Information Flow Security for Asynchronous, Distributed, and Mobile Applications

Felipe LUNA DEL AGUILA

OASIS Project
INRIA Sophia Antipolis
CNRS - I3S - Université Nice–Sophia Antipolis

Agenda

- Introduction

Agenda

- Introduction
- Context (*informal* and *formal* perspectives)

Agenda

- Introduction
- Context (*informal* and *formal* perspectives)
  - ProActive

Agenda

- Introduction
- Context (*informal* and *formal* perspectives)
  - ProActive
  - ASP calculus and communication reduction rules

Agenda

- Introduction
- Context ($informal$ and $formal$ perspectives)
  - ProActive
  - ASP calculus and communication reduction rules
- Objectives

Agenda

- Introduction
- Context (*informal* and *formal* perspectives)
  - ProActive
  - ASP calculus and communication reduction rules
- Objectives
- Related Security Mechanisms

Agenda

- Introduction
- Context (*informal* and *formal* perspectives)
  - ProActive
  - ASP calculus and communication reduction rules
- Objectives
- Related Security Mechanisms
- The ASP Security Model

Agenda

- Introduction
- Context ($informal$ and $formal$ perspectives)
  - ProActive
  - ASP calculus and communication reduction rules
- Objectives
- Related Security Mechanisms
- The ASP Security Model
- Implementation of the Security Model

Agenda

- Introduction
- Context (*informal* and *formal* perspectives)
  - ProActive
  - ASP calculus and communication reduction rules
- Objectives
- Related Security Mechanisms
- The ASP Security Model
- Implementation of the Security Model
- Conclusion

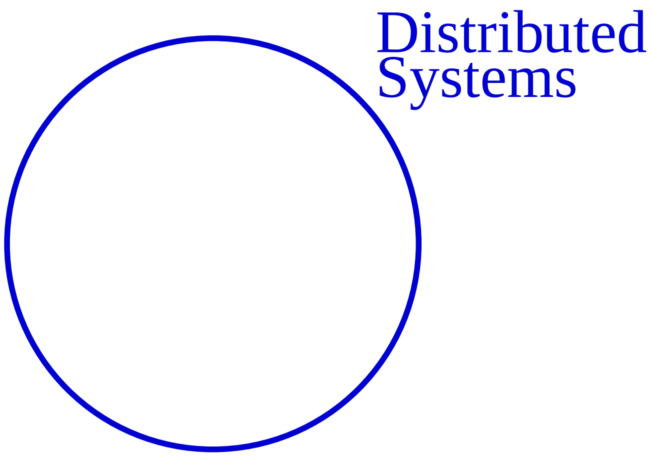
Introduction

Agenda

Recent paradigms

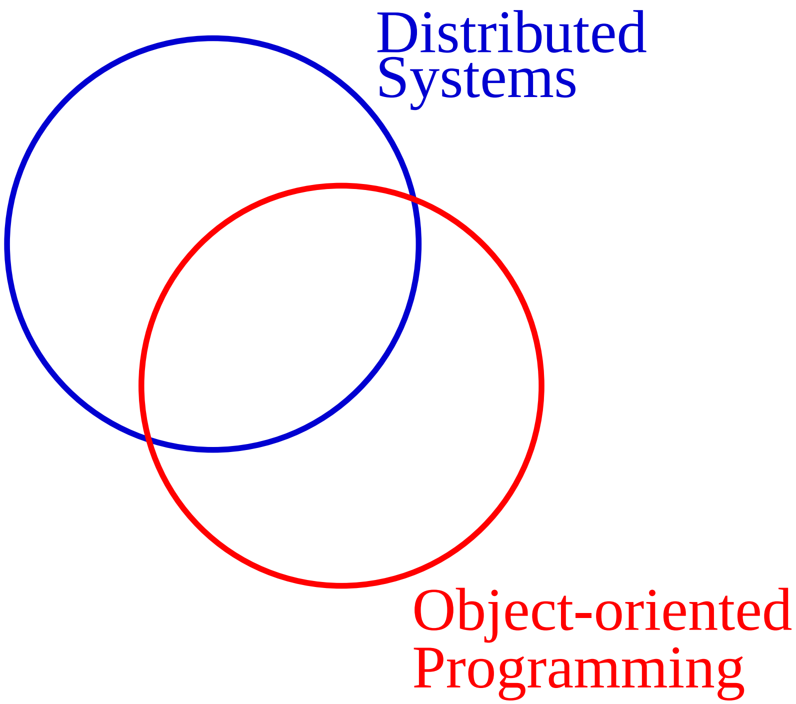
Introduction

Recent paradigms



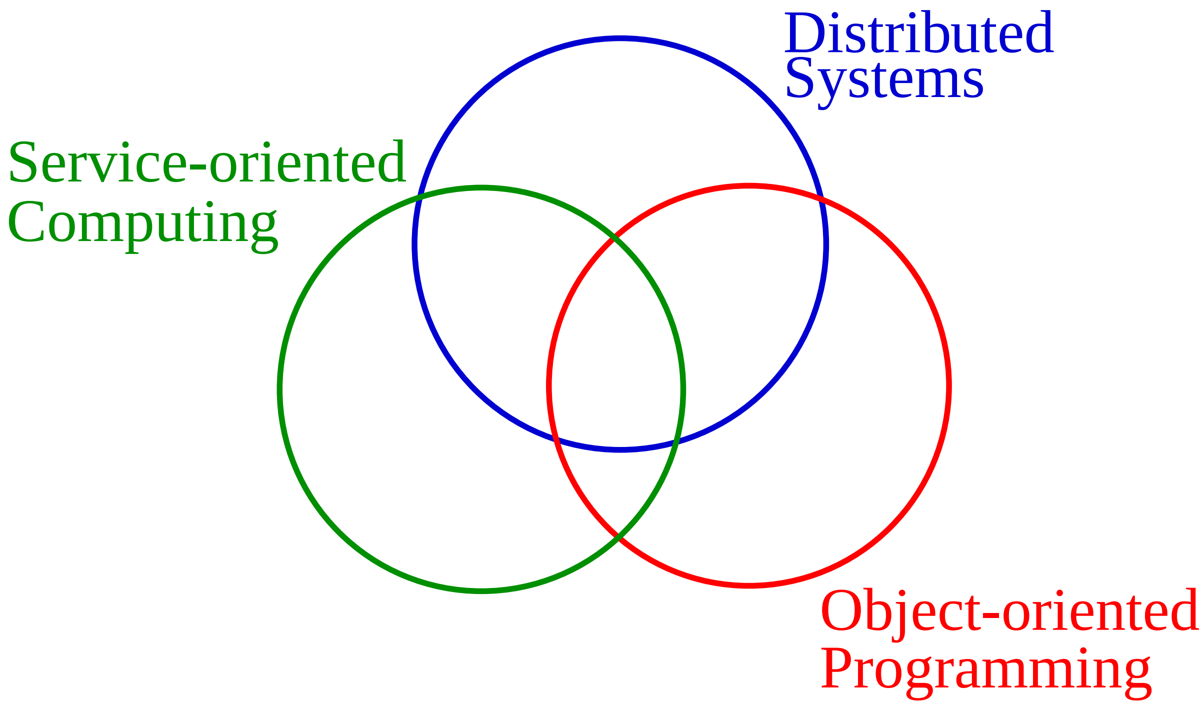
Introduction

Recent paradigms



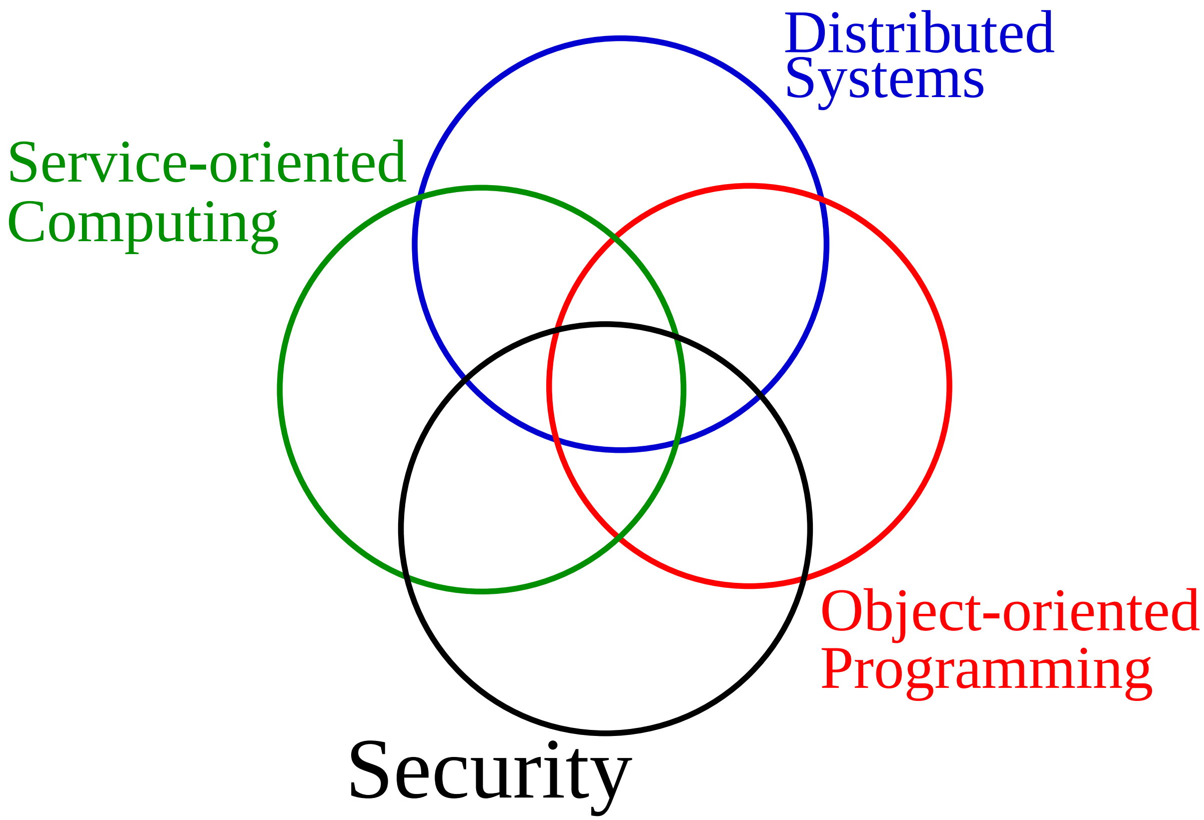
Introduction

Recent paradigms



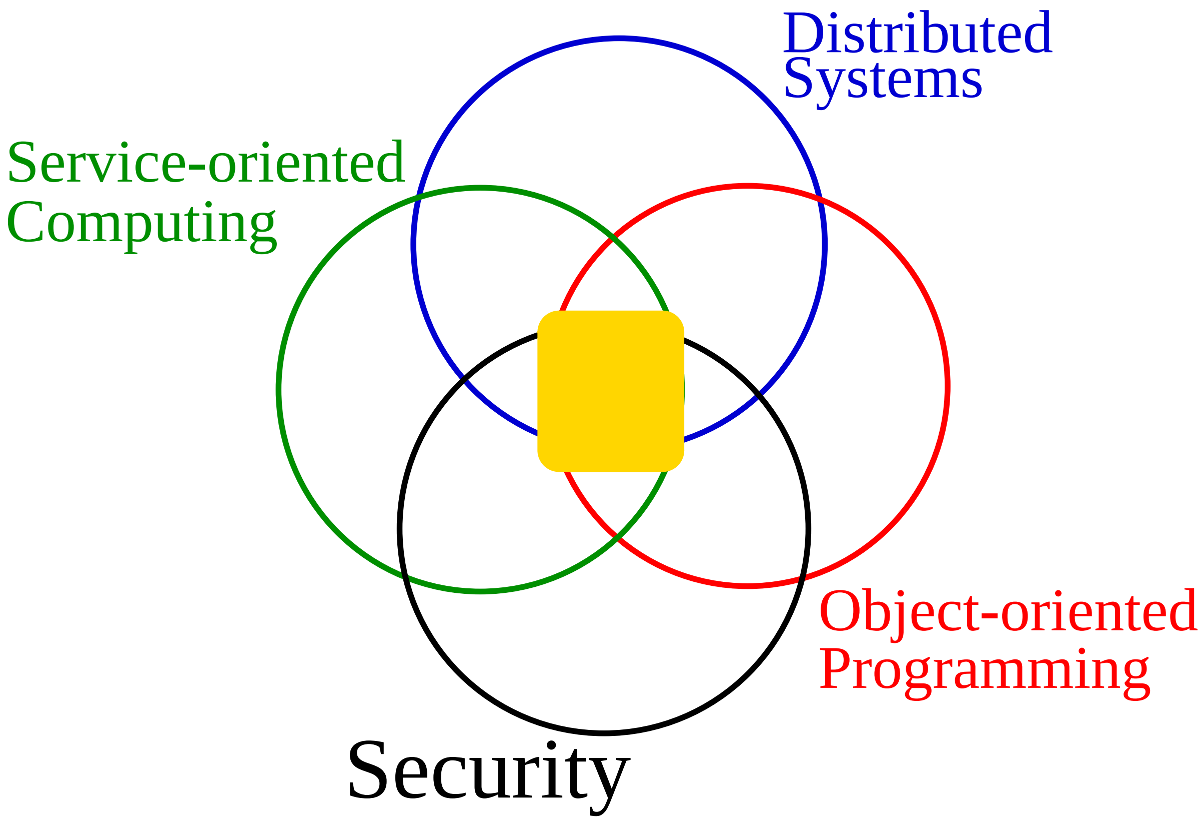
Introduction

Recent paradigms



Introduction

Recent paradigms



Security focused specifically on *Information Flow*

Context

ProActive main characteristics

- Middleware library for distributed applications

Context

ProActive main characteristics

- Middleware library for distributed applications
- 100% Java

Context

ProActive main characteristics

- Middleware library for distributed applications
- 100% Java
- Existence of passive and *active* objects

Context

ProActive main characteristics

- Middleware library for distributed applications
- 100% Java
- Existence of passive and *active* objects
- Asynchronous communications between *active* objects

Context

ProActive main characteristics

- Middleware library for distributed applications
- 100% Java
- Existence of passive and *active* objects
- Asynchronous communications between *active* objects
  - Principle of *wait-by-necessity* and *futures*:
    1. future reference

Context

ProActive main characteristics

- Middleware library for distributed applications
- 100% Java
- Existence of passive and *active* objects
- Asynchronous communications between *active* objects
  - Principle of *wait-by-necessity* and *futures*:
    1. future reference
       (ex.: http://www.anysite.com/anypage.html)

Context

ProActive main characteristics

- Middleware library for distributed applications
- 100% Java
- Existence of passive and *active* objects
- Asynchronous communications between *active* objects
  - Principle of *wait-by-necessity* and *futures*:
    1. future reference
       (ex.: http://www.anysite.com/anypage.html)
    2. future value

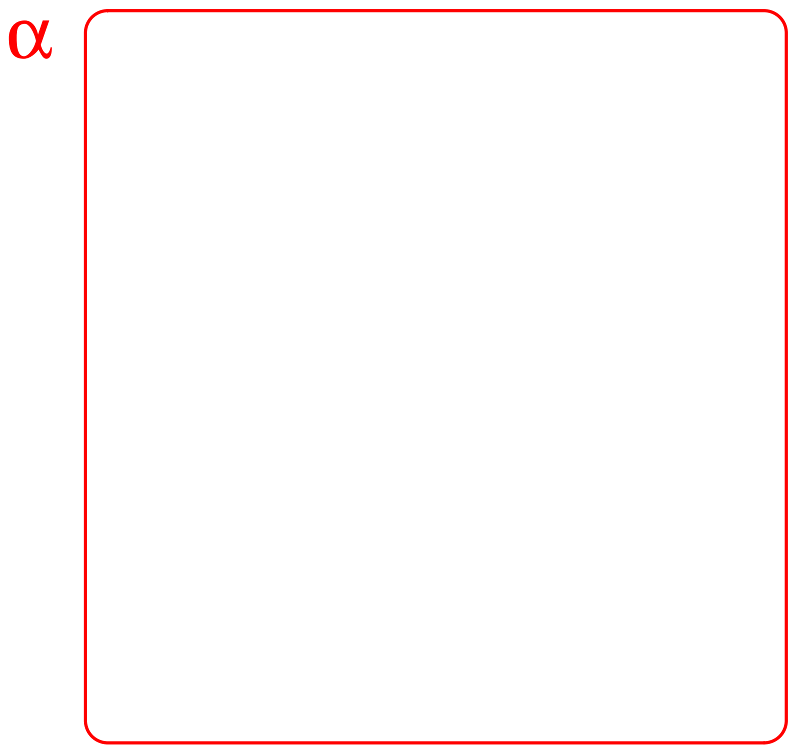
Context

ProActive main characteristics

- Middleware library for distributed applications
- 100% Java
- Existence of passive and *active* objects
- Asynchronous communications between *active* objects
  - Principle of *wait-by-necessity* and *futures*:
    1. future reference
       (ex.: http://www.anysite.com/anypage.html)
    2. future value
       (ex.: HTML error: 404 Not Authorized)

