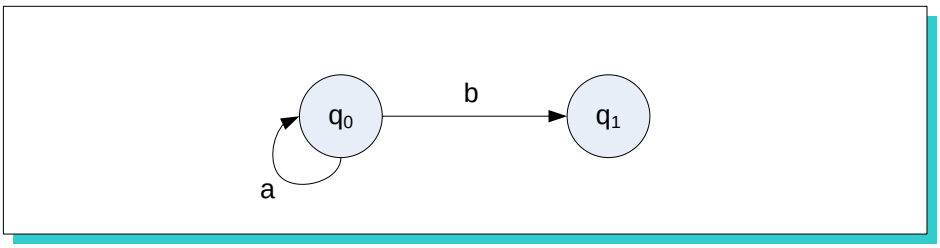

Figura 4.7: Trecho de autômato utilizado para exemplificar as ações adaptativas.

- **Ação adaptativa elementar de consulta**: realiza uma busca no autômato por produções cujos componentes sejam correspondentes aos valores passados a essa ação. É denotada por:

  $?[e, (q, a), A \rightarrow (q', a'), B]$ ou

  $?[e, (q, a, \beta), A \rightarrow (q', a', \beta'), B]$

  onde e é o autômato adaptativo, q é o estado corrente, a é o símbolo a ser consumido, β é o topo da pilha, q' é o estado de destino, a' é o novo símbolo, podendo ser o mesmo símbolo a ser consumido ou vazio ($a' = a$ ou $a' = \epsilon$), β' é o novo topo da pilha, A é a função adaptativa a ser executada antes da transição, e B é a função adaptativa a ser executada após a transição.

As ações de consulta utilizam-se de variáveis para armazenar o resultado destas consultas. Assim, as variáveis armazenam um conjunto de estados ou transições que correspondem aos parâmetros consultados. Admitindo o trecho de autômato da Figura 4.7, e utilizando as variáveis $?x$ e $?y$, a Tabela 4.8 exemplifica as ações adaptativas elementares de consulta.

Tabela 4.8: Exemplos de ações de consulta.

Consulta	**Significado**	**Resultado**
$?[e, (q_0, ?x) \rightarrow (q_1, \epsilon)]$	Quais são os símbolos que, a partir do estado q_0, levam ao estado q_1?	$?x = \{b\}$
$?[e, (q_0, b) \rightarrow (?x, \epsilon)]$	A partir do estado q_0, qual é o estado de destino, consumindo o símbolo b?	$?x = \{q_1\}$
$?[e, (q_0, ?x) \rightarrow (?y, \epsilon)]$	A partir do estado inicial q_0, consumindo qualquer símbolo, quais estados de destino possíveis existem?	$?x = \{a, b\}$, $?y = \{q_0, q_1\}$

Observe que, na terceira consulta, $?x$ e $?y$ contêm todos os símbolos possíveis, embora seja observado que só será possível obter os pares $(?x = a, ?y = q_0)$ e $(?x = b, ?y = q_1)$.

- **Ação adaptativa elementar de remoção**: remove uma produção de acordo com os valores passados a essa ação. É denotada por:

  $-[e, (q, a), A \rightarrow (q', a'), B]$ ou

  $-[e, (q, a, \beta), A \rightarrow (q', a', \beta'), B]$

  onde e é o autômato adaptativo, q é o estado corrente, a é o símbolo a ser consumido, β é o topo da pilha, q' é o estado de destino, a' é o novo símbolo podendo ser o mesmo símbolo a ser consumido ou vazio ($a' = a$ ou $a' = \epsilon$), β' é o novo topo da pilha, A é a função adaptativa a ser executada antes da transição, e B é a função adaptativa a ser executada após a transição.

  A ação adaptativa de remoção exclui elementos do autômato que correspondam aos parâmetros informados. Por exemplo, supondo o o trecho de autômato da Figura 4.7, a Tabela 4.9 exemplifica a execução da ação adaptativa

de remoção.

Tabela 4.9: Exemplo de ação de remoção.

Remoção	Significado
$-[e, (q_0, a) \rightarrow (q_0, \epsilon)]$	Remova a transição que parte do estado q_0, consumindo o símbolo a, e que leva ao próprio estado q_0

O trecho de autômato modificado por essa ação de remoção é ilustrado na Figura 4.8.

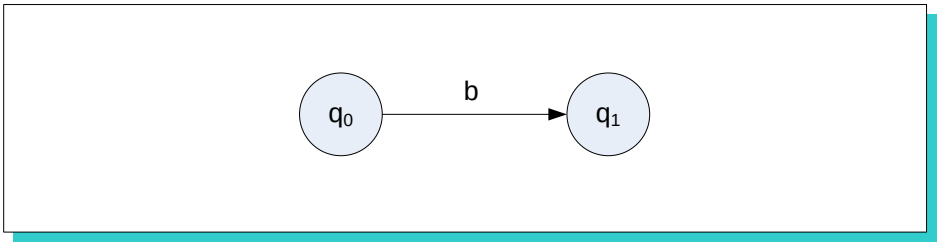

Figura 4.8: Trecho de autômato da Figura 4.7, modificado pela ação de remoção da Tabela 4.9.

- **Ação adaptativa elementar de inclusão**: inclui uma produção de acordo com os valores passados a essa ação. É denotada por:

  $+[e, (q, a), A \rightarrow (q', a'), B]$ ou

  $+[e, (q, a, \beta), A \rightarrow (q', a', \beta'), B]$

  onde e é o autômato adaptativo, q é o estado corrente, a é o símbolo a ser consumido, β é o topo da pilha, q' é o estado de destino, a' é o novo símbolo podendo ser o mesmo símbolo a ser consumido ou vazio ($a' = a$ ou $a' = \epsilon$), β' é o novo topo da pilha, A é a função adaptativa a ser executada antes da transição, e B é a função adaptativa a ser executada após a transição.

  A ação elementar de inclusão insere elementos novos no autômato, como transições e estados novos, de acordo com os argumentos fornecidos. Para a criação de um estado novo, utiliza-se um gerador, o qual receberá um identificador único que será o nome desse estado recém-criado. Por exemplo, supondo

o trecho de autômato da Figura 4.7, e utilizando um gerador g_1^*, a Tabela 4.10 ilustra as execuções de ações adaptativas de inclusão.

Tabela 4.10: Exemplo de ações de inclusão.

Inclusão	**Significado**
$+[e, (q_1, b) \rightarrow (q_1, \epsilon)]$	Insira uma transição que parta do estado q_1, consumindo o símbolo b, e que leve ao próprio estado q_1 (*loop*)
$+[e, (q_1, c) \rightarrow (g_1^*, \epsilon)]$	Insira uma transição que parta do estado q_1, consumindo o símbolo c, e que leve a um novo estado criado g_1^*

O trecho de autômato, modificado pelas ações de inclusão, é ilustrado na Figura 4.9.

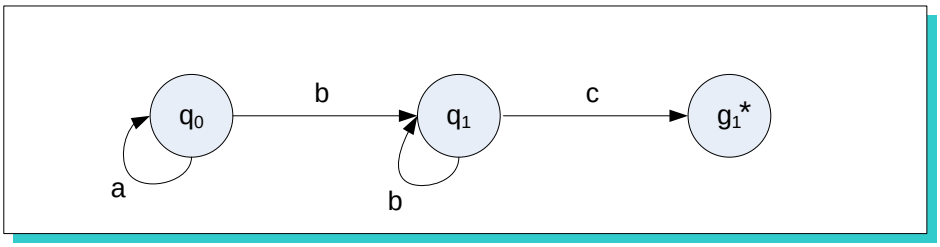

Figura 4.9: Trecho de autômato da Figura 4.7, modificado pelas ações de inclusão da Tabela 4.10.

Graficamente, é possível representar se uma função adaptativa será executada antes ou depois da transição, através da notação "**.**" apresentada na Figura 4.10. Em *(a)*, a função adaptativa A é executada antes da transição, enquanto que em *(b)* a função adaptativa B é executada após a transição. Em *(c)*, as duas funções são executadas, sendo A antes da transição, e B após.

Uma gramática que define a sintaxe de uma função adaptativa é apresentada no Código 4.4.1.

A relação de transição $\vdash$ entre configurações do Autômato Adaptativo é similar à empregada no Autômato de Pilha Estruturado, sendo que aqui deve ser considerada a presença das funções adaptativas.

A linguagem aceita por um autômato adaptativo M é dada por $L(M) = \{w \in$

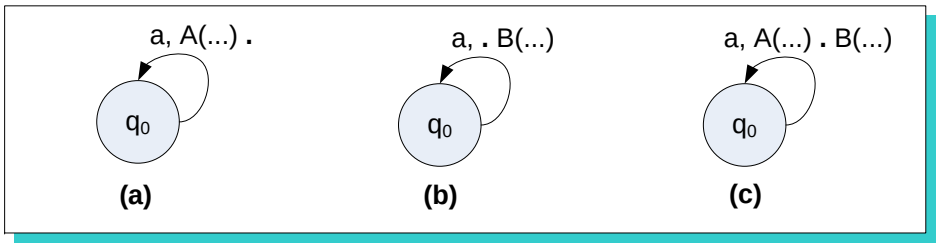

Figura 4.10: Notação gráfica para determinar o momento de execução das funções adaptativas.

Gramática *Função adaptativa*

Program ::= Id "(" Aa ParamList ")" "{" [VarDecList] StatementList "}"
ParamList ::= | "," Id ParamList
VarDecList ::= Var { "," Var } ":"
/ uma variável é uma variável normal ou gerador */*
Var ::= NormalVar | NewVar
/ identificador referente ao autômato adaptativo */*
Aa ::= Id
NormalVar ::= ?Id
NewVar ::= Id*
/ O valor pode ser um identificador, uma variável ou ϵ */*
Value ::= Id | Var | "ϵ"
StatementList ::= | Statement StatementList
Statement ::= Consulta | Remoção | Inclusão | returnStat
Consulta ::= "?" "[" Aa "," "(" Value "," Value ["," Value] ")" ["," Value] "$\rightarrow$"
"(" Value "," Value ["," Value] ")" ["," Value] "]"
Inclusão ::= Aa "=" "+" "[" Aa "," "(" Value "," Value ["," Value] ")" ["," Value]
"$\rightarrow$" "(" Value "," Value ["," Value] ")" ["," Value] "]"
Remoção ::= Aa "=" "–" "[" Aa "," "(" Value "," Value ["," Value] ")" ["," Value]
"$\rightarrow$" "(" Value "," Value ["," Value] ")" ["," Value] "]"
returnStat ::= "return" Aa

Código 4.4.1: Gramática de uma função adaptativa.

$\Sigma^*|(q_0, w, Z_0) \vdash^* (q_f, \epsilon, Z_0)$, onde $q_f \in F\}$.

Exemplo 4.11. Considere um autômato adaptativo M, responsável pelo reconhecimento de cadeias que pertençam à linguagem $L_4 = \{w \in \{a, b, c\}^* \mid w$ é da forma $a^n b^n c^n$, $n \geq 1\}$, ou seja, w é uma cadeia que contém um número de a's, seguido do mesmo número de b's e número de c's. O mapeamento P é definido na Tabela 4.11, onde I representa a função identidade.

O autômato adaptativo M é ilustrado na Figura 4.11.

Tabela 4.11: Mapeamento P do autômato adaptativo do exemplo 4.11.

Mapeamento P	
$P(q_0, a, Z_0) \rightarrow (q_1, \epsilon, Z_0, I, I)$	$P(q_1, a, Z_0) \rightarrow (q_1, \epsilon, Z_0, I, A(M, q_2, q_3))$
$P(q_1, b, Z_0) \rightarrow (q_2, \epsilon, Z_0, I, I)$	$P(q_2, c, Z_0) \rightarrow (q_3, \epsilon, Z_0, I, I)$

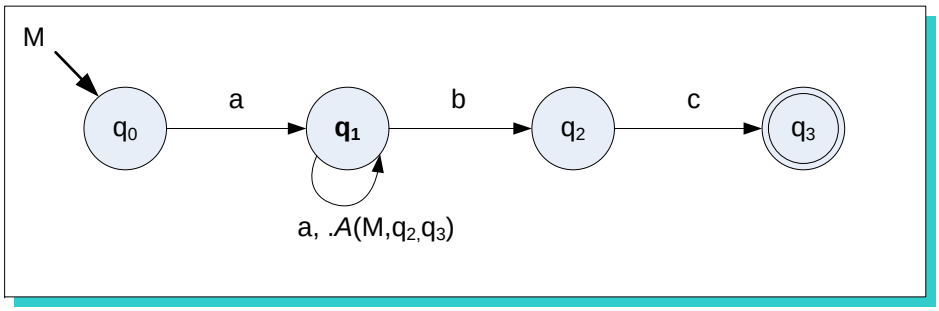

Figura 4.11: Exemplo de um autômato adaptativo.

O autômato adaptativo da Figura 4.11 utiliza uma função $A(e, p_1, p_2)$, definida no Código 4.4.2.

Supondo a cadeia $aabbcc$, deseja-se saber se esta pertence à linguagem L_4. A Tabela 4.12 mostra a seqüência de transições para o reconhecimento dessa cadeia.

Tabela 4.12: Seqüência de transições para o reconhecimento da cadeia $aabbcc$.

Origem	**Transição**	**Destino**
q_0	$(q_0, aabbcc, Z_0) \vdash_{P(q_0,a,Z_0)\rightarrow(q_1,\epsilon,Z_0,I,I)} (q_1, abbcc, Z_0)$	q_1
q_1	$(q_1, abbcc, Z_0) \vdash_{P(q_1,a,Z_0)\rightarrow(q_1,\epsilon,Z_0,I,A(M,q_2,q_3))} (q_1, bbcc, Z_0)$	q_1
q_1	$(q_1, bbcc, Z_0) \vdash_{P(q_1,b,Z_0)\rightarrow(q_{100},\epsilon,Z_0,I,I)} (q_{100}, bcc, Z_0)$	q_{100}
q_{100}	$(q_{100}, bcc, Z_0) \vdash_{P(q_{100},b,Z_0)\rightarrow(q_2,\epsilon,Z_0,I,I)} (q_2, cc, Z_0)$	q_2
q_2	$(q_2, cc, Z_0) \vdash_{P(q_2,c,Z_0)\rightarrow(q_{101},\epsilon,Z_0,I,I)} (q_{101}, c, Z_0)$	q_{101}
q_{101}	$(q_{101}, c, Z_0) \vdash_{P(q_{101},c,Z_0)\rightarrow(q_3,\epsilon,Z_0,I,I)} (q_3, \epsilon, Z_0)$	q_3

A configuração final do autômato para o reconhecimento da cadeia $aabbcc$ pode ser vista na Figura 4.12.

A função adaptativa $A(M, q_2, q_3)$ alterou o mapeamento P de acordo com a Ta-

Função adaptativa $A(e, p_1, p_2)$

$A(e, p_1, p_2) = \{$
$?x,\ ?y,\ g_1^*,\ g_2^*$:

/ Procura pelo estado que possui uma transição consumindo o símbolo* b *até* p_1 *e remove essa transição */*
$?[e, (?x, b) \rightarrow (p_1, \epsilon)] \qquad e = -[e, (?x, b) \rightarrow (p_1, \epsilon)]$

/ Procura pelo estado que possui uma transição consumindo o símbolo* c *até* p_2 *e remove essa transição */*
$?[e, (?y, c) \rightarrow (p_2, \epsilon)] \qquad e = -[e, (?y, c) \rightarrow (p_2, \epsilon)]$

/ Remove a transição de* q_1 **/*
$e = -[e, (q_1, a) \rightarrow (q_1, \epsilon), A(e, p_1, p_2)]$

/ Insere as novas transições */*
$e = +[e, (?x, b) \rightarrow (g_1^*, \epsilon)] \qquad e = +[e, (g_1^*, b) \rightarrow (p_1, \epsilon)]$
$e = +[e, (?y, c) \rightarrow (g_2^*, \epsilon)] \qquad e = +[e, (g_2^*, c) \rightarrow (p_2, \epsilon)]$

/ Insere uma nova transição em* q_1 **/*
$e = +[e, (q_1, a) \rightarrow (q_1, \epsilon), A(g_1, g_2)]$

return e
$\}$

Código 4.4.2: Função adaptativa $A(e, p_1, p_2)$ do exemplo 4.11.

