Implementação de políticas UCON em um núcleo de sistema operacional

Rafael Coninck Teigão, Carlos Maziero, Altair Santin*

Programa de Pós-Graduação em Informática
Pontifícia Universidade Católica do Paraná
Curitiba PR – Brasil
{teigao,maziero,santin}@ppgia.pucpr.br

**apresentador*

Visão Geral

- **Contexto**: Há uma necessidade para novos modelos de controle de acesso, para tratar Gerência de Confiança, Privacidade e DRM.

- **Problema**: O modelo UCON$_{ABC}$ foi apenas definido matematicamente, não há ainda implementações completas.

- **Nossa abordagem**: Criar uma gramática que representa políticas UCON$_{ABC}$, implementá-la em um protótipo e implantá-lo no núcleo de um SO.

- **Resultados**: O protótipo implantado controla acesso e uso em um SO com sucesso.

Motivação e Definição do Problema

- A disponibilidade dos dados mudou,
  mas os controles atuais não acompanharam a mudança.

- Há uma necessidade para se gerenciar Confiança,
  Privacidade e DRM.

- Novos modelos devem suportar modelos clássicos
  (e.g. DAC, MAC e RBAC).

- O modelo de Controle de Uso foi apenas apresentado
  formalmente.

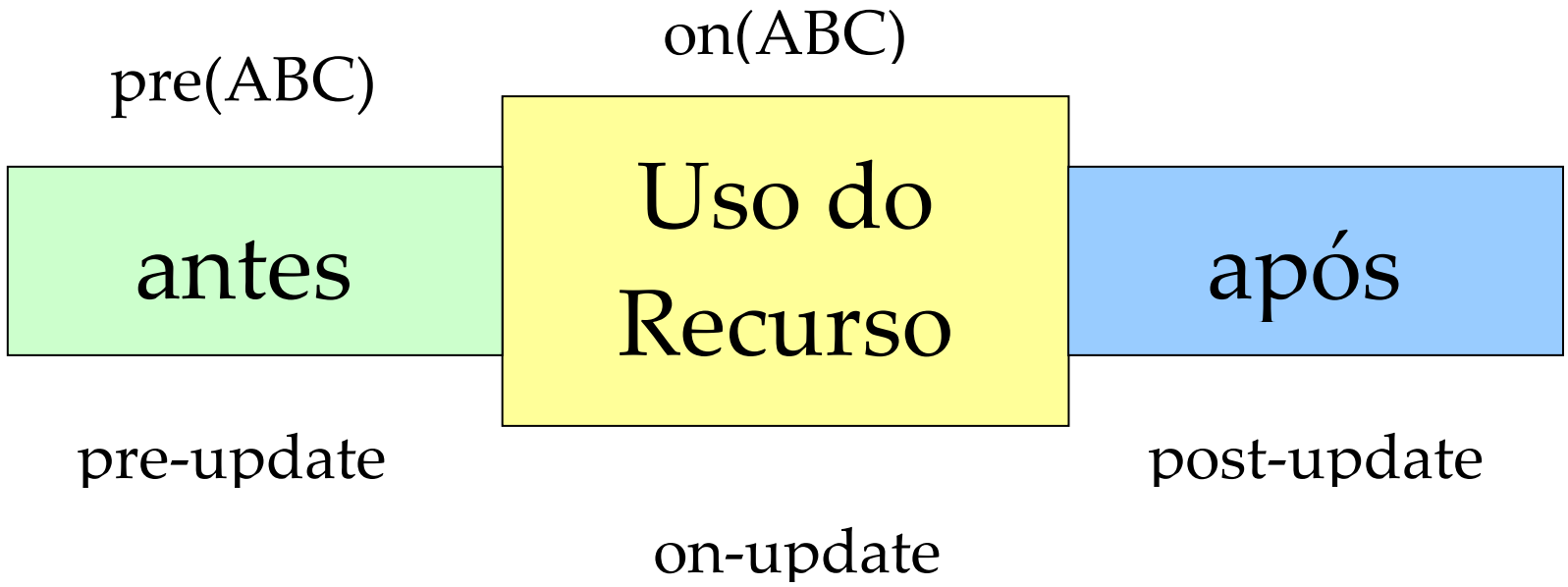
O Modelo de Controle de Uso UCON$_{ABC}$