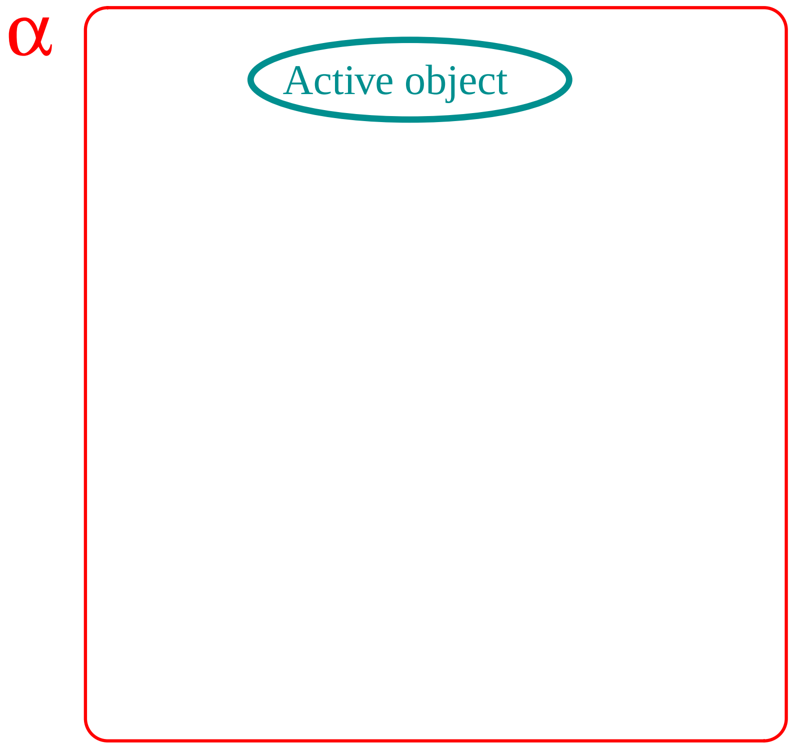
ASP semantics and ProActive

- The ASP language entities:
  - activity α



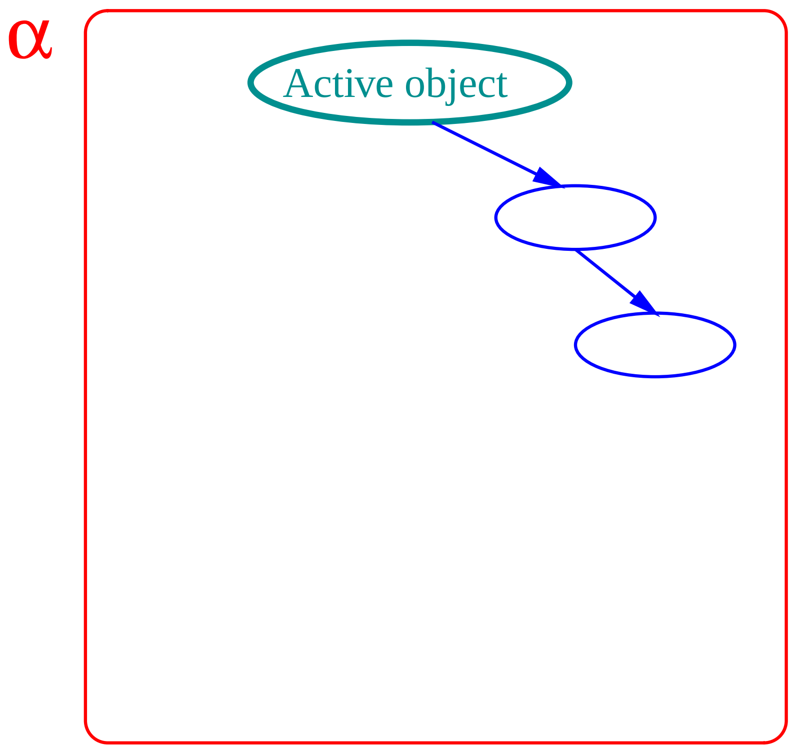
ASP semantics and ProActive

- The ASP language entities:
  - activity α, active object a



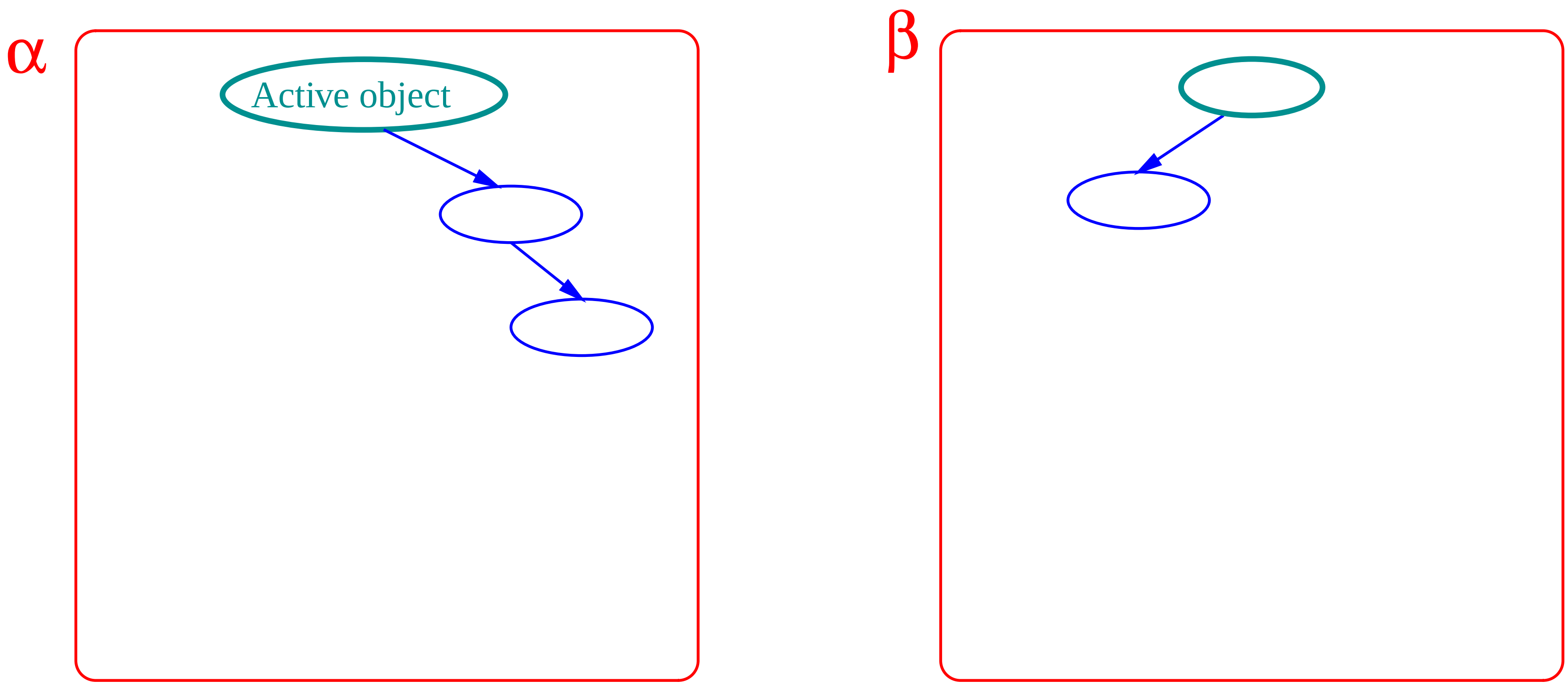
ASP semantics and ProActive

- The ASP language entities:
  - activity α, active object a, passive objects



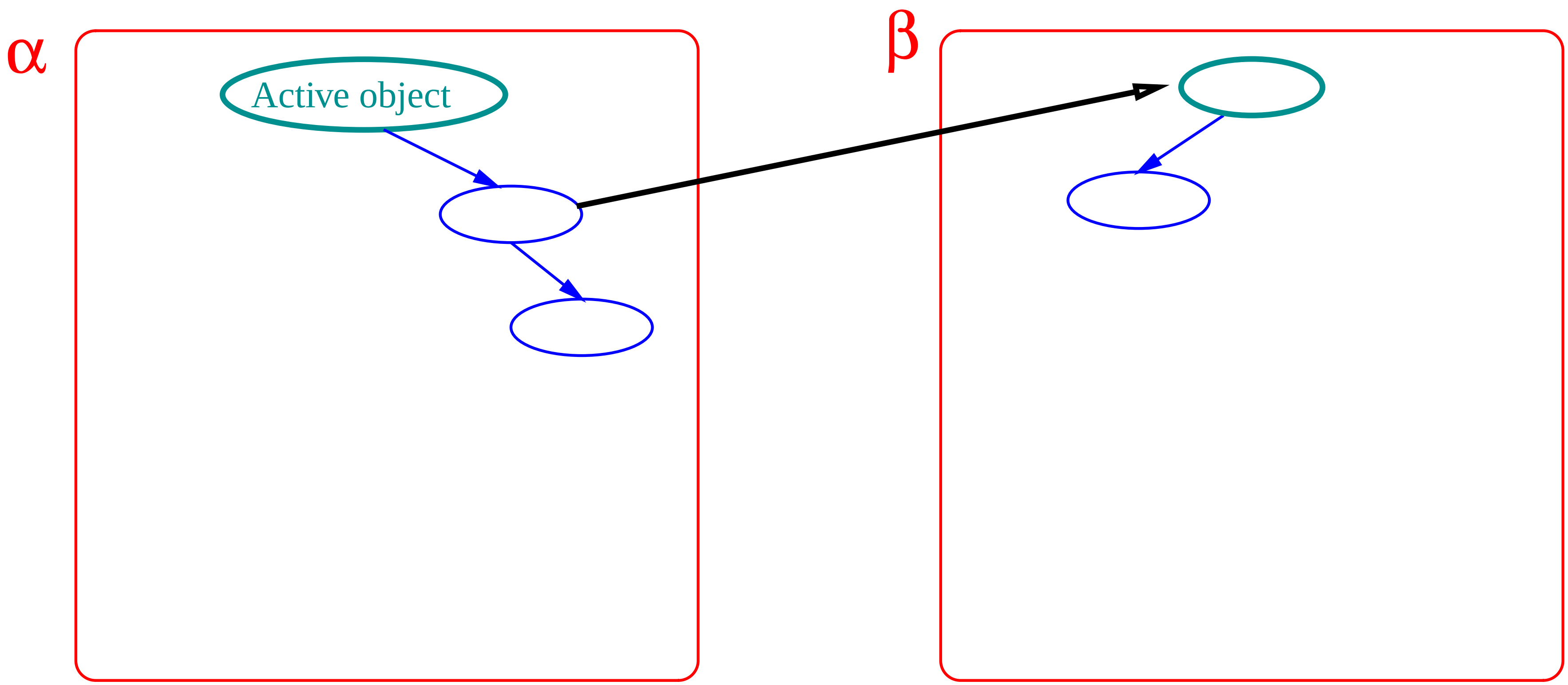
ASP semantics and ProActive

- The ASP language entities:
  - activity α, active object a, passive objects, activity β



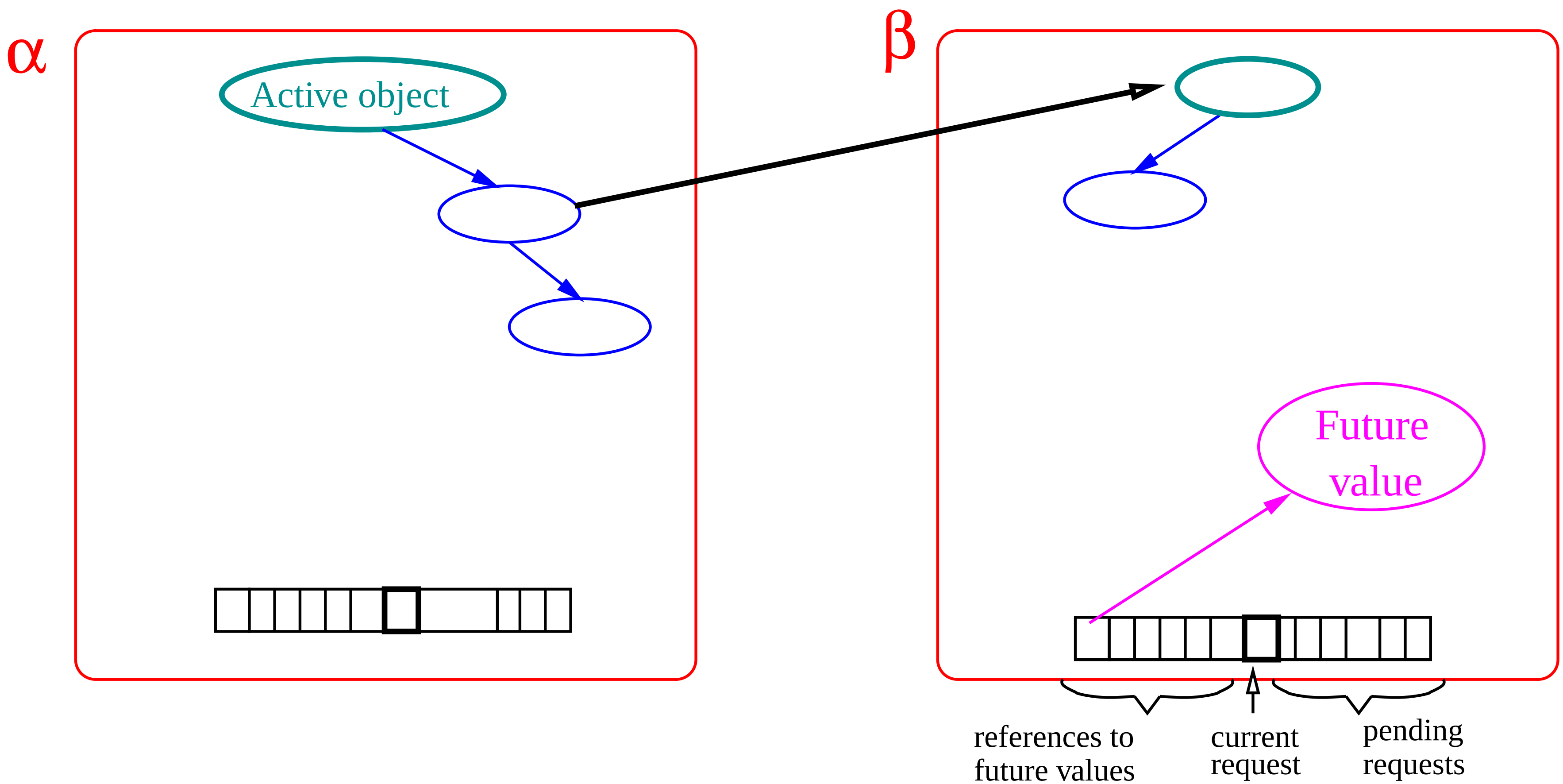
ASP semantics and ProActive

- The ASP language entities:
  - activity α, active object a, passive objects, activity β, active object reference $AO(\alpha)$



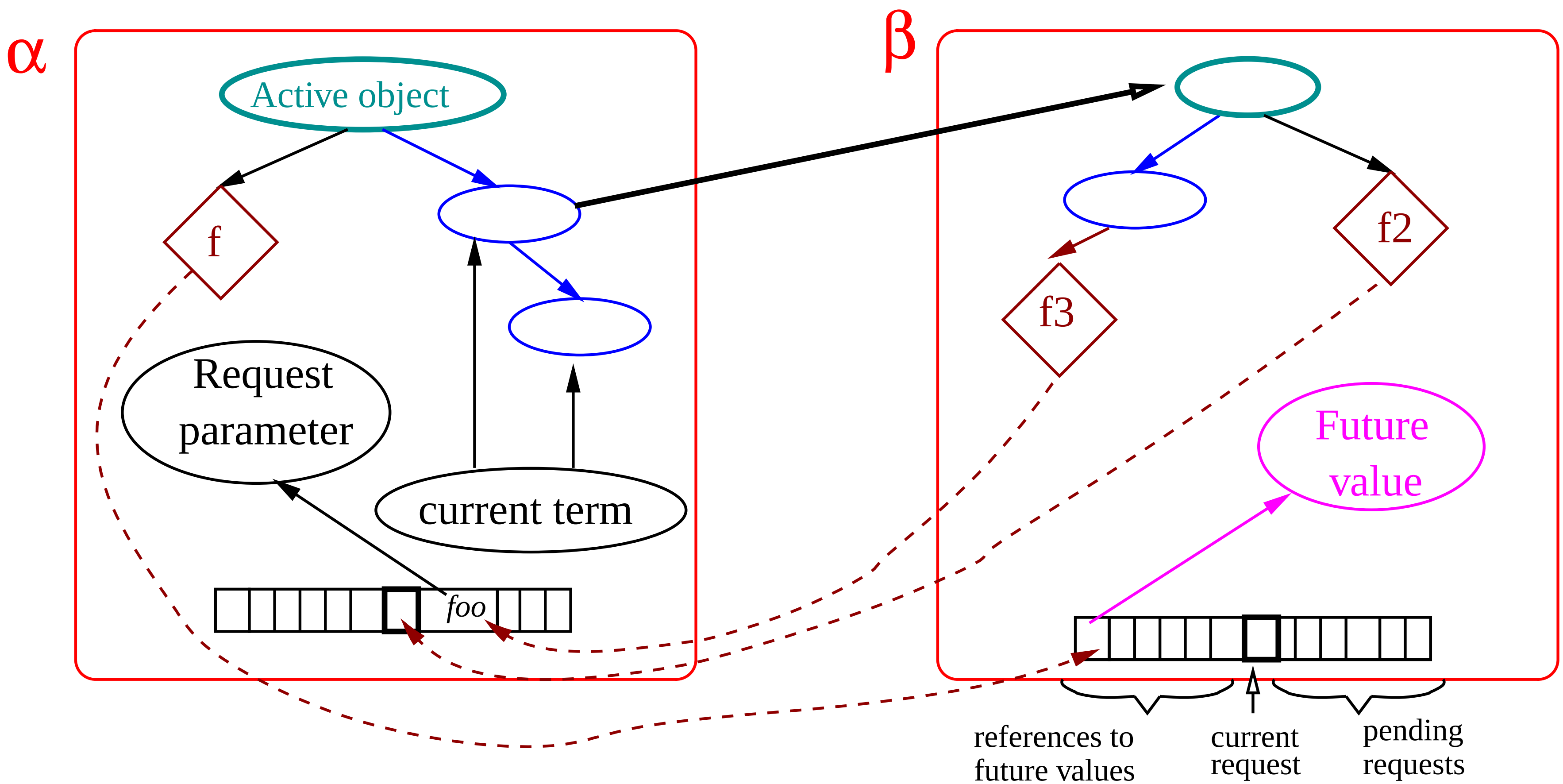
ASP semantics and ProActive

- The ASP language entities:
  - activity α, active object a, passive objects, activity β, active object reference $AO(\alpha)$, future $f_i^{\alpha\to\beta}$ and request queue (with pending, current, and completed requests)



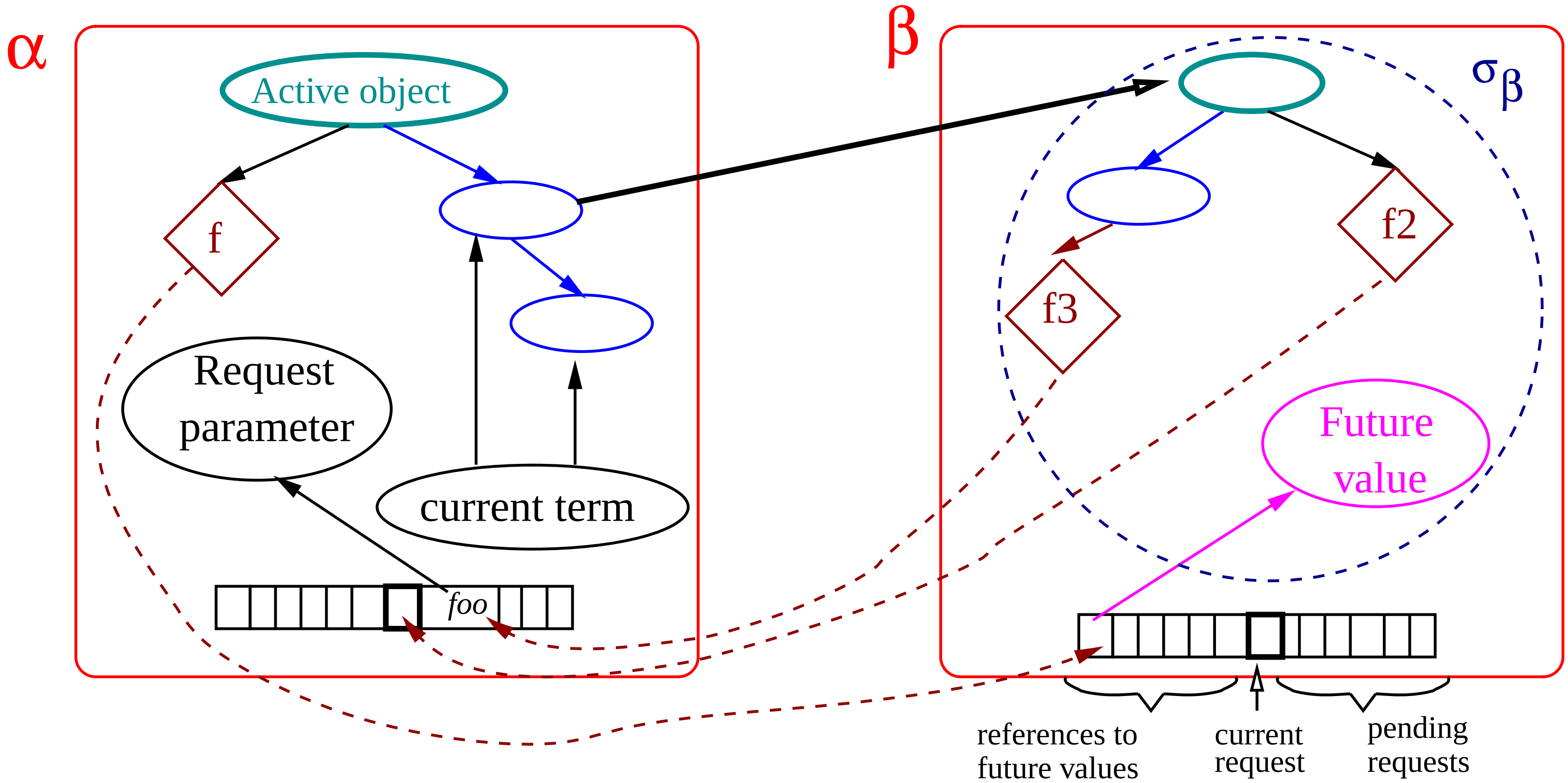
ASP semantics and ProActive

- The ASP language entities:
  - activity α, active object a, passive objects, activity β, active object reference $AO(\alpha)$, future $f_i^{\alpha\to\beta}$ and request queue (with pending, current, and completed requests), future references $fut(f_i^{\alpha\to\beta})$



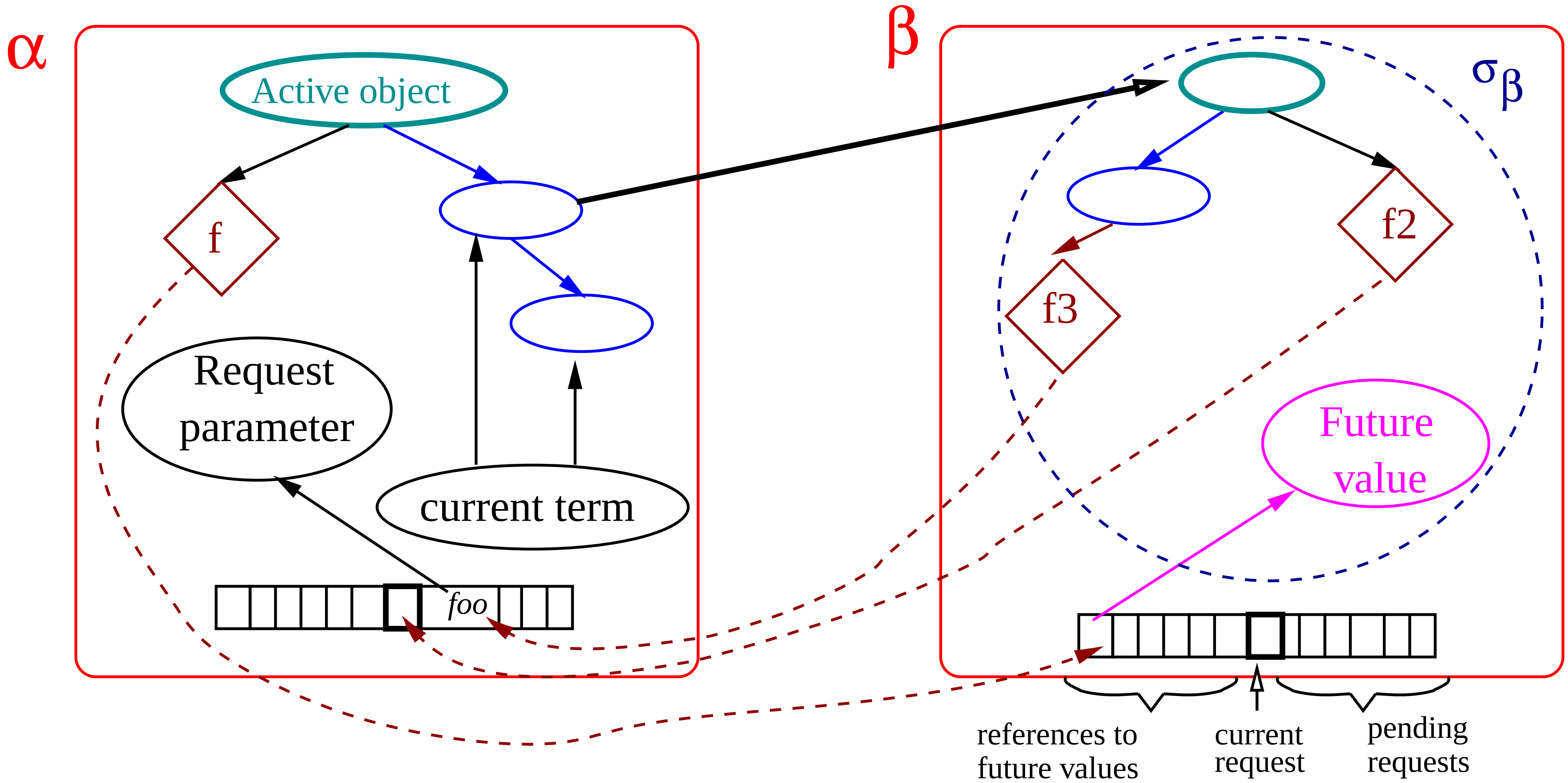
ASP semantics and ProActive

- The ASP language entities:
  - activity α, active object a, passive objects, activity β, active object reference $AO(\alpha)$, future $f_i^{\alpha\to\beta}$ and request queue (with pending, current, and completed requests), future references $fut(f_i^{\alpha\to\beta})$, and store σ



ASP semantics and ProActive

- The ASP language entities:
  - activity α, active object a, passive objects, activity β, active object reference $AO(\alpha)$, future $f_i^{\alpha\to\beta}$ and request queue (with pending, current, and completed requests), future references $fut(f_i^{\alpha\to\beta})$, and store σ



- Parallel configurations are then of the form: P,Q ::= $\alpha\ [a;\sigma;\iota;F;R;f] \parallel \beta\ [\cdots] \parallel \cdots$

ASP parallel reduction rules

ASP parallel reduction rules

$$\frac{(a,\sigma)\rightarrow_S(a',\sigma') \qquad \rightarrow_S \text{ does not clone a future}}{\alpha[a;\sigma;\iota;F;R;f] \parallel P \longrightarrow \alpha[a';\sigma';\iota;F;R;f] \parallel P}\ (\text{LOCAL})$$

$$\frac{\gamma \text{ fresh activity} \qquad \iota' \not\in dom(\sigma) \qquad \sigma' = \{\iota' \mapsto AO(\gamma)\} :: \sigma \qquad \sigma_\gamma = copy(\iota'',\sigma)}{\alpha[\mathcal{R}[Active(\iota'',m_j)];\sigma;\iota;F;R;f] \parallel P \longrightarrow \alpha[\mathcal{R}[\iota'];\sigma';\iota;F;R;f] \parallel \gamma[\iota''.m_j();\sigma_\gamma;\iota'';\emptyset;\emptyset;\emptyset] \parallel P}\ (\text{NEWACT})$$

$$\frac{\begin{array}{c}\sigma_\alpha(\iota) = AO(\beta) \qquad \iota'' \not\in dom(\sigma_\beta) \qquad f_i^{\alpha\rightarrow\beta} \text{ new future} \qquad \iota_f \not\in dom(\sigma_\alpha)\\ \sigma'_\beta = Copy\&Merge(\sigma_\alpha,\iota'\ ;\ \sigma_\beta,\iota'') \qquad \sigma'_\alpha = \{\iota_f \mapsto fut(f_i^{\alpha\rightarrow\beta})\} :: \sigma_\alpha\end{array}}{\begin{array}{c}\alpha[\mathcal{R}[\iota.m_j(\iota')];\sigma_\alpha;\iota_\alpha;F_\alpha;R_\alpha;f_\alpha] \parallel \beta[a_\beta;\sigma_\beta;\iota_\beta;F_\beta;R_\beta;f_\beta] \parallel P \longrightarrow\\ \alpha[\mathcal{R}[\iota_f];\sigma'_\alpha;\iota_\alpha;F_\alpha;R_\alpha;f_\alpha] \parallel \beta[a_\beta;\sigma'_\beta;\iota_\beta;F_\beta;R_\beta :: [m_j;\iota'';f_i^{\alpha\rightarrow\beta}];f_\beta] \parallel P\end{array}}\ (\text{REQUEST})$$

$$\frac{R = R' :: [m_j;\iota_r;f'] :: R'' \qquad m_j \in M \qquad \forall m \in M,\ m \not\in R'}{\alpha[\mathcal{R}[Serve(M)];\sigma;\iota;F;R;f] \parallel P \longrightarrow \alpha[\iota.m_j(\iota_r) \Uparrow f, \mathcal{R}[[]];\sigma;\iota;F;R' :: R'';f'] \parallel P}\ (\text{SERVE})$$

$$\frac{\iota' \not\in dom(\sigma) \qquad F' = F :: \{f \mapsto \iota'\} \qquad \sigma' = Copy\&Merge(\sigma,\iota\ ;\ \sigma,\iota')}{\alpha[\iota \Uparrow f', a;\sigma;\iota;F;R;f] \parallel P \longrightarrow \alpha[a;\sigma';\iota;F';R;f'] \parallel P}\ (\text{ENDSERVICE})$$

$$\frac{\sigma_\alpha(\iota) = fut(f_i^{\gamma\rightarrow\beta}) \qquad F_\beta(f_i^{\gamma\rightarrow\beta}) = \iota_f \qquad \sigma'_\alpha = Copy\&Merge(\sigma_\beta,\iota_f\ ;\ \sigma_\alpha,\iota)}{\begin{array}{c}\alpha[a_\alpha;\sigma_\alpha;\iota_\alpha;F_\alpha;R_\alpha;f_\alpha] \parallel \beta[a_\beta;\sigma_\beta;\iota_\beta;F_\beta;R_\beta;f_\beta] \parallel P \longrightarrow\\ \alpha[a_\alpha;\sigma'_\alpha;\iota_\alpha;F_\alpha;R_\alpha;f_\alpha] \parallel \beta[a_\beta;\sigma_\beta;\iota_\beta;F_\beta;R_\beta;f_\beta] \parallel P\end{array}}\ (\text{REPLY})$$

