Temporal Aspects as Security Automata

Peter Hui <phui@students.depaul.edu>

James Riely <jriely@cs.depaul.edu>

FOAL 2006

March 21, 2006

1 – Temporal Pointcuts

1. μABC— Minimal Aspect Based Calculus

2 – Temporal Pointcuts

Problem: Trap calls to function `RestrictedFunction()`

AspectJ solution:

```
pointcut restrictedPC():
      call( * myClass.RestrictedFunction() );

around() : restrictedPC(){
    System.out.println( "Sorry!" );
   }
```

3 – Problem

Problem: Trap calls to function `RestrictedFunction()`, but *only* if called from function `BadCaller()`

AspectJ solution:

```
pointcut myCflowPC():
    cflow( call( * myClass.BadCaller() ) ) &&
       call( * myClass.RestrictedFunction() );

around() : myCflowPC(){
    System.out.println( "Sorry!" );
   }
```

4 – Problem

Problem: Trap calls to function `RestrictedFunction()`, but only if `SomethingBad()` called *at any time in the past*

AspectJ solution:

????

5 – Related Work

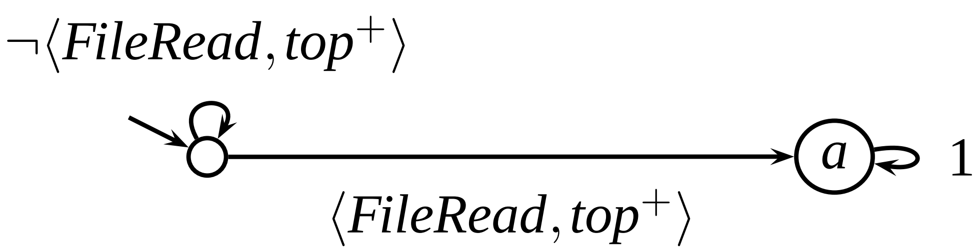
- Douence, Fradet, Südholt: *Stateful Aspects*
  1. Temporal pointcuts as regular expressions
  2. Variable binding within aspects
  3. Aspect interaction, conflict resolution
- Walker, Viggers: *Tracecuts*
  - Context-free temporal aspect specifications
- Allan, Avgustinov, et.al. : *Trace Matches with Free Variables*
  - Temporal aspects specified using regular expressions
  - Binding of free variables within the tracematch

6 – Related Work - Security Automata

- Schneider— Security Automata— *Enforceable Security Policies*
- Bauer, Ligatti, Walker— *Edit Automata*
- Bauer, Ligatti, Walker— *Polymer* (PLDI ’05)

7 – Our Approach

Read-Send example (Schneider)



a: “Error handling” advice— Fires if attempt is made to call FileSend

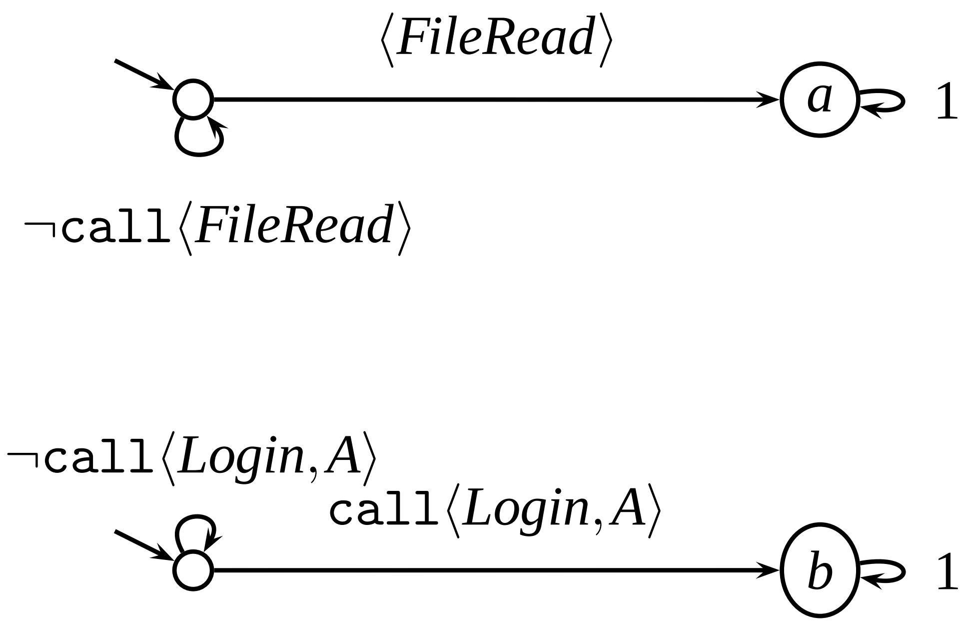
8 – Applications

- Security Policies
- Systems that:
  1. are modified dynamically (at runtime)
  2. don’t tolerate much downtime to accomodate policy modification

  e.g. phone switches, routers, servers, etc.

