Pushdown Systems and Weighted Pushdown Systems

Thomas Reps
University of Wisconsin

Joint work with
S. Jha, S. Schwoon, and S. Stubblebine

Topics

- Model checking of pushdown systems
- Context-sensitive dataflow analysis
- Authorization problems
- Authorization problems + privacy, recency, validity, and trust

- Jha, S. and Reps, T., Analysis of SPKI/SDSI certificates using model checking. In *Proc. of the 15th IEEE Computer Security Foundations Workshop*, 2002

- Schwoon, S., Jha, S., Reps, T., and Stubblebine, S., On generalized authorization problems. Submitted to *16th IEEE Computer Security Foundations Workshop*, 2003.

Outline

- Overview of SPKI/SDSI
  - Concepts
  - Certificate-analysis problems
- Translating SPKI/SDSI to PDSs
  - Translation
  - Solve certificate-analysis problems using model checking of PDSs

Motivation

- Traditionally, authorization is expressed using Access Control Lists or ACLs
  - Associate permissions with objects
  - For file F:

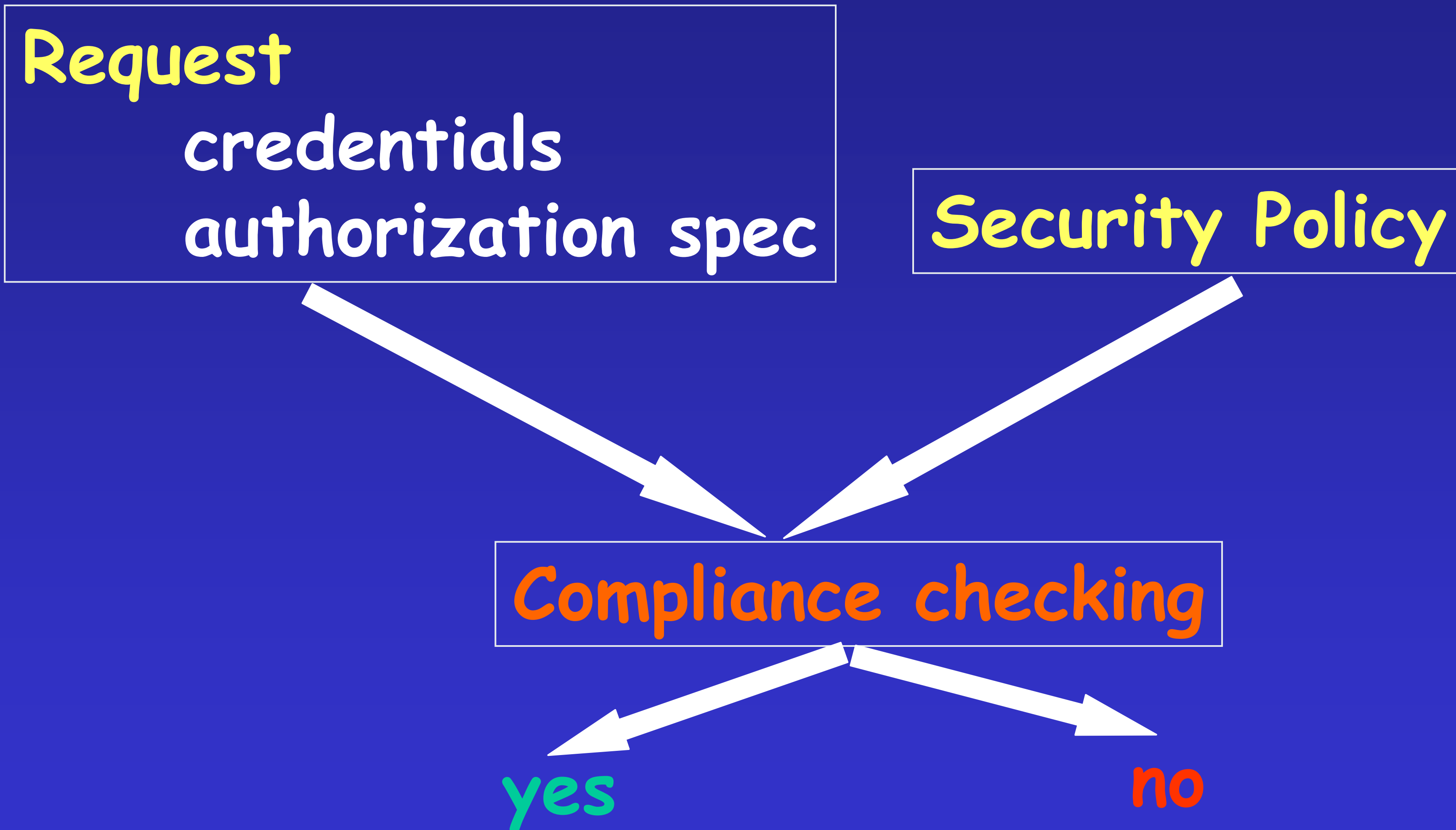
    reps : <r,w,x>

    jha : <r,x>

    reps-students : <r,x>

- Closed-world assumption
- Not appropriate in a distributed system

Trust Management



Trust Management Systems

- Request is the “proof” of authorization
- Several trust management systems
  - SPKI/SDSI
  - KeyNote
  - Referee, SD3, Binder, . . .

SPKI/SDSI

Principals (Public Keys)

R_H	Some host computer
K_{CS}	CS Department
K_{Bob}, K_{Alice}	Individuals

Local Names

K_{CS} faculty

K_{Bob} myStudents

Extended Names

K_{Bob} myStudents Spouses

Name Certs

Bob is a CS faculty member

K_{CS} faculty $\rightarrow K_{Bob}$

Alice is a student of Bob's

K_{Bob} myStudents $\rightarrow K_{Alice}$

Name Certs

Bob is a CS faculty

K_{CS} faculty $\to K$

Each Name cert also has a Validity specification

(usually a time interval)

Alice is a student of Bob's

K_{Bob} myStudents $\to K_{Alice}$

Auth Certs

A CS faculty member can use host R_H

R_H □ $\rightarrow$ K_{CS} faculty □

Can delegate

Bob allows his students to use host R_H

K_{Bob} □ $\rightarrow$ K_{Bob} myStudents ■

Cannot delegate

Alice allows access to her friends

K_{Alice} □ $\rightarrow$ K_{Alice} myFriends □

Example

```
# The lunch resource
  k_lunch_resource □

# AUTH CERT: Williamsburg Hosp. House lets conferences authorize lunch access
  k_lunch_resource □ → k_whh  conference □

# CIPSW is a conference at Williamsburg Hospitality House
  k_whh  conference → k_cipsw

# AUTH CERT: Conference organizers authorized to act on behalf of CIPSW
  k_cipsw □ → k_cipsw  organizer □

# The CIPSE organizers are . . .
  k_cipsw  organizer → k_wachter
  k_cipsw  organizer → k_toth

# AUTH CERT: Toth authorizes all attendees . . . but without delegation
  k_toth  □ → k_cipsw  attendee  ■

# List all attendees here
  k_cipsw  attendee → k_jha
  k_cipsw  attendee → k_reps
  k_cipsw  attendee → k_toth
  k_cipsw  attendee → k_wachter
  k_cipsw  attendee → k_clarke
```

Is k_wachter authorized to access k_lunch_resource?

Outline

- Overview of SPKI/SDSI
  - Concepts
  - Certificate-analysis problems
- Translating SPKI/SDSI to PDSs
  - Translation
  - Solve certificate-analysis problems using model checking of PDSs

Certificate Analysis

- Authorization access
  - Given a resource R and principal K, is K authorized to access R?
  - Solved by constructing a certificate chain that proves the authorization (certificate-chain discovery)

Certificate-Analysis Problems

- Shared access
  - Given two resources, R_1 and R_2, what principals can access both R_1 and R_2?
- Expiration vulnerability
  - What resources will principal K be prevented from accessing if certificate set C' expires?
- Universally guarded access
  - Is it the case that all authorizations that can be issued for a given resource R must involve a certificate signed by principal K?

More Certificate-Analysis Problems

- Many more . . .
  - Consult the CSFW '02 paper
- Main message
  - Model-checking algorithms for Pushdown Systems can be exploited to solve several certificate-analysis problems

Certificate Chain

R_H □

R_H □ → K_{CS} faculty □

K_{CS} fac

K_{Bob}

K_{Bob}

K_{Bob} udents → K_{Alice}

K_{Alice}

K_{Alice} □ → K_{Alice} myFriends □

Does not apply!