ASP parallel reduction rules

$$\frac{(a,\sigma) \to_S (a',\sigma') \qquad \to_S \text{ does not clone a future}}{\alpha[a;\sigma;\iota;F;R;f] \parallel P \longrightarrow \alpha[a';\sigma';\iota;F;R;f] \parallel P} \ (\text{LOCAL})$$

$$\frac{\gamma \text{ fresh activity} \qquad \iota' \notin dom(\sigma) \qquad \sigma' = \{\iota' \mapsto AO(\gamma)\} :: \sigma \qquad \sigma_\gamma = copy(\iota'',\sigma)}{\alpha[\mathcal{R}[Active(\iota'',m_j)];\sigma;\iota;F;R;f] \parallel P \longrightarrow \alpha[\mathcal{R}[\iota'];\sigma';\iota;F;R;f] \parallel \gamma[\iota''.m_j();\sigma_\gamma;\iota'';\emptyset;\emptyset;\emptyset] \parallel P} \ (\text{NEWACT})$$

$$\frac{\begin{array}{c}\sigma_\alpha(\iota) = AO(\beta) \qquad \iota'' \notin dom(\sigma_\beta) \qquad f_i^{\alpha\to\beta} \text{ new future} \qquad \iota_f \notin dom(\sigma_\alpha) \\ \sigma'_\beta = Copy\&Merge(\sigma_\alpha,\iota' \ ; \ \sigma_\beta,\iota'') \qquad \sigma'_\alpha = \{\iota_f \mapsto fut(f_i^{\alpha\to\beta})\} :: \sigma_\alpha\end{array}}{\begin{array}{c}\alpha[\mathcal{R}[\iota.m_j(\iota')];\sigma_\alpha;\iota_\alpha;F_\alpha;R_\alpha;f_\alpha] \parallel \beta[a_\beta;\sigma_\beta;\iota_\beta;F_\beta;R_\beta;f_\beta] \parallel P \longrightarrow \\ \alpha[\mathcal{R}[\iota_f];\sigma'_\alpha;\iota_\alpha;F_\alpha;R_\alpha;f_\alpha] \parallel \beta[a_\beta;\sigma'_\beta;\iota_\beta;F_\beta;R_\beta :: [m_j;\iota'';f_i^{\alpha\to\beta}];f_\beta] \parallel P\end{array}} \ (\text{REQUEST})$$

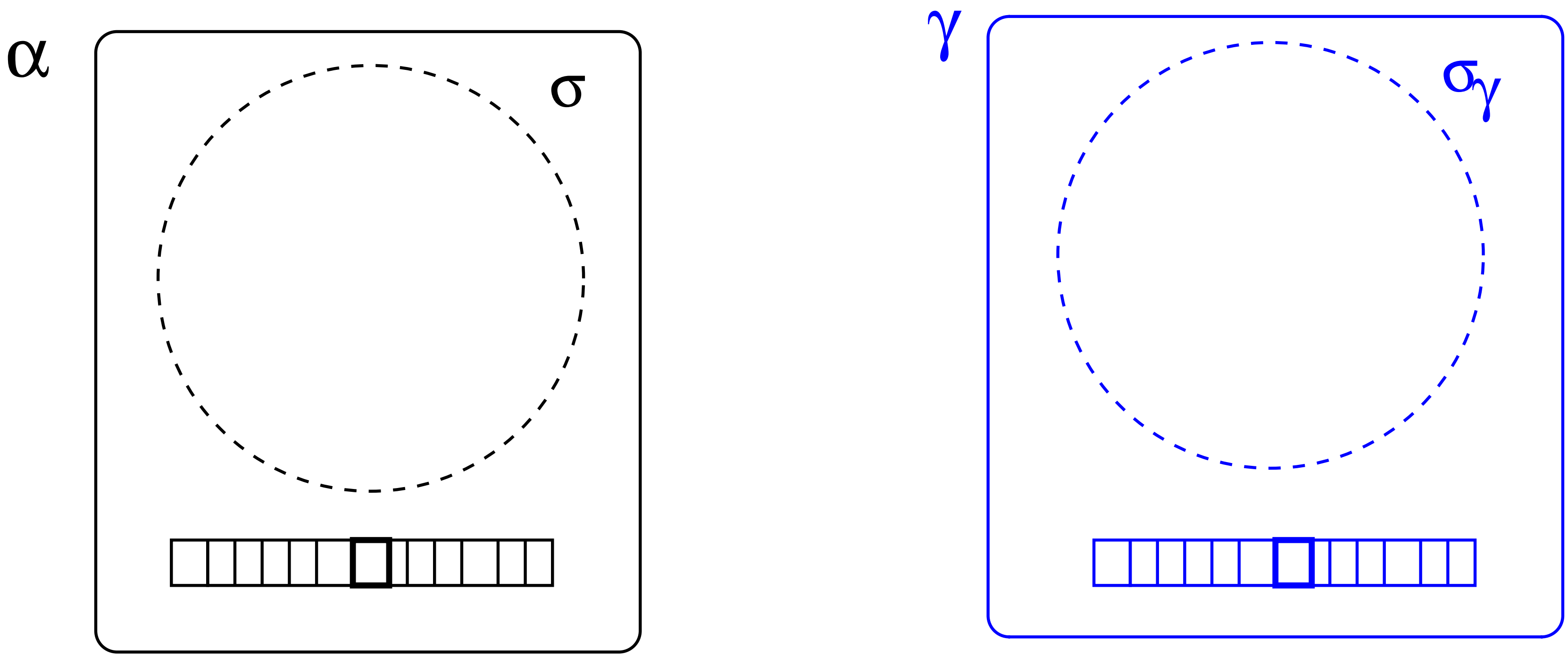
$$\frac{R = R' :: [m_j;\iota_r;f'] :: R'' \qquad m_j \in M \qquad \forall m \in M,\ m \notin R'}{\alpha[\mathcal{R}[Serve(M)];\sigma;\iota;F;R;f] \parallel P \longrightarrow \alpha[\iota.m_j(\iota_r) \Uparrow f, \mathcal{R}[[]];\sigma;\iota;F;R' :: R'';f'] \parallel P} \ (\text{SERVE})$$

$$\frac{\iota' \notin dom(\sigma) \qquad F' = F :: \{f \mapsto \iota'\} \qquad \sigma' = Copy\&Merge(\sigma,\iota \ ; \ \sigma,\iota')}{\alpha[\iota \Uparrow f', a;\sigma;\iota;F;R;f] \parallel P \longrightarrow \alpha[a;\sigma';\iota;F';R;f'] \parallel P} \ (\text{ENDSERVICE})$$

$$\frac{\sigma_\alpha(\iota) = fut(f_i^{\gamma\to\beta}) \qquad F_\beta(f_i^{\gamma\to\beta}) = \iota_f \qquad \sigma'_\alpha = Copy\&Merge(\sigma_\beta,\iota_f \ ; \ \sigma_\alpha,\iota)}{\begin{array}{c}\alpha[a_\alpha;\sigma_\alpha;\iota_\alpha;F_\alpha;R_\alpha;f_\alpha] \parallel \beta[a_\beta;\sigma_\beta;\iota_\beta;F_\beta;R_\beta;f_\beta] \parallel P \longrightarrow \\ \alpha[a_\alpha;\sigma'_\alpha;\iota_\alpha;F_\alpha;R_\alpha;f_\alpha] \parallel \beta[a_\beta;\sigma_\beta;\iota_\beta;F_\beta;R_\beta;f_\beta] \parallel P\end{array}} \ (\text{REPLY})$$

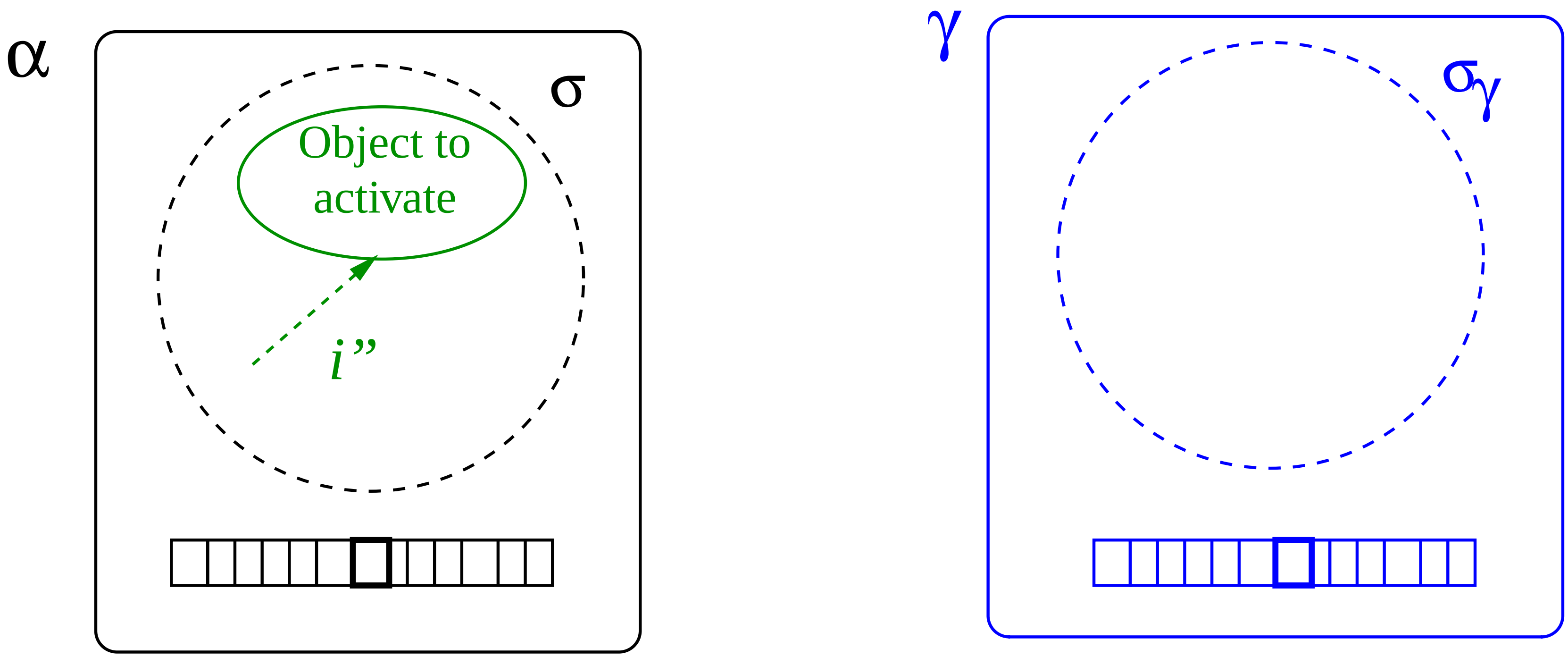
Reduction rules: 1) New Activity

γ fresh activity



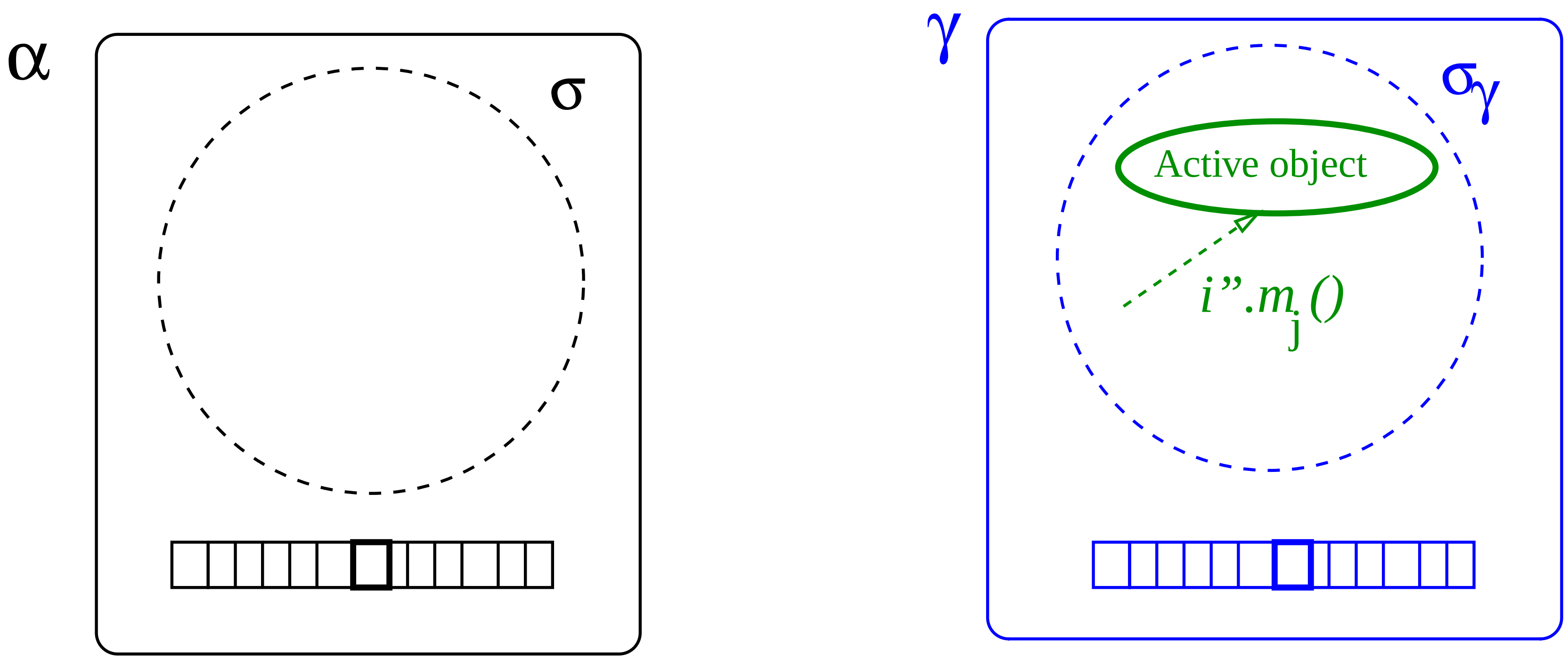
Reduction rules: 1) New Activity

γ fresh activity $copy(\iota'', \sigma)$



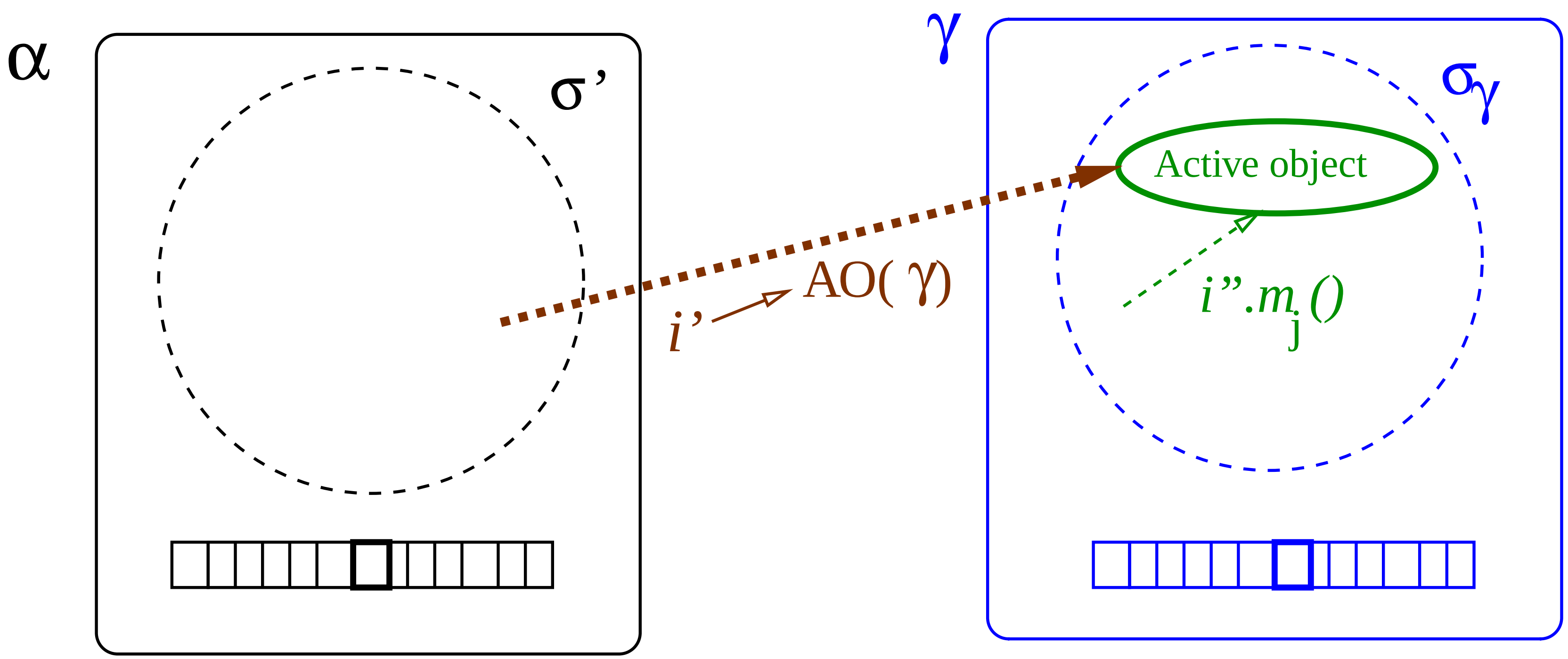
Reduction rules: 1) New Activity

γ fresh activity $\quad copy(\iota'', \sigma) = \sigma_\gamma$



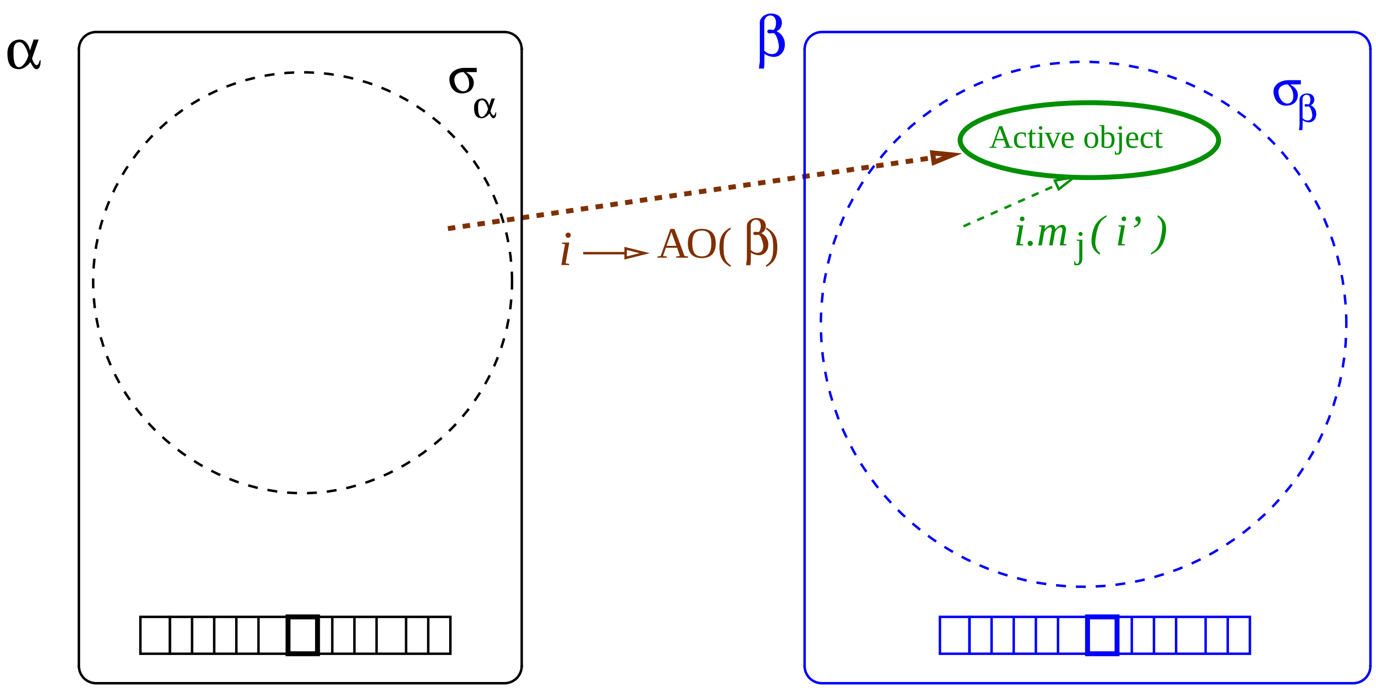
Reduction rules: 1) New Activity

$$\frac{\gamma \text{ fresh activity} \qquad copy(\iota'',\sigma) = \sigma_\gamma \qquad \iota' \notin dom(\sigma) \qquad \sigma' = \{\iota' \mapsto AO(\gamma)\} :: \sigma}{\alpha[\mathcal{R}[Active(\iota'', m_j)];\sigma;\iota;F;R;f] \parallel P \longrightarrow \alpha[\mathcal{R}[\iota'];\sigma';\iota;F;R;f] \parallel \gamma[\iota''.m_j();\sigma_\gamma;\iota'';\emptyset;\emptyset;\emptyset] \parallel P}$$