- Uma ampliação da Matriz de Acesso
- Introduz mutabilidade de atributos
- Validação das permissões é contínua

- 3 fatores de decisão:
  - **Autorização**: controle de acesso tradicional
  - **oBrigação**: ações que o usuário deve executar antes ou durante o acesso
  - **Condição**: requisitos ambientais que podem ser considerados

Mutabilidade dos Atributos e Continuidade da Avaliação

Continuidade da avaliação



Mutabilidade do atributo

A Gramática Proposta

- A linguagem deve expressar claramente os predicados funcionais de autorização, obrigação e condição.
- O acesso deve apenas ser permitido após todas as regras avaliadas retornarem *true*. Qualquer regra que retorne *false* nega o uso. Um conjunto vazio de regras não nega o acesso.
- Deve ser possível adicionar novas regras sem modificar os atributos do usuário (desde que os atributos necessários estejam presentes).
- O processo de avaliação das regras deve ser eficiente.
- A proposta deve ser implementável e facilmente integrável aos ambientes existentes.

Símbolos Terminais e Operadores

Variáveis e símbolos de valores

Símbolo	**Tipo**	**Descrição**
$nome	inteiro, string	Uma variável, iniciada com o símbolo $, seguido pelo o nome único
Dígitos	inteiro	Um valor inteiro constante
[string_1, ..., string_n]	string	Um conjunto constante de strings
o$slot	inteiro	Palavra-chave para acessar valores de obrigação
c$*nome*	inteiro	Palavra-chave para acessar valores do recurso *nome*, utilizada em condições

Operadores

Operador	Tipo	Funcionalidade
=	Atribuição	Atribui um valor a uma variável
== != < > <= >=	Comparação	Compara operandos
& |	Lógico	AND e OR
size	Conjunto	Retorna o tamanho de um conjunto
+ - * /	Aritimético	Operações numéricas básicas
+ *	Conjunto	União e interseção de conjuntos
(*expressão*)	Precedência	A expressão mais interna deve ser avaliada antes da expressão fora dos parênteses
#		Inicia um comentário

Exemplo de Uso 1

Limita a quantidade de usuários, com UCONpreA_{13}preC_0

Atributos do Objeto:

```
$users     = 0
$max_day   = 10
$max_night = 20
$day_start = 8
$day_end   = 18
```

Políticas pos:

```
$users = $users - 1
```

Políticas pre:

```
( ( (c$time > $day_start) & (c$time < $day_end) ) &
    ($users < $max_day) ) |
( ( (c$time < $day_start) | (c$time > $day_end) ) &
    ($users < $max_night) )
$users = $users + 1
```

Exemplo de Uso 2

RBAC Restrito – Separação de Tarefas

Atributos do usuário:

```
$roles = director manager teller
$active_roles = manager teller
```

Atributos do objeto:

```
$required_roles = teller manager
```

Políticas pre:

```
size ($required_roles * $roles) != 0
$active_role = $active_role + ($required_roles * $roles)
size ($active_role * $required_roles) == 1
```

Políticas on:

```
size ($active_role * $required_roles) == 1
```

Exemplo de Uso 3

Limitação da quantidade de usuários concorrentes baseado no tempo de uso, com UCONonA[123]

Atributos do usuário:

```
$total_usage = 0
$last_action = 0
```

Atributos do objeto:

```
$max_users = 10
$max_usage = 6
$users     = 0
```

Políticas pre:

```
$users < $max_users
$users = $users + 1
$last_action = c$time
```