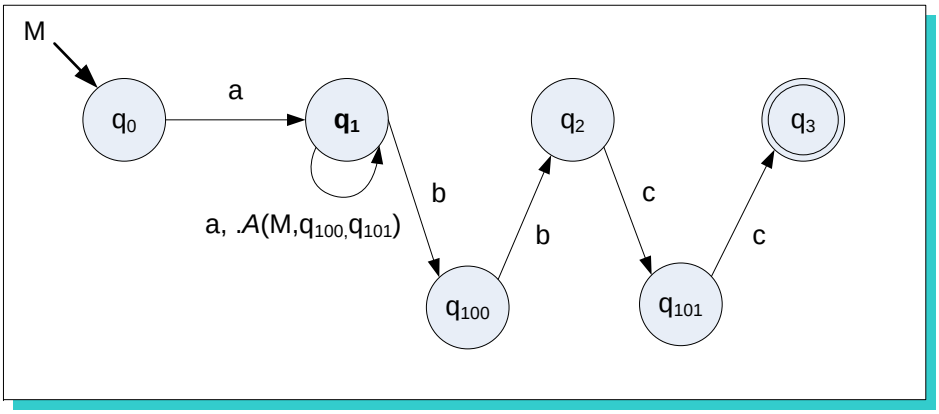

Figura 4.12: Configuração final do autômato adaptativo para o reconhecimento da cadeia $aabbcc$.

bela 4.13.

O estado q_3 pertence ao conjunto de estados de aceitação, logo, a cadeia é aceita

Tabela 4.13: Novo mapeamento P do autômato adaptativo do exemplo 4.11.

Mapeamento P	
$P(q_0, a, Z_0) \rightarrow (q_1, \epsilon, Z_0, I, I)$	$P(q_1, a, Z_0) \rightarrow (q_1, \epsilon, Z_0, I, A(M, q_{100}, q_{101}))$
$P(q_1, b, Z_0) \rightarrow (q_{100}, \epsilon, Z_0, I, I)$	$P(q_{100}, b, Z_0) \rightarrow (q_2, \epsilon, Z_0, I, I)$
$P(q_2, c, Z_0) \rightarrow (q_{101}, \epsilon, Z_0, I, I)$	$P(q_{101}, c, Z_0) \rightarrow (q_3, \epsilon, Z_0, I, I)$

e pertence à linguagem L_4. □

Resumo do Capítulo 4

Este Capítulo abordou os conceitos básicos da teoria de autômatos. Foram apresentados e definidos alguns dispostivos reconhecedores, como o autômato finito, o autômato de pilha, o autômato de pilha estruturado – uma variação do autômato de pilha tradicional, constituindo a base do autômato adaptativo – e o autômato adaptativo. A teoria de autômatos é uma área da computação teórica amplamente utilizada para representações formais.

No próximo Capítulo, será apresentado um modelo de controle de acesso baseado em autômatos adaptativos chamado *Modelo de Controle de Acesso Adaptativo* (MCAA).

CAPÍTULO

5

Modelo de Controle de Acesso Adaptativo

"Intelligence is the ability to adapt to change."

STEPHEN WILLIAM HAWKING

Muitos modelos de controle de acesso na literatura são amplamente utilizados, mas alguns deles possuem aplicação apenas em cenários específicos. Por exemplo, *Partition Rule Based Access Control* (PRBAC) (DAFA-ALLA et al., 2005) é utilizado quando uma política é definida para um domínio inteiro ou uma partição; por outro lado, *Local Rule Based Access Control* (LRBAC) (RAY et al., 2006) é útil em situações em que é desejável isolar pequenos grupos locais. Para obter uma maior abrangência de aplicabilidade, um modelo de controle de acesso deve permitir mudanças em suas características principais (política de acesso), se necessário. Tal possibilidade de alterações no modelo foi uma das razões para que o autômato adaptativo fosse

considerado para representar um modelo de controle de acesso.

Neste Capítulo, é apresentado o modelo de controle de acesso chamado *Modelo de Controle de Acesso Adaptativo* (MCAA). Este modelo utiliza um *autômato adaptativo de controle de acesso* para realizar o controle de acesso, e possui *comandos de privacidade* para codificar um conjunto de regras obtidas a partir de documentos de políticas de privacidade ou legislações. O MCAA é simples e genérico o suficiente para atender os requisitos de segurança e privacidade de um sistema.

5.1 Autômato Adaptativo de Controle de Acesso

Uma *Máquina de Estados Finitos* (MEF) pode ser utilizada para controlar o acesso a um dado recurso, permitindo o acesso ao aceitar uma cadeia que denota uma solicitação. Por exemplo, se a MEF aceita a cadeia $s_i o_j r$, isto seria interpretado da seguinte forma: o indivíduo s_i tem direito de leitura (r) para acessar o recurso o_j. Uma modelagem de controle de acesso pode ser feita por uma MEF se as políticas de acesso não variam com o tempo, mas isso não é usual. Quando há necessidade de mudança nas políticas, a MEF também deve ser alterada, e dessa forma, deve ser utilizado um outro tipo de dispositivo que atenda tais requisitos. O autômato adaptativo pode ser visto como uma extensão de uma MEF, pois utiliza um autômato de pilha estruturado com capacidade de modificação. É um modelo proposto na literatura e tem poder computacional de uma Máquina de Turing (ROCHA; NETO, 2001). O *Autômato Adaptativo de Controle de Acesso* (AACA) (CEREDA; ZORZO, 2008a), definido a seguir, é uma extensão do Autômato Adaptativo (NETO, 1993) (NETO, 1994), e foi proposto visando atender as necessidades de abrangência desejada para as finalidades deste trabalho.

Definição 5.1 (Autômato Adaptativo de Controle de Acesso)**.** Um *Autômato Adaptativo de Controle de Acesso* M é definido como $M = (Q, S, \Sigma, \Gamma, P, q_0,\ Z_0, F)$, tal que Q é o conjunto finito de estados, S é o conjunto de sub-máquinas, Σ é o alfabeto de entrada, Γ é o alfabeto de pilha, $\Gamma = Q \cup \{Z_0\}$, P é mapeamento

$P : Q^A \times (\Sigma^A \cup \{\epsilon\})^2 \times \Gamma^A \rightarrow Q^A \times (\Sigma^A \cup \{\epsilon\}) \times (\Gamma^A \cup \{\epsilon\}) \times H^0 \times H^0$, H^0 definido a seguir, $q_0 \in Q$ é o estado inicial, $Z_0 \in \Gamma$ é o símbolo inicial da pilha, $F \subset Q$ é o conjunto de estados finais. Os conjuntos Q^A, Σ^A, e Γ^A são todos enumeráveis, $Q \subset Q^A$, $\Sigma \subset \Sigma^A$, $\Gamma \subset \Gamma^A$, e $\Gamma^A = Q^A \cup \{Z_0\}$. Q^A é o conjunto de todos os estados possíveis, Σ^A é o alfabeto de todos os símbolos possíveis, Γ^A é o conjunto de todos os símbolos possíveis de pilha, $\Gamma^A = Q^A \cup \{Z_0\}$. Estes conjuntos "^{A}" ("*All*" – para todos) são necessários porque as *funções adaptativas* podem a) inserir novos estados q, $q \notin Q$ mas $q \in Q^A$, b) usar novos símbolos de pilha $\gamma \notin \Gamma$ mas $\gamma \in \Gamma^A$, e c) reconhecer novos símbolos de entrada s tal que $s \notin \Sigma$ mas $s \in \Sigma^A$. Em resumo, as *funções adaptativas* podem modificar o autômato, mas os novos símbolos que elas introduzem estão todos nos conjuntos "^{A}".

H^0 é o conjunto $\{f \mid f : E \times G_1 \times G_2 \times \ldots \times G_k \rightarrow E\}$, onde f é uma função, $k \in \mathbb{N}$ é o número de argumentos em f, e $G_i = Q^A \cup \Sigma^A \cup \Gamma^A$. E é $\{N \mid N$ é um autômato adaptativo de controle de acesso $N = (Q', \Sigma', \Gamma', P', q_0, Z_0, F)$, onde $Q' \subset Q^A$, $\Sigma' \subset \Sigma^A$, $\Gamma' \subset \Gamma^A$, $P' : Q^A \times (\Sigma^A \cup \{\epsilon\})^2 \times \Gamma^A \rightarrow Q^A \times (\Sigma^A \cup \{\epsilon\}) \times (\Gamma^A \cup \{\epsilon\}) \times H^0 \times H^0\}$. Observe que q_0, Z_0 e F são os mesmos para qualquer $N \in E$. □

Definição 5.2 (Sub-máquina do Autômato Adaptativo de Controle de Acesso)**.** S é o conjunto de sub-máquinas do autômato adaptativo de controle de acesso. Cada *sub-máquina* $s_i \in S$ 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, $\Sigma_i \subseteq \Sigma$ é o conjunto de símbolos de entrada, $P_i \subseteq P$ é o mapeamento, $q_{i0} \in Q_i$ é o estado de entrada, e $F_i \subseteq Q_i$ é o conjunto de estados finais. Se $s_i, s_j \in S$, $i \neq j$, $s_i = (Q_i, \Sigma_i, P_i, q_{i0}, F_i)$, $s_j = (Q_j, \Sigma_j, P_j, q_{j0}, F_j)$, então $Q_i \cap Q_j = \varnothing$. Assim, $P_i \cap P_j = \varnothing$. □

Utiliza-se $\vdash$ para representar uma transição, que é uma aplicação do mapeamento P. Foi utilizado $(q, w, \beta) \vdash (q', a', \beta')$ para $P(q, w, \beta) \rightarrow (q', a', \beta', A, B)$ onde w é uma cadeia com um ou dois símbolos de Σ. Se $\tilde{q}$, $\tilde{a}$, $\tilde{b}$ ou $\tilde{\beta}$ não pertencem à máquina atual, então $P(\tilde{q}, \tilde{a}, \tilde{\beta}) \rightarrow (\tilde{q}, \tilde{a}, \tilde{\beta}, I, I)$ ou $P(\tilde{q}, \tilde{a}\tilde{b}, \tilde{\beta}) \rightarrow (\tilde{q}, \tilde{a}\tilde{b}, \tilde{\beta}, I, I)$, onde I é a função identidade em E.

Graficamente, uma transição que utiliza um símbolo da cadeia de entrada como argumento de uma função adaptativa pode ser representada como na Figura 5.1, onde q_j é o estado de origem, q_k é o estado de destino, a é o símbolo a ser consumido, B é a função adaptativa a ser executada após o consumo do símbolo a, e e é o autômato adaptativo. A notação "•" indica que o próximo símbolo na cadeia de entrada será passado como argumento para a função adaptativa B, na posição indicada por "•".

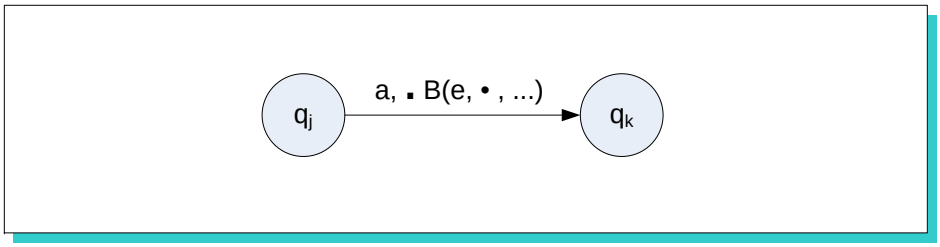

Figura 5.1: Notação gráfica de uma transição que utiliza um símbolo da cadeia de entrada como argumento de uma função adaptativa.

Uma transição convencional (que não passa um símbolo da cadeia de entrada a uma função adaptativa) ocorre quando o estado atual é q, a cadeia de entrada é aw, e existe um mapeamento $P(q, aw, \beta) \rightarrow (q', w, \beta, A, B)$, A e B são funções adaptativas, ambas opcionais; A é executada antes B é executada depois que a transição é disparada.

Se o estado atual é q, a cadeia é aw, e existe um mapeamento $P(q, aw, \beta) \rightarrow (q_{i0}, aw, \beta')$, com $\beta' \neq \epsilon$, então existe uma chamada para a sub-máquina $s_i = (Q_i, \Sigma_i, P_i, q_{i0}, F_i)$. Existe apenas uma sub-máquina s_i onde $q_{i0} \in Q_i$. Se $\beta' = q''$, então q'' é o estado para continuar o reconhecimento após o retorno da sub-máquina s_i.

O retorno de uma sub-máquina ocorre quando o estado corrente está no conjunto de estados finais dessa sub-máquina, não existe transição a ser aplicada e existe um estado na pilha. Formalmente, suponha que o estado corrente seja q e que a cadeia de entrada seja aw, onde $q \in Q_i$ e $s_i = (Q_i, \Sigma_i, P_i, q_{i0}, F_i)$. Se não existe transição $P(q, aw, \beta)$, $q \in F_i$, e existe um elemento $q' \in Q$ no topo da pilha, $q' \neq Z_0$,

então q' é removido da pilha e o estado corrente se torna q'. Note que o alfabeto de pilha Γ é $Q \cup \{Z_0\}$. O símbolo Z_0 está sempre no início da pilha. Se não existe outro símbolo na pilha, então nenhuma sub-máquina foi chamada.

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

5.2 Modelo de Controle de Acesso Adaptativo

O *Modelo de Controle de Acesso Adaptativo* (MCAA) é um modelo de controle de acesso que utiliza um *Autômato Adaptativo de Controle de Acesso*. Um modelo de controle de acesso verifica como os usuários, chamados *indivíduos* (ou *elementos ativos*), utilizam os recursos, chamados *objetos* (ou *elementos passivos*). O relacionamento *indivíduo-objeto* é caracterizado por alguns *direitos*. Por exemplo, um usuário da rede pode ter o direito de leitura (r) e escrita (w) em um arquivo o_1. O usuário é o *indivíduo*, o *objeto* é o arquivo o_1, e os *direitos* são rw.

Inicialmente, o autômato M do MCAA é definido com uma configuração que reflete o controle de acesso atual. Durante seu tempo de vida, M sofre alterações em sua topologia para receber novos indivíduos, objetos, direitos e relacionamentos entre indivíduos e objetos. Isto é feito através das *funções adaptativas* de M, que podem alterar o autômato. Essa natureza dinâmica é a razão pelo qual o MCAA foi utilizado ao invés de uma MEF.