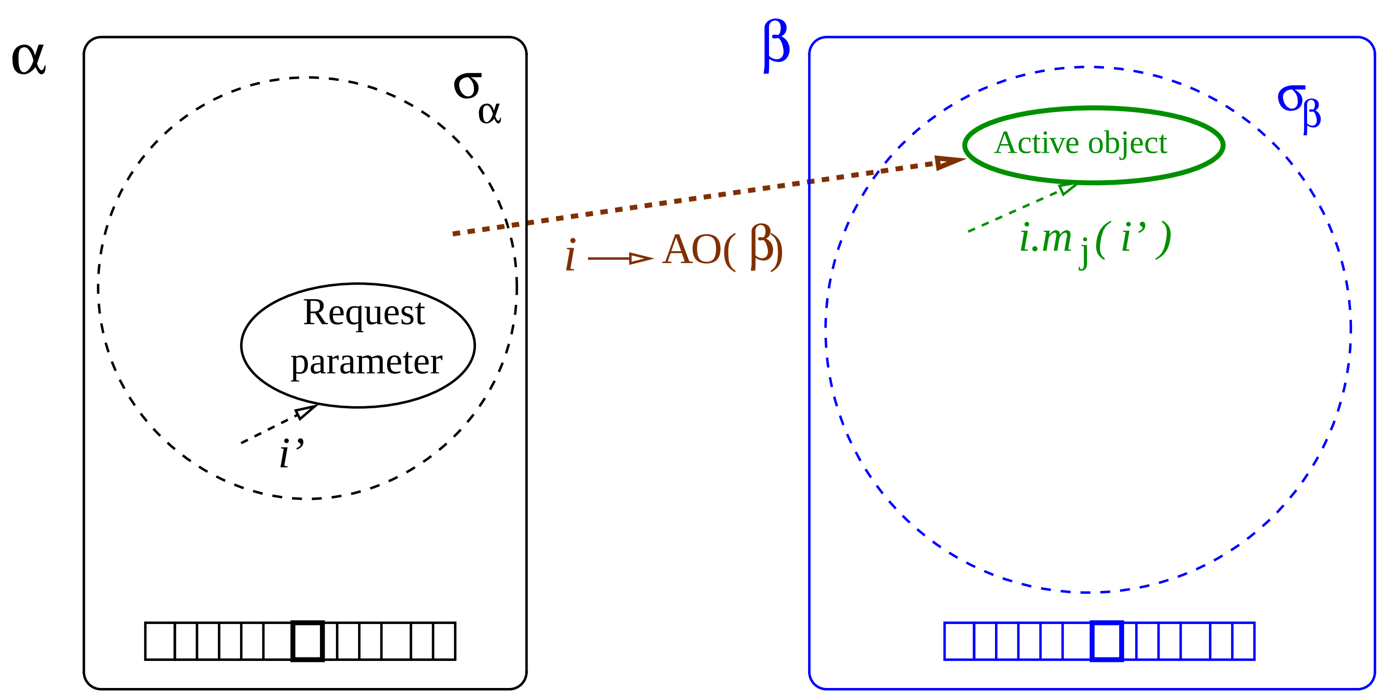
Reduction rules: 2) Request

$\sigma_\alpha(\iota) = AO(\beta)$



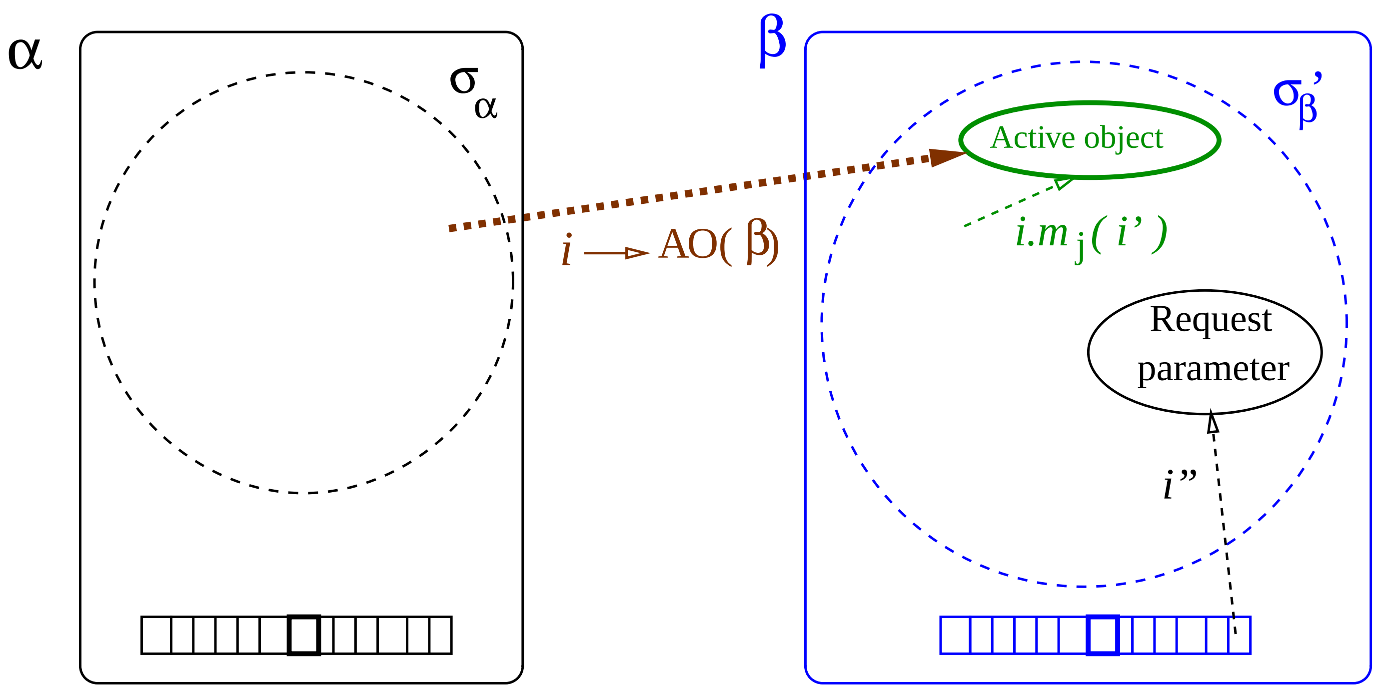
Reduction rules: 2) Request

$\sigma_\alpha(\iota) = AO(\beta) \qquad \iota'' \notin dom(\sigma_\beta) \qquad Copy\&Merge(\sigma_\alpha, \iota' \,;\, \sigma_\beta, \iota'')$



Reduction rules: 2) Request

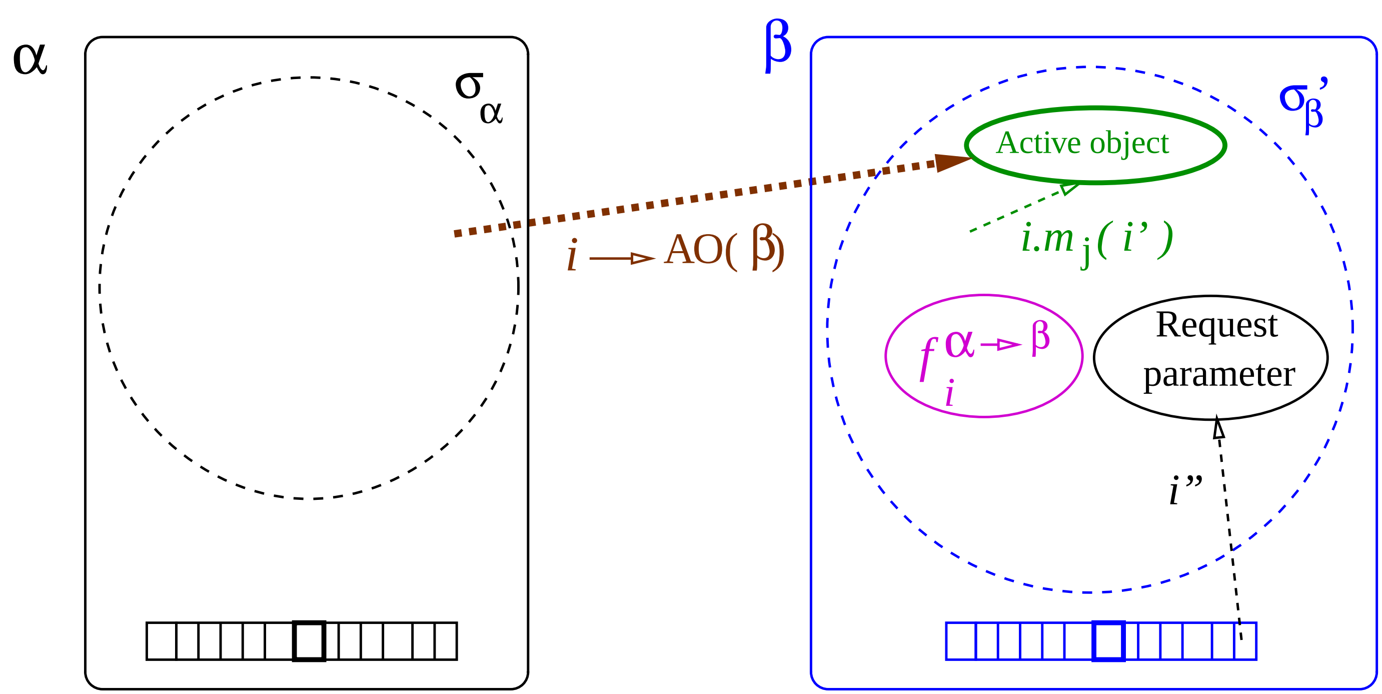
$\sigma_\alpha(\iota) = AO(\beta) \qquad \iota'' \notin dom(\sigma_\beta) \qquad Copy\&Merge(\sigma_\alpha, \iota' \,;\, \sigma_\beta, \iota'') = \sigma'_\beta$



Reduction rules: 2) Request

$\sigma_\alpha(\iota) = AO(\beta)$ $\quad \iota'' \notin dom(\sigma_\beta)$ $\quad Copy\&Merge(\sigma_\alpha, \iota' \,;\, \sigma_\beta, \iota'') = \sigma'_\beta$

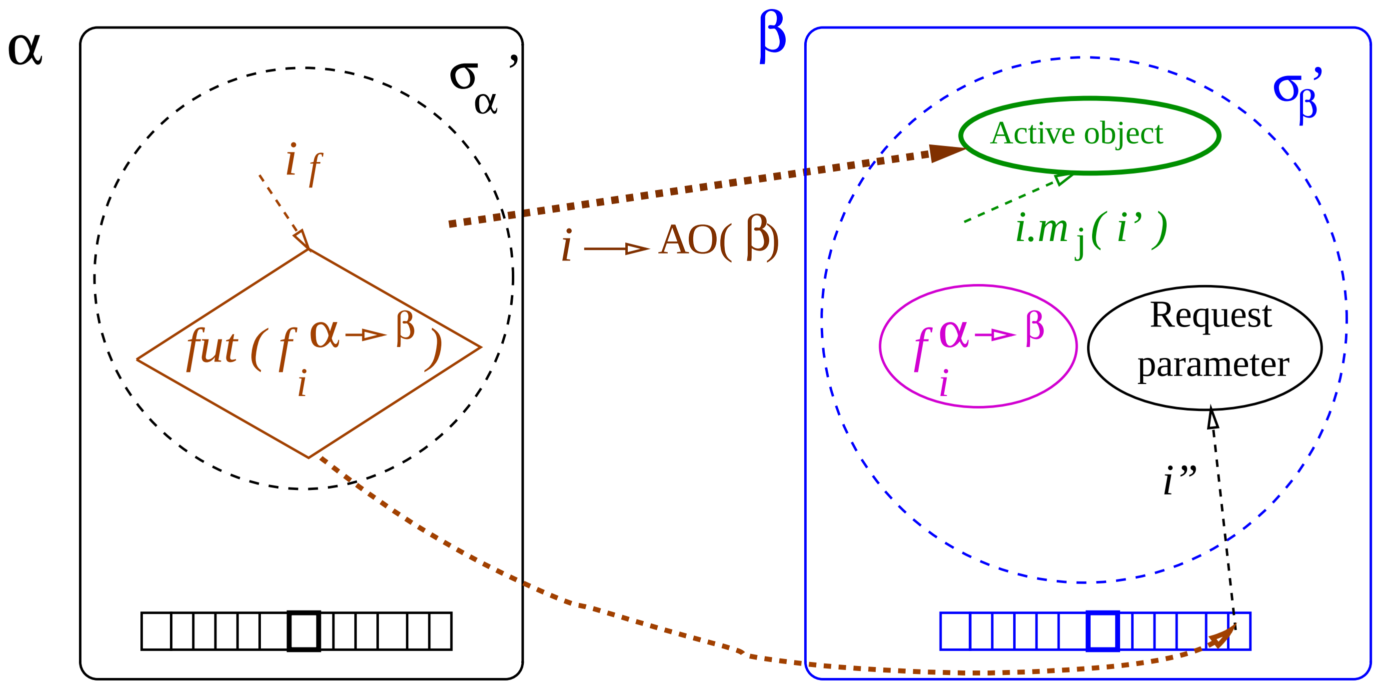
$f_i^{\alpha\to\beta}$ new future



Reduction rules: 2) Request

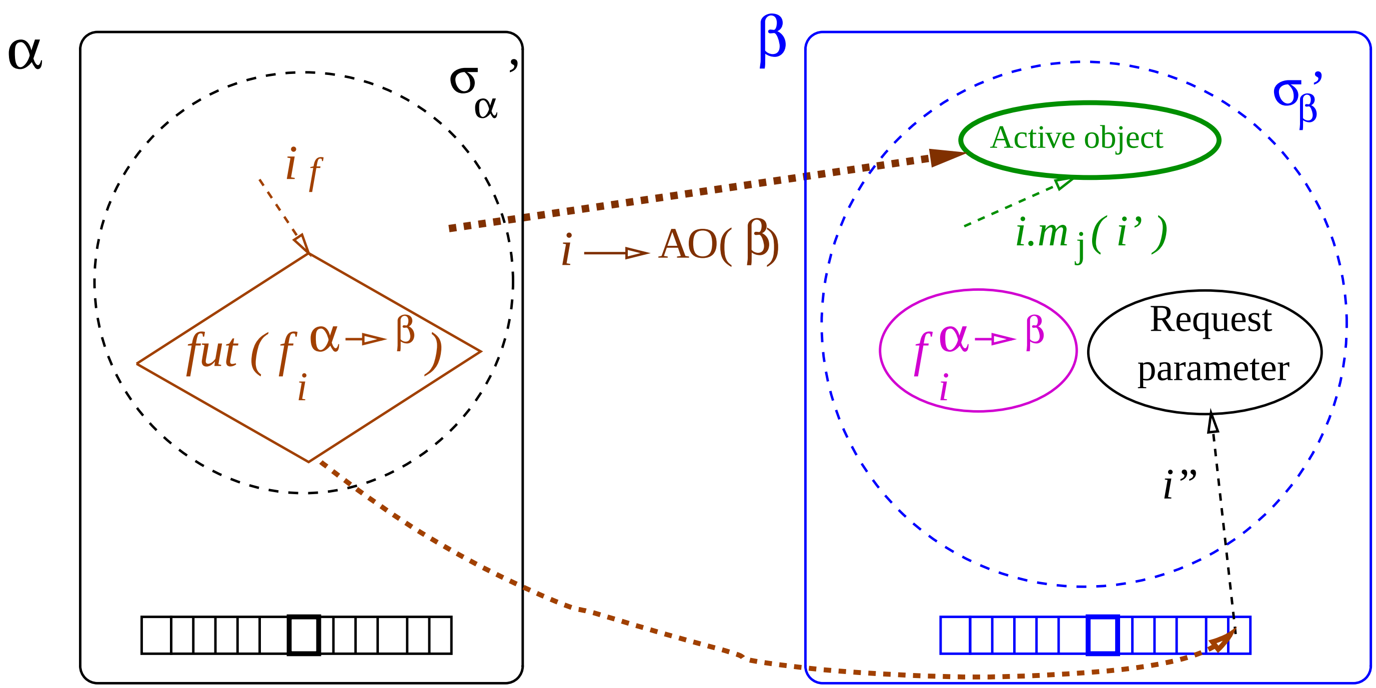
$\sigma_\alpha(\iota) = AO(\beta) \qquad \iota'' \notin dom(\sigma_\beta) \qquad Copy\&Merge(\sigma_\alpha, \iota' \,;\, \sigma_\beta, \iota'') = \sigma'_\beta$

$f_i^{\alpha \to \beta}$ new future $\qquad \iota_f \notin dom(\sigma_\alpha) \qquad \{\iota_f \mapsto fut(f_i^{\alpha\to\beta})\} :: \sigma_\alpha = \sigma'_\alpha$



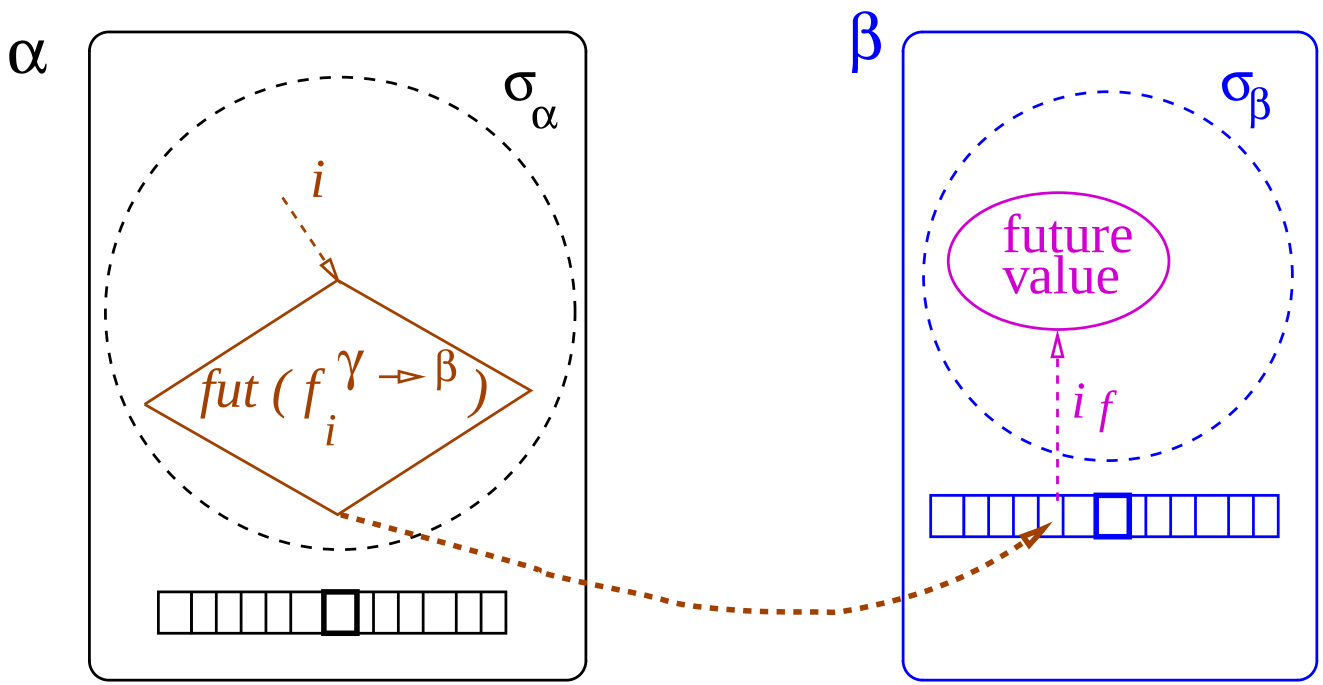
Reduction rules: 2) Request

$$\frac{\sigma_\alpha(\iota) = AO(\beta) \qquad \iota'' \not\in dom(\sigma_\beta) \qquad Copy\&Merge(\sigma_\alpha, \iota' \,;\, \sigma_\beta, \iota'') = \sigma'_\beta \\ f_i^{\alpha\rightarrow\beta} \text{ new future} \qquad \iota_f \not\in dom(\sigma_\alpha) \qquad \{\iota_f \mapsto fut(f_i^{\alpha\rightarrow\beta})\} :: \sigma_\alpha = \sigma'_\alpha}{\alpha[\mathcal{R}[\iota.m_j(\iota')]; \sigma_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P \longrightarrow \\ \alpha[\mathcal{R}[\iota_f]; \sigma'_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta[a_\beta; \sigma'_\beta; \iota_\beta; F_\beta; R_\beta :: [m_j; \iota''; f_i^{\alpha\rightarrow\beta}]; f_\beta] \parallel P}$$



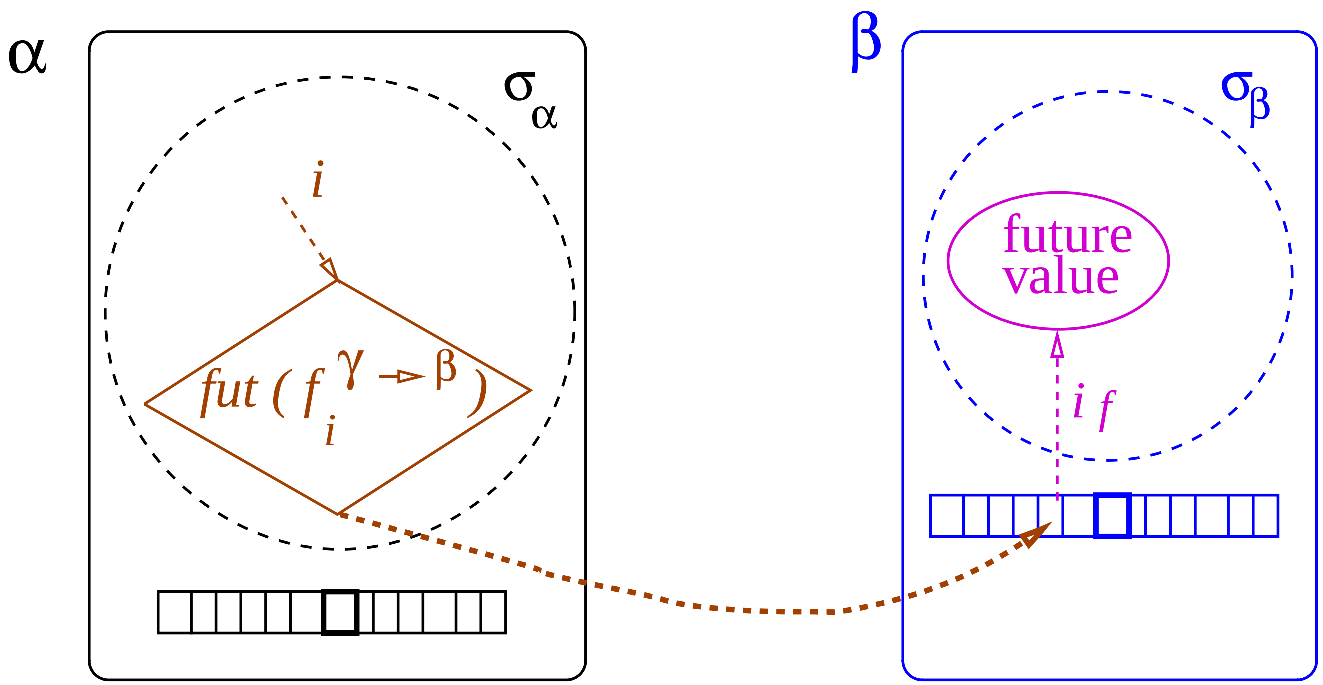
Reduction rules: 3) Reply

$\sigma_\alpha(\iota) = fut(f_i^{\gamma\rightarrow\beta}) \qquad F_\beta(f_i^{\gamma\rightarrow\beta}) = \iota_f$



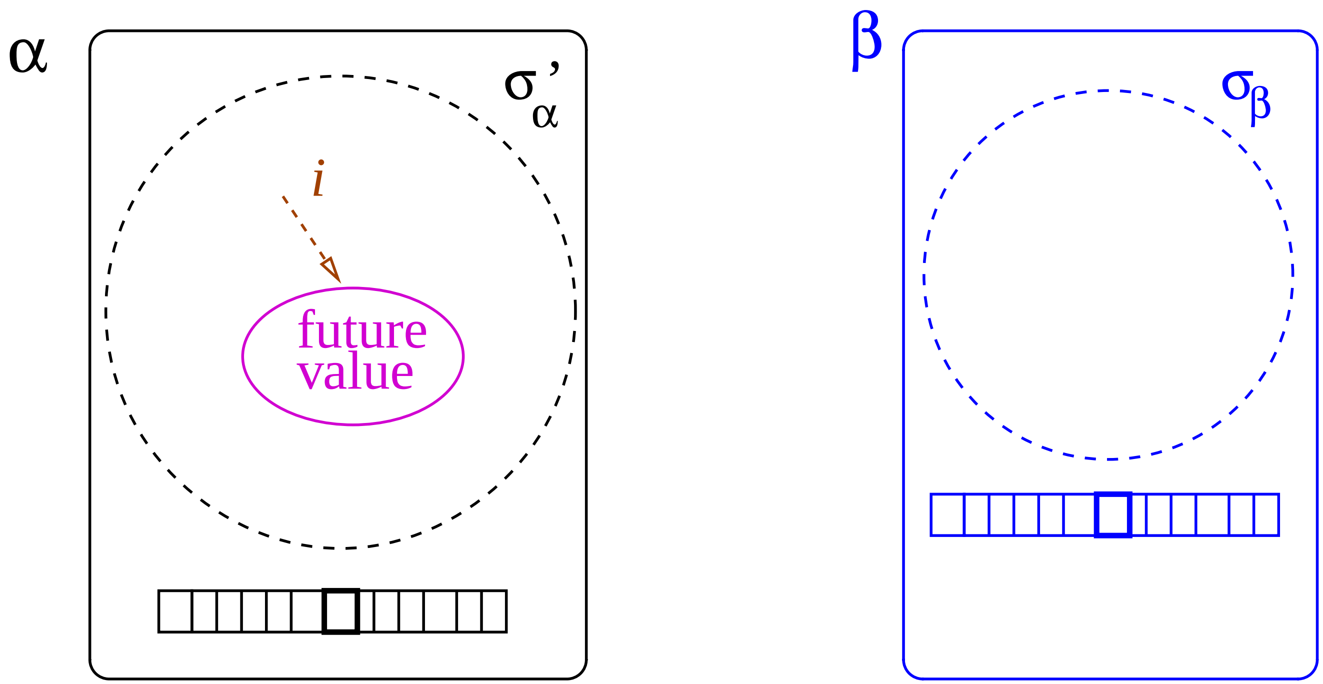
Reduction rules: 3) Reply

$\sigma_\alpha(\iota) = fut(f_i^{\gamma\to\beta})$ $\quad F_\beta(f_i^{\gamma\to\beta}) = \iota_f$ $\quad Copy\&Merge(\sigma_\beta, \iota_f \;;\; \sigma_\alpha, \iota)$