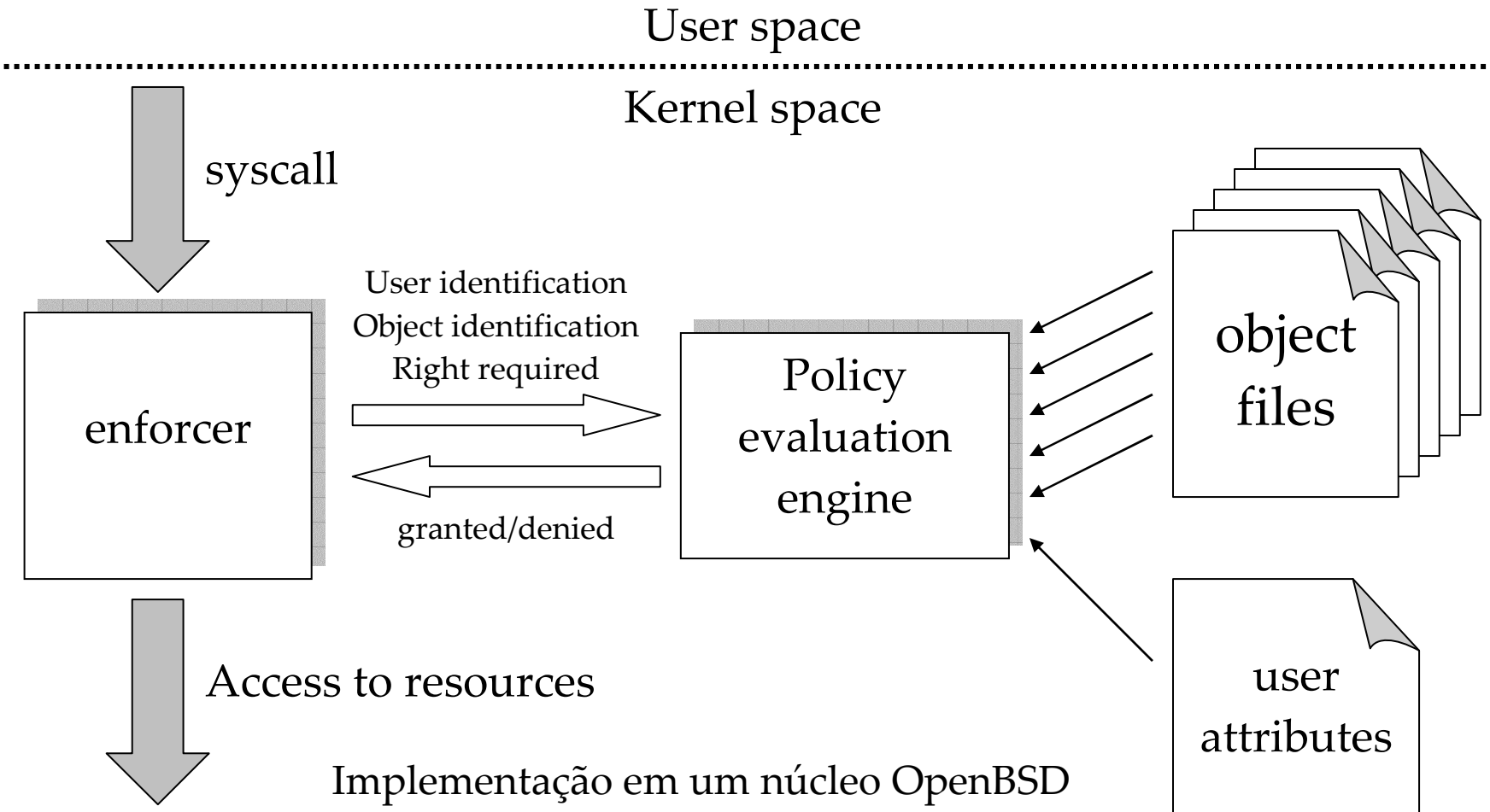
Políticas pos:

```
$total_usage = 0
$last_action = 0
$users = $users - 1
```

Políticas on:

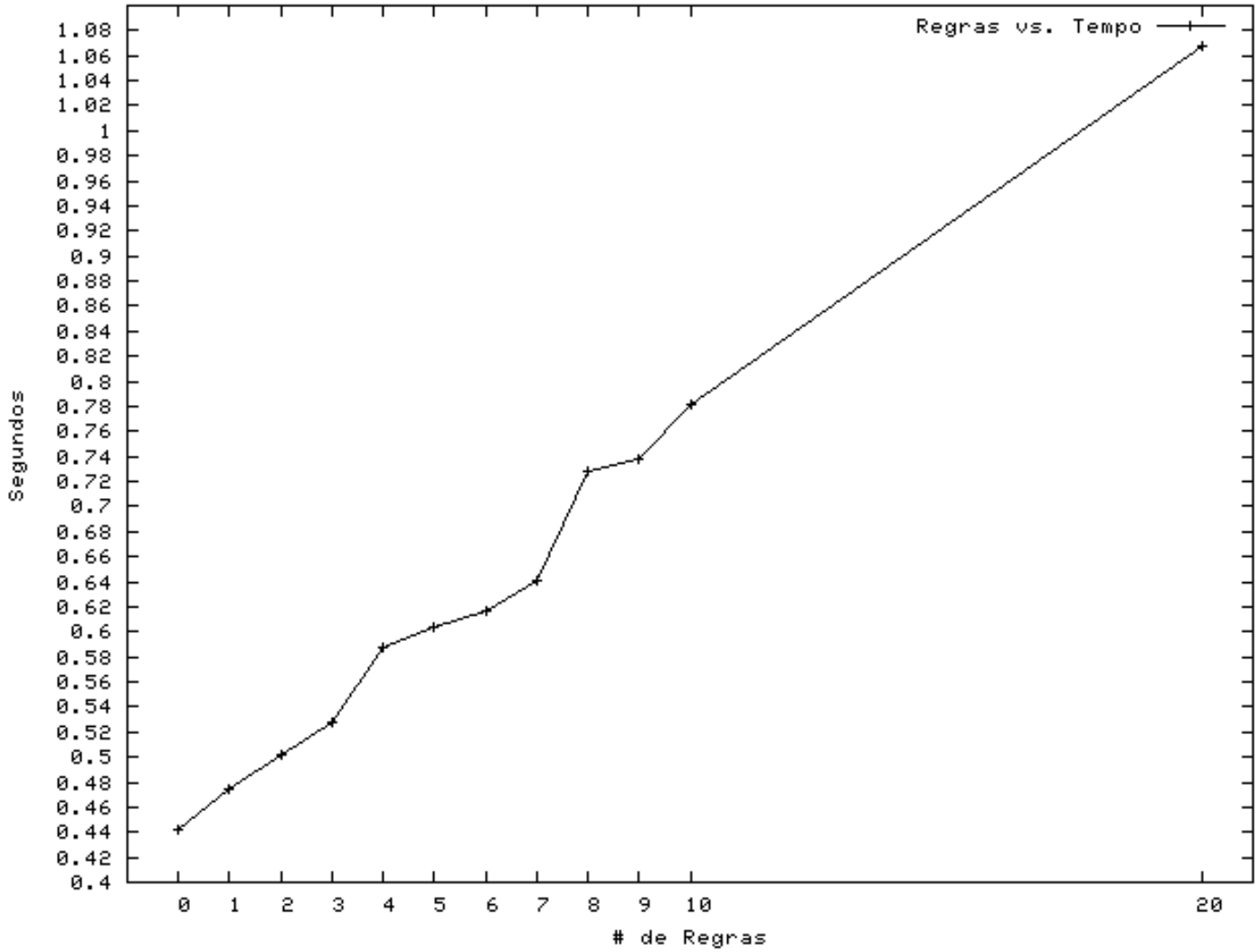
```
$total_usage = $total_usage + (c$time - $last_action)
$max_usage < $total_usage
$last_action = c$time
```

Implementação do Protótipo



Implementação em um núcleo OpenBSD
para controle de uso de arquivos
(intercepta chamadas open/close/read/write)

Resultados Obtidos



Relação Regras vs. Tempo para 18922 Chamadas

Trabalhos Futuros

- Compilação das políticas para melhorar o desempenho;
- Aumentar a quantidade de condições suportadas;
- Melhorar a representação das obrigações;
- Adaptar o mecanismo para suportar políticas de gestão autonômica de recursos.

Conclusão

Acreditamos que a gramática apresentada neste trabalho, juntamente com seu mecanismo de efetivação de políticas, pode facilitar a adoção do modelo UCONABC como uma solução viável para o controle de acesso, DRM e proteção de dados.