Suponha que o MCAA possua um conjunto $S = \{s_1 \dots s_{n_1}\}$ de indivíduos, um conjunto $O = \{o_1 \dots o_{n_2}\}$ de objetos, e um conjunto $R = \{r_1 \dots r_{n_3}\}$ de direitos. Assim, o autômato M usado pelo MCAA é projetado por um *administrador do sistema* para usar um alfabeto $\Sigma = S \cup O \cup R \cup \{\lambda_{o_1} \dots \lambda_{o_{n_2}}\} \cup \lambda_{change}$, onde $\lambda_{change} = \{\lambda_{NS}, \lambda_{NO}, \lambda_{RS}, \lambda_{RO}, \lambda_{RAO}, G, T\}$. O conjunto Σ deve ser um subconjunto de $\Sigma^A = \{w \mid w$ é um nome de indivíduo válido$\} \cup \{x \mid x$ é um nome de objeto válido$\} \cup \{y \mid y$ é um direito válido$\} \cup \{\lambda_x \mid x$ é um nome de objeto válido$\} \cup \lambda_{change}$. Os conjuntos S, O, e R são todos finitos. O conjunto $\{\lambda_{o_1} \dots \lambda_{o_{n_2}}\}$ é um conjunto auxiliar para

facilitar a remoção de um objeto e para inserir e remover direitos. Por exemplo, λ_{o_1} pode ser o rótulo de uma transição que leva a um estado que permite a remoção do objeto o_1 e inserir ou remover direitos em o_1. O conjunto λ_{change} representa as mudanças a serem realizadas na configuração do autômato.

Uma consulta sobre o relacionamento *indivíduo-objeto* pode ser submetida ao autômato M. Se M aceitá-la, o relacionamento é válido. Por exemplo, suponha que a cadeia de entrada seja $u = s_i o_j r w$. Se o autômato aceitar u, então isso significa que o indivíduo s_i tem direitos de leitura (r) e escrita (w) sobre o objeto o_j. A aceitação de uma cadeia u pelo autômato M, $u \in L(M)$, indica que a) a consulta está em conformidade com a configuração atual do sistema, ou que b) a operação realizada no autômato foi bem-sucedida. O item b) refere-se às cadeias de entrada que modificam M; essa modificação é feita toda vez que uma cadeia de entrada possuir um dos símbolos apresentados na Tabela 5.1 (conjunto λ_{change}). Por exemplo, se a cadeia de entrada é $s_1 \lambda_{NO} o_1$, então o objeto o_1 é criado e associado ao indivíduo s_1. Essa operação é realizada através de uma função adaptativa. Em cada transição que utiliza um dos símbolos do conjunto λ_{change}, existe uma função adaptativa que altera o autômato M (os símbolos G e T são uma exceção, pois nesse caso as funções adaptativas estarão associadas às transições seguintes).

Tabela 5.1: Descrição dos símbolos (conjunto λ_{change}) que representam alterações a serem realizadas na configuração do autômato.

Símbolo	**Descrição**
λ_{NS}	Novo indivíduo
λ_{NO}	Novo objeto
λ_{RS}	Remover indivíduo
λ_{RO}	Remover objeto
λ_{RAO}	Remover todos os objetos
G	Inserir direito
T	Remover direito

A Tabela 5.2 apresenta os formatos das cadeias aceitas pelo autômato M do MCAA. Uma cadeia de entrada que possui um dos símbolos da Tabela 5.1 deve seguir a sintaxe apresentada na Tabela 5.2; cadeias que não seguem tal sintaxe,

como $s_1 o_1 \lambda_{NO}$, não serão aceitas.

Tabela 5.2: Cadeias que são reconhecidas pelo autômato adaptativo de controle de acesso M do MCAA.

Cadeia	**Descrição**
$\lambda_{NS} a$	criação do indivíduo a
$s_i \lambda_{NO} b$	criação do objeto b associado ao indivíduo s_i
$s_i \lambda_{RS}$	remove do indivíduo s_i
$s_i \lambda_b \lambda_{RO}$	remove do objeto b associado ao indivíduo s_i
$\lambda_{RAO} b$	remove do objeto b do sistema
$s_i \lambda_b G r_k$	insere o direto r_k para o indivíduo s_i sobre o objeto b
$s_i \lambda_b T r_k$	remove o direito r_k para o indivíduo s_i sobre o objeto b
$s_i o_j$	verifica se o indivíduo s_i é relacionado com o objeto o_j
$s_i o_j r_k$	verifica se o indivíduo s_i tem direito r_k sobre o objeto o_j
$s_i o_j r_1 \dots r_k$	verifica se o indivíduo s_i tem direitos $r_1 \dots r_k$ sobre o objeto o_j

Exemplo 5.3. Seja M um AACA, onde o indivíduo *alice* tem direito de *leitura* sobre o objeto *file1* (Figura 5.2). O autômato M aceita cadeias de entrada que são verificações de acesso e também cadeias que modificam o autômato. Como exemplo deste segundo caso, pode-se criar o indivíduo *bob* submetendo a cadeia $\lambda_{NS}\langle bob \rangle$ ao autômato M. A transição rotulada por λ_{NS}, de q_0 a q_f, será disparada e a função adaptativa $F_{NS}(M,bob)$ será chamada, inserindo o indivíduo *bob* em M.

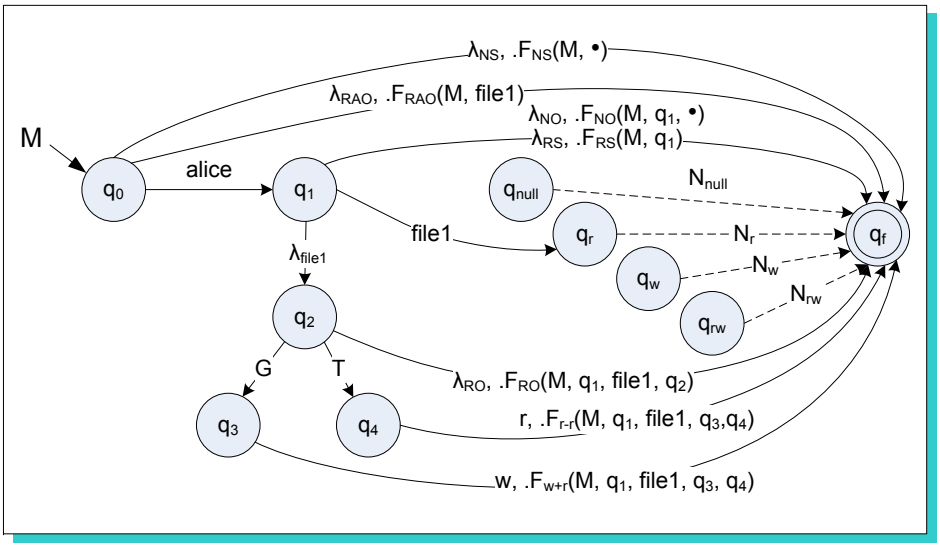

Figura 5.2: Exemplo de um autômato adaptativo de controle de acesso.

O AACA M possui um conjunto de quatro sub-máquinas, N_{null}, N_r, N_w e N_{rw}. Cada sub-máquina é responsável pelo reconhecimento dos direitos de um indivíduo s_i sobre um objeto o_j. Ela é chamada recursivamente para todo direito a ser verificado na cadeia e, se todos os direitos estiverem presentes, ela retornará cada chamada até chegar no estado final q_f. A Figura 5.3 ilustra a sub-máquina N_r, responsável por reconhecer o direito de *leitura*.

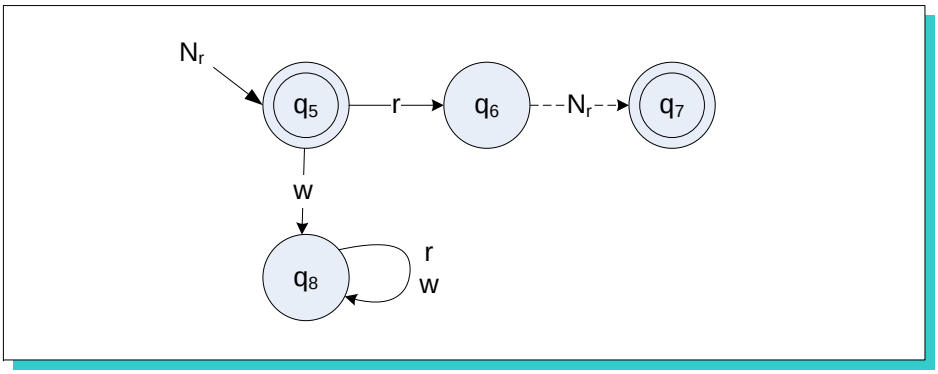

Figura 5.3: Sub-máquina N_r do exemplo 5.3.

As funções adaptativas de criação e remoção de indivíduos e objetos do exemplo 5.3 são apresentadas a seguir:

1. $F_{NS}(e, p_1)$: realiza a criação de um novo indivíduo, tomando como argumentos o autômato e o nome do indivíduo (por exemplo, *bob*). Definida no Código 5.2.1.

2. $F_{RS}(e, p_1)$: remove o indivíduo do autômato, tomando como argumentos o autômato e o estado referente ao indivíduo. Descrita no Código 5.2.2.

3. $F_{NO}(e, p_1, p_2)$: realiza a criação de um novo objeto associado a um indivíduo, tomando como argumentos o autômato, o estado referente ao indivíduo e nome do objeto (por exemplo, *file2*). Descrita no Código 5.2.3.

4. $F_{RO}(e, p_1, p_2, p_3)$: remove o objeto associado a um indivíduo, tomando como argumentos o autômato, o estado referente ao indivíduo, o nome do objeto e o

Função adaptativa $F_{NS}(e, p_1)$

$F_{NS}(e, p_1) = \{$
$g_1^* :$

/ Insere uma transição para o novo indivíduo* p_1 **/*
$e = +[e, (q_0, p_1) \rightarrow (g_1^*, \epsilon)]$

/ Insere uma transição para remover o indivíduo* p_1 **/*
$e = +[e, [g_1^*, \lambda_{RS}) \rightarrow (q_f, \epsilon), F_{RS}(e, g_1^*)]$

/ Insere uma transição para criar um objeto associado ao indivíduo* p_1 **/*
$e = +[e, (g_1^*, \lambda_{NO}) \rightarrow (q_f, \epsilon), F_{NO}(e, g_1^*, \bullet)]$

return e
$\}$

Código 5.2.1: Função adaptativa $F_{NS}(e, p_1)$ do exemplo 5.3.

Função adaptativa $F_{RS}(e, p_1)$

$F_{RS}(e, p_1) = \{$
$?x, ?y, ?z :$

/ Remove a transição de* q_0 *até* p_1 **/*
$?[e, (q_0, ?x) \rightarrow (p_1, \epsilon)]$
$e = -[e, (q_0, ?x) \rightarrow (p_1, \epsilon)]$

/ Remove as transições que partem de* p_1 **/*
$?[e, (p_1, ?y) \rightarrow (?z, \epsilon)]$
$e = -[e, (p_1, ?y) \rightarrow (?z, \epsilon)]$

return e
$\}$

Código 5.2.2: Função adaptativa $F_{RS}(e, p_1)$ do exemplo 5.3.

Função adaptativa $F_{NO}(e, p_1, p_2)$

$F_{NO}(e, p_1, p_2) = \{$
g_1^*, g_2^*, g_3^* :

/ Insere uma transição de* p_1 *até o estado* q_{null} **/*
$e = +[e, (p_1, p_2) \rightarrow (q_{null}, \epsilon)]$

/ Insere uma transição para remover o objeto */*
$e = +[e, (g_1^*, \lambda_{RO}) \rightarrow (q_f, \epsilon), F_{RO}(e, p_1, p_2, g_1^*)]$

/ Insere uma transição de* p_1 *até o estado auxiliar* g_1^* *e insere duas transições: de* g_1^* *até* g_2^* *(inserir direitos) e de* g_1^* *até* g_3^* *(remover direitos) */*
$e = +[e, (p_1, \lambda_{p_2}) \rightarrow (g_1^*, \epsilon)]$
$e = +[e, (g_1^*, G) \rightarrow (g_2^*, \epsilon)]$
$e = +[e, (g_1^*, T) \rightarrow (g_3^*, \epsilon)]$

/ Insere uma transição de* g_2^* *até* q_f *(inserir direito de escrita) e insere uma transição de* g_3^* *até* q_f *(remover direito de leitura) */*
$e = +[e, (g_2^*, w) \rightarrow (q_f, \epsilon), F_{w+r}(e, p_1, p_2, g_2^*, g_3^*)]$
$e = +[e, (g_3^*, r) \rightarrow (q_f, \epsilon), F_{r-r}(e, p_1, p_2, g_2^*, g_3^*)]$

return e
$\}$

Código 5.2.3: Função adaptativa $F_{NO}(e, p_1, p_2)$ do exemplo 5.3.

estado de propriedade do objeto (o estado de destino da transição com rótulo λ_{o_i}). Definida no Código 5.2.4.

5. $F_{RAO}(e, p_1)$: remove o objeto do sistema, tomando como argumentos o autômato e o nome do objeto (por exemplo, *file1*). É definida no Código 5.2.5.

Além das funções adaptativas de criação e remoção de indivíduos e objetos, o autômato do exemplo 5.3 possui duas funções adaptativas responsáveis pelo gerenciamento de direitos. Dependendo dos direitos associados a cada relacionamento *indivíduo-objeto*, essas funções são substituídas por outras, conforme as mudanças na configuração do controle de acesso. As duas funções adaptativas que gerenciam os direitos de *alice* sobre *file1* são apresentadas a seguir.

1. $F_{w+r}(e, p_1, p_2, p_3, p_4)$: adiciona o direito de escrita (w) em um relacionamento entre indivíduo e objeto, quando este já possui o direito de leitura (r). A função adaptativa toma como argumentos o autômato, o estado referente ao indivíduo, o nome do objeto, o estado de inclusão de direitos e o estado de remoção de direitos. É definida no Código 5.2.6.

2. $F_{r-r}(e, p_1, p_2, p_3, p_4)$: remove o direito de leitura (r) em um relacionamento entre indivíduo e objeto, quando este possui o direito de leitura (r). A função adaptativa toma como argumentos o autômato, o estado referente ao indivíduo, o nome do objeto, o estado de inclusão de direitos e o estado de remoção de direitos. É definida no Código 5.2.7.

Como exemplo de consulta ao autômato M, deseja-se saber se *alice* pode ler *file1*. Esta consulta é realizada submetendo-se a cadeia $u = \langle$*alice*$\rangle\langle$*file1*$\rangle\langle r\rangle$ a M. A Tabela 5.3 mostra a seqüência de transições para o reconhecimento dessa cadeia.

O estado q_f pertence ao conjunto de estados de aceitação e, portanto, a cadeia u é aceita pelo autômato, $u \in L(M)$. Isto significa que o indivíduo *alice* pode ler (r) o objeto *file1*. □

Função adaptativa $F_{RO}(e, p_1, p_2, p_3)$

$F_{RO}(e, p_1, p_2, p_3) = \{$
$?w, ?x, ?y, ?z :$

/ Remove a transição que parte de* p_1 *e consome* p_2 **/*
$?[e, (p_1, p_2) \rightarrow (?x, \epsilon)]$
$e = -[e, (p_1, p_2) \rightarrow (?x, \epsilon)]$

/ Remove a transição de* p_1 *até* p_3 **/*
$?[e, (p_1, ?y) \rightarrow (p_3, \epsilon)]$
$e = -[e, (p_1, ?y) \rightarrow (p_3, \epsilon)]$

/ Remove as transições que partem de* p_3 **/*
$?[e, (p_3, ?w) \rightarrow (?z, \epsilon)]$
$e = -[e, (p_3, ?w) \rightarrow (?z, \epsilon)]$

return e
$\}$

Código 5.2.4: Função adaptativa $F_{RO}(e, p_1, p_2, p_3)$ do exemplo 5.3.