Outline

- Overview of SPKI/SDSI
  - Concepts
  - Certificate-analysis problems
- Translating SPKI/SDSI to PDSs
  - Translation
  - Solve certificate-analysis problems using model checking of PDSs

Pushdown Systems and SPKI/SDSI

Think states

Locations

{ R_H, K_{CS}, K_{Bob}, K_{Alice} }

Stack Symbols

{ □ , ■ , faculty, myStudents }

Transition Rules

If location is K_{Bob} and the top of the stack is $\square$, then

- pop $\square$ off the stack
- transition to location K_{Bob}
- push myStudents $\blacksquare$ on the stack

$\langle R_H, \square$

$\langle K_{CS},$

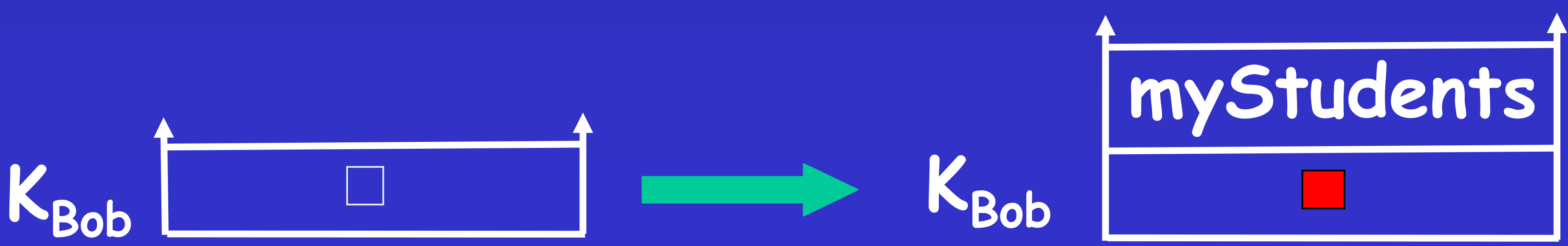
$\langle K_{Bob}, \square \rangle \to \langle K_{Bob}, \text{myStudents}\ \blacksquare \rangle$

$\langle K_{Bob}, \text{myStudents} \rangle \to \langle K_{Alice}, \varepsilon \rangle$

If location is K_{Bob} and the top of the stack is $\square$, then

- pop $\square$ off the stack
- transition to location K_{Bob}
- push myStudents $\blacksquare$ on the stack

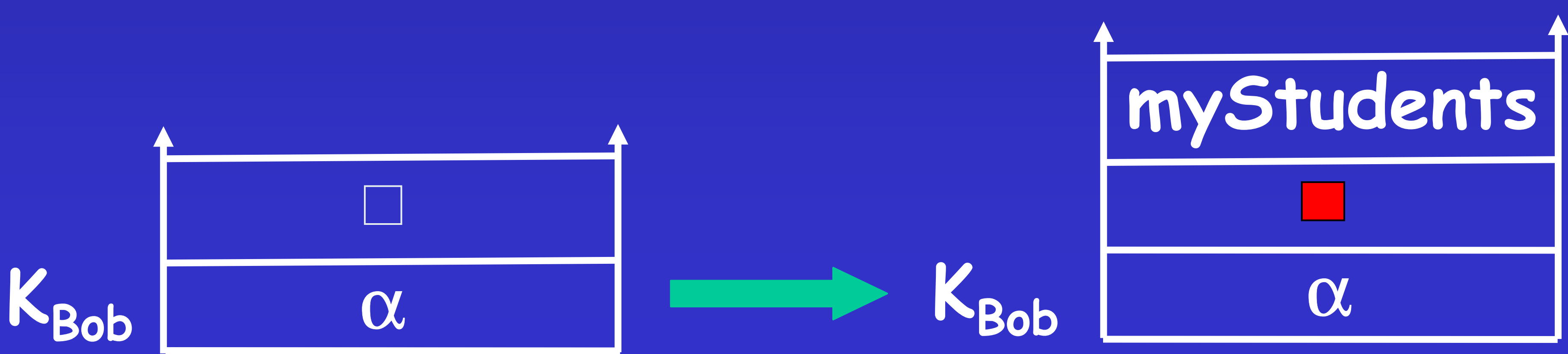
$$\langle K_{Bob}, \square \rangle \rightarrow \langle K_{Bob}, \text{myStudents}\ \blacksquare \rangle$$



If location is K_{Bob} and the top of the stack is $\square$, then

- pop $\square$ off the stack
- transition to location K_{Bob}
- push myStudents $\blacksquare$ on the stack

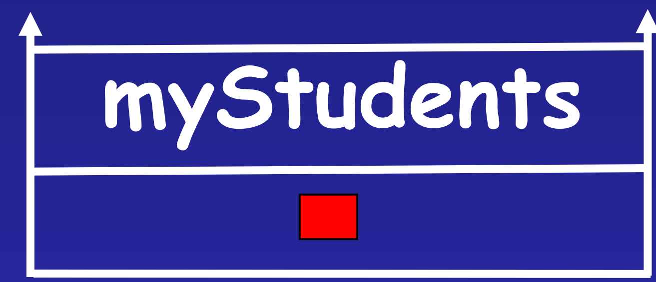
$$\langle K_{Bob}, \square \rangle \rightarrow \langle K_{Bob}, \text{myStudents} \blacksquare \rangle$$



PDS Terminology

Configuration

$\langle K_{Bob}$, myStudents ■ $\rangle$



$c \Rightarrow c'$

c′ follows from c by a transition rule

c **predecessor** of c′

c′ **successor** of c

$c_0 \Rightarrow c_1 \Rightarrow \ . \ . \ . \Rightarrow c_n$ **(a run)**

$c \Rightarrow^* c'$

reflexive transitive closure of $\Rightarrow$

A Certificate Chain is a Run

$\langle R_{H},$ □ $\rangle$

$\boxed{\langle R_{H,}\ □\rangle \rightarrow \langle K_{CS}, \text{faculty}\ □\rangle}$

$\langle K_{CS},$ faculty□ $\rangle$

$\boxed{\langle K_{CS}, \text{faculty}\rangle \rightarrow \langle K_{Bob}, \varepsilon\rangle}$

$\langle K_{Bob},$ □ $\rangle$

$\boxed{\langle K_{Bob}, □\ \rangle \rightarrow \langle K_{Bob}, \text{myStudents}\ ■\ \rangle}$

$\langle K_{Bob},$ myStudents ■ $\rangle$

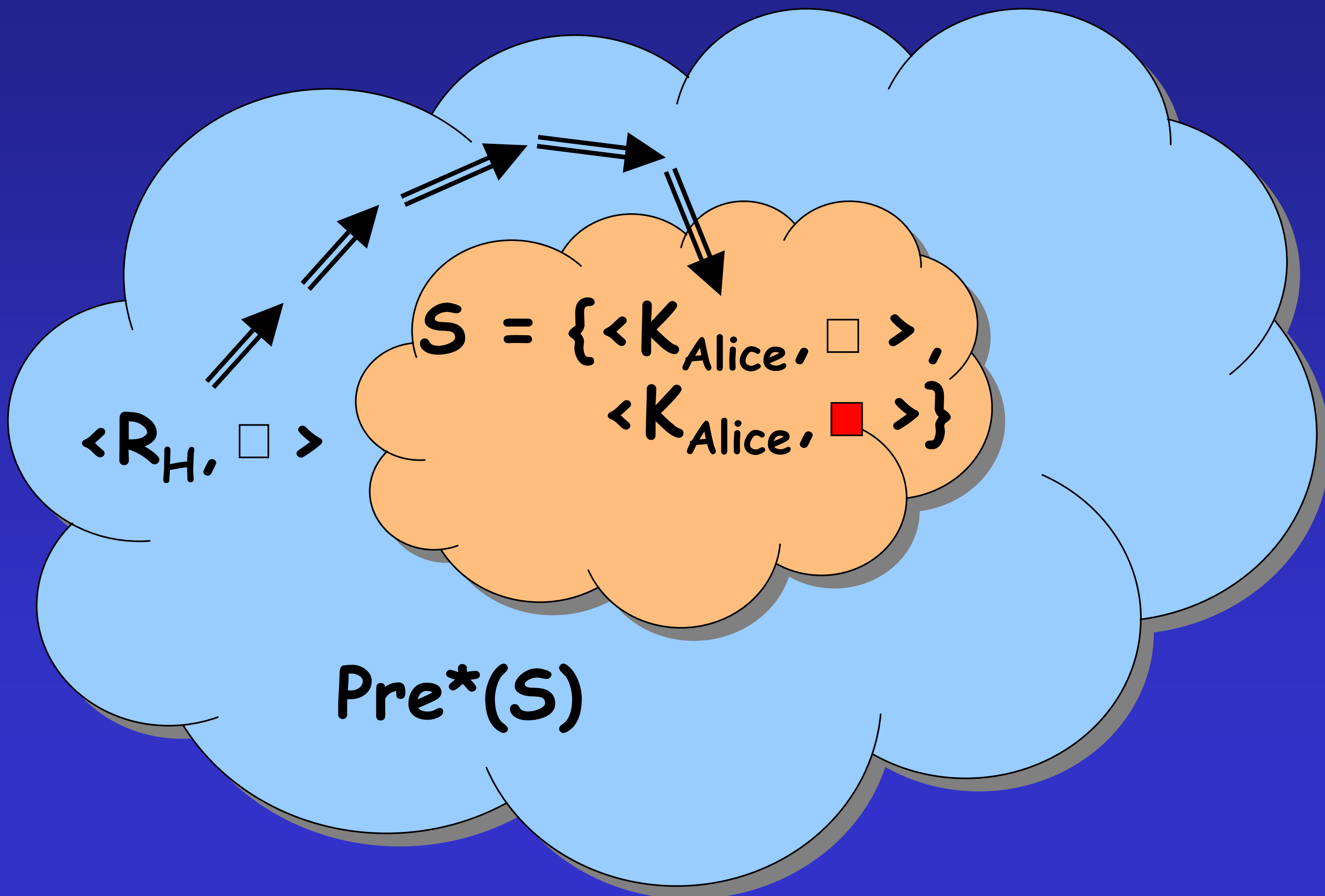
$\boxed{\langle K_{Bob}, \text{myStudents}\rangle \rightarrow \langle K_{Alice}, \varepsilon\ \rangle}$

$\langle K_{Alice},$ ■ $\rangle$

Outline

- Overview of SPKI/SDSI
  - Concepts
  - Certificate-analysis problems
- Translating SPKI/SDSI to PDSs
  - Translation
  - Solve certificate-analysis problems using model checking of PDSs

$Pre^*(\{\langle K_{Alice}, \square \rangle, \langle K_{Alice}, \blacksquare \rangle\})$