Reduction rules: 3) Reply

$$\frac{\sigma_\alpha(\iota) = fut(f_i^{\gamma\to\beta}) \qquad F_\beta(f_i^{\gamma\to\beta}) = \iota_f \qquad Copy\&Merge(\sigma_\beta, \iota_f \,;\, \sigma_\alpha, \iota) = \sigma'_\alpha}{\begin{array}{c}\alpha[a_\alpha;\sigma_\alpha;\iota_\alpha;F_\alpha;R_\alpha;f_\alpha] \parallel \beta[a_\beta;\sigma_\beta;\iota_\beta;F_\beta;R_\beta;f_\beta] \parallel P \longrightarrow \\ \alpha[a_\alpha;\sigma'_\alpha;\iota_\alpha;F_\alpha;R_\alpha;f_\alpha] \parallel \beta[a_\beta;\sigma_\beta;\iota_\beta;F_\beta;R_\beta;f_\beta] \parallel P\end{array}}$$



Objectives

Agenda

Main objective: Information Flow Control

Objectives

Main objective: Information Flow Control

1. To guarantee *data confidentiality*

   Confidentiality in MLS: follows the basic principle of *no write down, no read up*

Objectives

Main objective: Information Flow Control

1. To guarantee *data confidentiality*

   Confidentiality in MLS: follows the basic principle of *no write down, no read up*

2. Define a security policy to apply to asynchronous, distributed, and mobile applications

Objectives

Main objective: Information Flow Control

1. To guarantee *data confidentiality*

   Confidentiality in MLS: follows the basic principle of *no write down, no read up*

2. Define a security policy to apply to asynchronous, distributed, and mobile applications

   Main issue: Presence of asymmetric patterns of communications (result of the *future* and *wait-by necessity* concepts)

Objectives

Main objective: Information Flow Control

1. To guarantee *data confidentiality*

   Confidentiality in MLS: follows the basic principle of *no write down, no read up*

2. Define a security policy to apply to asynchronous, distributed, and mobile applications

   Main issue: Presence of asymmetric patterns of communications (result of the *future* and *wait-by necessity* concepts)

3. Provide a formal security model

Objectives

Main objective: Information Flow Control

1. To guarantee *data confidentiality*

   Confidentiality in MLS: follows the basic principle of *no write down, no read up*

2. Define a security policy to apply to asynchronous, distributed, and mobile applications

   Main issue: Presence of asymmetric patterns of communications (result of the *future* and *wait-by necessity* concepts)

3. Provide a formal security model which is verifiable mathematically

Objectives

Main objective: Information Flow Control

1. To guarantee *data confidentiality*

   Confidentiality in MLS: follows the basic principle of *no write down, no read up*

2. Define a security policy to apply to asynchronous, distributed, and mobile applications

   Main issue: Presence of asymmetric patterns of communications (result of the *future* and *wait-by necessity* concepts)

3. Provide a formal security model which is verifiable mathematically

4. Propose an architecture for the implementation of the security model

Related Security Mechanisms

Agenda

- Models

Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)

Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)

Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)
- Formal methods

Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)
- Formal methods
  - Non-interference

Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)
- Formal methods
  - Non-interference
  - π-calculus, Asynchronous-π, and Spi calculus

Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)
- Formal methods
  - Non-interference
  - π-calculus, Asynchronous-π, and Spi calculus
  - Ambients, and Mobile Ambients

Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)
- Formal methods
  - Non-interference
  - π-calculus, Asynchronous-π, and Spi calculus
  - Ambients, and Mobile Ambients
  - Seals, Boxed Ambients, and Secure Safe Ambients

Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)
- Formal methods
  - Non-interference
  - π-calculus, Asynchronous-π, and Spi calculus
  - Ambients, and Mobile Ambients
  - Seals, Boxed Ambients, and Secure Safe Ambients
- Informal approaches

Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)
- Formal methods
  - Non-interference
  - π-calculus, Asynchronous-π, and Spi calculus
  - Ambients, and Mobile Ambients
  - Seals, Boxed Ambients, and Secure Safe Ambients
- Informal approaches
  - Exceptions (in the security policy)

Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)
- Formal methods
  - Non-interference
  - π-calculus, Asynchronous-π, and Spi calculus
  - Ambients, and Mobile Ambients
  - Seals, Boxed Ambients, and Secure Safe Ambients
- Informal approaches
  - Exceptions (in the security policy), labels

Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)
- Formal methods
  - Non-interference
  - π-calculus, Asynchronous-π, and Spi calculus
  - Ambients, and Mobile Ambients
  - Seals, Boxed Ambients, and Secure Safe Ambients
- Informal approaches
  - Exceptions (in the security policy), labels, tickets

Related Security Mechanisms

- Models
  - Mandatory Access Controls (MAC)
  - Discretionary Access Controls (DAC)
- Formal methods
  - Non-interference
  - π-calculus, Asynchronous-π, and Spi calculus
  - Ambients, and Mobile Ambients
  - Seals, Boxed Ambients, and Secure Safe Ambients
- Informal approaches
  - Exceptions (in the security policy), labels, tickets, certificates . . .

The ASP Security Model

Syntax and semantics of the security framework

The ASP Security Model

Syntax and semantics of the security framework

- $\mathcal{S}$ set of activities acting as subjects, where $\alpha, \beta, \gamma, \ldots \in \mathcal{S}$

The ASP Security Model

Syntax and semantics of the security framework

- $\mathcal{S}$ set of activities acting as subjects, where $\alpha, \beta, \gamma, ... \in \mathcal{S}$
- $\mathcal{D}$ set of data objects sent as arguments in REQUEST actions: $Rq_{\alpha \rightarrow \beta}(d)$

The ASP Security Model

Syntax and semantics of the security framework

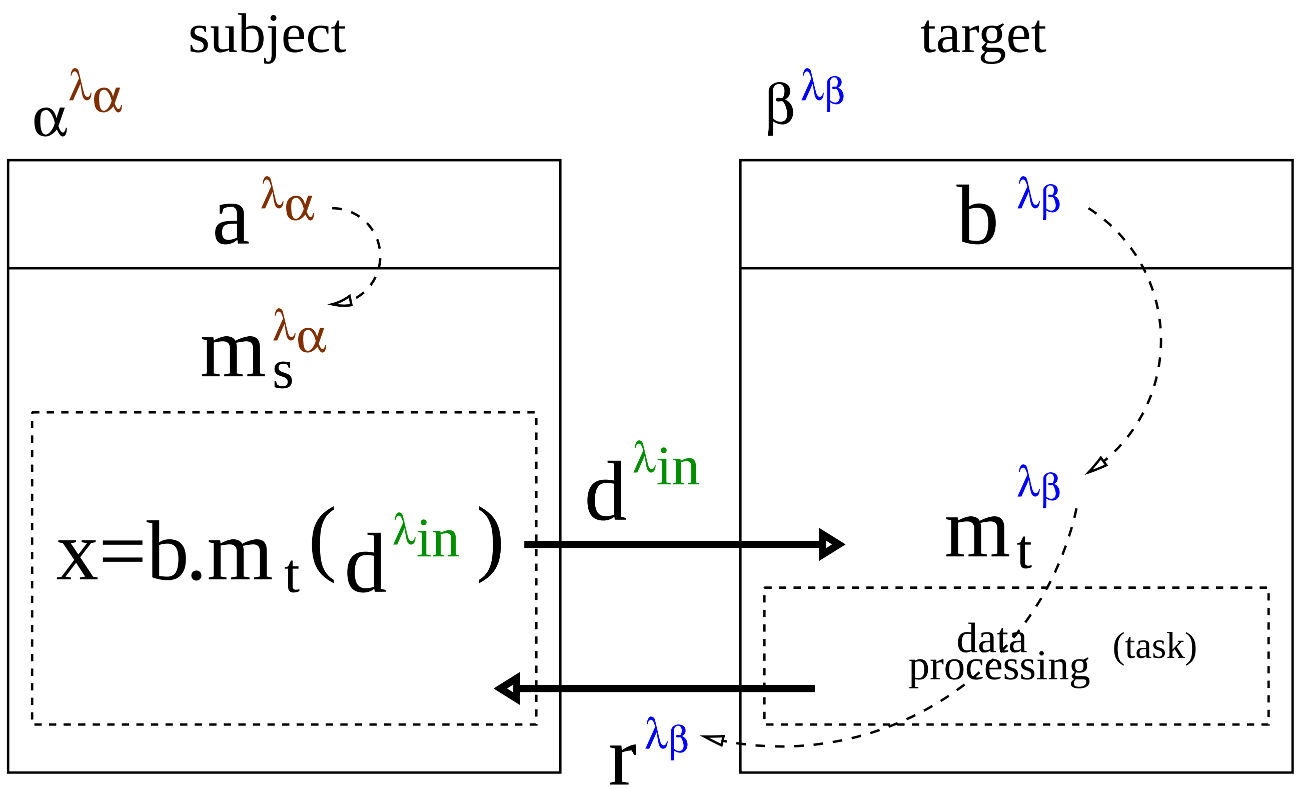
- $\mathcal{S}$ set of activities acting as subjects, where $\alpha, \beta, \gamma, \ldots \in \mathcal{S}$
- $\mathcal{D}$ set of data objects sent as arguments in REQUEST actions: $Rq_{\alpha \rightarrow \beta}(d)$
- $\mathcal{R}$ is the set of objects associated to *futures*, and returned in REPLY actions: $Rp_{\beta \rightarrow \alpha}(r)$

The ASP Security Model *(cntd.)*

- $\mathcal{L}$ finite set of security levels λ, partially ordered by the relation $\leq$, where $\forall i \in \mathcal{S} \cup \mathcal{D}, \lambda_i \in \mathcal{L}$

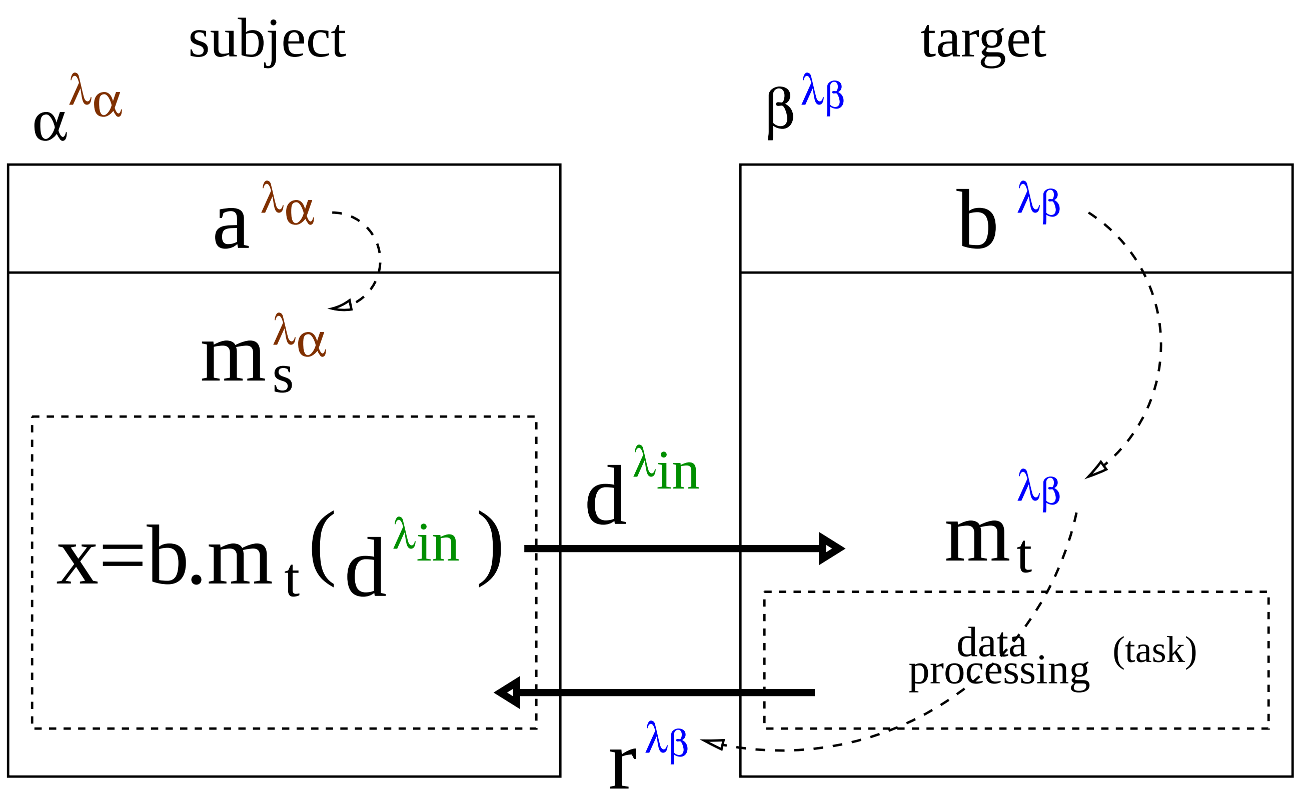
The ASP Security Model *(cntd.)*

- $\mathcal{L}$ finite set of security levels λ, partially ordered by the relation $\leq$, where $\forall i \in \mathcal{S} \cup \mathcal{D}, \lambda_i \in \mathcal{L}$



The ASP Security Model *(cntd.)*

- $\mathcal{L}$ finite set of security levels λ, partially ordered by the relation $\leq$, where $\forall i \in \mathcal{S} \cup \mathcal{D}, \lambda_i \in \mathcal{L}$



- Transmissions of d and r are restricted by the security rules

The ASP Security Model *(cntd.)*

Agenda

Actions

- $\mathcal{A}$ set of actions, where $a \in \mathcal{A}$

The ASP Security Model *(cntd.)*

Agenda

Actions

- $\mathcal{A}$ set of actions, where $a \in \mathcal{A}$
- ASP actions are now rewritten to include security properties:

The ASP Security Model *(cntd.)*

Actions

- $\mathcal{A}$ set of actions, where $a \in \mathcal{A}$
- ASP actions are now rewritten to include security properties:

  $Nw(\gamma, \lambda_\gamma)$ is a modified NEWACT

The ASP Security Model *(cntd.)*

Actions

- $\mathcal{A}$ set of actions, where $a \in \mathcal{A}$
- ASP actions are now rewritten to include security properties:

  $Nw(\gamma, \lambda_\gamma)$ is a modified NEWACT
  $Rq_{\alpha \rightarrow \beta}(d, \lambda_{in})$ is a modified REQUEST

The ASP Security Model *(cntd.)*

Actions

- $\mathcal{A}$ set of actions, where $a \in \mathcal{A}$
- ASP actions are now rewritten to include security properties:

  $Nw(\gamma, \lambda_\gamma)$ is a modified NEWACT
  $Rq_{\alpha \to \beta}(d, \lambda_{in})$ is a modified REQUEST
  $Rp_{\beta \to \alpha}(r)$ is an unchanged REPLY

The ASP Security Model *(cntd.)*

Actions

- $\mathcal{A}$ set of actions, where $a \in \mathcal{A}$
- ASP actions are now rewritten to include security properties:

  $Nw(\gamma, \lambda_\gamma)$ is a modified NEWACT
  $Rq_{\alpha \to \beta}(d, \lambda_{in})$ is a modified REQUEST
  $Rp_{\beta \to \alpha}(r)$ is an unchanged REPLY

- In general,
  $a = \{Nw(\gamma, \lambda_\gamma), Rq_{\alpha \to \beta}(d, \lambda_{in}), Rp_{\beta \to \alpha}(r)\}$

The ASP Security Model *(cntd.)*

Additional entities

- $\mathcal{M}$ matrix of explicit (discretionary) rights

The ASP Security Model *(cntd.)*

Additional entities

- $\mathcal{M}$ matrix of explicit (discretionary) rights

$$\mathcal{M} = \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$$

The ASP Security Model *(cntd.)*

Additional entities

- $\mathcal{M}$ matrix of explicit (discretionary) rights

$$\mathcal{M} = \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$$

$\mathcal{P}(\mathcal{A})$ set of actions whose assignation of a security level is explicitly allowed

The ASP Security Model *(cntd.)*

Additional entities

- $\mathcal{M}$ matrix of explicit (discretionary) rights

  $\mathcal{M} = \mathcal{S} \times \mathcal{S} \to \mathcal{P}(\mathcal{A})$

  $\mathcal{P}(\mathcal{A})$ set of actions whose assignation of a security level is explicitly allowed

  In general, $p \in \mathcal{P}(\mathcal{A})$ if and only if $p = \{Nw(\gamma, \lambda_\gamma), Rq_{\alpha\to\beta}(d, \lambda_{in})\}$

The ASP Security Model *(cntd.)*

Additional entities

- $\mathcal{M}$ matrix of explicit (discretionary) rights

  $\mathcal{M} = \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$

  $\mathcal{P}(\mathcal{A})$ set of actions whose assignation of a security level is explicitly allowed

  In general, $p \in \mathcal{P}(\mathcal{A})$ if and only if $p = \{Nw(\gamma, \lambda_\gamma), Rq_{\alpha \rightarrow \beta}(d, \lambda_{in})\}$

- $\mathcal{T}$ set of authorized (access) transmissions

The ASP Security Model *(cntd.)*

Additional entities

- $\mathcal{M}$ matrix of explicit (discretionary) rights

  $\mathcal{M} = \mathcal{S} \times \mathcal{S} \rightarrow \mathcal{P}(\mathcal{A})$

  $\mathcal{P}(\mathcal{A})$ set of actions whose assignation of a security level is explicitly allowed

  In general, $p \in \mathcal{P}(\mathcal{A})$ if and only if $p = \{Nw(\gamma, \lambda_\gamma), Rq_{\alpha \rightarrow \beta}(d, \lambda_{in})\}$

- $\mathcal{T}$ set of authorized (access) transmissions

  $\mathcal{T} = \mathcal{S} \times \mathcal{S} \times \mathcal{A}$

Application of the security framework

Agenda

Application of the security framework

1.- Base statements:

- Creation (and migration) of new activities are secure
- Emission of requests, with modifiable security levels in the data sent, are secure
- Emission of replies are secure

Application of the security framework

1.- Base statements:

- Creation (and migration) of new activities are secure
- Emission of requests, with modifiable security levels in the data sent, are secure
- Emission of replies are secure

2.- Support concepts:

- Elementary flows of information
- *Flow-paths*

Application of the security framework

1.- Base statements:

- Creation (and migration) of new activities are secure
- Emission of requests, with modifiable security levels in the data sent, are secure
- Emission of replies are secure

2.- Support concepts:

- Elementary flows of information
- *Flow-paths*

3.- Results:

Confidentiality, from end-to-end in a flow-path, is guaranteed

Secure Activity Creation

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

Secure Activity Creation

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

$$\forall \alpha, \gamma \in \mathcal{S}$$

Secure Activity Creation

Agenda

$\forall \alpha, \gamma \in \mathcal{S}$: $(\alpha, \gamma, Nw(\gamma, \lambda_\gamma)) \in \mathcal{T} \Longleftrightarrow$

A new activity action is considered secure iff:

Secure Activity Creation

Agenda

$$\forall \alpha, \gamma \in \mathcal{S}\colon (\alpha, \gamma, Nw(\gamma, \lambda_\gamma)) \in \mathcal{T} \iff (\lambda_\alpha \leq \lambda_\gamma)$$

A new activity action is considered secure iff:

1. The new activity has a higher security level compared to its creator

Secure Activity Creation

$$\forall \alpha, \gamma \in \mathcal{S}\colon (\alpha, \gamma, Nw(\gamma, \lambda_\gamma)) \in \mathcal{T} \iff (\lambda_\alpha \leq \lambda_\gamma) \vee Nw(\gamma, \lambda_\gamma) \in \mathcal{M}(\alpha, \gamma)$$

A new activity action is considered secure iff:

1. The new activity has a higher security level compared to its creator
2. or, in case the new activity has a lower security (i.e. a downgrade), the creation action must be explicitly allowed

Secure Activity Creation

$$\forall \alpha, \gamma \in \mathcal{S}\colon (\alpha, \gamma, Nw(\gamma, \lambda_\gamma)) \in \mathcal{T} \iff (\lambda_\alpha \leq \lambda_\gamma) \vee Nw(\gamma, \lambda_\gamma) \in \mathcal{M}(\alpha, \gamma)$$

A new activity action is considered secure iff:

1. The new activity has a higher security level compared to its creator

2. or, in case the new activity has a lower security (i.e. a downgrade), the creation action must be explicitly allowed

Special case: Migration of an existing activity

Secure Request Transmission

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

Secure Request Transmission

$$\forall \alpha, \beta \in \mathcal{S}$$

Agenda

Secure Request Transmission

$$\forall \alpha, \beta \in \mathcal{S}\colon (\alpha, \beta, Rq_{\alpha \to \beta}(d, \lambda_{in})) \in \mathcal{T} \iff$$

A request transmission action is considered secure iff:

Secure Request Transmission

$$\forall \alpha, \beta \in \mathcal{S}\colon (\alpha, \beta, Rq_{\alpha\rightarrow\beta}(d, \lambda_{in})) \in \mathcal{T} \iff$$
$$(\lambda_{in} \leq \lambda_{\beta})\wedge$$

A request transmission action is considered secure iff:

1. Data is "released" to an authorized target, *AND*

Secure Request Transmission

$$\forall \alpha, \beta \in \mathcal{S}\colon (\alpha, \beta, Rq_{\alpha \to \beta}(d, \lambda_{in})) \in \mathcal{T} \iff$$
$$(\lambda_{in} \leq \lambda_\beta) \wedge$$
$$\left((\lambda_\alpha \leq \lambda_{in}) \right.$$

A request transmission action is considered secure iff:

1. Data is "released" to an authorized target, *AND*
2. Either:
   - The data has a higher level than the sender

Secure Request Transmission

$$\forall \alpha, \beta \in \mathcal{S}\colon (\alpha, \beta, Rq_{\alpha\to\beta}(d, \lambda_{in})) \in \mathcal{T} \iff (\lambda_{in} \leq \lambda_\beta) \wedge \left(\begin{array}{l} (\lambda_\alpha \leq \lambda_{in}) \\ \vee\ ((\lambda_\alpha > \lambda_{in}) \wedge Rq_{\alpha\to\beta}(d, \lambda_{in}) \in \mathcal{M}(\alpha, \beta)) \end{array} \right.$$

A request transmission action is considered secure iff:

1. Data is "released" to an authorized target, *AND*
2. Either:
   - The data has a higher level than the sender
   - If data has a lower level than the sender (i.e. a downgrade), the action must be explicitly allowed