Função adaptativa $F_{RAO}(e, p_1)$

$F_{RAO}(e, p_1) = \{$
$?x, ?y, ?z :$

/ Remove todas as transições que consomem* p_1 **/*
$?[e, (?x, p_1) \rightarrow (?y, \epsilon)]$
$e = -[e, (?x, p_1) \rightarrow (?y, \epsilon)]$

/ Remove todas as transições que consomem* λ_{p_1} **/*
$?[e, (?y, \lambda_{p_1}) \rightarrow (?z, \epsilon)]$
$e = -[e, (?y, \lambda_{p_1}) \rightarrow (?z, \epsilon)]$

return e
$\}$

Código 5.2.5: Função adaptativa $F_{RAO}(e, p_1)$ do exemplo 5.3.

Função adaptativa $F_{w+r}(e, p_1, p_2, p_3, p_4)$

$F_{w+r}(e, p_1, p_2, p_3, p_4) = \{$

/ Remove a transição de* p_1 *consumindo* p_2 *até* q_r*, e insere uma nova transição de* p_1 *consumindo* p_2 *até* q_{rw} **/*
$e = -[e, (p_1, p_2) \rightarrow (q_r, \epsilon)]$
$e = +[e, (p_1, p_2) \rightarrow (q_{rw}, \epsilon)]$

/ Remove todas as transições referentes ao gerenciamento de direitos desse relacionamento e insere outras transições que reflitam a nova configuração do controle de acesso */*
$e = -[e, (p_4, r) \rightarrow (q_f, \epsilon), F_{r-r}(e, p_1, p_2, p_3, p_4)]$
$e = -[e, (p_3, w) \rightarrow (q_f, \epsilon), F_{w+r}(e, p_1, p_2, p_3, p_4)]$
$e = +[e, (p_4, r) \rightarrow (q_f, \epsilon), F_{r-rw}(e, p_1, p_2, p_3, p_4)]$
$e = +[e, (p_4, w) \rightarrow (q_f, \epsilon), F_{w-rw}(e, p_1, p_2, p_3, p_4)]$

return e
$\}$

Código 5.2.6: Função adaptativa $F_{w+r}(e, p_1, p_2, p_3, p_4)$ do exemplo 5.3.

Função adaptativa $F_{r-r}(e, p_1, p_2, p_3, p_4)$

$F_{r-r}(e, p_1, p_2, p_3, p_4) = \{$

/ Remove a transição de* p_1 *consumindo* p_2 *até* q_r*, e insere uma nova transição de* p_1 *consumindo* p_2 *até* q_{null} **/*
$e = -[e, (p_1, p_2) \rightarrow (q_r, \epsilon)]$
$e = +[e, (p_1, p_2) \rightarrow (q_{null}, \epsilon)]$

/ Remove todas as transições referentes ao gerenciamento de direitos desse relacionamento e insere outras transições que reflitam a nova configuração do controle de acesso */*
$e = -[e, (p_4, r) \rightarrow (q_f, \epsilon), F_{r-r}(e, p_1, p_2, p_3, p_4)]$
$e = -[e, (p_3, w) \rightarrow (q_f, \epsilon), F_{w+r}(e, p_1, p_2, p_3, p_4)]$
$e = +[e, (p_3, r) \rightarrow (q_f, \epsilon), F_{r+}(e, p_1, p_2, p_3, p_4)]$
$e = +[e, (p_3, w) \rightarrow (q_f, \epsilon), F_{w+}(e, p_1, p_2, p_3, p_4)]$

return e
$\}$

Código 5.2.7: Função adaptativa $F_{r-r}(e, p_1, p_2, p_3, p_4)$ do exemplo 5.3.

Tabela 5.3: Seqüência de transições para reconhecer a cadeia $\langle alice \rangle \langle file1 \rangle \langle r \rangle$.

Origem	**Pilha (antes)**	**Transição**	**Pilha (depois)**	**Destino**
q_0	Z_0	$(q_0, \langle alice \rangle \langle file1 \rangle \langle r \rangle, Z_0) \vdash (q_1, \langle file1 \rangle \langle r \rangle, Z_0)$	Z_0	q_1
q_1	Z_0	$(q_1, \langle file1 \rangle \langle r \rangle, Z_0) \vdash (q_r, \langle r \rangle, Z_0)$	Z_0	q_r
q_r	Z_0	$(q_r, \langle r \rangle, Z_0) \vdash (q_5, \langle r \rangle, q_f Z_0)$	$q_f Z_0$	q_5
q_5	$q_f Z_0$	$(q_5, \langle r \rangle, q_f) \vdash (q_6, \epsilon, q_f)$	$q_f Z_0$	q_6
q_6	$q_f Z_0$	$(q_6, \epsilon, q_f) \vdash (q_5, \epsilon, q_7 q_f)$	$q_7 q_f Z_0$	q_5
q_5	$q_7 q_f Z_0$	$(q_5, \epsilon, q_7) \vdash (q_7, \epsilon, \epsilon)$	$q_f Z_0$	q_7
q_7	$q_f Z_0$	$(q_7, \epsilon, q_f) \vdash (q_f, \epsilon, \epsilon)$	Z_0	q_f

O autômato adaptativo de controle de acesso $M = (Q, S, \Sigma, \Gamma, P, q_0, Z_0, F)$ deve ser modelado pelo administrador do sistema de acordo com algumas *regras de construção*, apresentadas a seguir.

Para cada indivíduo $s_j \in S$, existe uma transição de q_0 até q_{s_j} consumindo s_j, $q_0, q_{s_j} \in Q$. Para cada indivíduo $s_j \in S$ e para cada objeto $o_i \in O$ relacionados entre si, existe uma transição para um estado que representa os direitos de s_j em o_i. No exemplo 5.3, *alice* é o indivíduo s_j, o estado q_1 é o estado q_{s_j}, o objeto *file1* é o_i, e o estado q_r representa o direito de leitura. Assim, a transição $q_1 \rightarrow q_r$ significa que *alice* tem direito de leitura sobre *file1*. Se houvesse uma transição de q_1 para q_{rw}, significaria que *alice* possuiria os direitos de leitura (r) e escrita (w) sobre *file1*.

Em geral, existe um estado para cada combinação de direitos e um estado que representa um relacionamento sem direitos estabelecidos (q_{null}). No exemplo 5.3, existem dois direitos (r e w) e quatro estados para representá-los (q_r, q_w, q_{rw} e q_{null}). Se existem n direitos, serão necessários $(\sum_{i=1}^{n} C_{n,i}) + 1$ estados, onde $C_{n,i}$ é uma combinação simples de n elementos, i a i, representada pela seguinte fórmula:

$$C_{n,i} = \frac{n!}{i!(n-i)!} \tag{5.1}$$

$C_{n,i}$ (Fórmula 5.1) denota o número total de combinações de n elementos tomados i a i, onde i é o número de elementos presentes em cada combinação.

Se existe um estado para cada combinação de direitos, conseqüentemente ha-

verá uma sub-máquina associada a cada estado para realizar a verificação dos direitos. A Figura 5.4 ilustra a configuração de uma sub-máquina genérica N_i, responsável por verificar os direitos de um indivíduo sobre um objeto. Por exemplo, uma sub-máquina N_{rx}, utilizando um conjunto de direitos $R = \{r, w, x\}$, teria $\{r, x\}$ como conjunto de direitos permitidos e $\{w\}$ como conjunto de direitos proibidos.

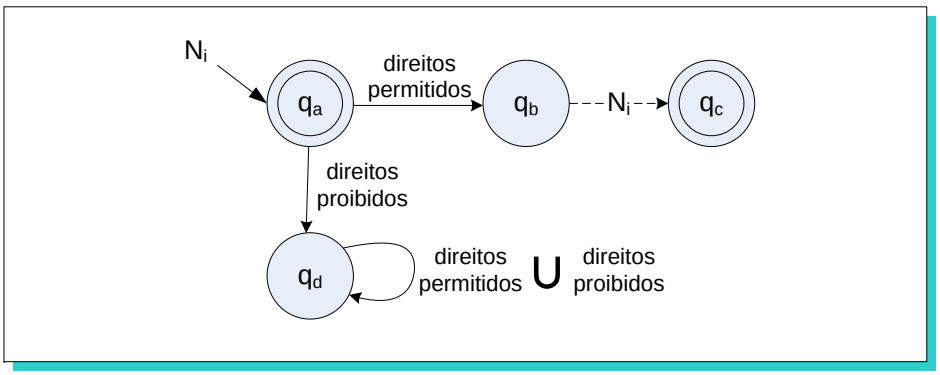

Figura 5.4: Sub-máquina genérica N_i.

Um objeto recém-criado o_i associado a um indivíduo s_j não tem permissões definidas *a priori*, conforme ilustrado na função $F_{NO}(e, p_1, p_2)$ (Código 5.2.3). Em outras palavras, a transição de q_{s_j} consumindo o_i terá como estado de destino q_{null}. No exemplo 5.3, ao criar o objeto *file2* para *alice*, haverá uma transição de q_1 consumindo *file2* até q_{null}.

Ao alterar os direitos de um indivíduo sobre um objeto, as funções adaptativas de inclusão e remoção de direitos referentes a esse relacionamento são substituidas por outras, de modo a refletir a configuração atual do controle de acesso. Por exemplo, suponha que o direito de *leitura* de *alice* sobre *file1* (exemplo 5.3) seja removido, através da submissão da cadeia $\langle alice \rangle \langle \lambda_{file1} \rangle \langle T \rangle \langle r \rangle$ ao autômato M. Ao consumir o último símbolo (r), a função adaptativa $F_{r-r}(M, q_1, file1, q_3, q_4)$ (Código 5.2.7) será chamada. Esta função removerá a transição de q_3 consumindo r, $F_{r-r}(M, q_1, file1, q_3, q_4)$ até q_f e a transição de q_4 consumindo w, $F_{w+r}(M, q_1, file1, q_3, q_4)$ até q_f. Ela também inserirá duas transições partindo de q_3 até q_f: uma consumindo w e chamando a função adaptativa $F_{w+}(M, q_1, file1, q_3, q_4)$, e outra consumindo r e

chamando a função adaptativa $F_{r+}(M, q_1, \textit{file1}, q_3, q_4)$. Neste caso, como o relacionamento entre *alice* e *file1* não possui nenhum direito, não é mais possível remover direitos (não existem transições de q_4 para q_f), mas agora é possível inserir direitos de *leitura* e *escrita* (r e w), através das funções adaptativas $F_{r+}(M, q_1, \textit{file1}, q_3, q_4)$ e $F_{w+}(M, q_1, \textit{file1}, q_3, q_4)$, respectivamente.

Seja n_{fi} o número de funções adaptativas de inclusão de direitos e n_{fr} o número de funções adaptativas de remoção de direitos de um determinado relacionamento *indivíduo-objeto*, com $n_{fi} + n_{fr} = n$, onde n é o total de direitos. n_{fr} é igual ao número de direitos presentes nesse relacionamento, pois as funções adaptativas de remoção de direitos só poderão remover direitos existentes no relacionamento. Por exemplo, se *alice* possui direito de *leitura* sobre *file1*, $n = 2$ (r e w), $n_{fr} = 1$, então existe uma função adaptativa de inclusão de direitos e uma função adaptativa de remoção de direitos. Caso *alice* possua os direitos de *leitura* e *escrita* sobre *file1*, $n = 2$, $n_{fr} = 2$, então não haverá função adaptativa de inclusão de direitos.

O número total de funções adaptativas de gerenciamento de direitos (inclusão e remoção) do autômato M, representado por $n_{functions}$, pode ser obtido através da seguinte fórmula:

$$n_{functions} = 2 \sum_{i=0}^{n-1} C_{n,i}(n - i) \tag{5.2}$$

A Fórmula 5.2 utiliza o total de direitos, representado por n. $C_{n,i}$ é uma combinação simples, apresentada na Fórmula 5.1. No exemplo 5.3, com o conjunto de direitos $R = \{r, w\}$, $n = 2$, existirão oito funções adaptativas de gerenciamento de direitos (inclusão e remoção), listadas na Tabela 5.4.

Tabela 5.4: Funções adaptativas de gerenciamento de direitos do exemplo 5.3.

Inclusão	**Remoção**
$F_{r+}(e, p_1, p_2, p_3, p_4)$	$F_{r-r}(e, p_1, p_2, p_3, p_4)$
$F_{w+}(e, p_1, p_2, p_3, p_4)$	$F_{w-w}(e, p_1, p_2, p_3, p_4)$
$F_{r+w}(e, p_1, p_2, p_3, p_4)$	$F_{r-rw}(e, p_1, p_2, p_3, p_4)$
$F_{w+r}(e, p_1, p_2, p_3, p_4)$	$F_{w-rw}(e, p_1, p_2, p_3, p_4)$

O conjunto R de direitos pode conter qualquer tipo de direito definido pelo administrador do sistema. Alguns exemplos são *leitura*, *escrita*, *execução*, *posse*, e assim por diante.

5.3 Comandos de privacidade

Para oferecer garantia de privacidade, um modelo de controle de acesso necessita de um conjunto de regras obtidas a partir de documentos de políticas de privacidade ou legislação. Tais regras são incorporadas ao modelo e analisadas durante uma requisição de acesso.

Para analisar um conjunto de regras, o MCAA deve permitir consultas ao autômato do modelo. Entretanto, o autômato não pode ser acessado arbitrariamente. Ele pode somente ser utilizado por um conjunto de *comandos de privacidade*, $C = \{c_1 \dots c_i\}$, $i \in \mathbb{N}$. Esses comandos são codificações de um *conjunto de regras* obtidas a partir de uma legislação ou política de privacidade escrita em linguagem natural. O Código 5.3.1 apresenta um exemplo de um conjunto de regras obtidas a partir de um parágrafo de um documento de política de privacidade.

Exemplo *Conjunto de regras*

Parágrafo:
"Um usuário pode copiar um arquivo para outro usuário, se esse primeiro for o dono do arquivo a ser copiado. O arquivo copiado deverá ser apenas para leitura."

Conjunto de regras
Sejam
A, B : usuários
C : arquivo a ser copiado

Se usuário A é dono do arquivo C **então**
início
Usuário A copia o arquivo C para B
Usuário B recebe direito de leitura para o objeto recém-copiado
fim

Código 5.3.1: Exemplo de um conjunto de regras obtidas a partir de um parágrafo de um documento de política de privacidade.

A partir de um conjunto de regras, é possível codificá-las em *comandos de privacidade* que verificarão se uma determinada situação pode ser caracterizada como violação de privacidade. Esses comandos são criados pelo administrador do sistema e refletem uma determinada política de privacidade ou legislação (ou um conjunto de documentos). Os comandos de privacidade podem controlar o acesso a recursos, abertura e leitura de documentos confidenciais, alteração e remoção de registros em um banco de dados, e assim por diante.

Os comandos de privacidade são escritos em uma linguagem de comandos de privacidade chamada *Pricomlan* (CEREDA; ZORZO, 2008a). Essa linguagem permite verificar se um indivíduo tem alguns direitos sobre um objeto, verificar relacionamentos, adicionar e remover indivíduos, objetos e direitos, entre outros. A linguagem *Pricomlan* possui uma sintaxe formal e proporciona uma facilidade para o administrador do sistema em codificar um conjunto de regras em comandos de privacidade. A gramática da linguagem *Pricomlan* é apresentada no Código 5.3.2.

Gramática *Linguagem Pricomlan*

```
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"
```

Código 5.3.2: Gramática da linguagem *Pricomlan*.