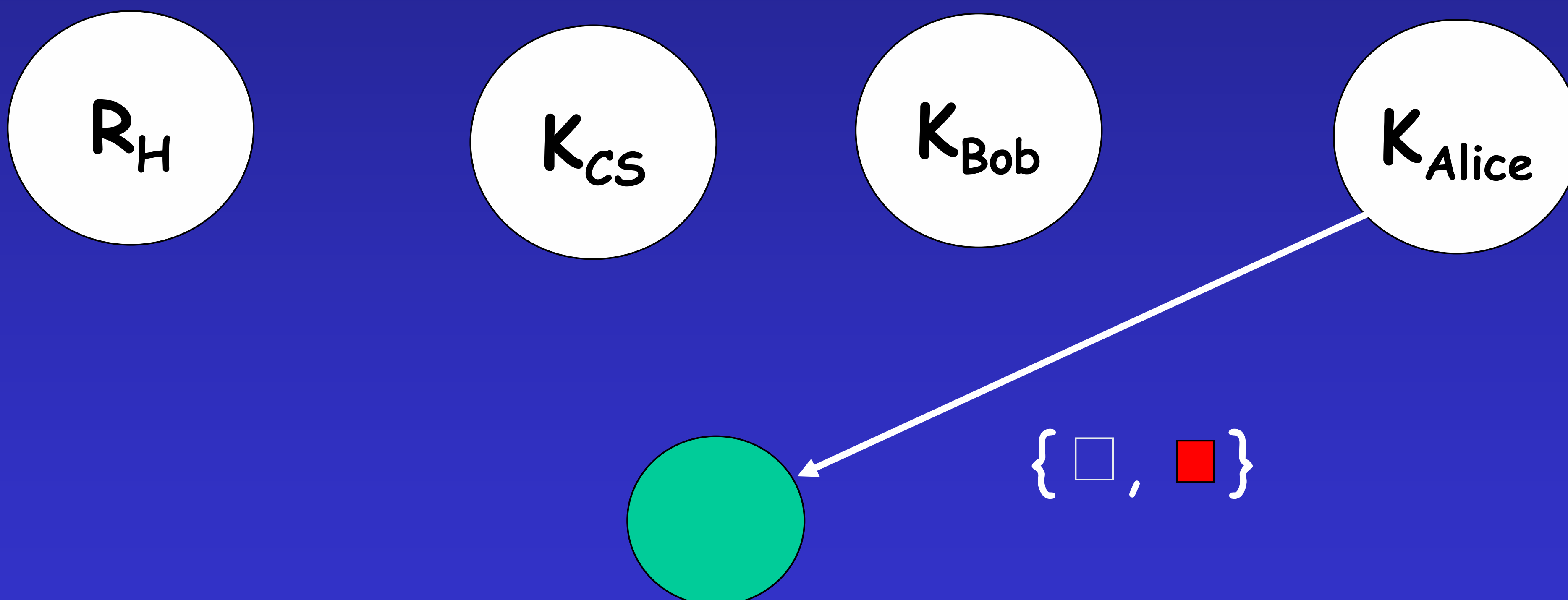
Representation Issue

- The set of configurations pre*(S) can be infinite
- Example
  - $\langle K,A\rangle \rightarrow \langle K,\varepsilon\rangle$
  - pre* ({$\langle K,A\rangle$}) = { $K\,A^i \mid i \geq 1$ }
- Solution in the PDS literature:

  Represent a set of configurations with an automaton

$$\begin{array}{c} \vdots \\ \langle K,AAA\rangle \\ \downarrow \\ \langle K,AA\rangle \\ \downarrow \\ \langle K,A\rangle \\ \downarrow \\ \langle K,\varepsilon\rangle \end{array}$$

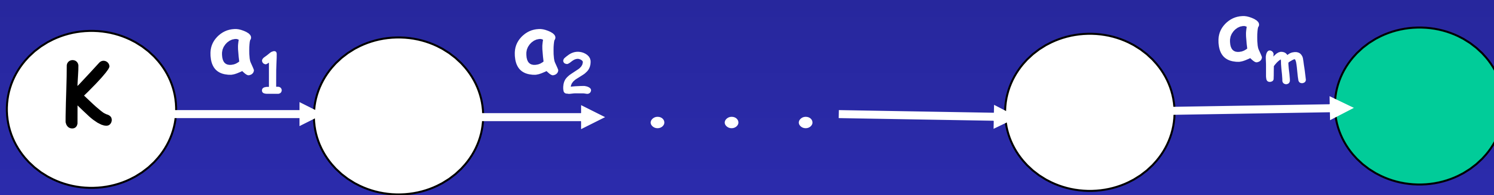
$\{\langle K_{Alice}, \square \rangle, \langle K_{Alice}, \blacksquare \rangle\}$



What Does the Automaton Represent?

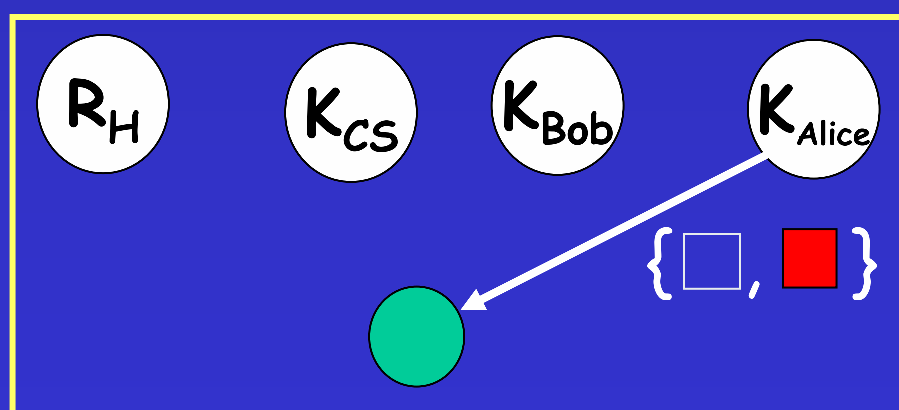
- A set of configurations:

  $\langle K, a_1 \ldots a_m \rangle$ is in the set if



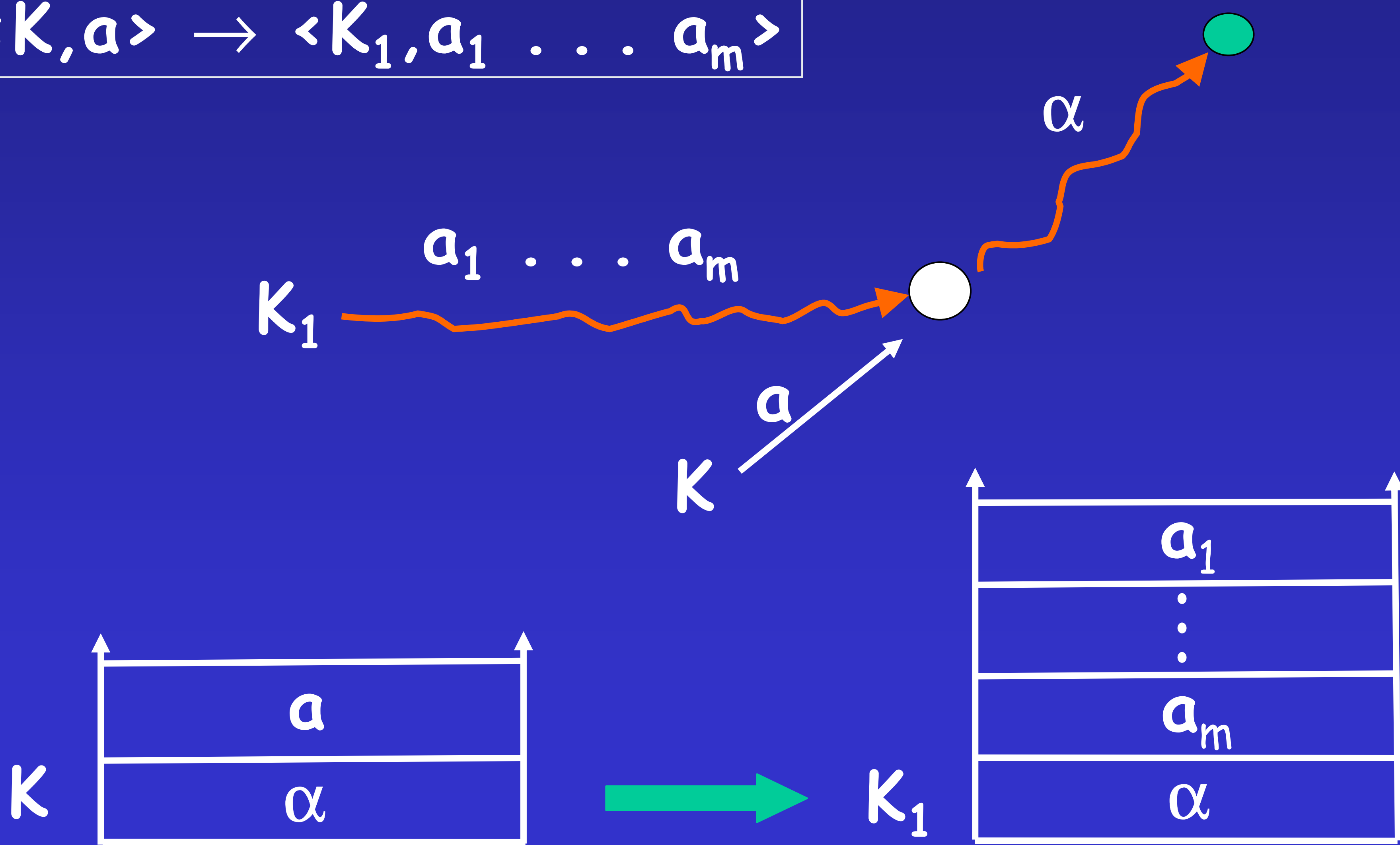
- Initial automaton represents

  $\{\langle K_{Alice}, \square \rangle, \langle K_{Alice}, \blacksquare \rangle\}$