Secure Request Transmission

$$\forall \alpha, \beta \in \mathcal{S}\colon (\alpha, \beta, Rq_{\alpha \to \beta}(d, \lambda_{in})) \in \mathcal{T} \iff (\lambda_{in} \le \lambda_\beta) \wedge \left(\begin{array}{l} (\lambda_\alpha \le \lambda_{in}) \\ \vee \; ((\lambda_\alpha > \lambda_{in}) \wedge Rq_{\alpha \to \beta}(d, \lambda_{in}) \in \mathcal{M}(\alpha, \beta)) \\ \vee \; \exists \gamma, \delta, f_i, \; d = fut(f_i^{\gamma \to \delta}) \end{array} \right)$$

A request transmission action is considered secure iff:

1. Data is "released" to an authorized target, AND
2. Either:
   - The data has a higher level than the sender
   - If data has a lower level than the sender (i.e. a downgrade), the action must be explicitly allowed
   - The data is a *future reference*

Secure Reply Transmission

Agenda

Secure Reply Transmission

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

$$\forall \alpha, \beta \in \mathcal{S}$$

Secure Reply Transmission

$$\forall \alpha, \beta \in \mathcal{S}\text{: } (\alpha, \beta, Rp_{\beta\rightarrow\alpha}(r)) \in \mathcal{T} \iff$$

A reply transmission action is considered secure iff:

Secure Reply Transmission

$$\forall \alpha, \beta \in \mathcal{S}\colon (\alpha, \beta, Rp_{\beta \to \alpha}(r)) \in \mathcal{T} \iff (\lambda_\beta \leq \lambda_\alpha)$$

A reply transmission action is considered secure iff:

1. The data contained in the reply r (hence of level λ_β) can be released to the corresponding receiving subject (with λ_α)

Secure Reply Transmission

$$\forall \alpha, \beta \in \mathcal{S}\colon (\alpha, \beta, Rp_{\beta \to \alpha}(r)) \in \mathcal{T} \iff$$
$$(\lambda_\beta \leq \lambda_\alpha) \vee (\exists \gamma, \delta, f_i,\ r = fut(f_i^{\gamma \to \delta}))$$

A reply transmission action is considered secure iff:

1. The data contained in the reply r (hence of level λ_β) can be released to the corresponding receiving subject (with λ_α)

2. or, if the data in the reply is only a reference to a future

Secure ASP reduction rules

Secure ASP reduction rules

$$\frac{\gamma \text{ fresh activity} \qquad \iota' \notin dom(\sigma) \qquad \sigma' = \{\iota' \mapsto AO(\gamma)\} :: \sigma \qquad \sigma_\gamma = copy(\iota'', \sigma) \qquad (\alpha, \gamma, Nw(\gamma, \lambda_\gamma)) \in \mathcal{T}}{\alpha^\lambda[\mathcal{R}[Active^{\lambda a}(\iota'', m_j)]; \sigma; \iota; F; R; f] \parallel P \longrightarrow \alpha^\lambda[\mathcal{R}[\iota']; \sigma'; \iota; F; R; f] \parallel \gamma^{\lambda a}[\iota''.m_j(); \sigma_\gamma; \iota''; \emptyset; \emptyset; \emptyset] \parallel P} \text{ (SecNEWACT)}$$

$$\frac{\sigma_\alpha(\iota) = AO(\beta) \qquad \iota'' \notin dom(\sigma_\beta) \qquad f_i^{\alpha\to\beta} \text{ new future} \qquad \iota_f \notin dom(\sigma_\alpha) \qquad \sigma'_\beta = Copy\&Merge(\sigma_\alpha, \iota' \; ; \; \sigma_\beta, \iota'') \qquad \sigma'_\alpha = \{\iota_f \mapsto fut(f_i^{\alpha\to\beta})\} :: \sigma_\alpha \qquad (\alpha, \beta, Rq_{\alpha\to\beta}(\sigma_\alpha(\iota'), \lambda_{in})) \in \mathcal{T}}{\alpha^{\lambda\alpha}[\mathcal{R}[\iota.m_j(\iota'^{\lambda in})]; \sigma_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta^{\lambda\beta}[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P \longrightarrow \alpha^{\lambda\alpha}[\mathcal{R}[\iota_f]; \sigma'_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta^{\lambda\beta}[a_\beta; \sigma'_\beta; \iota_\beta; F_\beta; R_\beta :: [m_j; \iota''; f_i^{\alpha\to\beta}]; f_\beta] \parallel P} \text{ (SecREQUEST)}$$

$$\frac{\sigma_\alpha(\iota) = fut(f_i^{\gamma\to\beta}) \qquad F_\beta(f_i^{\gamma\to\beta}) = \iota_f \qquad \sigma'_\alpha = Copy\&Merge(\sigma_\beta, \iota_f \; ; \; \sigma_\alpha, \iota) \qquad (\beta, \alpha, Rp_{\beta\to\alpha}(\sigma_\beta(\iota_f))) \in \mathcal{T}}{\alpha^{\lambda\alpha}[a_\alpha; \sigma_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta^{\lambda\beta}[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P \longrightarrow \alpha^{\lambda\alpha}[a_\alpha; \sigma'_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta^{\lambda\beta}[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P} \text{ (SecREPLY)}$$

Secure ASP reduction rules

$$\frac{\begin{array}{c}\gamma \text{ fresh activity} \qquad \iota' \notin dom(\sigma) \qquad \sigma' = \{\iota' \mapsto AO(\gamma)\} :: \sigma \\ \sigma_\gamma = copy(\iota'', \sigma) \qquad (\alpha, \gamma, Nw(\gamma, \lambda_\gamma)) \in \mathcal{T}\end{array}}{\begin{array}{c}\alpha^{\lambda}[\mathcal{R}[Active^{\lambda_a}(\iota'', m_j)]; \sigma; \iota; F; R; f] \parallel P \longrightarrow \\ \alpha^{\lambda}[\mathcal{R}[\iota']; \sigma'; \iota; F; R; f] \parallel \gamma^{\lambda_a}[\iota''.m_j(); \sigma_\gamma; \iota''; \emptyset; \emptyset; \emptyset] \parallel P\end{array}} \text{(SecNEWACT)}$$

$$\frac{\begin{array}{c}\sigma_\alpha(\iota) = AO(\beta) \qquad \iota'' \notin dom(\sigma_\beta) \qquad f_i^{\alpha\to\beta} \text{ new future} \\ \iota_f \notin dom(\sigma_\alpha) \qquad \sigma'_\beta = Copy\&Merge(\sigma_\alpha, \iota' \ ; \ \sigma_\beta, \iota'') \\ \sigma'_\alpha = \{\iota_f \mapsto fut(f_i^{\alpha\to\beta})\} :: \sigma_\alpha \qquad (\alpha, \beta, Rq_{\alpha\to\beta}(\sigma_\alpha(\iota'), \lambda_{in})) \in \mathcal{T}\end{array}}{\begin{array}{c}\alpha^{\lambda_\alpha}[\mathcal{R}[\iota.m_j(\iota'^{\lambda_{in}})]; \sigma_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta^{\lambda_\beta}[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P \longrightarrow \\ \alpha^{\lambda_\alpha}[\mathcal{R}[\iota_f]; \sigma'_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta^{\lambda_\beta}[a_\beta; \sigma'_\beta; \iota_\beta; F_\beta; R_\beta :: [m_j; \iota''; f_i^{\alpha\to\beta}]; f_\beta] \parallel P\end{array}} \text{(SecREQUEST)}$$

$$\frac{\begin{array}{c}\sigma_\alpha(\iota) = fut(f_i^{\gamma\to\beta}) \qquad F_\beta(f_i^{\gamma\to\beta}) = \iota_f \\ \sigma'_\alpha = Copy\&Merge(\sigma_\beta, \iota_f \ ; \ \sigma_\alpha, \iota) \qquad (\beta, \alpha, Rp_{\beta\to\alpha}(\sigma_\beta(\iota_f))) \in \mathcal{T}\end{array}}{\begin{array}{c}\alpha^{\lambda_\alpha}[a_\alpha; \sigma_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta^{\lambda_\beta}[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P \longrightarrow \\ \alpha^{\lambda_\alpha}[a_\alpha; \sigma'_\alpha; \iota_\alpha; F_\alpha; R_\alpha; f_\alpha] \parallel \beta^{\lambda_\beta}[a_\beta; \sigma_\beta; \iota_\beta; F_\beta; R_\beta; f_\beta] \parallel P\end{array}} \text{(SecREPLY)}$$

Parallel configurations are now of the form:

$$P, Q ::= \alpha^{\lambda_\alpha}[a; \sigma; \iota; F; R; f] \parallel \beta^{\lambda_\beta}[\cdots] \parallel \cdots$$

The Secure Information Flow

The Secure Information Flow

Agenda

- The concept *elementary flow of information* is based on the "release" or transmission of information from an activity

The Secure Information Flow

- The concept *elementary flow of information* is based on the "release" or transmission of information from an activity

- Hence, it is derived the **secure information flow** notion:

$$\frac{(\alpha, \beta, Rq_{\alpha \to \beta}(\sigma(\iota'), \lambda_{in})) \in \mathcal{T}}{Sec\varphi_{\emptyset}(\alpha, \beta)} \qquad \frac{(\beta, \alpha, Rp_{\beta \to \alpha}(\sigma_{\alpha}(\iota_f))) \in \mathcal{T}}{Sec\varphi_{\emptyset}(\beta, \alpha)}$$

$$\frac{(\alpha, \gamma, Nw(\gamma, \lambda_{\gamma})) \in \mathcal{T}}{Sec\varphi_{\emptyset}(\alpha, \gamma)}$$

The Secure Information Flow

- The concept *elementary flow of information* is based on the "release" or transmission of information from an activity
- Hence, it is derived the **secure information flow** notion:

$$\frac{(\alpha, \beta, Rq_{\alpha\rightarrow\beta}(\sigma(\iota'), \lambda_{in})) \in \mathcal{T}}{Sec\varphi_{\emptyset}(\alpha, \beta)} \qquad \frac{(\beta, \alpha, Rp_{\beta\rightarrow\alpha}(\sigma_{\alpha}(\iota_f))) \in \mathcal{T}}{Sec\varphi_{\emptyset}(\beta, \alpha)}$$

$$\frac{(\alpha, \gamma, Nw(\gamma, \lambda_{\gamma})) \in \mathcal{T}}{Sec\varphi_{\emptyset}(\alpha, \gamma)}$$

The syntax $Sec\varphi_{\emptyset}(\alpha, \beta)$ means there is a secure flow ($Sec\varphi$), with no other intermediate activities ($\emptyset$), happening between activities α and β

The Secure Path for Information Flow

Agenda

The Secure Path for Information Flow

- A *flow of information* is composed of several elementary flows happening in a sequential order

The Secure Path for Information Flow

- A *flow of information* is composed of several elementary flows happening in a sequential order

- A *flow-path* (fp) is produced when intermediate activities are present in between the communication of two given activities (i.e. the end points)

The Secure Path for Information Flow

- A *flow of information* is composed of several elementary flows happening in a sequential order
- A *flow-path* (fp) is produced when intermediate activities are present in between the communication of two given activities (i.e. the end points)
- Formally, the *secure path for information flow* is:

$$\frac{Sec\varphi_{fp_1}(\alpha,\gamma) \qquad Sec\varphi_{fp_2}(\gamma,\beta)}{Sec\varphi_{fp_1.\gamma.fp_2}(\alpha,\beta)}$$

The Secure Path for Information Flow

- A *flow of information* is composed of several elementary flows happening in a sequential order
- A *flow-path* (fp) is produced when intermediate activities are present in between the communication of two given activities (i.e. the end points)
- Formally, the *secure path for information flow* is:

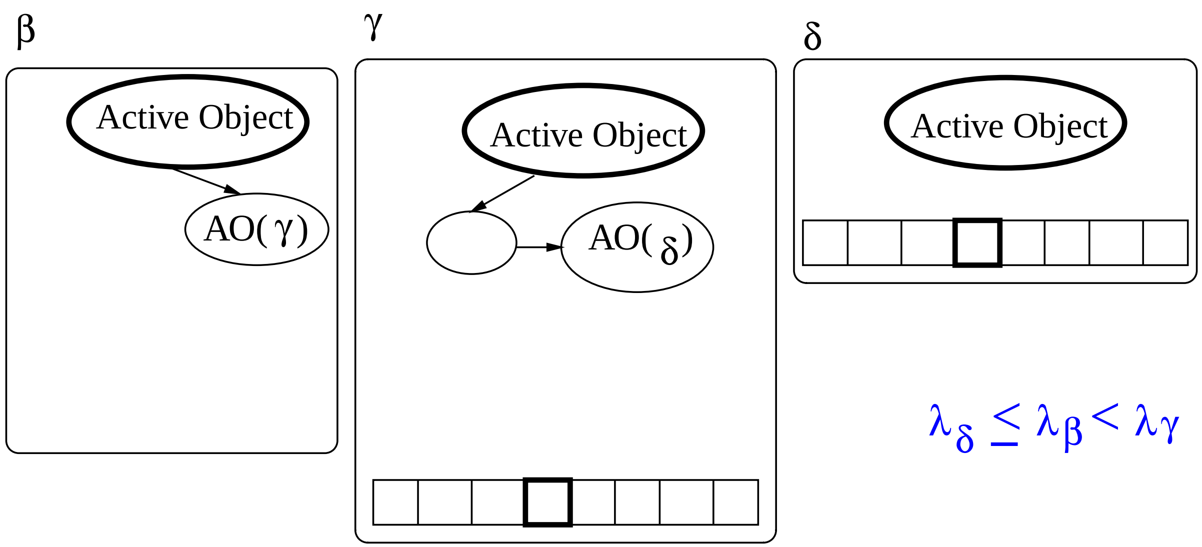
$$\frac{Sec\varphi_{fp_1}(\alpha,\gamma) \qquad Sec\varphi_{fp_2}(\gamma,\beta)}{Sec\varphi_{fp_1.\gamma.fp_2}(\alpha,\beta)}$$

- There is a secure information flow from end-to-end on any flow path when:

$$Sec\varphi_{\gamma_1\cdots\gamma_n}(\alpha,\beta) \iff$$
$$Sec\varphi_{\emptyset}(\alpha,\gamma_1) \wedge Sec\varphi_{\emptyset}(\gamma_1,\gamma_2) \wedge \cdots \wedge Sec\varphi_{\emptyset}(\gamma_n,\beta)$$

Service-Oriented Computing and *futures*

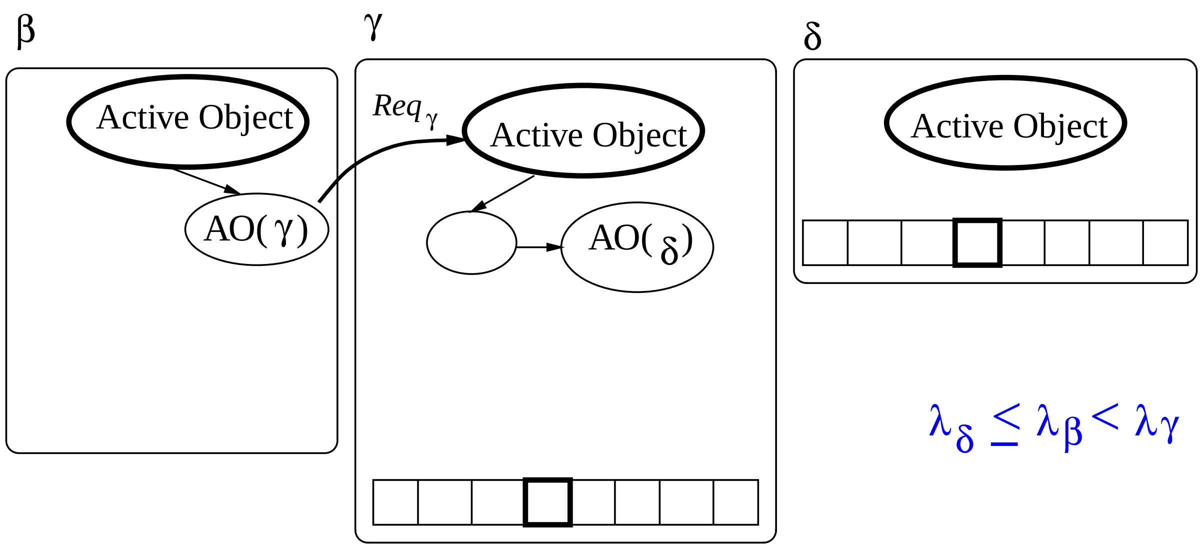
Impossible future updates with symmetric patterns of communications



$\lambda_\delta \leq \lambda_\beta < \lambda_\gamma$

Service-Oriented Computing and *futures*

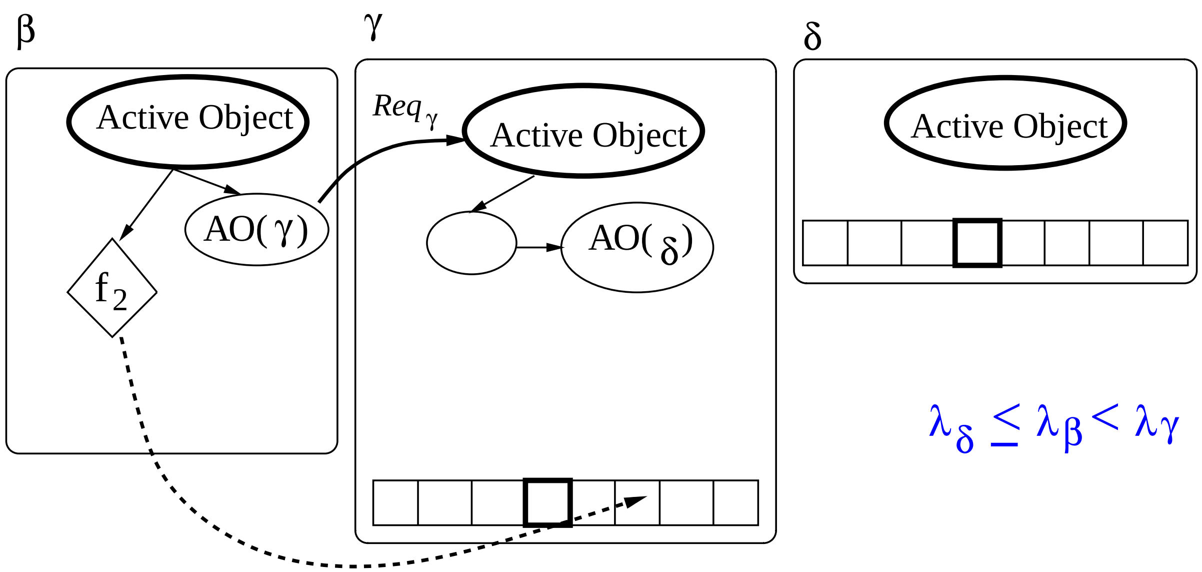
Impossible future updates with symmetric patterns of communications



$\lambda_\delta \leq \lambda_\beta < \lambda_\gamma$

Service-Oriented Computing and *futures*

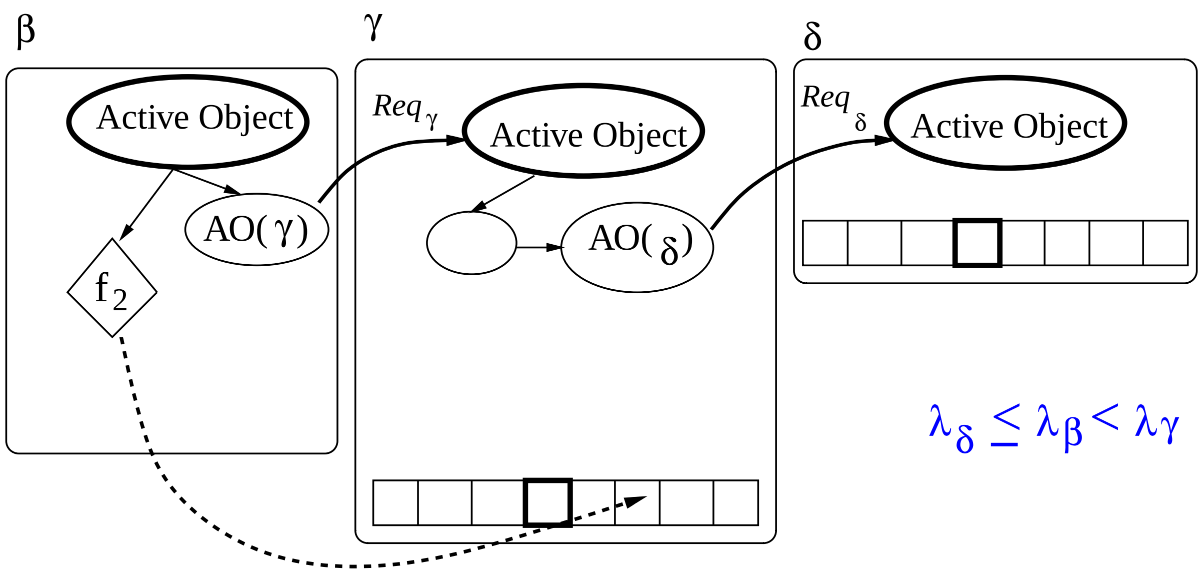
Impossible future updates with symmetric patterns of communications



$\lambda_\delta \leq \lambda_\beta < \lambda_\gamma$

Service-Oriented Computing and *futures*

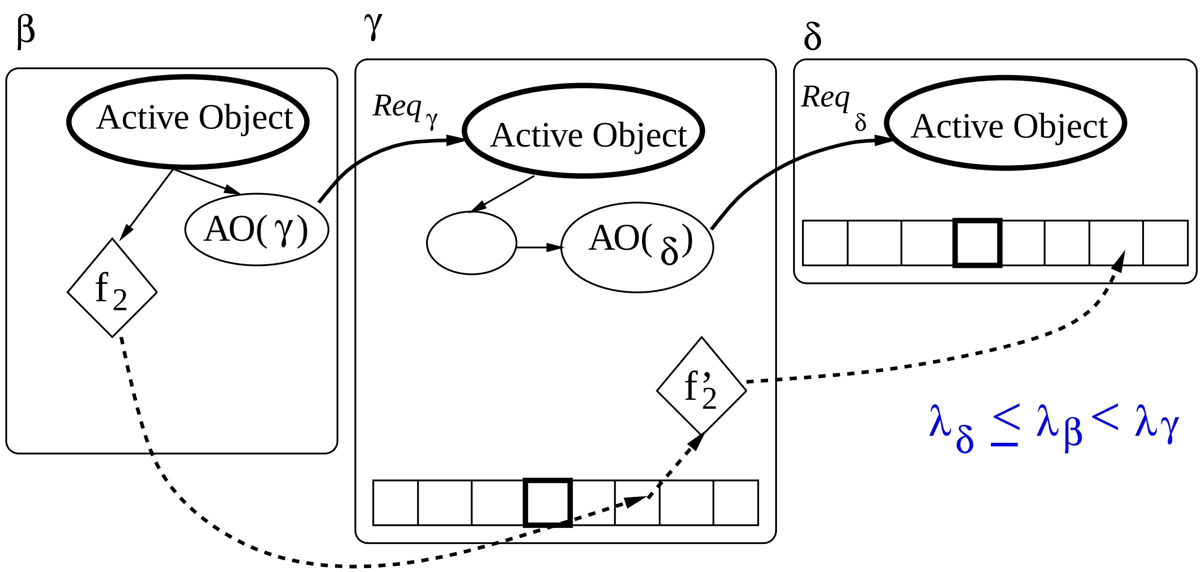
Impossible future updates with symmetric patterns of communications