Um comando de privacidade é semelhante a uma função de uma linguagem imperativa como C ou Pascal. O comando toma argumentos (pelo menos um) e retorna "*true*" ou "*false*". No corpo do comando, a pseudo-variável "*auto*" refere-se

ao autômato adaptativo de controle de acesso que o modelo utiliza. Além de alguns comandos usuais, tais como condições e retornos, a linguagem *Pricomlan* dispõe de três funções pré-definidas: *eval*, *exec* e *equiv*.

A função *eval(u, e)* retorna "*true*" se a cadeia u é aceita pelo autômato e, $u \in L(e)$, ou "*false*" caso contrário.

A função *exec(u, e)* recebe o autômato e e a cadeia u. A função retorna o próprio autômato e se $u \notin L(e)$ ou se u é apenas uma cadeia de consulta (nenhum dos símbolos da Tabela 5.1 está em u). Se a cadeia de entrada u contém um dos símbolos da Tabela 5.1 e $u \in L(e)$, então *exec(u, e)* retornará o autômato modificado.

A função *equiv(e, f)* retorna "*true*" se os autômatos e e f são equivalentes, ou "*false*" caso contrário. Dois autômatos são equivalentes se eles possuirem os mesmos componentes (conjunto de estados, conjunto de sub-máquinas, alfabeto, alfabeto da pilha, transições, símbolo inicial, símbolo inicial da pilha e conjunto de estados finais). O algoritmo para *equiv* roda em tempo polinomial.

As cadeias submetidas às funções são definidas através da concatenação de símbolos pertencentes ao alfabeto Σ e os argumentos do comando de privacidade c_i. De acordo com a gramática da linguagem *Pricomlan* (Código 5.3.2), cada símbolo $v_j \in \Sigma$ é representado na forma <v_j>, e um argumento α é representado na forma <[α]>. Na execução do comando de privacidade, cada argumento é substituído pelo valor que foi atribuído. Por exemplo, se o argumento y for igual à *alice*, a cadeia <[y]><o_j<r> representará <*alice*><o_j<r>.

O conjunto de regras do Código 5.3.1 pode ser codificado em um comando de privacidade chamado `fileCopy(a, b, c)`, apresentado no Código 5.3.3. O direito o representa *posse* (do inglês *ownership*).

O comando de privacidade *fileCopy(a, b, c)* (Código 5.3.3) toma como argumentos dois indivíduos *a* e *b* e o objeto *c*. A primeira verificação é se *a* é o dono de *c* (linha 3). Se a condição for falsa, o código altera o fluxo para o bloco de *senão* (linha 8) e retorna "*false*" (linha 9). Se *a* é o dono de *c*, cria-se um relacionamento entre o indivíduo *b* e o objeto *c* (linha 4), inserindo o direito de leitura (linha 5). Na linha

Comando de privacidade *fileCopy(a, b, c)*

```
1   fileCopy(a, b, c)
2   início
3      se eval(<[a]><[c]><o>, auto) então {
4         auto = exec(<[b]><λ_NO><[c]>, auto);
5         auto = exec(<[b]><λ_[c]><G><r>, auto);
6         retornar true;
7      }
8      senão {
9         retornar false;
10     }
11  fim
```

Código 5.3.3: Comando de privacidade *fileCopy(a, b, c)*.

seguinte, a função retorna "*true*" (linha 6), indicando que o comando *fileCopy(a, b, c)* foi executado com sucesso. No exemplo 5.3, e supondo que o indivíduo *bob* esteja inserido no sistema, a execução do comando *fileCopy("alice", "bob", "file1")* retornaria "*false*", pois a primeira verificação (linha 3) falharia, já que *alice* tem apenas o direito de *leitura* sobre *file1*, e não o de *posse*.

A definição de *prova de conformidade* é apresentada por LI et al. (2005). Essa definição afirma que se uma consulta é verdadeira em um determinado sistema, a consulta está em conformidade com as políticas descritas (para esse sistema). Seja P um conjunto de documentos de políticas, chamado de *estado* do sistema, juntamente com uma relação de dedução $\Vdash$. Dado um estado P e uma determinada consulta Q, a relação $P \Vdash Q$ denota que Q é verdadeiro em P. Quando Q é proveniente de uma requisição de acesso, $P \Vdash Q$ significa que Q é permitido em P (LI et al., 2005). De modo semelhante, se um comando de privacidade $c_i \in C$ retorna verdadeiro em um autômato adaptativo de controle de acesso M, c_i está em conformidade com a configuração atual do autômato e com as políticas que ele descreve.

Os comandos de privacidade representam o conjunto de regras de um determinado sistema, enquanto que as funções adaptativas representam o modelo em si e suas características (política de acesso). Esta *separação de representação* confere

ao modelo a capacidade de alterar suas características principais (através das funções adaptativas) sem interferir significativamente nos comandos de privacidade. Entretanto, isso só é possível quando o conjunto de regras e as características do modelo são compatíveis. Por exemplo, o modelo pode ser projetado para representar uma política DAC (SANDHU; MUNAWER, 1998) e posteriormente, ser alterado para uma política MAC (OSBORN et al., 2000) sem a necessidade de alterações nos comandos de privacidade. Os comandos de privacidade apenas submetem cadeias ao autômato; o comportamento das funções adaptativas (isto é, como cada função adaptativa procederá com uma determinada ação) geralmente não importa para os comandos de privacidade.

Por exemplo, suponha que as funções adaptativas de gerenciamento de direitos do exemplo 5.3 tenham sido alteradas para que toda vez que um direito do relacionamento entre um indivíduo s_j e um objeto o_i for alterado, os direitos de todos os relacionamentos que possuam o objeto o_i também sejam alterados. Se *alice* e *bob* têm relacionamento com *file1*, ao submeter a cadeia $\langle alice\rangle\langle\lambda_{file1}\rangle\langle G\rangle\langle w\rangle$, *bob* também terá os direitos de seu relacionamento com *file1* alterados. Essa modificação nas funções adaptativas de gerenciamento de direitos representa o conceito de *classes de indivíduos*.

Existem casos, entretanto, em que não é possível manter a *separação de representação*. Em situações onde as mudanças no comportamento das funções adaptativas implicam em consultas com construções diferentes ou ações que não serão mais aceitas, os comandos devem ser avaliados para manter a coerência e conformidade. Por exemplo, se a função adaptativa de criação de um objeto (F_{NO}) for alterada para não permitir que tal objeto seja removido posteriormente, um comando que necessite remover um objeto retornará sempre "`false`". Nesse caso, especificamente, o conjunto de regras (representado pelo comando de privacidade) não é compatível com as características do modelo (representadas pelas funções adaptativas); dessa forma, a separação de representação pode ocorrer apenas em casos onde as regras e o modelo não sejam conflitantes.

Outra característica do MCAA é a capacidade de reuso dos comandos de privacidade. Um comando de privacidade pode ser utilizado com diversos argumentos e também ser utilizado em outra instância do MCAA, contanto que tais modelos sejam compatíveis entre si. Por exemplo, o comando de privacidade `fileCopy(a, b, c)` (Código 5.3.3) poderia ser utilizado em outro sistema que utilizasse um MCAA semelhante (e compatível) ao exemplo 5.3.

Do ponto de vista de implementação de um modelo de controle de acesso, é apresentado no Apêndice A um arcabouço para privacidade chamado SPA^3 (CEREDA; ZORZO, 2008b), que utiliza o MCAA, apresentado neste Capítulo, em conjunto com outras funcionalidades relevantes para a implementação.

O MCAA é um modelo de controle de acesso simples, porém genérico o suficiente para tratar os requisitos de privacidade e segurança de um sistema e, com isso, abrange diversas classes de problemas.

Resumo do Capítulo 5

Este Capítulo apresentou um modelo de controle de acesso baseado em autômatos adaptativos, chamado *Modelo de Controle de Acesso Adaptativo* (MCAA). O modelo utiliza um *autômato adaptativo de controle de acesso* (AACA) para realizar o controle de acesso, e possui *comandos de privacidade*, escritos em uma linguagem chamada *Pricomlan*, para codificar um conjunto de regras obtidas a partir de documentos de políticas de privacidade ou legislações, escritas em linguagem natural.

No próximo Capítulo, será apresentada uma avaliação do MCAA através de quatro análises realizadas.

CAPÍTULO 6

Avaliação

"A demonstração está no coração da Matemática, e isso é o que a distingue das outras ciências."

SIMON SINGH

Neste Capítulo, serão apresentadas quatro análises para avaliar o *Modelo de Controle de Acesso Adaptativo* (MCAA). A primeira análise consiste em uma comparação do MCAA com um dos modelos matriciais de controle de acesso, o HRU. A segunda análise trata da complexidade de aceitação de uma cadeia pelo autômato adaptativo de controle de acesso do MCAA. A terceira análise apresenta um estudo de caso de um ambiente hospitalar, descrito no Apêndice B, e compara a representatividade de alguns modelos de controle de acesso com o MCAA. Por fim, a quarta análise apresenta uma comparação formal através de uma teoria proposta por TRIPUNITARA; LI (2007).

As quatro análises permitem inferir a viabilidade do MCAA em termos de efici-

ência, representatividade e poder expressivo.

6.1 Comparação com modelos matriciais

Estruturalmente, o MCAA possui uma eficiência em tempo e espaço quando comparado com modelos matriciais, como o modelo HRU (HARRISON et al., 1976) e o modelo de Graham-Denning (GRAHAM; DENNING, 1972). Para esta análise, o modelo HRU foi escolhido devido à sua simplicidade de uso e semelhança com os conjuntos de indivíduos, objetos e direitos do MCAA.

As operações de criação e remoção de indivíduos são semelhantes nos dois modelos.

No MCAA, as relações nulas entre indivíduos e objetos não são representadas; assim, um objeto estará sempre relacionado a i indivíduos, $1 \leq i \leq n$, $i, n \in \mathbb{N}$, n é o número atual de indivíduos no sistema. 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 ($\varnothing$).

Exemplo 6.1. Seja 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 6.1 mostra a representação das relações entre indivíduos e objetos do exemplo no modelo HRU. O símbolo $\bullet$ indica as relações válidas, enquanto o símbolo $\varnothing$ representa as relações nulas.

Tabela 6.1: Representação das relações entre indivíduos e objetos no modelo HRU do exemplo 6.1.

	o_1	o_2	$\cdots$	o_k
s_1	$\bullet$	$\varnothing$	$\cdots$	$\varnothing$
s_2	$\varnothing$	$\bullet$	$\cdots$	$\varnothing$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
s_k	$\varnothing$	$\varnothing$	$\cdots$	$\bullet$

No modelo HRU do exemplo 6.1, 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 total de 900 relações. No MCAA, apenas as relações válidas são consideradas; dessa forma, considerando o exemplo 6.1, existirão apenas 30 relações no sistema. □

Seja T_{MCAA} o total de relações do MCAA, 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, $T_{MCAA} = T_{HRU}$; em todas as outras situações, $T_{MCAA} < T_{HRU}$.

Ao remover um objeto, o MCAA 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.

As operações de inserção e remoção de direitos são semelhantes nos dois modelos.

O MCAA permite que consultas sobre os direitos de um determinado objeto sejam realizadas em uma única cadeia, na forma $s_j o_k r_1 \dots r_n$, $j, k, n \in \mathbb{N}$, onde s_j é o indivíduo, o_k é o objeto, e $r_1 \dots r_n$ são os direitos a serem verificados. 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.

Exemplo 6.2. Deseja-se saber se o indivíduo s_i possui n direitos, $r_1 \dots r_n$, sobre o objeto o_j. A Figura 6.1 mostra as consultas necessárias para a verificação do exemplo através do modelo HRU neste exemplo.

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

Figura 6.1: Conjunto de consultas no modelo HRU do exemplo 6.2.

De acordo com a Figura 6.1, seriam necessárias n consultas para verificar se o indivíduo s_i possui n direitos sobre o objeto o_j. No MCAA, a verificação pode ser

realizada em apenas uma consulta, conforme a Figura 6.2.

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

Figura 6.2: Consulta de direitos no MCAA do exemplo 6.2.

As consultas no MCAA são realizadas em apenas uma cadeia a ser submetida ao autômato adaptativo de controle de acesso. No modelo HRU é necessária uma consulta para cada direito a ser verificado. Assim, se $n = 10$, serão necessárias 10 consultas no modelo HRU, e apenas 1 no MCAA. □

O MCAA possui tamanho de representação muito menor do que o dos modelos matriciais. O tempo de resposta para uma consulta entre o MCAA e os modelos matriciais é equivalente, entretanto o MCAA permite que as consultas sejam realizadas em apenas uma cadeia a ser submetida ao autômato.

6.2 Complexidade de aceitação de uma cadeia

Para analisar a complexidade de aceitação de uma cadeia, considere uma cadeia de entrada de comprimento n. Um autômato adaptativo qualquer toma $\mathcal{O}(n)$ passos, uma vez que o tempo utilizado por qualquer função adaptativa é constante (NETO; IWAI, 1998). Um *autômato adaptativo de controle de acesso* (AACA) consome o mesmo número de passos de um autômato adaptativo, pois a diferença entre os dois resume-se na característica do AACA em poder passar um símbolo de entrada como argumento para uma função que se tornará uma função adaptativa. Como um AACA é determinístico e toda função adaptativa executa em tempo constante, o número de passos também será $\mathcal{O}(n)$.

Se existir uma função adaptativa arbitrária que depende da entrada, o número de passos pode não ser limitado. No caso extremo, o algoritmo nunca encerrará. Essa situação nunca acontecerá no AACA, pois o número de passos tomados por uma função adaptativa não depende da cadeia de entrada.

O número de estados adicionados por uma entrada de comprimento n é $\mathcal{O}(n)$, considerando que existe um número constante de transições adicionadas. Este será sempre o caso do AACA por causa do modo que os estados e transições são manipulados pelas cadeias de entrada descritas na Tabela 5.2 (Capítulo 5, pág. 71). Estas são as únicas cadeias que usam as funções adaptativas e, portanto, podem alterar o autômato.

6.3 Estudo de caso de um ambiente hospitalar

Para comparar a representatividade de alguns modelos de controle de acesso com o MCAA, foi realizado um estudo de caso de um cenário de ambiente hospitalar, especificado em FISCHER-HÜBNER (2001). No ambiente hospitalar, foi considerado o tratamento médico em um centro cirúrgico, onde os dados pessoais do paciente são processados nos seguintes passos (FISCHER-HÜBNER, 2001): a) admissão do paciente, b) instruções de diagnóstico e tratamento por um especialista em exames, c) operação por um cirurgião, terapia ou transferência para outro centro de tratamento médico, d) alta do paciente, e e) transferência das informações de cobrança para a companhia de plano de saúde do paciente.

A descrição do estudo de caso é detalhada no Apêndice B.

O MCAA representou de modo direto o conjunto de regras do estudo de caso do ambiente hospitalar, apresentado no Apêndice B. Os demais modelos apresentados (MACP e MPT) apresentaram a descentralização das informações de acesso e a repetição de avaliações das regras de autorização. O MCAA apresentou um mapeamento direto da definição das regras em linguagem natural para um comando de privacidade, facilitando a representação e otimizando as validações. Além disso, é genérico o suficiente para tratar outras classes de problemas, não somente a situação exposta no estudo de caso.

A capacidade de mapeamento das outras abordagens em comandos de privacidade e sua representatividade conferem ao MCAA a característica de ser *multi-*

paradigma, podendo ser utilizado em mecanismos de segurança com reforço em privacidade.