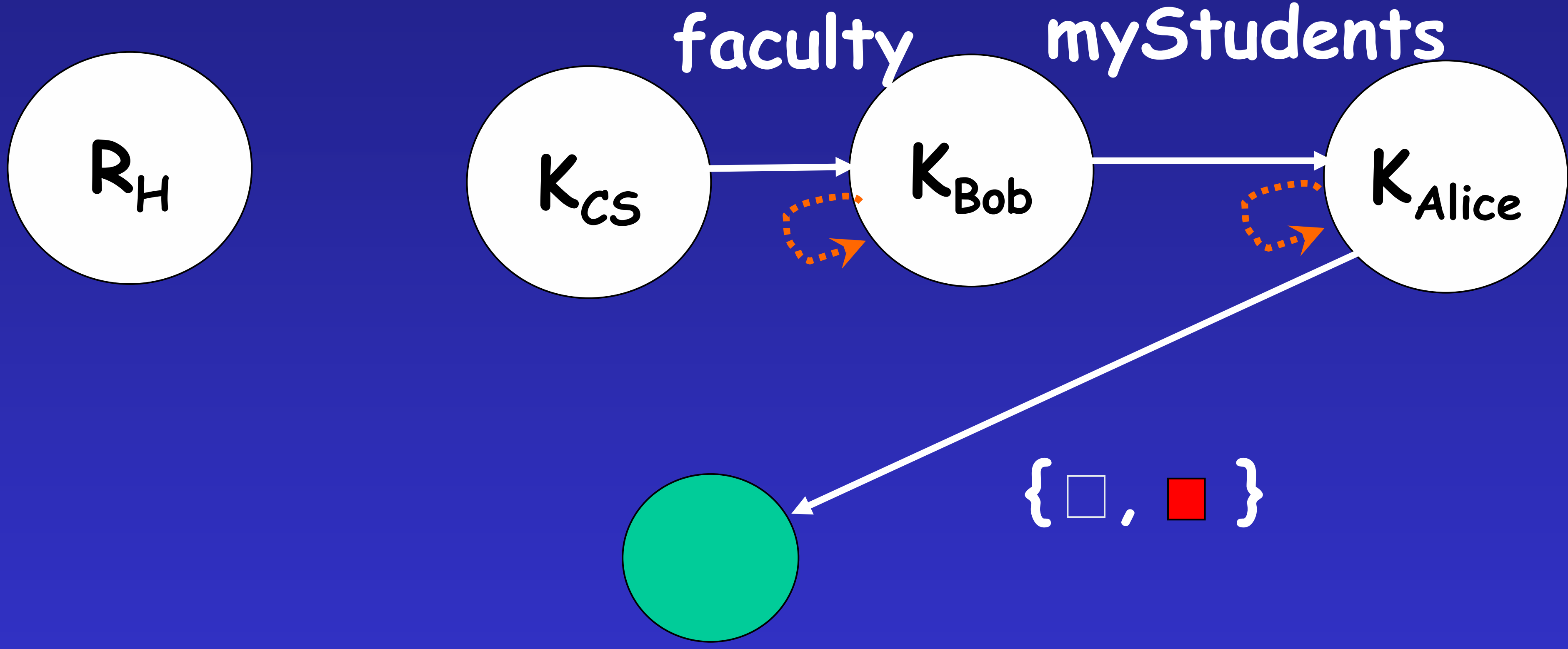


Update Rule

$\langle K, a \rangle \rightarrow \langle K_1, a_1 \; . \; . \; . \; a_m \rangle$



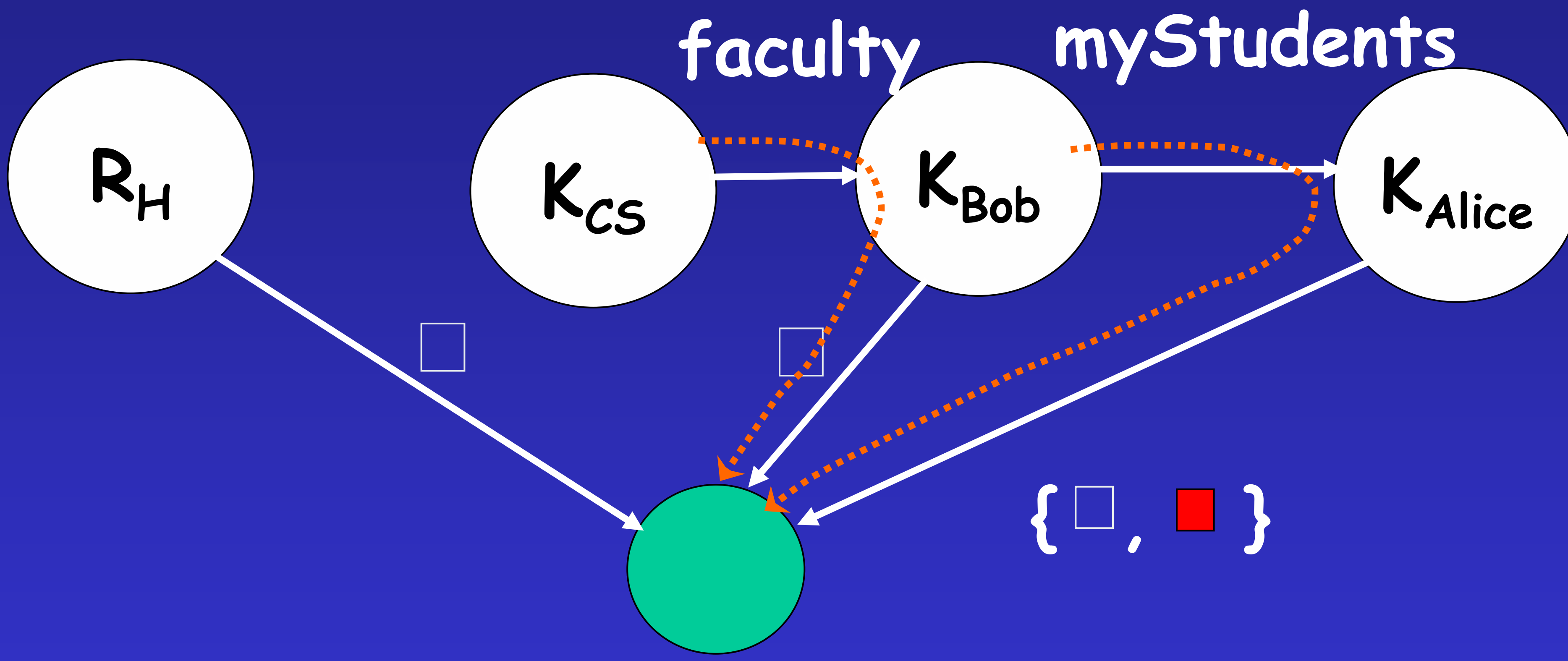
$Pre^*(\{\langle K_{Alice}, \square \rangle, \langle K_{Alice}, \blacksquare \rangle\})$



$\langle K_{Bob}, \text{myStudents} \rangle \rightarrow \langle K_{Alice}, \varepsilon \rangle$

$\langle K_{CS}, \text{faculty} \rangle \rightarrow \langle K_{Bob}, \varepsilon \rangle$

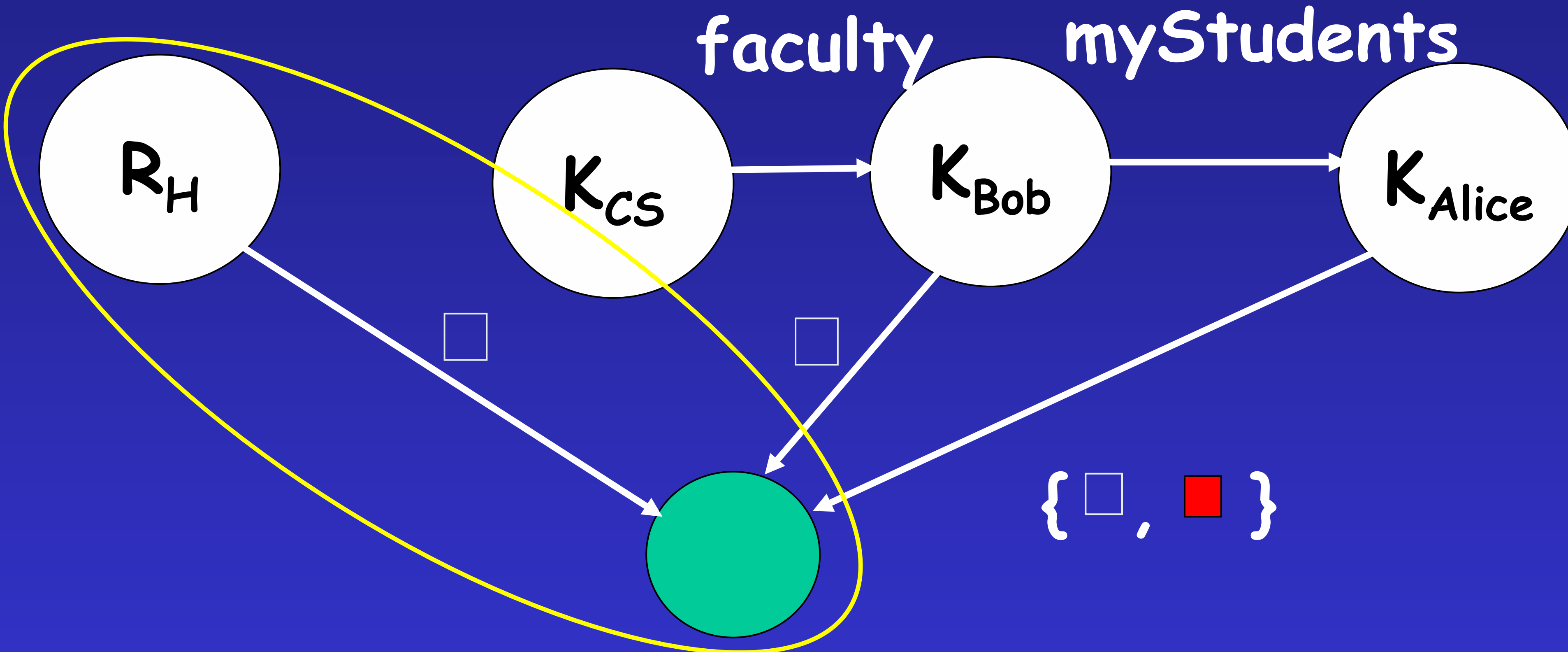
Pre*({$\langle K_{Alice}$, □$\rangle$, $\langle K_{Alice}$, ■$\rangle$})