$\lambda_\delta \leq \lambda_\beta < \lambda_\gamma$

Service-Oriented Computing and *futures*

Impossible future updates with symmetric patterns of communications

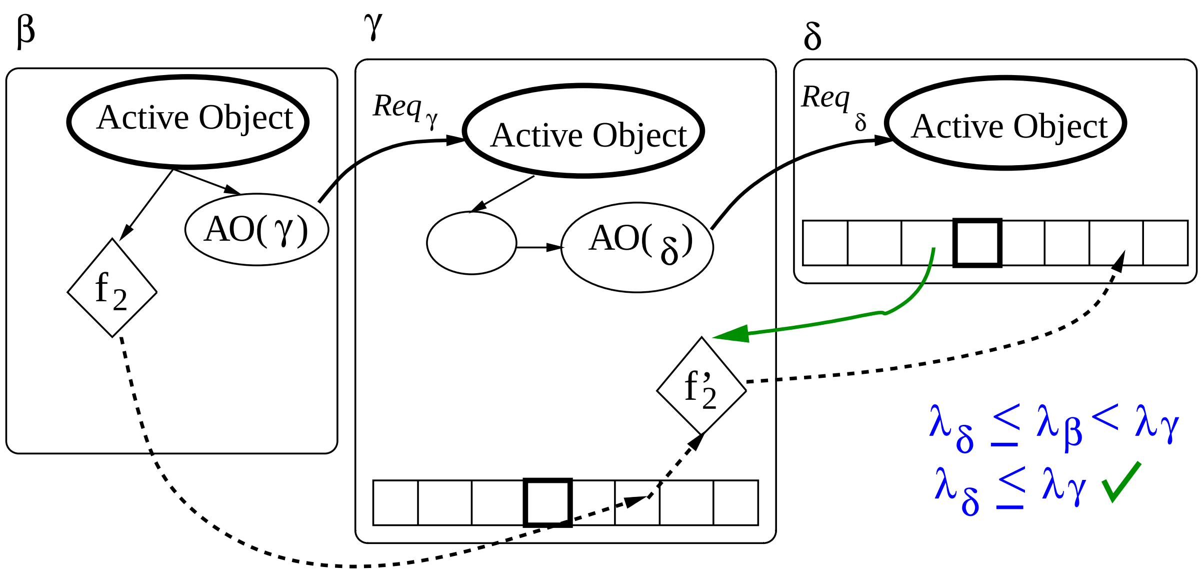


$\lambda_\delta \leq \lambda_\beta < \lambda\gamma$

Service-Oriented Computing and *futures*

Agenda

Impossible future updates with symmetric patterns of communications



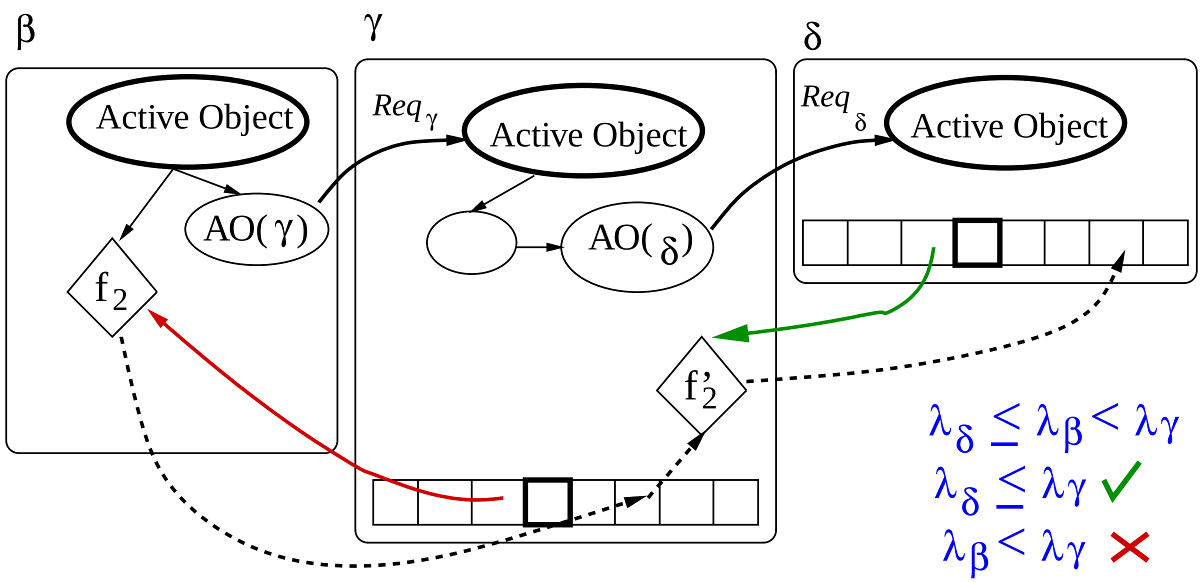
$\lambda_\delta \leq \lambda_\beta < \lambda_\gamma$

$\lambda_\delta \leq \lambda_\gamma$ ✓

Service-Oriented Computing and *futures*

Agenda

Impossible future updates with symmetric patterns of communications



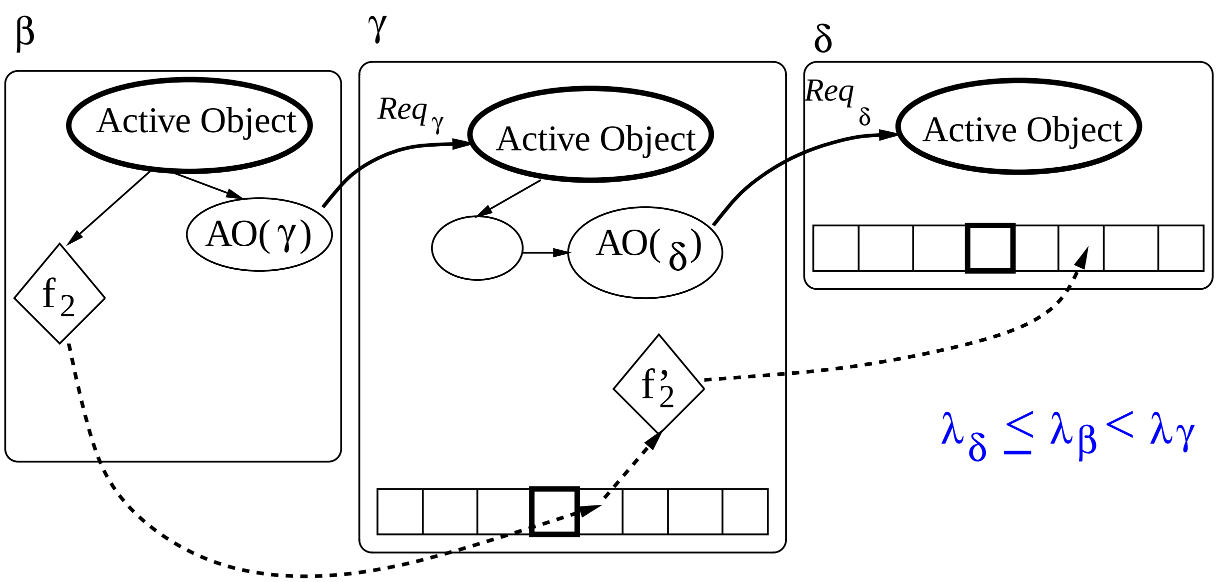
$\lambda_\delta \leq \lambda_\beta < \lambda_\gamma$

$\lambda_\delta \leq \lambda_\gamma$ ✓

$\lambda_\beta < \lambda_\gamma$ ✗

Service-Oriented Computing and *futures* (contd.)

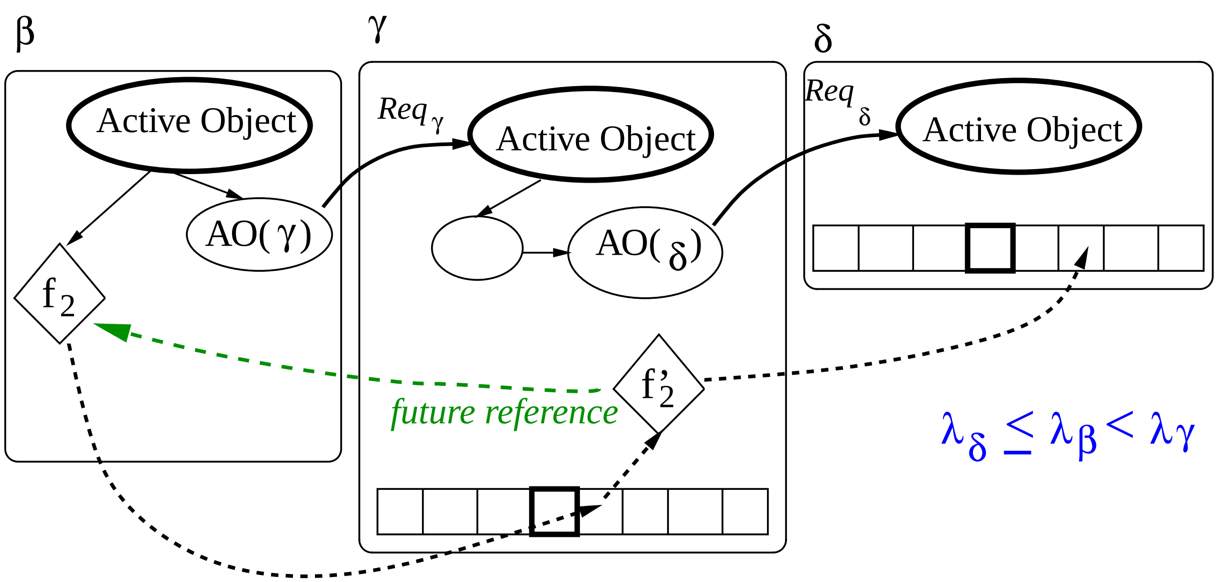
Future updates are possible in asymmetric patterns of communications



$\lambda_\delta \leq \lambda_\beta < \lambda_\gamma$

Service-Oriented Computing and *futures* (contd.)

Future updates are possible in asymmetric patterns of communications

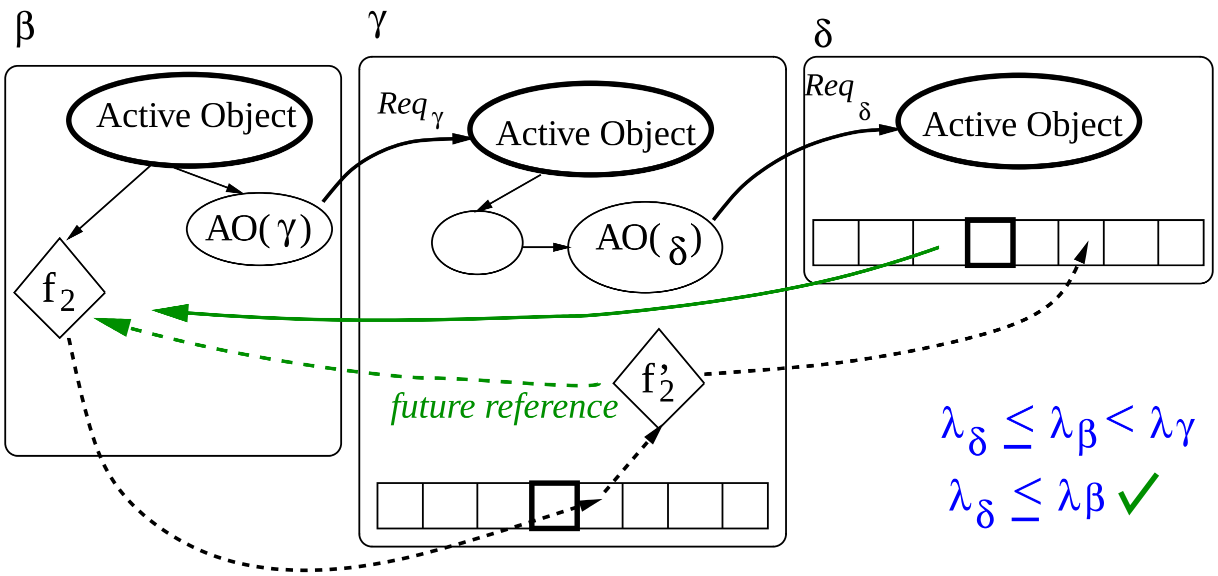


Service-Oriented Computing and *futures* (contd.)

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

Future updates are possible in asymmetric patterns of communications

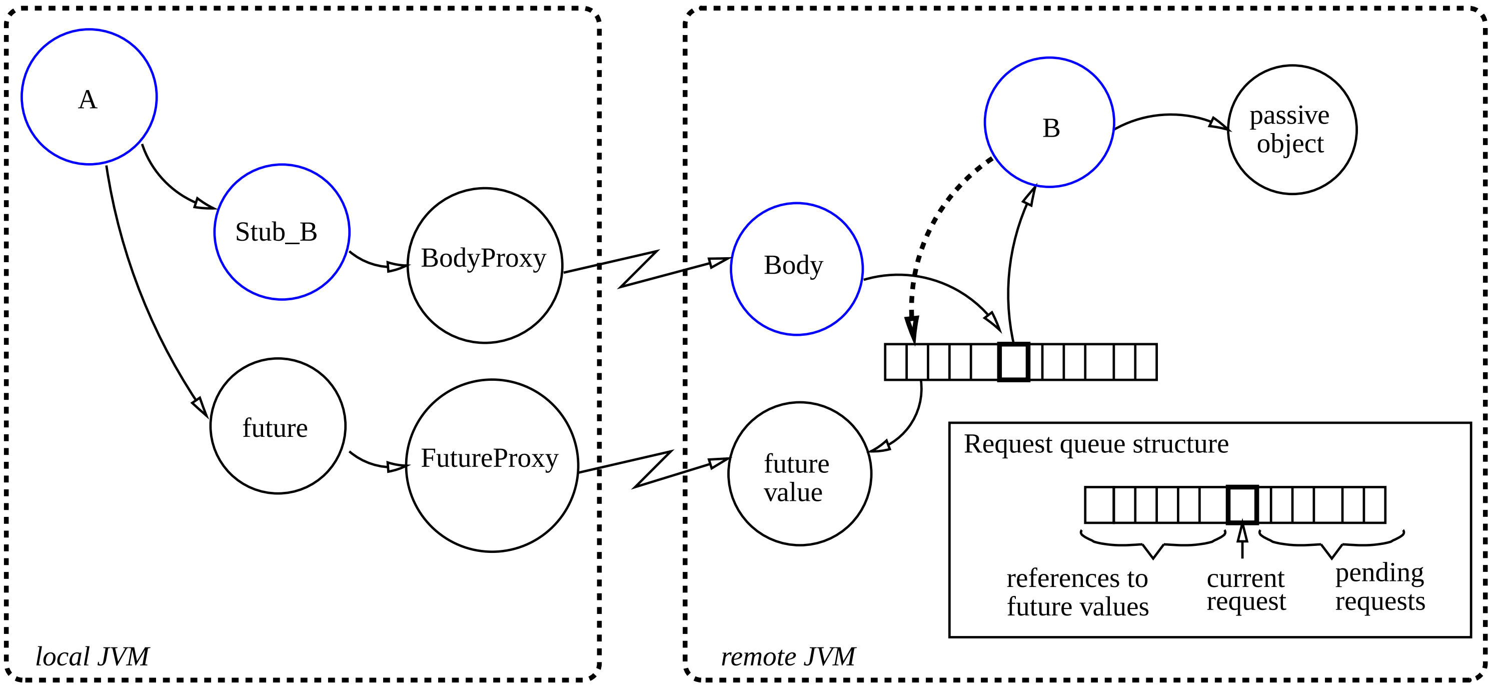


$\lambda_\delta \leq \lambda_\beta < \lambda_\gamma$

$\lambda_\delta \leq \lambda_\beta$ ✓

Implementation of the Security Model

Architecture of *active objects*



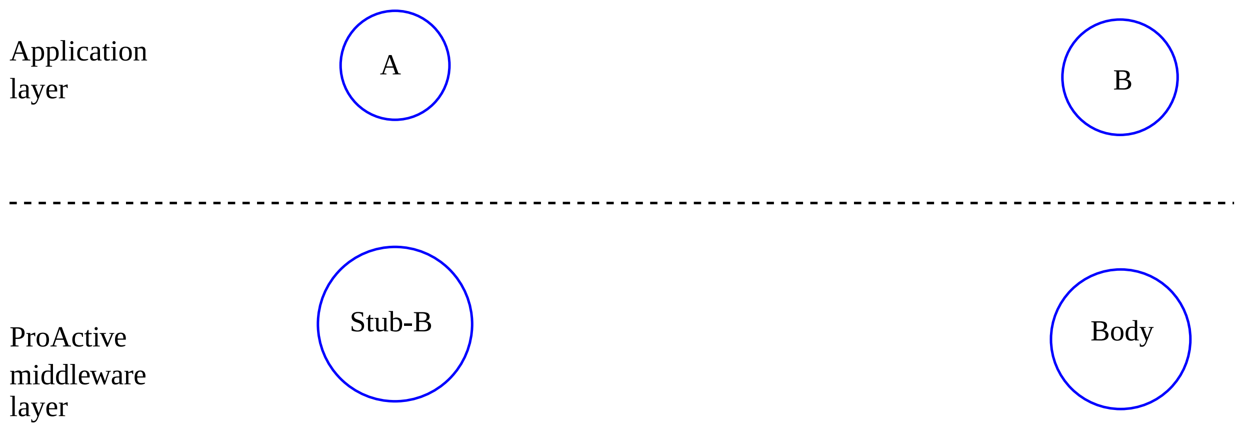
ArtINRIA

Implementation of the Security Model *(contd.)*

Agenda

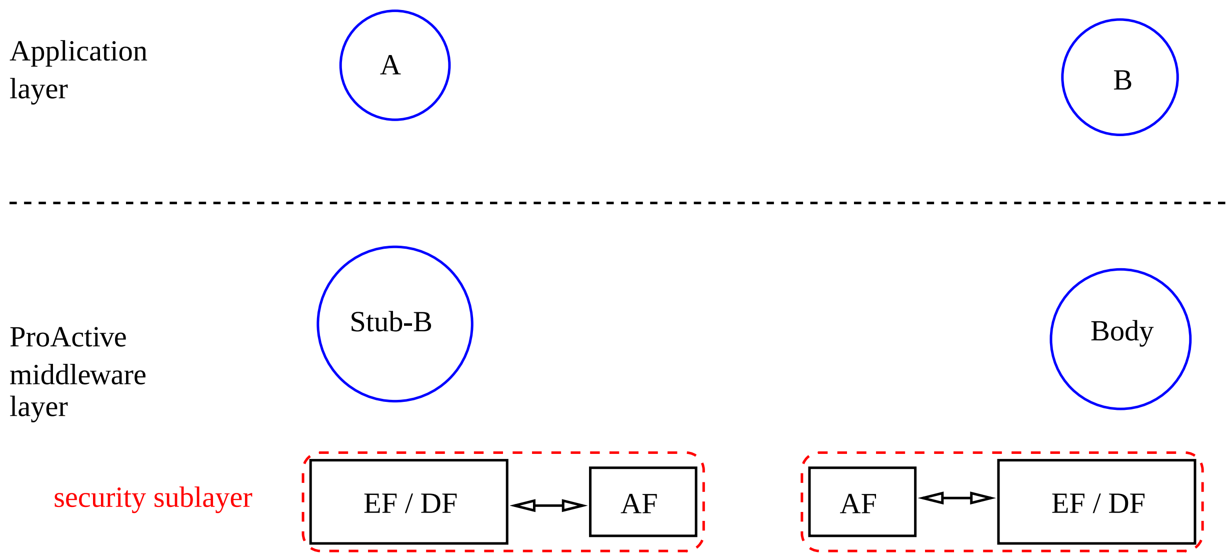
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

Security schema for *active objects*



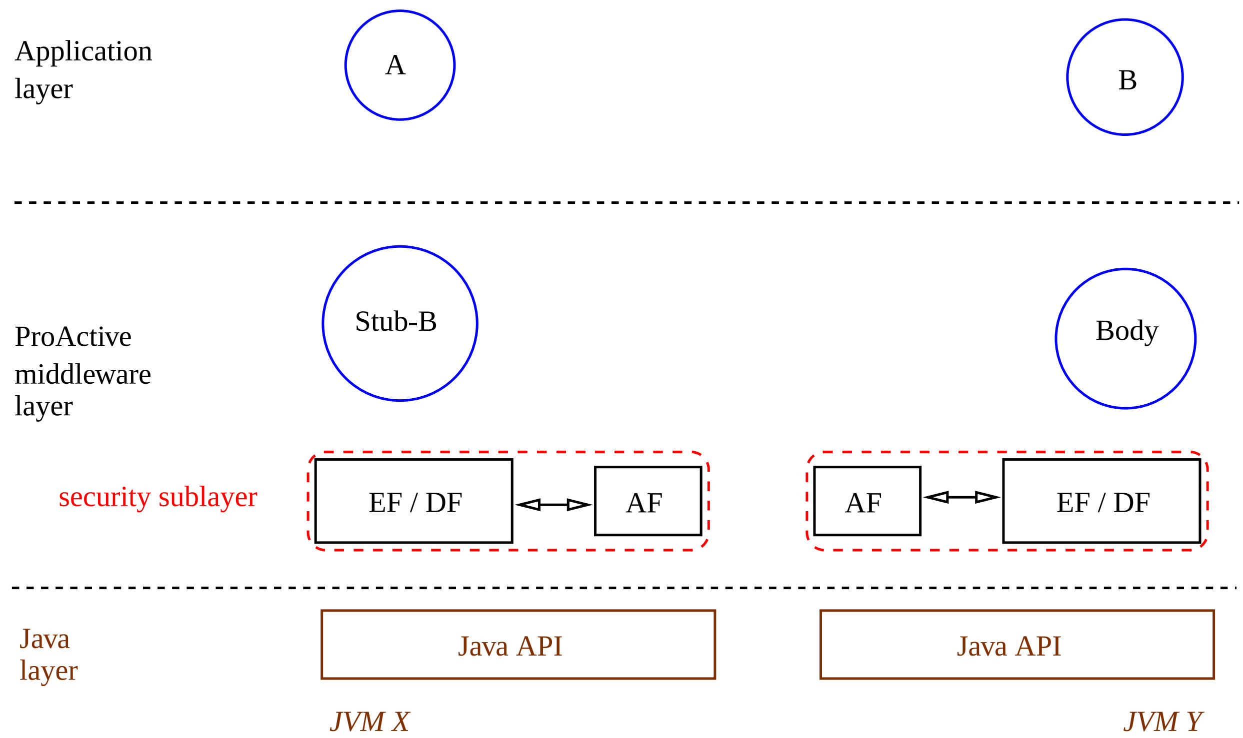
Implementation of the Security Model *(contd.)*