6.4 Comparação formal entre modelos

A teoria de comparação de modelos de controle de acesso, proposta por TRIPUNITARA; LI (2007), permite comparar o poder de expressão dos modelos de controle de acesso.

Definição 6.3 (Esquema de Controle de Acesso)**.** Um *esquema de controle de acesso* é uma quádrupla $\langle \Gamma, Q, \Vdash, \Psi \rangle$, onde Γ é o conjunto finito de estados, Q é o conjunto finito de consultas, $\Vdash$: $\Gamma \times Q \rightarrow \{$ *true, false* $\}$ é a relação de dedução, e Ψ é o conjunto de regras de estado-transição.

Um estado $\gamma \in \Gamma$ contém todas as informações necessárias para tomar decisões de controle de acesso em um tempo determinado. A relação $\Vdash$ determina se uma consulta é verdadeira em um determinado estado. Quando uma consulta $q \in Q$ é realizada a partir de uma requisição de acesso, $\gamma \Vdash q$ significa que a requisição de acesso q é permitida no estado γ, e $\gamma \nVdash q$ significa que q é proibida.

Uma regra de estado-transição $\psi \in \Psi$ determina como o sistema de controle de acesso muda de estado. ψ define uma relação (denotada por $\xrightarrow{\psi}$) em Γ. Um estado γ_j é alcançável a partir do estado γ_i se $\gamma_j \xrightarrow{\psi}^{*} \gamma_i$ (TRIPUNITARA; LI, 2007). □

De acordo com a definição apresentada por TRIPUNITARA; LI (2007), um sistema de controle de acesso em um esquema de controle de acesso $\langle \Gamma, Q, \Vdash, \Psi \rangle$ é dado pelo par $\langle \gamma, \psi \rangle$, onde $\gamma \in \Gamma$ é o estado atual do sistema e $\psi \in \Psi$ é a regra de estado-transição que governa as mudanças de estado do sistema. Um modelo de controle de acesso é um conjunto de esquemas de controle de acesso.

Definição 6.4 (Redução de Equivalência de Estados)**.** Dado um mapeamento de X para Y, $\sigma : (\Gamma^X \times \Psi^X) \cup Q^X \rightarrow (\Gamma^Y \times \Psi^Y) \cup Q^Y$, X e Y são esquemas de controle de acesso, os estados γ^X e γ^Y são equivalentes sob o mapeamento quando para cada

$q^X \in Q^X$, $\gamma^X \Vdash^X q^X$ se e somente se $\gamma^Y \Vdash^Y \sigma(q^X)$. Um mapeamento σ de X para Y é dito ser uma *redução de equivalência de estados* se para cada $\gamma^X \in \Gamma^X$ e para cada $\psi^X \in \Psi^X$, $\langle \gamma^Y, \psi^Y \rangle$ = $\sigma(\langle \gamma^X, \psi^X \rangle)$ tem-se as duas propriedades a seguir:

- para cada estado γ_1^X no esquema X tal que $\gamma^X \overset{\psi^{X*}}{\rightarrow} \gamma_1^X$, existe um estado γ_1^Y tal que $\gamma^Y \overset{\psi^{Y*}}{\rightarrow} \gamma_1^Y$ e γ_1^X e γ_1^Y são equivalentes sob o mapeamento σ.
- para cada estado γ_1^Y no esquema Y tal que $\gamma^Y \overset{\psi^{Y*}}{\rightarrow} \gamma_1^Y$, existe um estado γ_1^X tal que $\gamma^X \overset{\psi^{X*}}{\rightarrow} \gamma_1^X$ e γ_1^X e γ_1^Y são equivalentes sob o mapeamento σ.

□

O mapeamento σ é uma redução de equivalência de estados se para cada $\gamma^X \in \Gamma^X$ e para cada $\psi^X \in \Psi^X$, qualquer estado γ_1^X alcançável por ψ^X, começando em γ^X pode ser associado a um estado γ_1^Y alcançável por γ^Y, onde $(\gamma^Y, \psi^Y) = \sigma(\langle \gamma^X, \psi^X \rangle)$, e γ_1^X e γ_1^Y produzem respostas iguais para consultas equivalentes, isto é, $\gamma_1^X \Vdash^X q^X$ se e somente se $\gamma_1^Y \Vdash^Y \sigma(q^X)$.

Definição 6.5 (Redução)**.** Sejam dois esquemas de controle de acesso $X = \langle \Gamma^X, Q^X, \Vdash^X, \Psi^X \rangle$ e $Y = \langle \Gamma^Y, Q^Y, \Vdash^Y, \Psi^Y \rangle$. Um mapeamento de X para Y, σ, é dito ser uma *redução* de X para Y se para cada $\gamma^X \in \Gamma^X$ e para cada $\psi^X \in \Psi^X$, $\langle \gamma^Y, \psi^Y \rangle$ = $\sigma(\langle \gamma^X, \psi^X \rangle)$ tem-se a seguinte propriedade:

- para cada estado γ_1^X no esquema X tal que $\gamma^X \overset{\psi^{X*}}{\rightarrow} \gamma_1^X$ e para cada consulta q^X em X existe um estado γ_1^Y em Y, dependente de q^X, tal que $\gamma^Y \overset{\psi^{Y*}}{\rightarrow} \gamma_1^Y$ e $\gamma^X \Vdash^X q^X$ se e somente se $\gamma_1^Y \Vdash^Y \sigma(q^X)$. Para cada q^X deve-se obter um γ_1^Y diferente.

□

A Definição 6.4 difere da Definição 6.5 no sentido de que a primeira requer que, para cada estado alcançável em X, exista um estado equivalente em Y que retorne a mesma resposta para cada consulta. Na Definição 6.5, é requerida a existência de um estado equivalente para cada consulta, entretanto, os estados equivalentes podem ser diferentes para consultas diferentes (TRIPUNITARA; LI, 2007).

Definição 6.6 (Comparando o Poder Expressivo de Modelos de Controle de Acesso)**.** Dados dois modelos de controle de acesso M e M', M' é pelo menos tão expressivo quanto M se para cada esquema em M existe uma redução de equivalência de estados (ou uma redução) para um esquema em M'. Em adição, se para cada esquema em M' existe uma redução de equivalência de estados (ou uma redução) para um esquema em M, M e M' são *equivalentes em poder expressivo*. Se M' é pelo menos tão expressivo quanto M', e existe um esquema A em M' tal que para qualquer esquema B em M não existe nenhuma redução de equivalência de estados (ou uma redução) de A para B, então M' é *estritamente mais expressivo* que M (TRIPUNITARA; LI, 2007). □

A representação do MCAA como um esquema de controle de acesso é definida como: Γ é o conjunto de todos os autômatos adaptativos de controle de acesso possíveis, Q é o conjunto das cadeias possíveis (Tabela 5.2, pág. 71), a relação $\Vdash$ é definida como $\gamma \Vdash u$ se e somente se $u \in L(\gamma)$, u é uma cadeia, e a regra de estado-transição ψ é o conjunto C de comandos de privacidade.

Teorema 6.7. *O Modelo de Controle de Acesso Adaptativo (MCAA) possui poder expressivo equivalente ao Modelo de Autorização Contextual baseado em Papéis (MACP) e ao Modelo de Privacidade baseado em Tarefas (MPT).*

Prova. Para comparar formalmente os modelos MCAA, MACP e MPT, considere os seguintes elementos: γ^{MCAA} é o estado atual do MCAA, γ_1^{MCAA} é o estado alcançável por ψ^{MCAA}, γ^{MACP} é o estado atual do MACP, γ_1^{MACP} é o estado alcançável por ψ^{MACP}, γ^{MPT} é o estado que atual do MPT e γ_1^{MPT} é o estado alcançável por ψ^{MPT}. Em outras palavras, $\gamma^{MCAA} \overset{\psi^{MCAA^*}}{\rightarrow} \gamma_1^{MCAA}$, $\gamma^{MACP} \overset{\psi^{MACP^*}}{\rightarrow} \gamma_1^{MACP}$ e $\gamma^{MPT} \overset{\psi^{MPT^*}}{\rightarrow} \gamma_1^{MPT}$. A partir do exemplo do estudo de caso apresentado na seção 6.3 e detalhado no Apêndice B, é possível observar que:

- $(\gamma^{MCAA}, \psi^{MCAA}) = \sigma(\langle \gamma^{MACP}, \psi^{MACP} \rangle)$, γ_1^{MCAA} e γ_1^{MACP} produzem respostas iguais para consultas equivalentes, $\gamma_1^{MCAA} \Vdash^{MCAA} q^{MCAA}$ e $\gamma_1^{MACP} \Vdash^{MACP}$

$\sigma(q^{MCAA})$; os estados γ_1^{MCAA} e γ_1^{MACP} são equivalentes sob o mapeamento σ (Definição 6.4).

- $(\gamma^{MACP}, \psi^{MACP}) = \sigma(\langle\gamma^{MCAA}, \psi^{MCAA}\rangle)$, γ_1^{MACP} e γ_1^{MCAA} produzem respostas iguais para consultas equivalentes, $\gamma_1^{MACP} \Vdash^{MACP} q^{MACP}$ e $\gamma_1^{MCAA} \Vdash^{MCAA} \sigma(q^{MACP})$.

- $(\gamma^{MCAA}, \psi^{MCAA}) = \sigma(\langle\gamma^{MPT}, \psi^{MPT}\rangle)$, γ_1^{MCAA} e γ_1^{MPT} produzem respostas iguais para consultas equivalentes, $\gamma_1^{MCAA} \Vdash^{MCAA} q^{MCAA}$ e $\gamma_1^{MPT} \Vdash^{MPT} \sigma(q^{MCAA})$; os estados γ_1^{MCAA} e γ_1^{MPT} são equivalentes sob o mapeamento σ (Definição 6.4).

- $(\gamma^{MPT}, \psi^{MPT}) = \sigma(\langle\gamma^{MCAA}, \psi^{MCAA}\rangle)$, γ_1^{MPT} e γ_1^{MCAA} produzem respostas iguais para consultas equivalentes, $\gamma_1^{MPT} \Vdash^{MPT} q^{MPT}$ e $\gamma_1^{MCAA} \Vdash^{MCAA} \sigma(q^{MPT})$.

As observações anteriores demonstram que o MCAA possui redução de equivalência de estados para o MACP e o MPT assim como o MACP e o MPT também apresentam reduções de estados para o MCAA. De acordo com a Definição 6.6, dois modelos M e M' são equivalentes em poder expressivo quando existe uma redução de equivalência de estados (ou uma redução) de M para M' e uma redução de equivalência de estados (ou uma redução) de M' para M. Assim:

- $(\gamma^{MCAA}, \psi^{MCAA}) = \sigma(\langle\gamma^{MACP}, \psi^{MACP}\rangle)$, $(\gamma^{MACP}, \psi^{MACP}) = \sigma(\langle\gamma^{MCAA}, \psi^{MCAA}\rangle)$, portanto o MCAA e o MACP são *equivalentes em poder expressivo*.

- $(\gamma^{MCAA}, \psi^{MCAA}) = \sigma(\langle\gamma^{MPT}, \psi^{MPT}\rangle)$, $(\gamma^{MPT}, \psi^{MPT}) = \sigma(\langle\gamma^{MCAA}, \psi^{MCAA}\rangle)$, portanto o MCAA e o MPT são *equivalentes em poder expressivo*.

□

Como apresentado no Teorema 6.7, pode-se concluir que o MCAA demonstrou ser equivalente em poder expressivo aos modelos MACP e MPT, ratificando-o como uma solução viável para ambientes computacionais que necessitem reforçar os aspectos de segurança e privacidade.

6.5 Considerações

O mapeamento direto de documentos de políticas de privacidade em comandos, a eficiência em relação aos modelos matriciais, a representatividade do modelo, o reuso de comandos de privacidade e o poder expressivo equivalente a outros modelos demonstram que o MCAA é uma solução eficiente para garantia de privacidade em sistemas computacionais.

Resumo do Capítulo 6

Este Capítulo apresentou quatro análises para avaliar o *Modelo de Controle de Acesso Adaptativo* (MCAA). Os resultados obtidos nas análises foram relevantes e validam o MCAA como um modelo viável de controle de acesso com reforço em privacidade.

No próximo Capítulo, as conclusões referentes a este trabalho serão apresentadas.

CAPÍTULO 7

Conclusões

"I have no special talent. I am only passionately curious."

ALBERT EINSTEIN

O *Modelo de Controle Acesso Adaptativo* (MCAA) apresenta-se como uma nova abordagem para modelos de controle de acesso com reforço em privacidade. Seu poder expressivo é comparável a outros modelos existentes na literatura, e a característica de *separação de representação* permite ao modelo alterar suas regras de acordo com um determinado cenário. Esta característica confere ao modelo uma ampla área de aplicação.

Partindo-se de um documento de privacidade (legislação, políticas, entre outros) escrito em linguagem natural, é possível reduzí-lo a um conjunto de regras para serem codificadas em comandos de privacidade. Esses comandos são utilizados pelo MCAA e verificam se uma determinada situação pode ser caracterizada como violação de privacidade, de acordo com o documento sobre o qual os comandos

foram definidos.

A separação de representação do MCAA permite que as funções adaptativas sejam substituídas sem interferir nos comandos de privacidade, e vice-versa. Além disso, o alto nível de abstração confere aos comandos de privacidade a capacidade de reuso e a *prova de conformidade* assegura que tais comandos procederão como esperado.

As modificações no controle de acesso são facilmente transformadas em modificações no autômato através da submissão de cadeias de entrada. As cadeias de entrada representam consultas e alterações no autômato, o qual está formalmente definido.

A linguagem *Pricomlan* possui uma sintaxe formal e simplificada, o que proporciona uma facilidade para o administrador do sistema em codificar um conjunto de regras em comandos de privacidade.

A utilização do MCAA como modelo de controle de acesso no SPA^3 (Apêndice A) confere ao arcabouço a possibilidade de registro de todas as ações realizadas no sistema – as que foram bem-sucedidas e as que falharam. Tal característica permite a realização de auditorias e garante a privacidade dos indivíduos e de suas informações pessoais.

O MCAA possibilita uma representação eficiente e compacta para o controle de acesso. A utilização de um autômato adaptativo permite que as alterações realizadas em sua topologia reflitam no estado corrente do controle de acesso. Além disso, a separação de representação permite que sua utilização seja expandida para outros contextos além da privacidade, pois o modelo é genérico o suficiente para permitir outras classes de problemas.

7.1 Trabalhos futuros

O trabalho apresentado evidencia que alguns estudos posteriores podem contribuir para as áreas de Privacidade e de Tecnologias Adaptativas.

Sugere-se um estudo sobre os riscos de privacidade em ambientes ubíqüos que ofereçam serviços baseados em localização (LBS), e a tentativa de utilização de um modelo de controle de acesso que ofereça suporte a tais ambientes.

Na área de Tecnologias Adaptativas, sugere-se uma extensão do autômato adaptativo, na qual seja possível que uma função adaptativa possa também tomar como argumento uma outra função adaptativa e alterá-la, retornando assim uma nova função adaptativa.

APÊNDICE

A

Um Arcabouço para Privacidade

"Great things are not done by impulse, but a series of small things brought together."

VINCENT VAN GOGH

Este Apêndice apresenta um arcabouço para privacidade, chamado *Sistema de Privacidade Auditável com Autômato Adaptativo* (SPA^3), que contribui para a construção simplificada de sistemas com reforço nos aspectos de privacidade. O SPA^3 utiliza o *Modelo de Controle de Acesso Adaptativo* (MCAA), apresentado no Capítulo 5, como controle de acesso.