$\langle K_{Bob}$, □$\rangle \rightarrow \langle K_{Bob}$, myStudents ■$\rangle$

$\langle R_H$, □$\rangle \rightarrow \langle K_{CS}$, faculty □$\rangle$

$Pre^*(\{\langle K_{Alice}, \square\rangle, \langle K_{Alice}, \blacksquare\rangle\})$



$\langle R_H, \square\rangle \in Pre^*(\{\langle K_{Alice}, \square\rangle, \langle K_{Alice}, \blacksquare\rangle\})$

Demo

```
# The lunch resource
  (k_lunch_resource <delegation>)

# AUTH CERT: Williamsburg Hosp. House lets conferences authorize lunch access
  k_lunch_resource  <delegation> --> k_whh  <conference delegation>

# CIPSW is a conference at Williamsburg Hospitality House
  k_whh  <conference>  --> k_cipsw  <>

# AUTH CERT: Conference organizers authorized to act on behalf of CIPSW
  k_cipsw  <delegation> --> k_cipsw  <organizer  delegation>

# The CIPSE organizers are . . .
  k_cipsw  <organizer>  --> k_wachter  <>
  k_cipsw  <organizer>  --> k_toth  <>

# AUTH CERT: Toth authorizes all attendees . . . but without delegation
  k_toth  <delegation>  -->  k_cipsw  <attendee  no_delegation>

# List all attendees here
  k_cipsw  <attendee>  -->  k_jha  <>
  k_cipsw  <attendee>  -->  k_reps  <>
  k_cipsw  <attendee>  -->  k_toth  <>
  k_cipsw  <attendee>  -->  k_wachter  <>
  k_cipsw  <attendee>  -->  k_clarke  <>
```

Demo

Time and Space Complexity

- n_K : number of principals
- $|C|$: sum of the lengths of the right-hand sides of the certs in C
- Pre*
  - Time complexity: $O(n_K^2 \, |C|)$
  - Space complexity: $O(n_K \, |C|)$
- Post*
  - Time and space complexity: $O(n_K^2 \, (n_K+|C|))$

Other Certificate Analysis Problems

- Authorized access 2
  - ‹R, □› is in pre* ({ c(N}) })
  - Where N = K A_1 … A_m is an extended name
  - c(N) is equal to ‹K, A_1 … A_m ›
  - Note: N need not be a key
- Expiration vulnerability 1
  - R1= pre*[C] ({‹K, □›,‹K, ■›})
  - R2 = pre*[C-C′] ({‹K,□›, ‹K, ■›})
  - { R | ‹R, □› is in R1 – R2 }

Related Work

- Certificate-chain discovery [Clarke et.al. 99]
  - Name-reduction closure
  - No mechanism to represent infinite sets of configurations
  - Only solves one certificate-analysis problem
- Our paper
  - Infinite sets of configurations represented by automata
  - PDS model checking solves many certificate-analysis problems

Related Work

- Certificate-chain discovery [Clarke et.al. 99]
  - Name-reduction closure
  - No mechanism to represent infinite sets of configurations
  - Only solves one certificate-analysis problem
- Semantics of SPKI/SDSI
  - [Abadi 98], [Howell & Kotz 00], [Halpern & Meyden 01]
- Our paper
  - PDS model checking solves many certificate-analysis problems
  - SPKI/SDSI semantics for free

Contributions

- Observed
  - SPKI/SDSI certs = PDS transition rules
  - SPKI/SDSI names = PDS configurations
- Harnessed theory of PDS model checking
  - PDS model checking solves many certificate-analysis problems
- SPKI/SDSI semantics for free

Topics

- Model checking of pushdown systems
- Context-sensitive dataflow analysis
- Authorization problems
- Authorization problems + privacy, recency, validity, and trust

Reachability
Reachability
+ a value

- Jha, S. and Reps, T., Analysis of SPKI/SDSI certificates using model checking. In *Proc. of the 15th IEEE Computer Security Foundations Workshop*, 2002
- Schwoon, S., Jha, S., Reps, T., and Stubblebine, S., On generalized authorization problems. Submitted to *16th IEEE Computer Security Foundations Workshop*, 2003.

Privacy using a Weighted PDS

$$\langle R_{\text{Insurance}}, \square \rangle \rightarrow \langle K_H, \text{patient}\ \blacksquare \rangle \qquad \text{I}$$

$$\langle K_H, \text{patient} \rangle \rightarrow \langle K_{\text{AIDS}}, \text{patient} \rangle \qquad \text{I}$$

$$\langle K_H, \text{patient} \rangle \rightarrow \langle K_{\text{IM}}, \text{patient} \rangle \qquad \text{I}$$

$$\langle K_{\text{AIDS}}, \text{patient} \rangle \rightarrow \langle K_{\text{Alice}}, \varepsilon \rangle \qquad \text{S}$$

$$\langle K_{\text{IM}}, \text{patient} \rangle \rightarrow \langle K_{\text{Alice}}, \varepsilon \rangle \qquad \text{I}$$

$$\begin{array}{c} \text{S} \\ | \\ \text{I} \end{array}$$

Privacy using a Weighted PDS

$\langle R_{\mathit{Insurance}}, \square \rangle$

$\Big\downarrow$ I $\quad \langle R_{\mathit{Insurance}}, \square \rangle \rightarrow \langle K_H, \mathit{patient}\ \blacksquare \rangle$

$\langle K_H, \mathit{patient}\ \blacksquare \rangle$

$\Big\downarrow$ I $\quad \langle K_H, \mathit{patient} \rangle \rightarrow \langle K_{\mathit{AIDS}}, \mathit{patient} \rangle$

$\langle K_{\mathit{AIDS}}, \mathit{patient}\ \blacksquare \rangle$

$\Big\downarrow$ S $\quad \langle K_{\mathit{AIDS}}, \mathit{patient} \rangle \rightarrow \langle K_{\mathit{Alice}}, \varepsilon \rangle$

$\langle K_{\mathit{Alice}}, \blacksquare \rangle$

Privacy using a Weighted PDS

$\langle R_{Insurance}, \square \rangle$

$\downarrow$ I

$\langle K_{H}, \text{patient} \ \blacksquare \rangle$

S

$\downarrow$ I

$\langle K_{AIDS}, \text{patient} \ \blacksquare \rangle$

$\downarrow$ S

$\langle K_{Alice}, \blacksquare \rangle$

$$I \otimes I \otimes S = S$$

Privacy using a Weighted PDS

$\langle R_{\text{Insurance}}, \square \rangle$

$\Downarrow$ I $\quad \langle R_{\text{Insurance}}, \square \rangle \rightarrow \langle K_H, \text{patient}\ \blacksquare \rangle$

$\langle K_H, \text{patient}\ \blacksquare \rangle$

$\Downarrow$ I $\quad \langle K_H, \text{patient} \rangle \rightarrow \langle K_{IM}, \text{patient} \rangle$

$\langle K_{IM}, \text{patient}\ \blacksquare \rangle$

$\Downarrow$ I $\quad \langle K_{IM}, \text{patient} \rangle \rightarrow \langle K_{\text{Alice}}, \varepsilon \rangle$

$\langle K_{\text{Alice}}, \blacksquare \rangle$

Privacy using a Weighted PDS

$\langle R_{Insurance}, \square \rangle$

I

$\langle K_H, \text{patient } \blacksquare \rangle$

I I

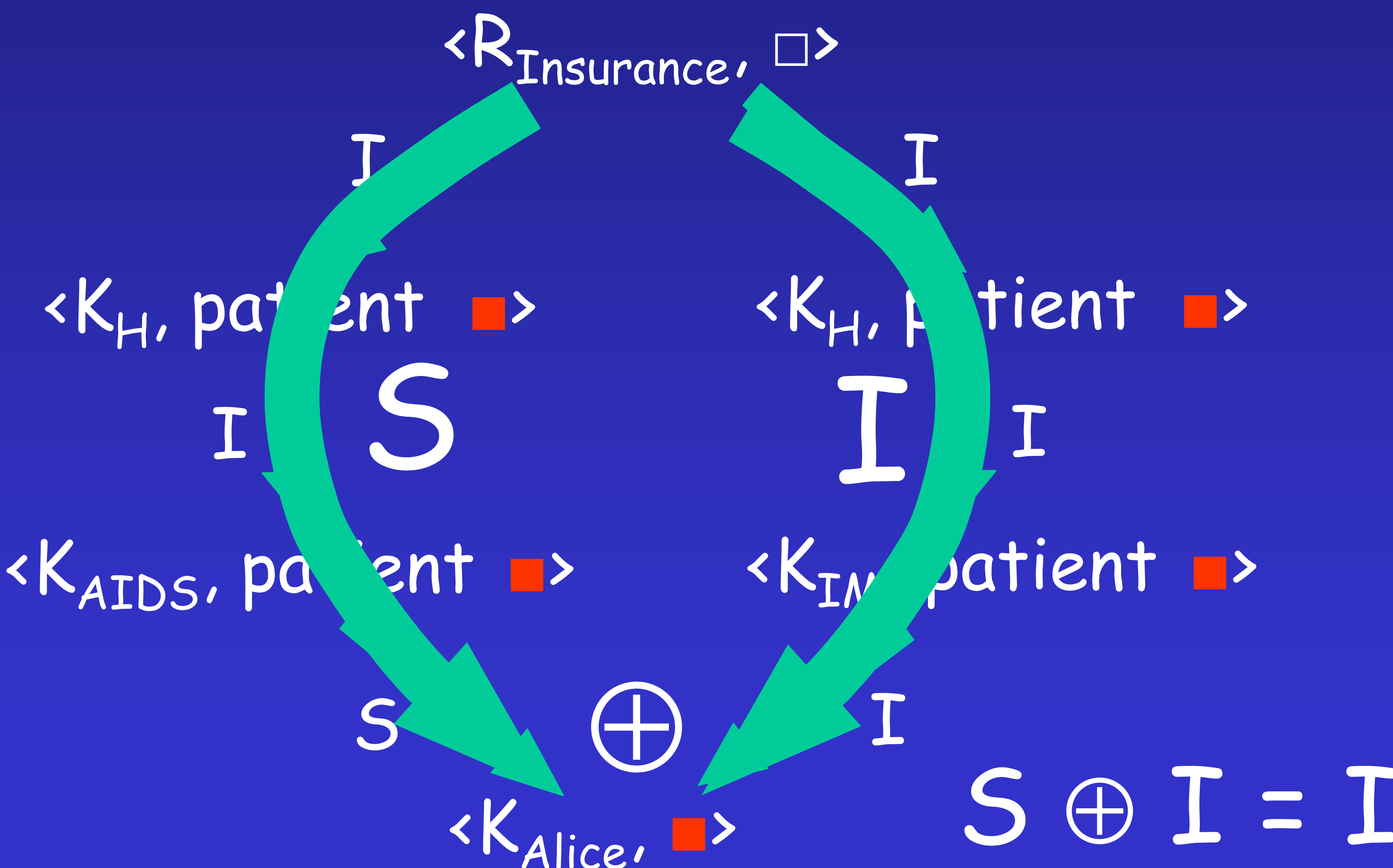
$\langle K_{IM}, \text{patient } \blacksquare \rangle$