9 – Our Approach

For example, what does it mean to merge:

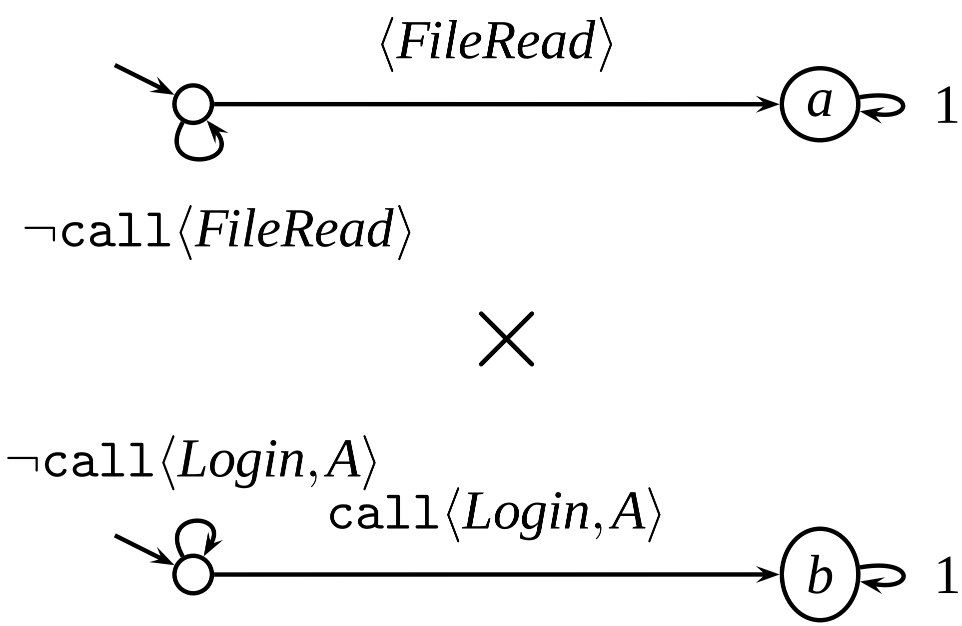


One interpretation: new policy "covers" same time period as old policy— but, requires recording of *entire* program history— impractical

Our solution: new policy takes effect at time of declaration

10 – Our Approach

- Merged automaton is simply the product automaton (standard product construction).
- (Slightly) tricky part: semantically, *What does this mean?*



11 – Syntax

a–e, u–w	Advice Names
f–t, x–z	Role Names
$\sigma, \rho, \theta ::= \langle \bar{p} \rangle$	Role tuple (atomic event)
α	Atomic Event Pointcut
	(boolean logic over atomic events)
ϕ, ψ, χ	Temporal Pointcuts
	(regexps over atomic pointcuts)

12 – Syntax

$D, E ::=$	Declarations
$\texttt{role}\ p < q$	Role declaration
$\mathsf{adv}\ a[\phi\,\alpha] = u(\bar{p})\ N$	Advice declaration *u bound to proceed,* *$\bar{p}$ bound to advised event*
$M ::=$	Terms
$\bar{D}; \bar{a}\langle\sigma\rangle$	Declarations, advised event

13 – Configurations

- In the history-based semantics: a configuration is

  $$\bar{\sigma}; \bar{D} \triangleright M$$

- In the automaton-based semantics: a configuration is

  $$\mathcal{A}; \Phi; \bar{D} \triangleright M$$

- Automaton $\mathcal{A}$ provides an abstraction of $\bar{\sigma}$
  - Φ is the current state in $\mathcal{A}$
  - $\bar{D}$ are the declared roles, advice
  - M is the current term
- Automaton $\mathcal{A}$ provides mechanism to encapsulate particular "flavor" of temporal aspects— e.g., can replace with PDA to use context-free specifications

14 – Example

On calling σ followed by an attempted ρ, trigger θ:

$$\triangleright \underbrace{\mathsf{adv}\ a[\sigma\,\rho] = \theta;}_{\textit{aspect decl}} \sigma; \rho$$

$$\begin{aligned}
&\longrightarrow \underbrace{\mathsf{adv}\ a[\sigma\,\rho] = \theta}\ \triangleright\ \sigma; \rho \\
&\longrightarrow \underbrace{\sigma}_{\textit{hist}}; \mathsf{adv}\ a[\underbrace{\sigma}_{\textit{hist}}\ \underbrace{\rho}_{\textit{evt}}] = \theta\ \triangleright\ \underbrace{\rho}_{\textit{evt}} \\
&\longrightarrow \sigma; \mathsf{adv}\ a[\sigma\,\rho] = \theta\ \triangleright\ \theta
\end{aligned}$$

15 – Example

Encoding of sequencing: $N;M$

- N constrained to end with `return` (encoded as $\texttt{call}\langle \mathit{continue}, p\rangle$ for role p).