A.1 Sistema de Privacidade Auditável

Um *Sistema de Privacidade Auditável* (SPA) (MAY et al., 2006) 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*[1].

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 informa o indivíduo acerca de ações realizadas por outros indivíduos no sistema, e no *logging* o sistema monitora 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, apresentado no Capítulo 3, é um modelo de controle de acesso que foi utilizado no Sistema de Privacidade Auditável proposto por MAY et al. (2006), através das seis operações primitivas presentes no modelo, com a adição de duas novas operações que realizam a notificação (**inform** s **of** t) e *logging* (**log** t). Alterações no modelo HRU foram necessárias para que este implementasse totalmente as características de um Sistema de Privacidade Auditável (MAY et al., 2006).

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 (MAY et al., 2006), como o MCAA.

A.2 SPA com Autômato Adaptativo

O *Sistema de Privacidade Auditável com Autômato Adaptativo* (SPA^3) tem como objetivo simplificar a especificação dos aspectos de privacidade de um determinado

[1]Palavra de origem latina que significa "*após o fato*".

sistema. O arcabouço pode ser utilizado em vários domínios, devido à abrangência do MCAA.

O autômato adaptativo de controle de acesso é representado no arcabouço utilizando a linguagem de marcação XML. O arquivo XML possui a configuração inicial do autômato adaptativo de controle de acesso (AACA) a ser utilizado. As *tags* do arquivo XML representam as transições do autômato, que podem ser na forma $P(q, a, \beta) \rightarrow (q', a', A, B)$ ou $P(q, ab, \beta) \rightarrow (q', a', A, B(b))$. Um exemplo de arquivo XML pode ser visualizado no Código A.2.1 (CEREDA; ZORZO, 2008b).

Arquivo XML *Representação do AACA*

```
<automato>
...
   <transicao>
     <origem> q </origem>
     <simbolo> a </simbolo>
     <topo> β </topo>
     <destino> q' </destino>
     <novosimbolo> a' </novosimbolo>
     <novotopo> β' </novotopo>
     <funcao1> A(...) </funcao1>
     <funcao2> B(...) </funcao2>
   </transicao>
...
</automato>
```

Código A.2.1: Exemplo de arquivo XML para representação do autômato adaptativo de controle de acesso (CEREDA; ZORZO, 2008b).

A construção do arcabouço requer o arquivo XML, contendo o autômato adaptativo, e os comandos de privacidade, de acordo com a Figura A.1.

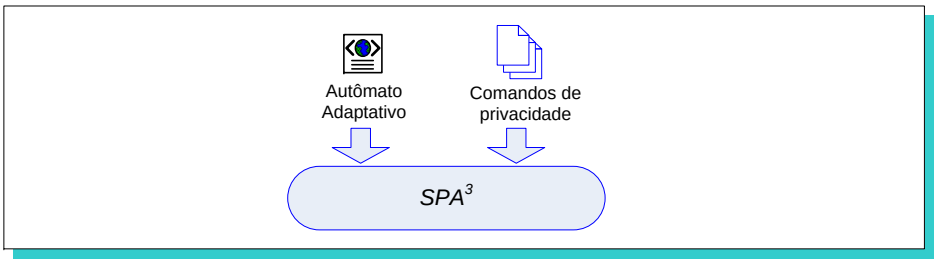

Figura A.1: Construção do arcabouço.

Exemplo A.1. Seja o comando de privacidade *fileCopy(a, b, c)*, apresentado no Código 5.3.3 (Capítulo 5, pág. 84). Deseja-se verificar se *alice* pode copiar *file1* para *bob*. O Código A.2.2 apresenta um trecho de código escrito em Java responsável pela execução do comando de privacidade *fileCopy("alice", "bob", "file1")*.

Código Java *Trecho de código da execução do comando* `fileCopy`

```
...
1   AdaptiveAutomaton automato = new AdaptiveAutomaton();
2   automato.load("automato.xml");

3   PrivacyCommand priv = new PrivacyCommand(automato);
4   priv.loadPrivacyCommand("fileCopy");
5   priv.setParameters("alice, bob, file1");

6   boolean resultado = priv.execute();
7   if (resultado == true) System.out.println("Ação permitida");
8   else System.out.println("Ação proibida");
...
```

Código A.2.2: Trecho de código escrito em Java responsável pela execução do comando de privacidade *fileCopy("alice", "bob", "file1")*.

Conforme o Código A.2.2, a primeira parte consiste em instanciar e carregar o AACA (linhas 1 e 2); a segunda parte carrega o comando de privacidade *fileCopy* e define os argumentos deste comando (linhas 3 a 5); a última parte consiste em executar o comando de privacidade *fileCopy*, armazenar o valor de retorno em uma variável lógica e imprimir um texto na tela de acordo com o conteúdo dessa variável (linhas 6 a 8). □

O SPA^3 permite que os comandos de privacidade sejam executados e avaliados em um sistema, aumentando o controle sobre as informações que trafegam e minimizando situações que caracterizem violação de privacidade.

A.3 Considerações

A utilização de um arcabouço é justificada pela facilidade proporcionada no desenvolvimento de sistemas. Um arcabouço para privacidade contribui para a constru-

ção simplificada de sistemas com reforço em privacidade de modo flexível e extensível.

O SPA^3 provê uma solução que reforça os aspectos de privacidade de um determinado sistema. A utilização do formato XML para representação do autômato adaptativo proporciona um padrão bem definido, simples e formal para ser utilizado como entrada do arcabouço. A modelagem do autômato adaptativo através de XML é clara e concisa, além de permitir uma especificação completa do mesmo. Os comandos de privacidade possuem uma sintaxe de alto nível e de fácil representação, contribuindo para a especificação de situações de violação de privacidade em um sistema.

O SPA^3 é uma solução viável para incorporar reforços de privacidade nos sistemas. Os comandos de privacidade são reutilizáveis e podem ser executados a qualquer momento em um sistema, conferindo ao arcabouço um nível de abstração para ser utilizado em diversas situações.

APÊNDICE

B

Estudo de Caso de um Ambiente Hospitalar

"A leitura é uma fonte inesgotável de prazer mas, por incrível que pareça, a quase totalidade não sente esta sede."

CARLOS DRUMMOND DE ANDRADE

Este Apêndice apresenta a descrição do estudo de caso de um ambiente hospitalar, originalmente especificado em (FISCHER-HÜBNER, 2001) e utilizado neste trabalho para comparar os modelos MCAA, MACP e MPT.

B.1 Descrição do estudo de caso

De acordo com a especificação do estudo de caso de um ambiente hospitalar (FISCHER-HÜBNER, 2001), foi considerado o tratamento médico em um centro

cirúrgico, onde os dados pessoais do paciente são processados nos seguintes passos: a) admissão do paciente, b) instruções de diagnóstico e tratamento por um especialista em exames, c) operação por um cirurgião, terapia ou transferência para outro centro de tratamento médico, d) alta do paciente, e e) transferência das informações de cobrança para a companhia de plano de saúde do paciente.

A operação escolhida para representar o estudo de caso, dentro do domínio hospitalar, foi a realização do diagnóstico de um paciente, feita por um especialista em exames. As etapas dessa operação são:

1. Fazer diagnóstico do paciente.
2. Criar arquivo de diagnóstico.
3. Criar arquivo de instrução de tratamento.
4. Adicionar linhas ao arquivo de protocolo de tratamento.

Para que a operação de diagnóstico de um paciente seja realizada com sucesso, é necessário obedecer às seguintes regras definidas para o domínio hospitalar:

- O usuário precisa ter permissão para criar os arquivos de diagnóstico e instrução de tratamento.
- O paciente necessita ter um protocolo de tratamento previamente criado.
- O usuário precisa ter permissão para adicionar linhas ao protocolo de tratamento.
- É preciso garantir que o protocolo de tratamento pertence ao paciente.
- As etapas da operação de realização de diagnóstico devem ser permitidas.

Para uma análise comparativa com o MCAA, o estudo de caso foi também mapeado para o Modelo de Autorização Contextual baseado em Papéis (MACP) e o Modelo de Privacidade baseado em Tarefas (MPT), apresentados no Capítulo 3. A

escolha destes modelos justifica-se por suas utilizações em estudos de casos de ambientes hospitalares, e também pelas características de cada modelo.

B.2 Representação no MCAA

A representação da operação de realização do diagnóstico de um paciente em um comando de privacidade do MCAA, escrito em *Pricomlan*, é apresentada no Código B.2.1.

O comando de privacidade *realizarDiagnóstico* (Código B.2.1) recebe os seguintes argumentos (linha 1): *p1* é o usuário que está cadastrando, *p2* é o nome do paciente, *p3* é o arquivo de diagnóstico, *p4* é o arquivo de instrução de tratamento e *p5* é o arquivo de protocolo de tratamento. Na linha 3, verifica-se se usuário que está cadastrando (*p1*) é um especialista em exames. Se o usuário é um especialista, é verificado se o contexto do diagnóstico, com a ação *criar*, tem os direitos de criação (*c*), leitura (*r*), escrita (*w*) e *append* (*a*) (linha 4). O próximo passo, na linha 5, é verificar se o paciente (*p2*) já possui um protocolo de tratamento (*p5*) associado a ele. Nas linhas 6 e 12, os arquivos de diagnóstico e instrução de tratamento são criados. Se as condições anteriores foram satisfeitas, o autômato do modelo recebe a nova configuração (linha 18) e o comando retorna *true* (linha 19). Os demais retornos indicam que o protocolo de tratamento não existe (linha 23), o contexto não permite realizar essa operação (linha 27) ou que o usuário não tem permissão para realizar o diagnóstico (linha 31).

B.3 Representação no MACP

A representação da operação do diagnóstico de um paciente através do MACP é apresentada no Código B.3.1.

No MACP (Código B.3.1), é necessário definir uma autorização contextual para cada etapa da operação a ser realizada. A definição da autorização está associada ao papel de usuário (linha 1: *Especialista*) e sua execução depende da avaliação

Comando de privacidade *realizarDiagnóstico(p1, p2, p3, p4, p5)*

```
1  realizarDiagnóstico(p1, p2, p3, p4, p5)
2  início
3    se eval(<[p1]><especialista>, auto) então {
4      se eval(<diag><criar><c><r><w><a>, auto) então {
5        se eval(<[p2]><[p5]>, auto) então {
6          tmp = exec(<paciente><λNO><[p3]>, auto);
7          se equiv(auto, tmp) então {
8            retornar false;
9          }
10         senão {
11           auto = tmp;
12           tmp = exec(<paciente><λNO><[p4]>, auto);
13         }
14         se equiv(auto, tmp) então {
15           retornar false;
16         }
17         senão {
18           auto = tmp;
19           retornar true;
20         }
21       }
22       senão {
23         retornar false;
24       }
25     }
26     senão {
27       retornar false;
28     }
29   }
30   senão {
31     retornar false;
32   }
33 fim
```

Código B.2.1: Comando de privacidade referente à operação de realização do diagnóstico de um paciente.

da regra de autorização, representada pela expressão lógica *exp-abs(p1, p2)*, onde *p1* representa o paciente e *p2* o arquivo de protocolo de tratamento. No exemplo, existem três regras de autorização: criar arquivo de diagnóstico (linhas 2 a 5), criar arquivo de instrução de tratamento (linhas 6 a 9) e adicionar informações ao arquivo de protocolo de tratamento (linhas 10 a 13). A operação de realização do

MACP *Realização do diagnóstico de um paciente*

```
1  Especialista:
2  < exp-abs(p1, p2) {
3     p1 in pacCtx.pacientes_internados & p2 in pacCtx.protocolo_tratamento &
4     pacCtx.permissoes("diag","criar", "create & read & write & append")
5  }, criar, arquivo_de_diagnóstico, forte >,
6  < exp-abs(p1, p2) {
7     p1 in pacCtx.pacientes_internados & p2 in pacCtx.protocolo_tratamento &
8     pacCtx.permissoes("diag","criar", "create & read & write & append")
9  }, criar, arquivo_de_instrução_de_tratamento, forte >,
10  < exp-abs(p1, p2) {
11    p1 in pacCtx.pacientes_internados & p2 in pacCtx.protocolo_tratamento &
12    pacCtx.permissoes("diag","criar", "create & read & write & append")
13 }, append, arquivo_de_protocolo_de_tratamento, forte >.
```

Código B.3.1: Representação da operação de realização do diagnóstico de um paciente no MACP.

diagnóstico de um paciente necessita de três chamadas às autorizações contextuais para ser executada satisfatoriamente.

B.4 Representação no MPT

No MPT, um indivíduo não pode acessar um objeto de dados pessoais de maneira arbitrária. Para executar uma tarefa, é necessário executar procedimentos de transformação que acessarão objetos de modo controlado. Assim, para cada etapa a ser executada para realizar o diagnóstico de um paciente, um procedimento de transformação será executado (FISCHER-HÜBNER, 2001). A Tabela B.1 ilustra os propósitos e acessos necessários para a realização da operação do diagnóstico de um paciente no MPT.

O MPT trata apenas do processamento dos dados pessoais e reforça os princípios básicos de privacidade. Assim, sua implementação requer uma combinação de outros modelos para prover o controle de acesso propriamente dito (FISCHER-HÜBNER, 2001). O Código B.4.1 ilustra os procedimentos de transformação e de criação de objetos no MPT.

De acordo com o Código B.4.1, *Execute-TP*$(S_i, transp_j)$ (linha 1) informa que o

Tabela B.1: Propósitos e acessos necessários para a realização da operação do diagnóstico de um paciente no MPT (FISCHER-HÜBNER, 2001).

Tarefa	**Classe do objeto**	**Propósito da Tarefa**	**Acessos**
Diagnóstico	Diagnóstico	pm_create	$\{c\}$
Diagnóstico	Diagnóstico	Editor	$\{r, w, a\}$
Diagnóstico	Diagnóstico	Visualizar	$\{r\}$
Diagnóstico	Prot. Tratamento	Editor	$\{a\}$
Diagnóstico	Req. Tratamento	pm_create	$\{c\}$
Diagnóstico	Req. Tratamento	Editor	$\{r, w, a\}$

MPT *Procedimentos de transformação e de criação de objetos*

```
1  Execute-TP(S_i, transp_j)
2     if transp_j ∈ ATP(CT(S_i)) ∧ CTP(S_i) = NIL then
3        CA* = CA − (CA ∩ ({S_i} × O × A))
4        CTP*(S_i) = transp_j

5  create-object(S_i, o-class_k)
6     if (CT(S_i), o-class_k, CTP(S_i), create) ∈ NA∧
7      T-Purpose(CT(S_i)) ∈ O-Purposes(o-class_k) then
8        OP* = OP ∪ {O_new}
9        O* = O ∪ {O_new}
10       Class*(O_new) = o-class_k
```

Código B.4.1: Representação dos procedimentos de transformação e de criação de objetos no MPT (FISCHER-HÜBNER, 2001)

indivíduo S_i deseja executar um procedimento de transformação $transp_j$, enquanto que *create-object*$(S_i, o\text{-}class_k)$ (linha 5) informa que o indivíduo S_i deseja criar um objeto de dados pessoais, denotado por O_{new}, de uma classe $o\text{-}class_k$. Os elementos apresentados no Código B.4.1 são os seguintes: O é o conjunto de objetos, $CT(S_i)$ é a tarefa executada atualmente pelo indivíduo S_i, OP é o conjunto de objetos que possuem dados pessoais, $T\text{-}Purpose(T_i)$ é o propósito da tarefa T_i, $CTP(S_i)$ é o procedimento de transformação executado atualmente pelo indivíduo S_i, $ATP(T_i)$ é o conjunto de procedimentos de transformação autorizados para a tarefa T_i, CA é o conjunto de direitos atuais, A é o conjunto de direitos de acesso, $O\text{-}Purposes(o\text{-}class_i)$ são os propósitos para os objetos da classe $class_i$, NA indica os acessos necessários e $(T_i, o\text{-}class_j, transp_k, x) \in NA$ indica a necessidade de um indivíduo com uma tarefa atual T_i para acessar um objeto da classe $o\text{-}class_j$ em modo x

através do procedimento de transformação $transp_k$ (FISCHER-HÜBNER, 2001).