I

$\langle K_{Alice}, \blacksquare \rangle$

$$\mathrm{I} \otimes \mathrm{I} \otimes \mathrm{I} = \mathrm{I}$$

Privacy using a Weighted PDS



More Expressive Memory-Safety Policies

Opportunity: “Checking System Rules” [Engler]

v.unknown:

[(v = malloc(_)) == 0] $\rightarrow_t$ v.null

$\rightarrow_f$ v.notNull

[v = malloc(_)] $\rightarrow$ v.unknown

v.unknown, v.null, v.notNull:

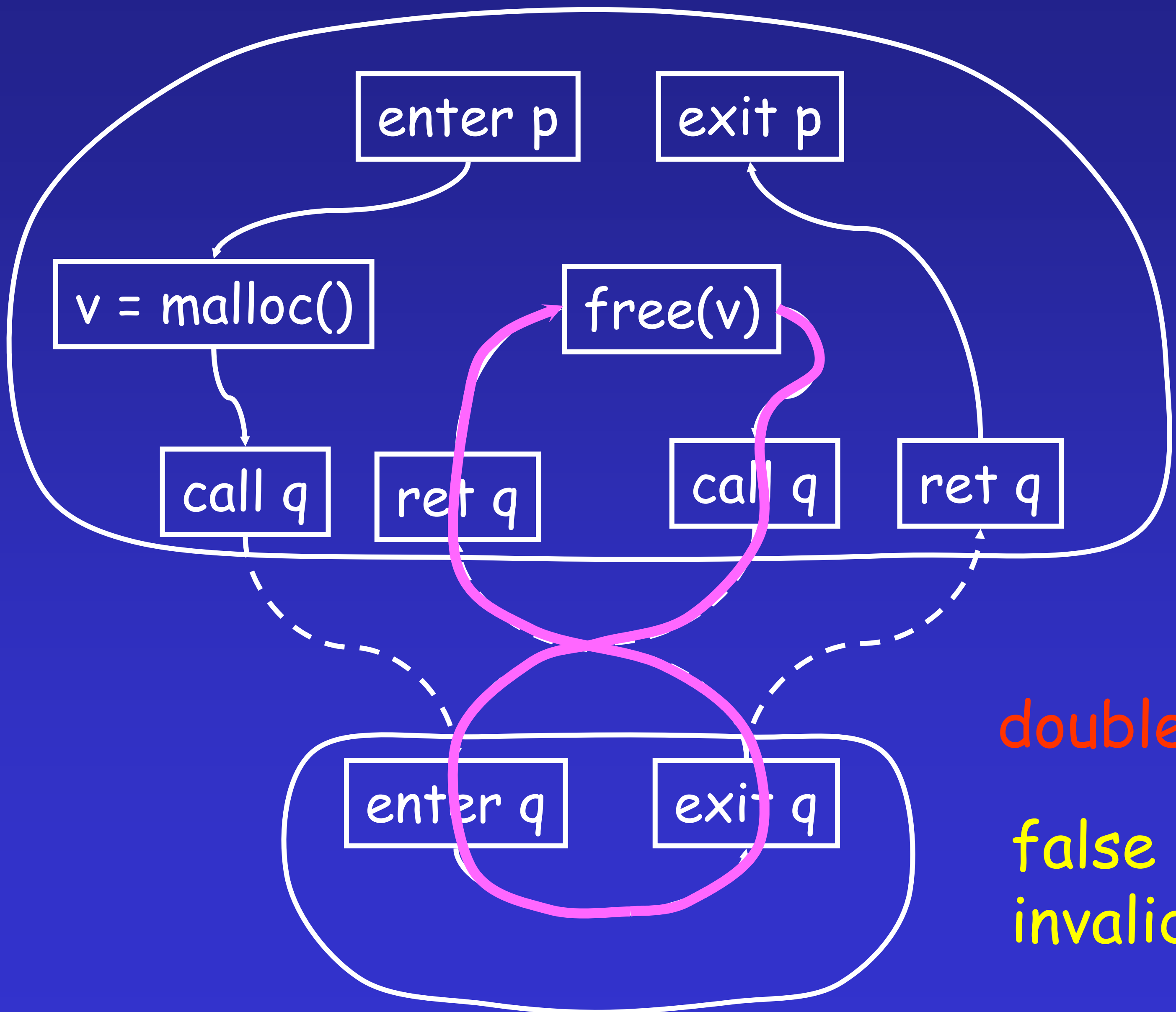
[free(v)] $\rightarrow$ v.freed

v.freed:

[free(v)] $\rightarrow$ “double free!”

[v] $\rightarrow$ “use after free!”