Security schema for *active objects*



Implementation of the Security Model *(contd.)*

Security schema for *active objects*

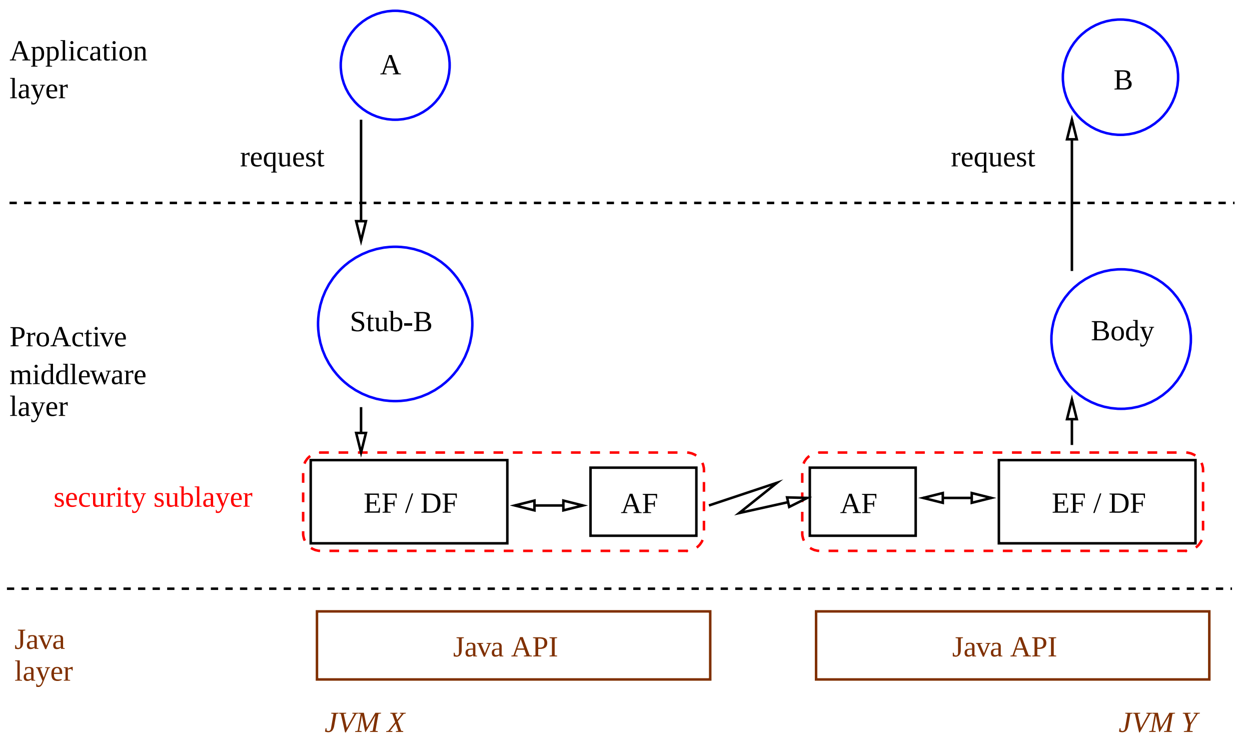


Implementation of the Security Model *(contd.)*

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

Security schema for *active objects*

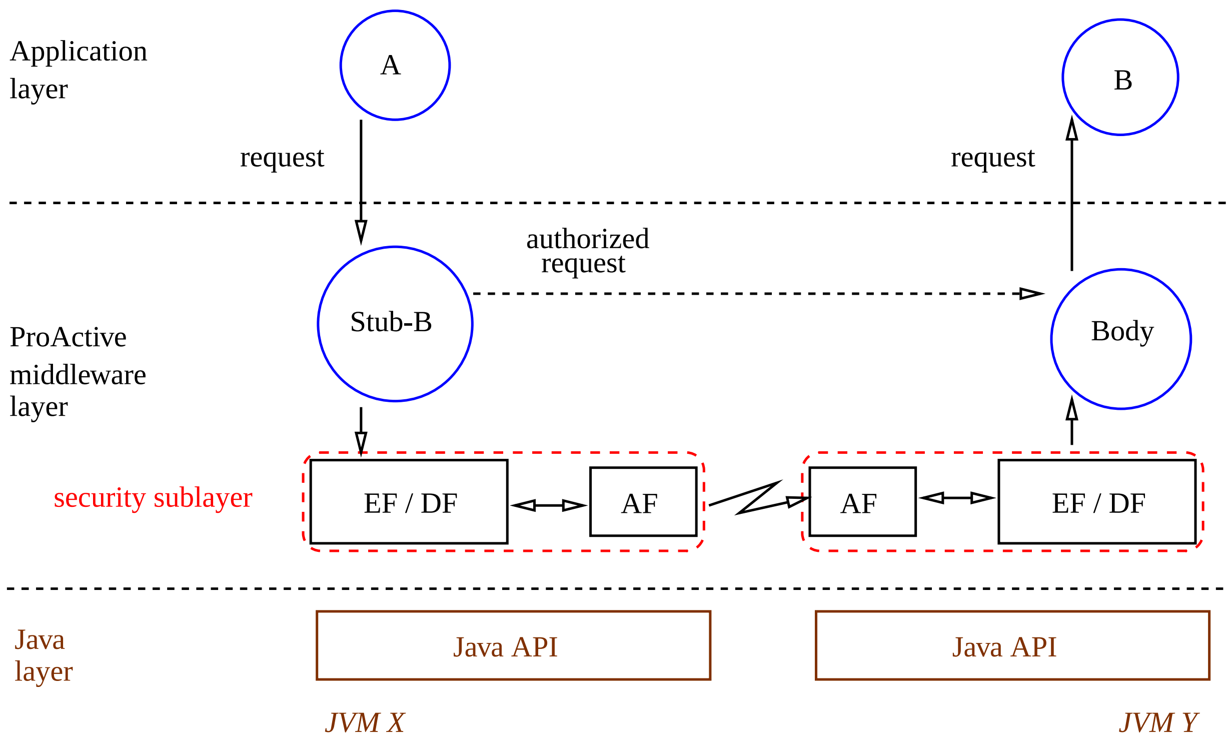


Implementation of the Security Model *(contd.)*

Agenda

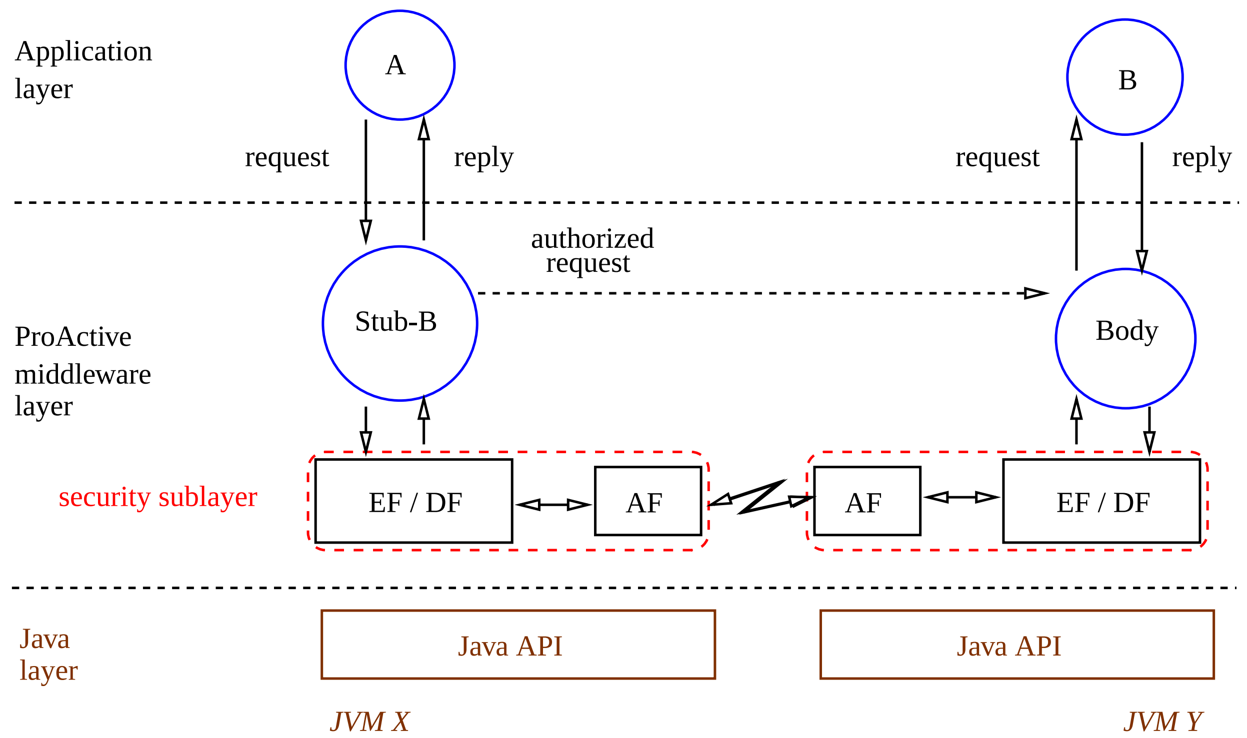
- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

Security schema for *active objects*



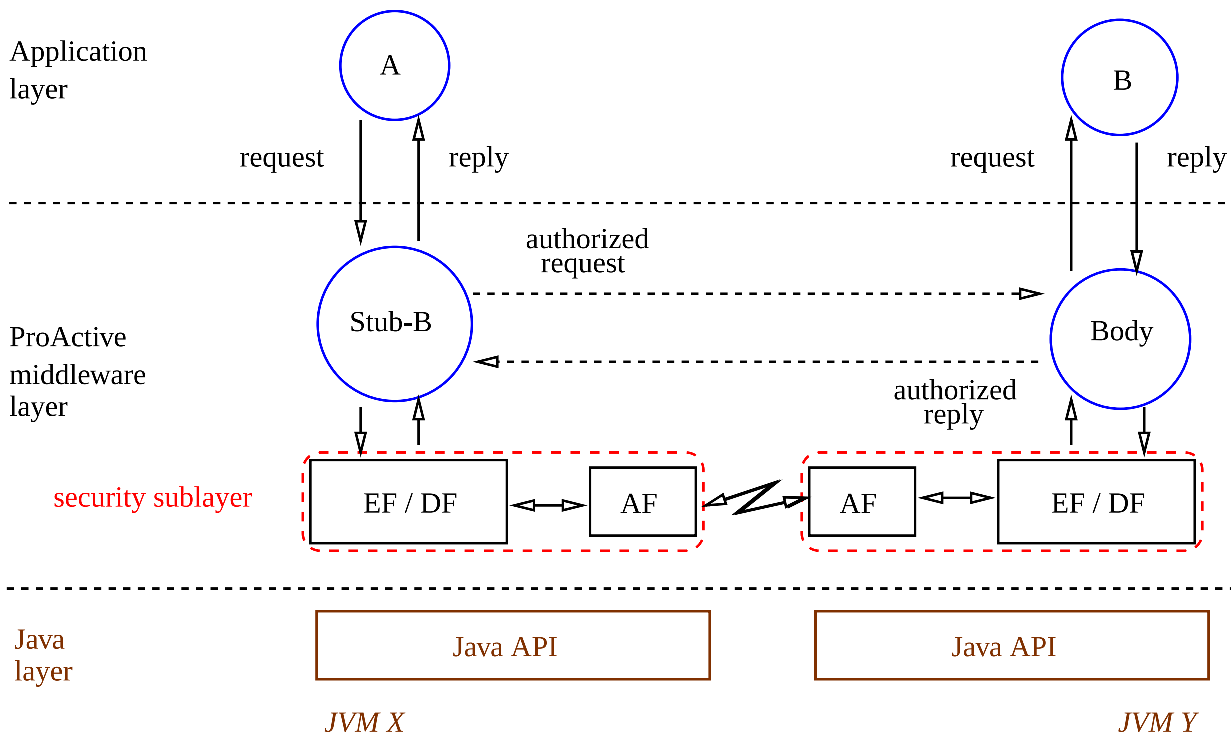
Implementation of the Security Model *(contd.)*

Security schema for *active objects*



Implementation of the Security Model *(contd.)*

Security schema for *active objects*



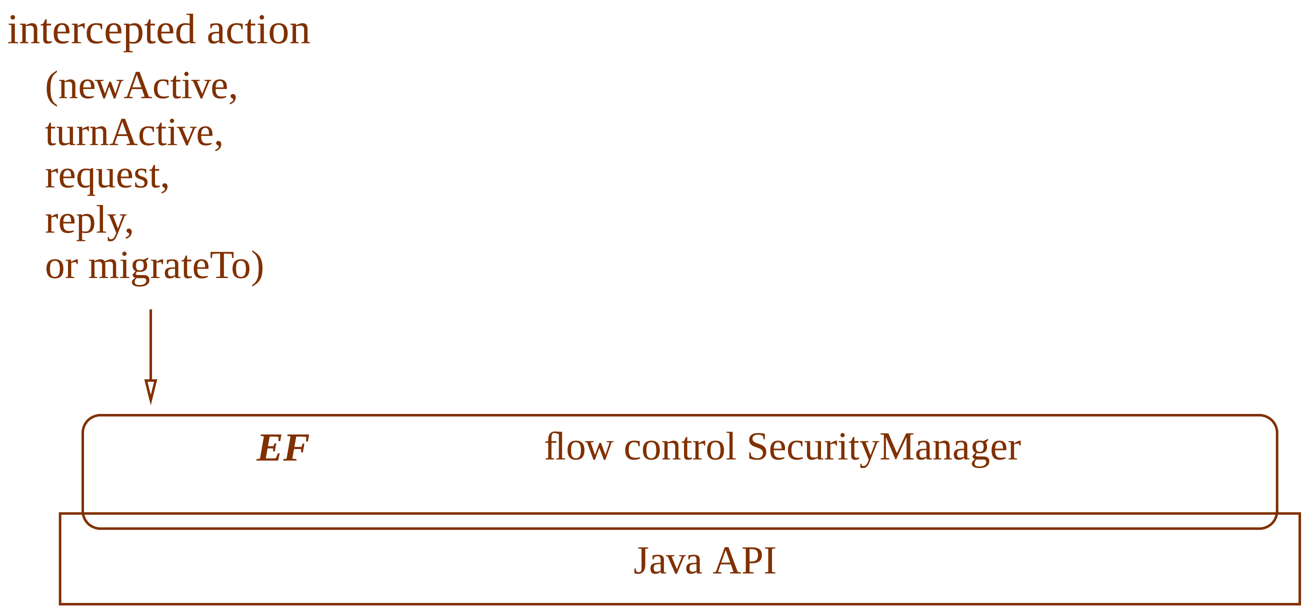
Implementation of the Security Model *(contd.)*

Agenda

Detailed Security sub-layer

Implementation of the Security Model *(contd.)*

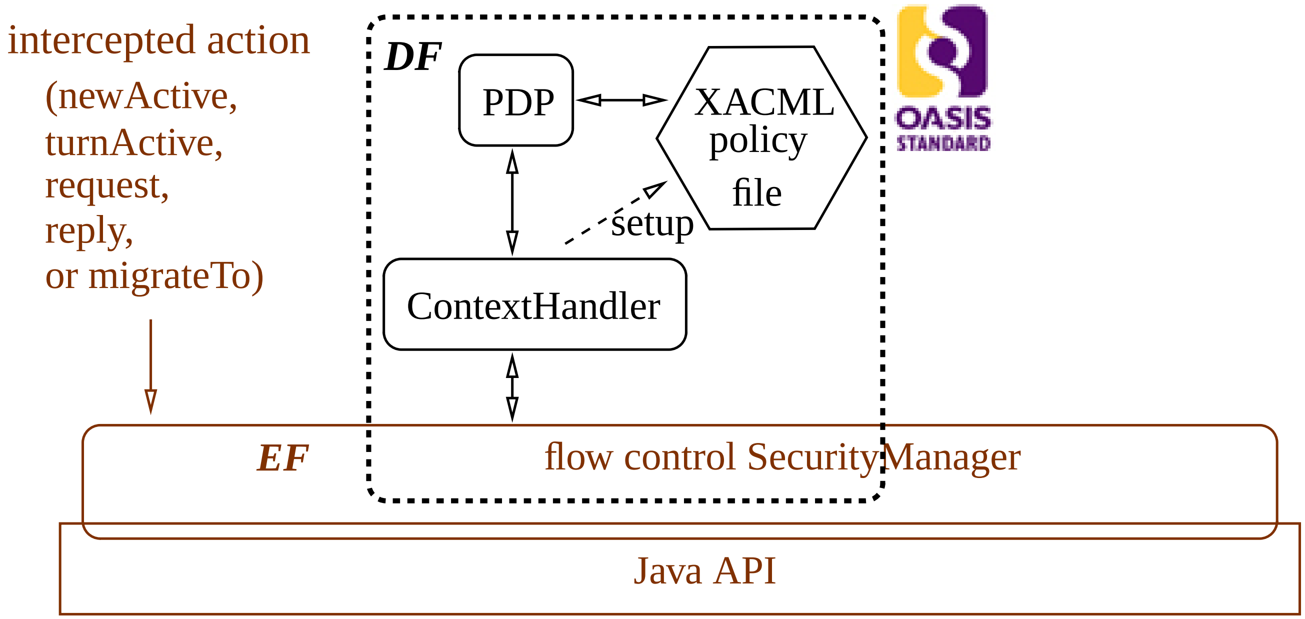
Detailed Security sub-layer



- EF = flow control mechanism as a Java Security Manager

Implementation of the Security Model *(contd.)*

Detailed Security sub-layer

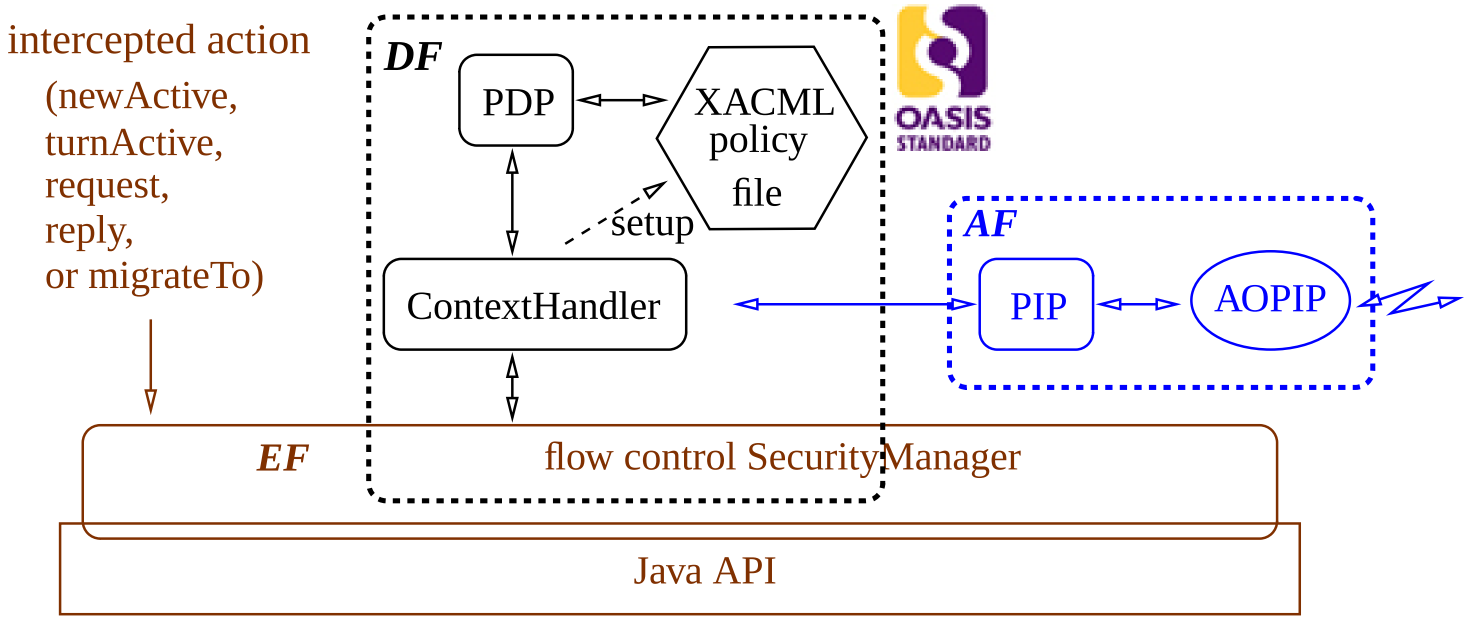


- EF = flow control mechanism as a Java Security Manager
- DF = Context Handler + Policy Decision Point + XACML file

Implementation of the Security Model *(contd.)*

Agenda

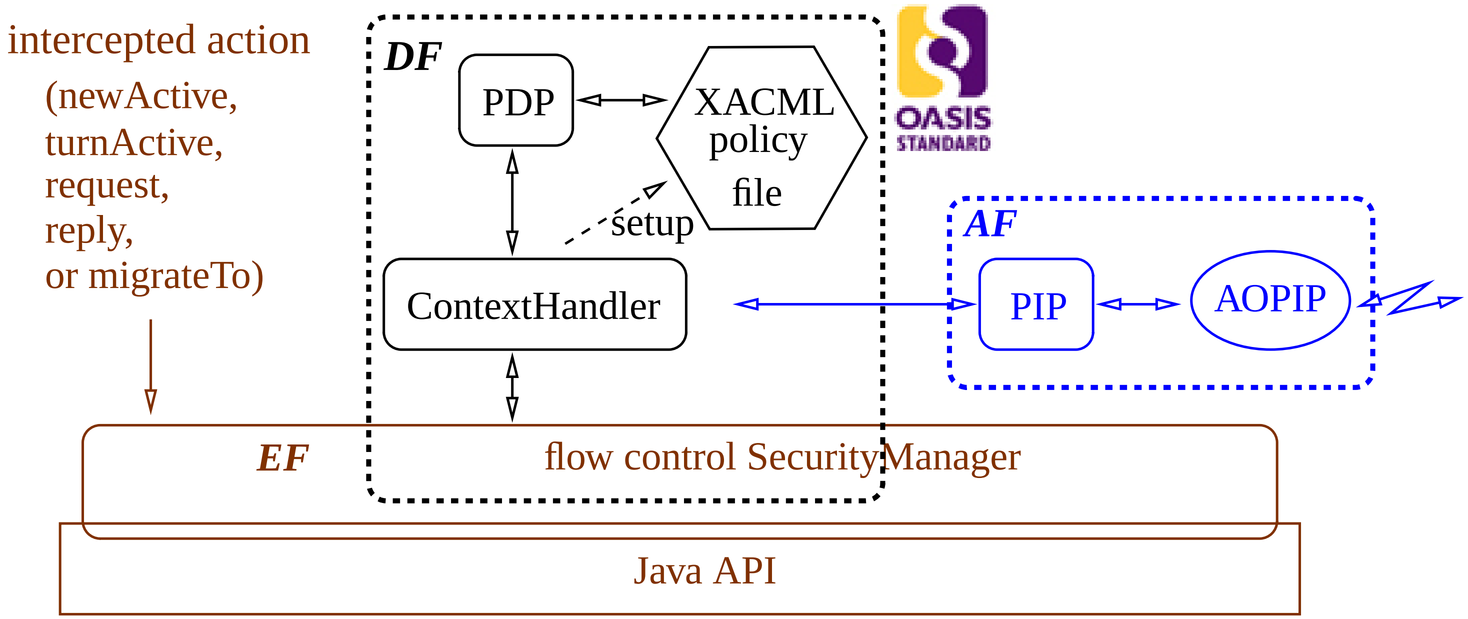
Detailed Security sub-layer



- EF = flow control mechanism as a Java Security Manager
- DF = Context Handler + Policy Decision Point + XACML file
- AF = Policy Information Point + *active object* PIP

Implementation of the Security Model *(contd.)*

Detailed Security sub-layer



- EF = flow control mechanism as a Java Security Manager
- DF = Context Handler + Policy Decision Point + XACML file
- AF = Policy Information Point + *active object* PIP

Conclusions

Agenda

Conclusions

- Expresiveness:
  - Assignation of specific security levels to request parameters and created activities

Conclusions

- Expresiveness:
  - Assignation of specific security levels to request parameters and created activities
- Scalability:
  - Dynamic checks performed only at activity creation, and inter-activity communications

Conclusions

- Expresiveness:
  - Assignation of specific security levels to request parameters and created activities
- Scalability:
  - Dynamic checks performed only at activity creation, and inter-activity communications
- Extendable:
  - XACML features provide a finer control on the discretionary access control

Perspectives

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

Perspectives

Agenda

- TCSEC/ITSEC/CC level A/EAL7 can be attained (i.e. formal design and verification)

Perspectives

- TCSEC/ITSEC/CC level A/EAL7 can be attained (i.e. formal design and verification)
- Further study of covert channels in distributed systems

Perspectives

- TCSEC/ITSEC/CC level A/EAL7 can be attained (i.e. formal design and verification)
- Further study of covert channels in distributed systems
- Static type checking in Java can be complemented with our model

Perspectives

- TCSEC/ITSEC/CC level A/EAL7 can be attained (i.e. formal design and verification)
- Further study of covert channels in distributed systems
- Static type checking in Java can be complemented with our model
- The security mechanism can be applied to the Components paradigm

Q&A

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

Questions ?

Thank you for your attention

Q&A

Agenda

- Introduction
- Context
- Objectives
- Mechanisms
- ASP Security Model
- Implementation
- Conclusions

Questions ?

Thank you for your attention