Para realizar a operação do diagnóstico de um paciente no MPT, será necessário executar a tarefa *Diagnóstico* 5 vezes (Tabela B.1), uma para cada propósito da tarefa (*Diagnóstico* → *pm_create*, *Diagnóstico* → *Editor*, *Prot. Tratamento* → *Editor*, *Req. Tratamento* → *pm_create*, *Req. Tratamento* → *Editor*). O propósito *Visualizar* foi omitido por não fazer parte da descrição do estudo de caso.

B.5 Considerações

O MCAA representou de modo direto o conjunto de regras do estudo de caso, além de apresentar um mapeamento direto da definição das regras em linguagem natural para um comando de privacidade. Tal capacidade de mapeamento das outras abordagens em comandos de privacidade e sua representatividade conferem ao MCAA a característica de ser *multiparadigma*.

Referências Bibliográficas

ACKERMAN, M.; CRANOR, L. F. **Privacy critics – safeguarding users' personal data**. Disponível em <http://www.webtechniques.com/archives/1999/09/ackerman>. Acesso em: 20 jan. 2007.

BARKLEY, J. et al. Role based access control for the world wide web. In: 20TH NATIONAL COMPUTER SECURITY CONFERENCE, 1997. **Proceedings...** Baltimore, MD, USA, 1997. p. 1–11.

BELL, D. E.; LAPADULA, L. J. **Secure computer systems: Mathematical foundations and model**, 1974. Technical Report ESD–TR–278.

BIBA, K. J. **Integrity Considerations for Secure Computer Systems**, 1977. Technical Report TR–3153.

BISKUP, J.; LEINEWEBER, T. State-dependent security decisions for distributed object-systems. In: 15TH ANNUAL WORKING CONFERENCE ON DATABASE AND APPLICATION SECURITY, 2002. **Proceedings...** Norwell, MA, USA, 2002. p. 105–118.

BRASIL, 1988. **Constituição da República Federativa do Brasil**. Senado,

Brasília, Distrito Federal. Disponível em <http://www.senado.gov.br/sf/ ▼ legislacao/const/>. Acesso em: 03 jan. 2007.

BRASIL, 2002. **Código Civil Brasileiro**. Brasília, Distrito Federal. Disponível em <http://www.planalto.gov.br/CCIVIL/leis/2002/L10406.htm>. Acesso em: 03 fev. 2007.

BREWER, D. F. C.; NASH, M. J. The chinese wall security policy. In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY, 1989. **Proceedings...** Oakland, CA, USA, 1989. p. 206–214.

CEREDA, P. R. M.; ZORZO, S. D. Formalismo de modelo de controle de acesso utilizando autômato adaptativo. **IEEE Latin America Transactions**, 2008a. A ser publicado.

CEREDA, P. R. M.; ZORZO, S. D. SPA^3: um framework para privacidade. In: II WTA 2008: SEGUNDO WORKSHOP DE TECNOLOGIA ADAPTATIVA, 2008b. **Proceedings...** Escola Politécnica da Universidade de São Paulo, São Paulo, Brasil, 2008b. p. 9–13.

CHAUM, D. L. Untraceable electronic mail, return addresses, and digital pseudonyms. **Commun. ACM**. v. 24, n. 2, p. 84–90, 1981.

CHOMSKY, N. Three models for the description of language. **IEEE Transactions on Information Theory**. v. 2, n. 3, p. 113–124, 1956.

CHOMSKY, N. **Syntactic structures**. The Hague: Mouton, 1957.

CLARK, D. D.; WILSON, D. R. A comparison of commercial and military computer security policies. In: IEEE SYMPOSIUM ON SECURITY AND PRIVACY, 1987. **Proceedings...** Oakland, CA, USA, 1987. p. 184–194.

COYLE, K., 1999. **A social analysis of the Platform for Privacy Preferences (P3P)**. The Platform for Privacy Preferences (P3P) Project. Disponível em <http://www.w3.org/P3P/>. Acesso em: 02 fev. 2007.

CRANOR, L. et al., 2002. **The Platform for Privacy Preferences 1.0 (P3P 1.0) Specification**. World Wide Web Consortium (W3C). Disponível em <http://www.w3.org/TR/P3P/>. Acesso em: 02 fev. 2007.

DAFA-ALLA, A. F. A. et al. PRBAC: An extended role based access control for privacy preserving data mining. In: 4TH ANNUAL ACIS INTERNATIONAL CONFERENCE ON COMPUTER AND INFORMATION SCIENCE, 2005. **Proceedings...** Washington, DC, USA, 2005. p. 68–73.

DZIERZAWSKI, D. Automatic access controls in the defense message system (DMS). In: IEEE MILITARY COMMUNICATIONS CONFERENCE – MILCOM, 1999. **Proceedings...** Atlantic City, NJ, USA, 1999. p. 1262–1266.

FERNANDES, C. H. **A Privacidade na Sociedade da Informação**. Disponível em <http://www.linux.ime.usp.br/~carloshf/0302-mac339/fase2/node2.html>. Acesso em: 13 jan. 2007.

FERRAIOLO, D. F.; KUHN, D. R. Role based access control. In: 15th National Computer Security Conference, 1992. Baltimore, MD, USA, 1992. p. 544–554.

FERRAIOLO, D. F.; KUHN, D. R.; CHANDRAMOULI, R. **Role-Based Access Control**. Norwood, MA, USA: Artech House Publishers, 2003.

FISCHER-HÜBNER, S. **IT-security and privacy: design and use of privacy-enhancing security mechanisms**. New York, NY, USA: Springer-Verlag New York, Inc., 2001.

FISCHER-HÜBNER, S.; YNGSTRÖM, L.; HOLVAST, J. Addressing vulnerability and privacy problems generated by the use of IT-security mechanisms. In: IFIP 12TH WORLD COMPUTER CONGRESS ON EDUCATION AND SOCIETY – INFORMATION PROCESSING '92, 1992. **Proceedings...** Amsterdam, The Netherlands, 1992. p. 314–321.

FRIEDMAN, B.; KHAN JR., P. H.; HOWE, D. C. Trust online. **Commun. ACM**. v. 43, n. 12, p. 34–40, 2000.

FTC, 2000. **Fair information practices in the electronic marketplace: a report to congress**. Federal Trade Commission.

GABBER, E. et al., 1997. **How to Make Personalized Web Browsing Simple, Secure and Anonymous**. Bell Laboratories, Lucent Technologies. Disponível em <http://www.bell-labs.com/project/lpwa/papers.html>. Acesso em: 13 jan. 2007.

GABBER, E. et al. Consistent, yet anonymous, web access with LPWA. **Commun. ACM**. v. 42, n. 2, p. 42–47, 1999.

GOLDBERG, I.; WAGNER, D.; BREWER, E. Privacy-enhancing technologies for the internet. In: 42ND IEEE SPRING COMPUTER CONFERENCE – COMPCON, 1997. **Proceedings...** San Jose, CA, USA, 1997. p. 103–110.

GOLDSCHLAG, D.; REED, M.; SYVERSON, P., 1996. **Anonymous Connections and Onion Routing**. Assurance Computer Systems, Naval Research Laboratory, Washington, DC.

GRAHAM, G. S.; DENNING, P. J. Protection – principles and practice. In: AFIPS SPRING JOINT COMPUTER CONFERENCE, 1972. **Proceedings...** Anaheim, CA, USA, 1972. p. 417–429.

GRANDE, R. E. **Sistema de Integração de Técnicas de Proteção de Privacidade que permitem Personalização**, 138 p. Dissertação (Mestrado). Departamento de Computação, Universidade Federal de São Carlos, São Carlos, 2006.

GUNTER, C. A.; MAY, M. J.; STUBBLEBINE, S. G. A formal privacy system and its application to location based services. In: PRIVACY ENHANCING TECHNOLOGIES, 2004. **Proceedings...** Toronto, Canadá, 2004. p. 256–282.

HARRISON, M. A.; RUZZO, W. L.; ULLMAN, J. D. Protection in operating systems. **Commun. ACM**, New York, NY, USA. v. 19, n. 8, p. 461–471, 1976.

HOFFMAN, J. Implementing RBAC on a type enforced system. In: ACSAC '97 – 13TH ANNUAL COMPUTER SECURITY APPLICATIONS CONFERENCE, 1997. **Proceedings...** Washington, DC, USA, 1997. p. 158.

HOPCROFT, J. E.; ULLMAN, J. D.; MOTWANI, R. **Introdução à teoria de Autômatos, Linguagens e Computação**, 2th. ed. Rio de Janeiro: Elsevier, 2002.

ISHITANI, L. **Uma Arquitetura para Controle de Privacidade na Web**, 92 p. Tese (Doutorado). Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Minas Gerais, 2003.

JOSHI, J. B. D. et al. Digital government security infrastructure design challenges. **IEEE Computer**. v. 34, n. 2, p. 66–72, 2001a.

JOSHI, J. B. D. et al. Security models for web-based applications. **Commun. ACM**. v. 44, n. 2, p. 38–44, 2001b.

LAMPSON, B. Protection. In: 5TH ANNUAL PRINCETON CONFERENCE ON INFORMATION SCIENCES AND SYSTEMS, 1971. **Proceedings...** Princeton University, Princeton, NJ, USA, 1971. p. 437–443.

LEWIS, H. R.; PAPADIMITRIOU, C. H. **Elements of the theory of computation**. New Jersey: Prentice-Hall, Inc., 1998.

LI, N.; MITCHELL, J. C.; WINSBOROUGH, W. H. Beyond proof-of-compliance: security analysis in trust management. **Journal of the ACM**, New York, NY, USA. v. 52, n. 3, p. 474–514, 2005.

LOBATO, L. L.; ZORZO, S. D. Avaliação dos mecanismos de privacidade e personalização na web. In: XXXII CONFERÊNCIA LATINOAMERICANA DE INFORMÁTICA – CLEI 2006, 2006. **Proceedings...** Santiago, Chile, 2006.

MAY, M. J.; GUNTER, C. A.; LEE, I. Privacy APIs: Access control techniques to analyze and verify legal privacy policies. In: CSFW '06 – 19TH IEEE WORKSHOP ON COMPUTER SECURITY FOUNDATIONS, 2006. **Proceedings...** Washington, DC, USA, 2006. p. 85–97.

MOTTA, G. H. M. B. **Um modelo de autorização contextual para o controle de acesso ao prontuário eletrônico do paciente em ambientes abertos e distribuídos**, 212 p. Tese (Doutorado). Escola Politécnica da Universidade de São Paulo, São Paulo, 2004.

MOTTA, G. H. M. B.; FURUIE, S. S. A contextual role-based access control authorization model for electronic patient record. **IEEE Transactions on Information Technology in Biomedicine**. v. 7, p. 202–207, 2003.

NCSC, 1985. **Department of Defense Trusted Computer Systems Evaluation Criteria**. DoD National Computer Security Center. DoD 5200.28-STD.

NETO, J. J. **Contribuições à Metodologia de Construção de Compiladores**, 272 p. Tese (Livre Docência). Escola Politécnica da Universidade de São Paulo, São Paulo, 1993.

NETO, J. J. Adaptive automata for context-dependent languages. **SIGPLAN Notices**, New York, NY, USA. v. 29, n. 9, p. 115–124, 1994.

NETO, J. J. Solving complex problems efficiently with adaptive automata. **Lecture Notes in Computer Science**. v. 2088, p. 340–342, 2001.

NETO, J. J. Um levantamento da evolução da adaptatividade e da tecnologia adaptativa. **IEEE Latin America Transactions**. v. 5, n. 7, p. 496–505, 2007.

NETO, J. J.; IWAI, M. K. Adaptive automata for syntax learning. In: XXIV CONFERÊNCIA LATIONAMERICANA DE INFORMÁTICA – CLEI 1998, 1998. **Proceedings...** Quito, Equador, 1998.

OECD, 1980. **Recommendation of the council concerning guidelines governing the protection of privacy and transborder flows of personal data**. Organization for Economic Co-Operation and Development – OECD. Disponível em <http://www.datenschutz-berlin.de/gesetze/internat/>. Acesso em: 02 fev. 2007.

OSBORN, S.; SANDHU, R.; MUNAWER, Q. Configuring role-based access control to enforce mandatory and discretionary access control policies. **ACM Transactions on Information and System Security**, New York, NY, USA. v. 3, n. 2, p. 85–106, 2000.

PFITZMANN, A.; KÖHNTOPP, M. Anonymity, unobservability, and pseudeonymity – a proposal for terminology. In: INTERNATIONAL WORKSHOP ON DESIGNING PRIVACY ENHANCING TECHNOLOGIES, 2001. **Proceedings...** New York, NY, USA, 2001. p. 1–9.

RAY, I.; KUMAR, M.; YU, L. **LRBAC: A Location-Aware Role-Based Access Control Model**. Springer Berlin / Heidelberg, 2006.

REITER, M. K.; RUBIN, A. D., 1997. **Crowds: Anonymity for Web Transactions**. AT & T Labs – Research.

ROCHA, R. L. A.; NETO, J. J. Autômato adaptativo, limites e complexidade em comparação com máquina de Turing. In: SECOND CONGRESS OF LOGIC APPLIED TO TECHNOLOGY – LAPTEC, 2001. **Proceedings...** São Paulo, Brasil, 2001. p. 33–48.

ROSENBERG, R. S. **The Social Impact of Computers**. Academic Press Inc., 1992.

SAMARATI, P.; VIMERCATI, S. C. Access control: Policies, models, and mechanisms. **Lecture Notes in Computer Science**. v. 2171, p. 137–196, 2001.

SANDHU, R. et al. Role-based access control models. **IEEE Computer**. v. 29, n. 2, p. 38–47, 1996.

SANDHU, R.; MUNAWER, Q. How to do discretionary access control using roles. In: RBAC '98: THIRD ACM WORKSHOP ON ROLE-BASED ACCESS CONTROL, 1998. **Proceedings...** New York, NY, USA, 1998. p. 47–54.

SHUBINA, A.; SMITH, S. Using caching for browsing anonymity. **ACM SIGEcom Exchanges**. v. 4, n. 2, 2003.

TELTZROW, M.; KOBSA, A. Communication of privacy and personalization in e-business. In: WORKSHOP WHOLES: A MULTIPLE VIEW OF INDIVIDUAL PRIVACY IN A NETWORKED WORLD, 2004. **Proceedings...** Stockholm, Sweden, 2004.

TRIPUNITARA, M. V.; LI, N. A theory for comparing the expressive power of access control models. **Journal of Computer Security (JCS)**. v. 15, n. 2, p. 231–272, 2007.

TURING, A. M. On computable numbers, with an application to the Entscheidungsproblem. In: LONDON MATHEMATICAL SOCIETY, 1936. **Proceedings...** London, UK, 1936. p. 230–265.

"É graça divina começar bem. Graça maior persistir na caminhada certa. Mas graça das graças é não desistir nunca."

DOM HÉLDER CÂMARA