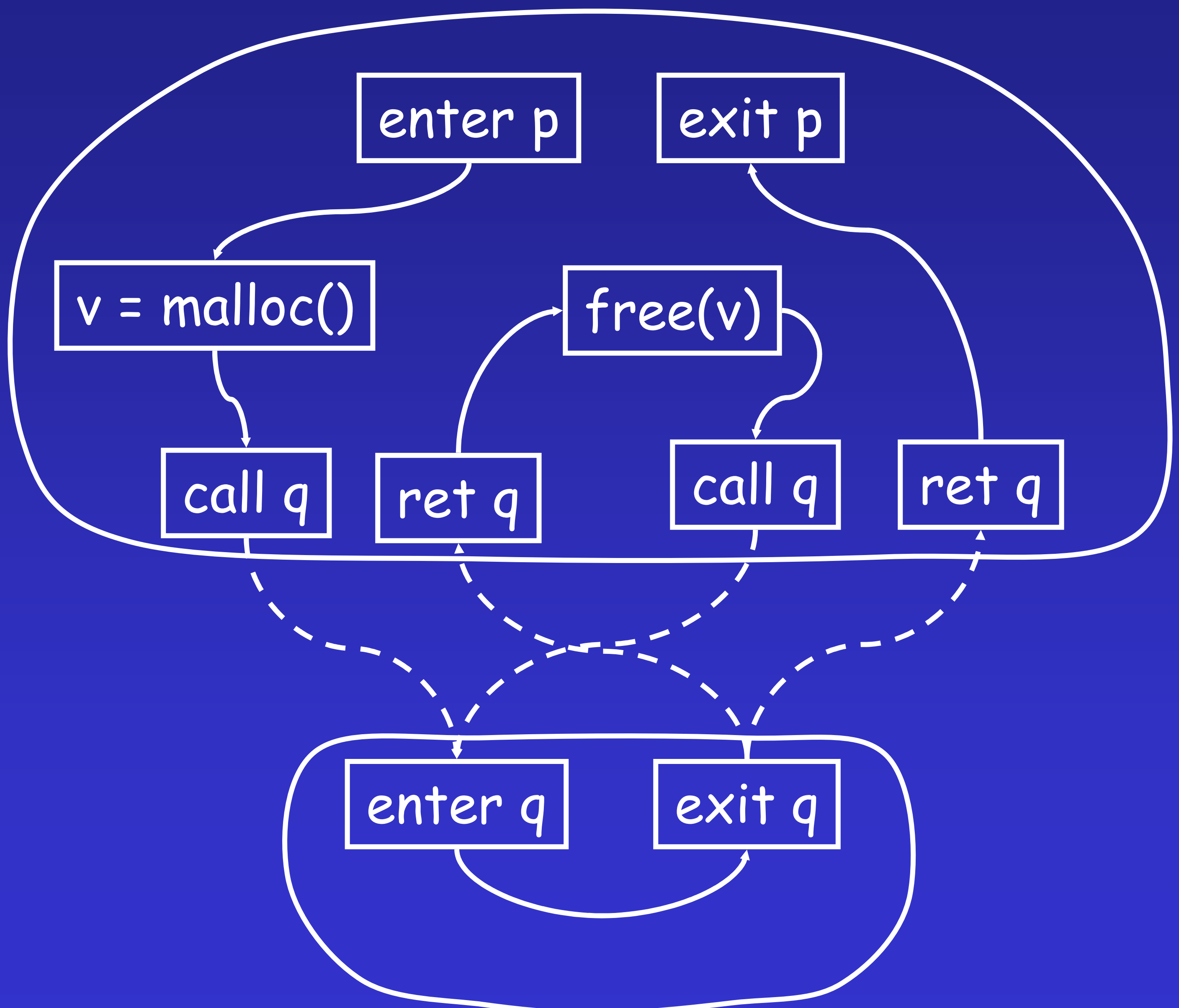
The Need for Context Sensitivity



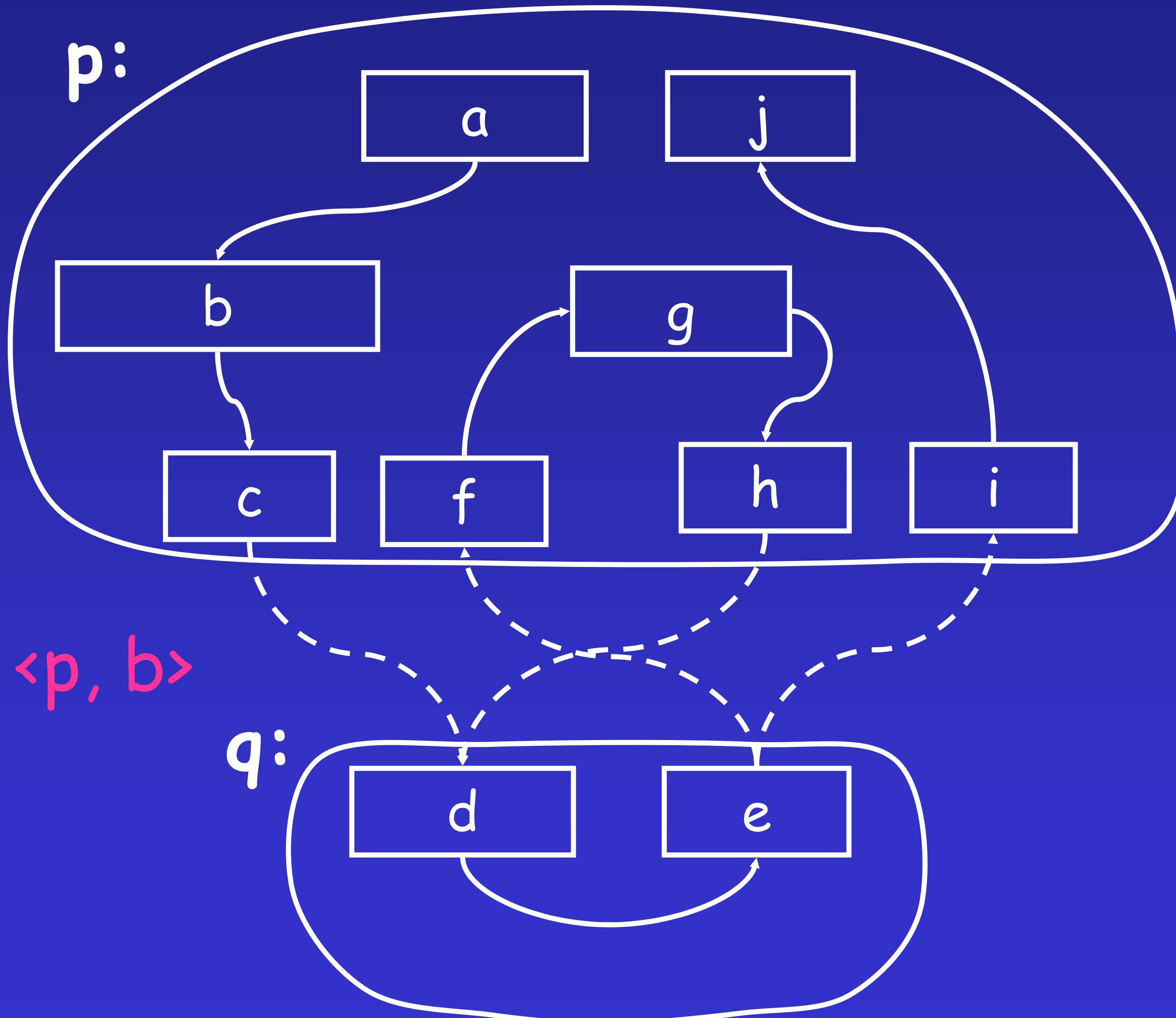
double free!

false alarm:
invalid path!

Hierarchical Graph = PDS

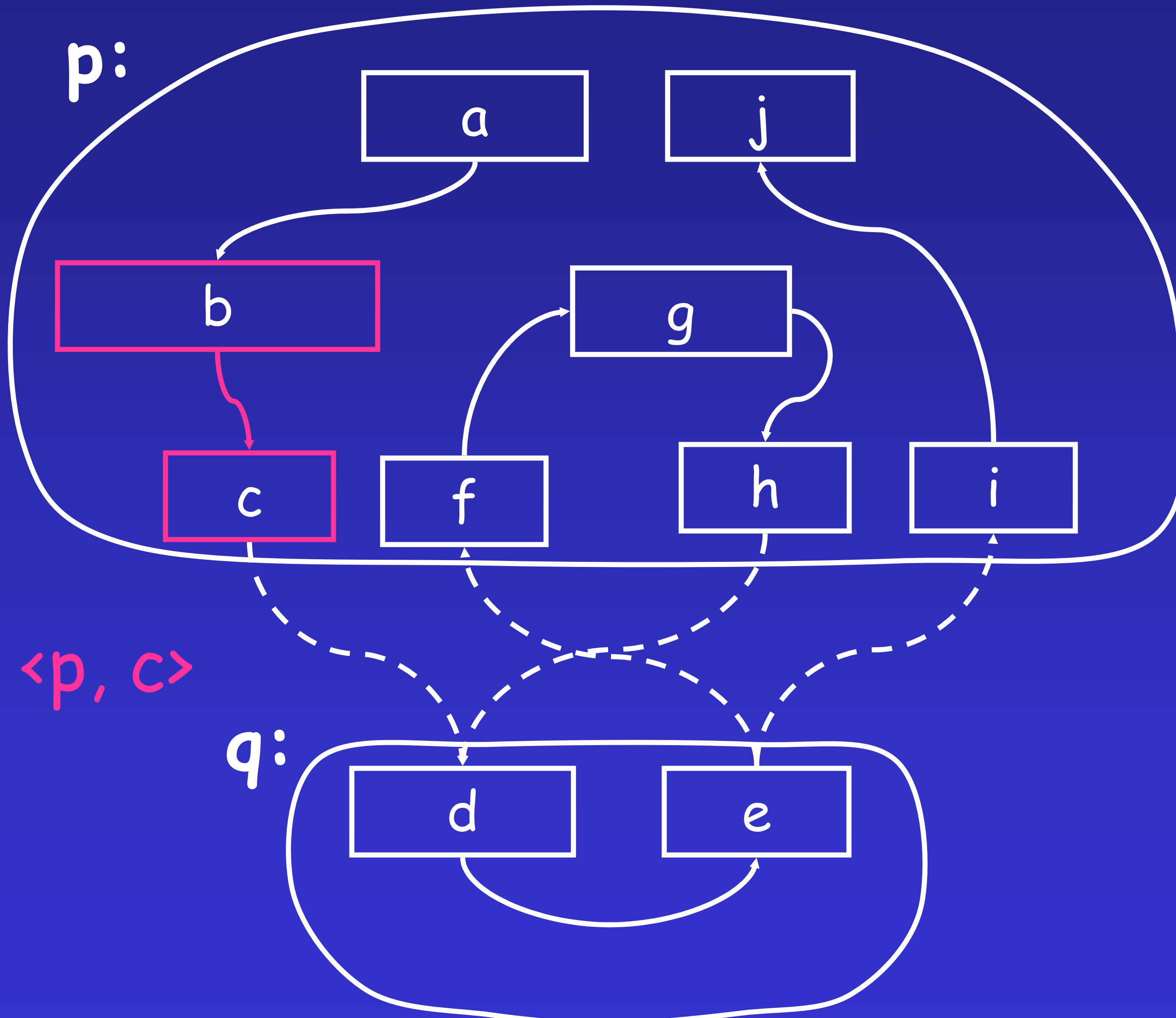


Hierarchical Graph = PDS



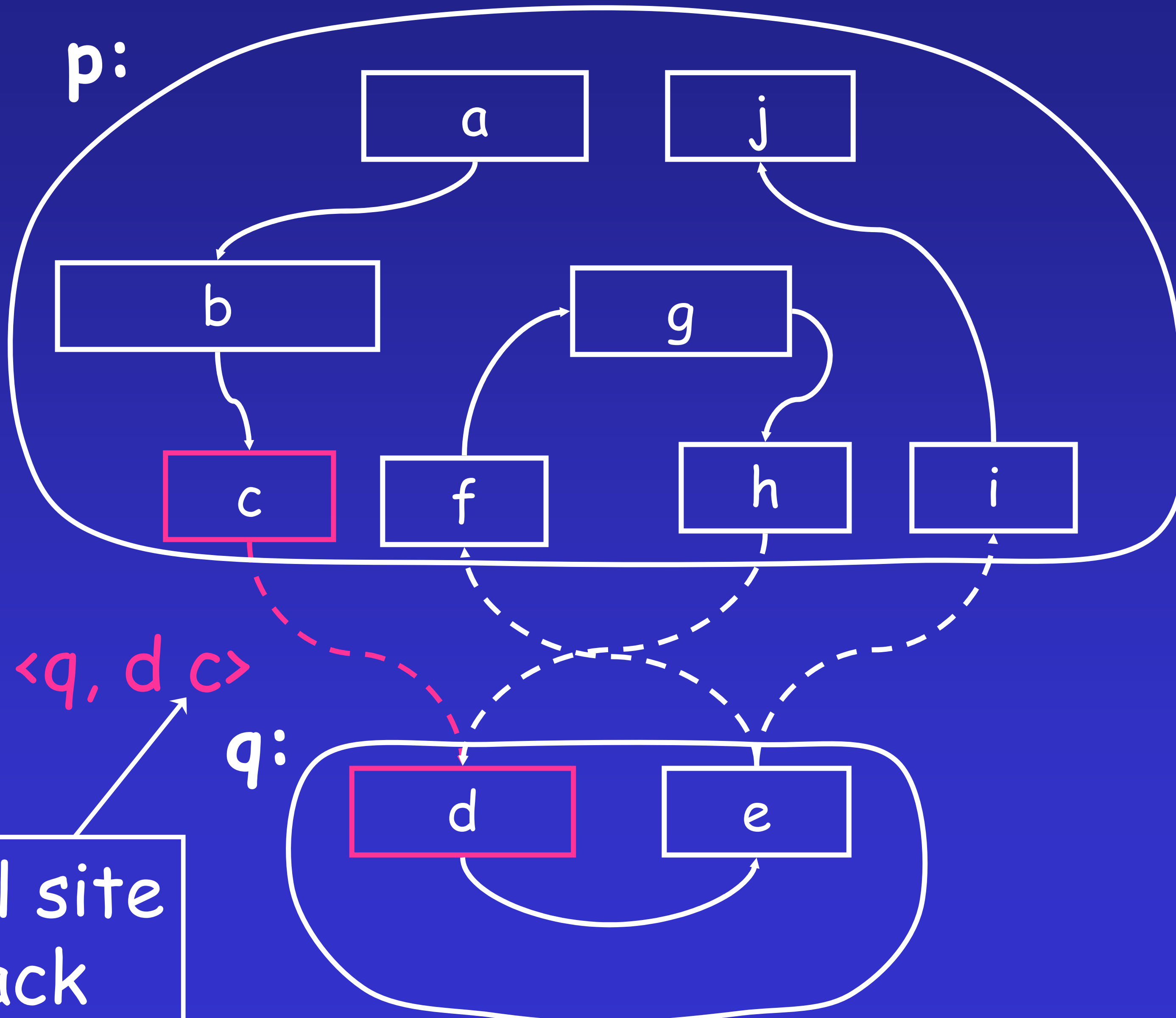
$\langle p, a \rangle \rightarrow \langle p, b \rangle$