$\rhd\ M;N$

$$\longrightarrow \rhd\ \underbrace{\mathit{role}\ c;}\underbrace{\mathsf{adv}\ \_[\langle c, {+}\mathit{top}\rangle] = (\_, x)\ M;}\underbrace{N\{^{c}/_{\mathsf{continue}}\}}$$

$$\longrightarrow^{2}\ \underbrace{\mathit{role}\ c;}\underbrace{\mathsf{adv}\ \_[\langle c, {+}\mathit{top}\rangle] = (\_, x)\ M}\ \rhd\ N\{^{c}/_{\mathsf{continue}}\}$$

$$\longrightarrow^{*}\ \underbrace{\mathit{role}\ c;}\underbrace{\mathsf{adv}\ \_[\langle c, {+}\mathit{top}\rangle] = (\_, x)\ M}\ \rhd\ \texttt{call}\langle c, p\rangle$$

$$\longrightarrow\ \underbrace{\mathit{role}\ c;}\underbrace{\mathsf{adv}\ \_[\langle c, {+}\mathit{top}\rangle] = (\_, x)\ M}\ \rhd\ M$$

16 – Semantics

(EVAL-ADV)

$$\frac{\overline{D} \ni \mathsf{adv}\ a = u(\overline{x})\ N}{\overline{D} \rhd \overline{b}, a\langle\overline{p}\rangle \longrightarrow \overline{D} \rhd N\{\overline{b}/u, \overline{p}/\overline{x}\}}$$

17 – Semantics

(EVAL-CALL)

$$\frac{[\overline{\ }a] = [a \mid \overline{D} \ni \mathsf{adv}\ a[\phi\,\alpha] \text{ and } \overline{D} \vdash \overline{\sigma}, \langle \overline{\ }p \rangle \ \mathsf{sat}\ \phi\,\alpha]}{\overline{\sigma}; \overline{D} \rhd \overline{b}, \mathsf{call}\langle \overline{\ }p \rangle \longrightarrow \overline{\sigma}; \overline{D} \rhd \overline{b}, \overline{\ }a\langle \overline{\ }p \rangle}$$

(EVAL-CALL)

$$\frac{[\overline{\ }a] = \left[a \;\middle|\; \begin{array}{l} \langle \Phi, (\overline{b}, a, \overline{b}') \rangle \in \mathcal{A} \\ \overline{D} \ni \mathsf{adv}\ a[\alpha] \\ \overline{D} \vdash \langle \overline{\ }p \rangle \ \mathsf{sat}\ \alpha \end{array} \right]}{\mathcal{A}; \Phi; \overline{D} \rhd \overline{b}, \mathsf{call}\langle \overline{\ }p \rangle \longrightarrow \mathcal{A}; \Phi; \overline{D} \rhd \overline{b}, \overline{\ }a\langle \overline{\ }p \rangle}$$

18 – Semantics

(EVAL-DEC-ADV)

$$\frac{}{\begin{array}{l}\quad\overline{\sigma};\overline{D} \rhd \mathsf{adv}\ a[\phi\,\alpha]{=}u(\overline{\ }x)\ N\ ;\ M\\ \longrightarrow \overline{\sigma};\overline{D},\mathsf{adv}\ a[1^{|\overline{\sigma}|}\,\phi\,\alpha]{=}u(\overline{\ }x)\ N \rhd M\end{array}}$$

(EVAL-DEC-ADV)

$$\frac{}{\begin{array}{l}\quad\mathcal{A};\Phi;\overline{D} \rhd \mathsf{adv}\ a[\phi\,\alpha]{=}u(\overline{\ }x)\ N\ ;\ M\\ \longrightarrow \nu(\mathcal{A},\phi,a);\langle\Phi,\phi\rangle;\overline{D},(\mathsf{adv}\ a[\alpha]{=}u(\overline{\ }x)\ N) \rhd M\end{array}}$$

$\nu(\mathcal{A},\phi,a) \triangleq$ Product automaton $\mathcal{A}\times\phi$, with advice a attached to ϕ's final states

19 – End