Hierarchical Graph = PDS



$\langle p, b\rangle \rightarrow \langle p, c\rangle$

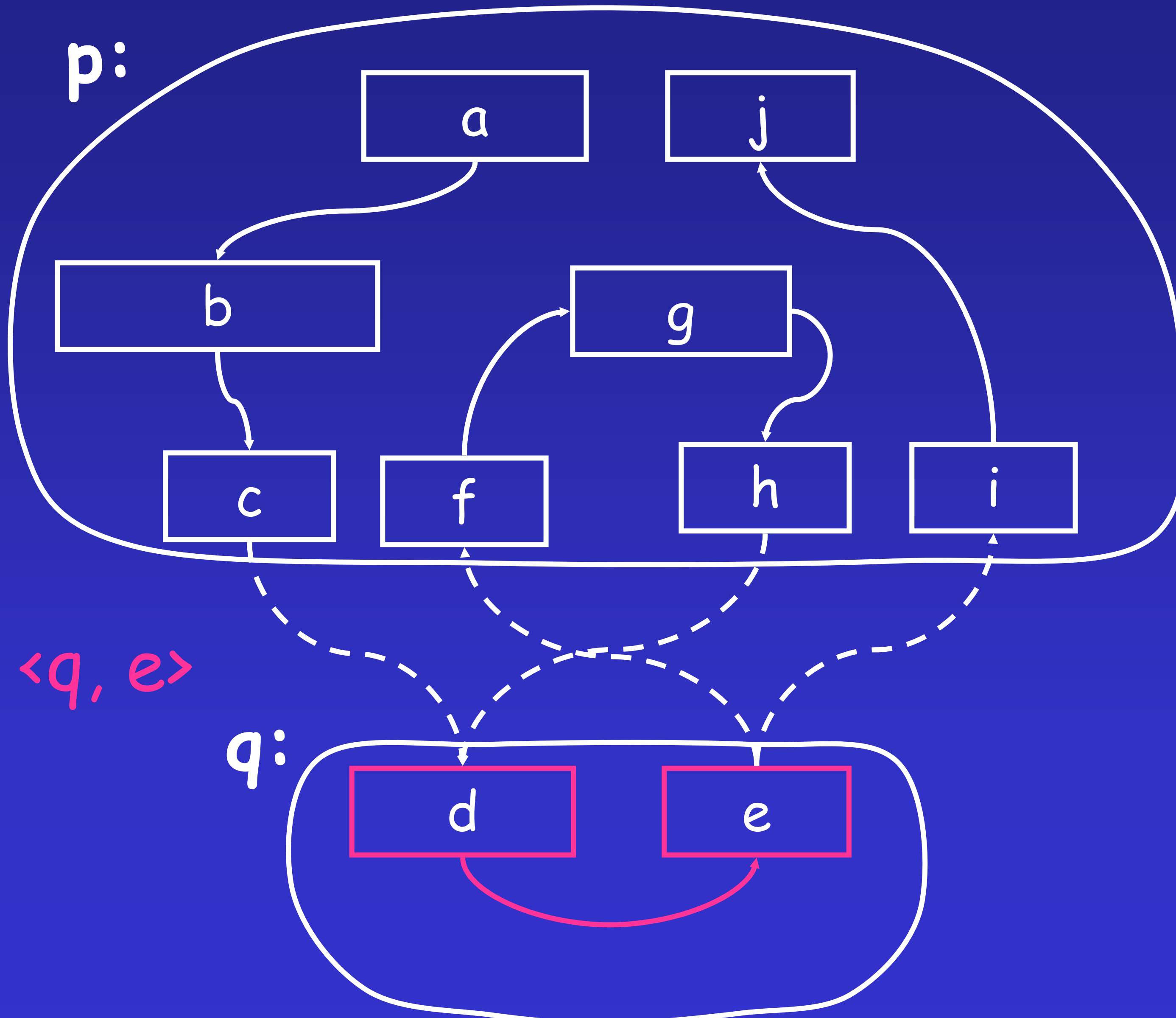
Hierarchical Graph = PDS



<p, c> → <q, d c>

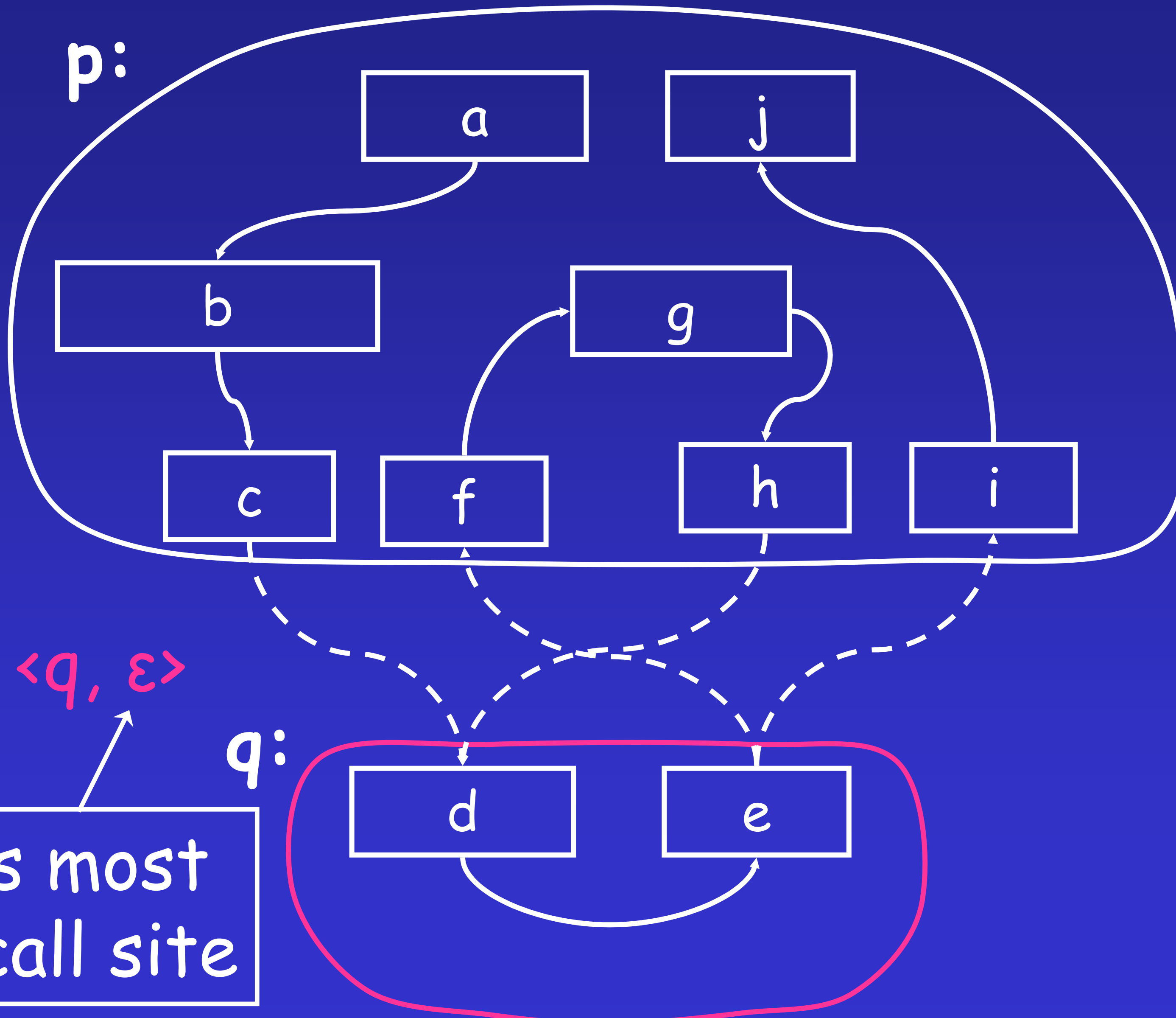
save call site on stack

Hierarchical Graph = PDS



$\langle q, d\rangle \rightarrow \langle q, e\rangle$

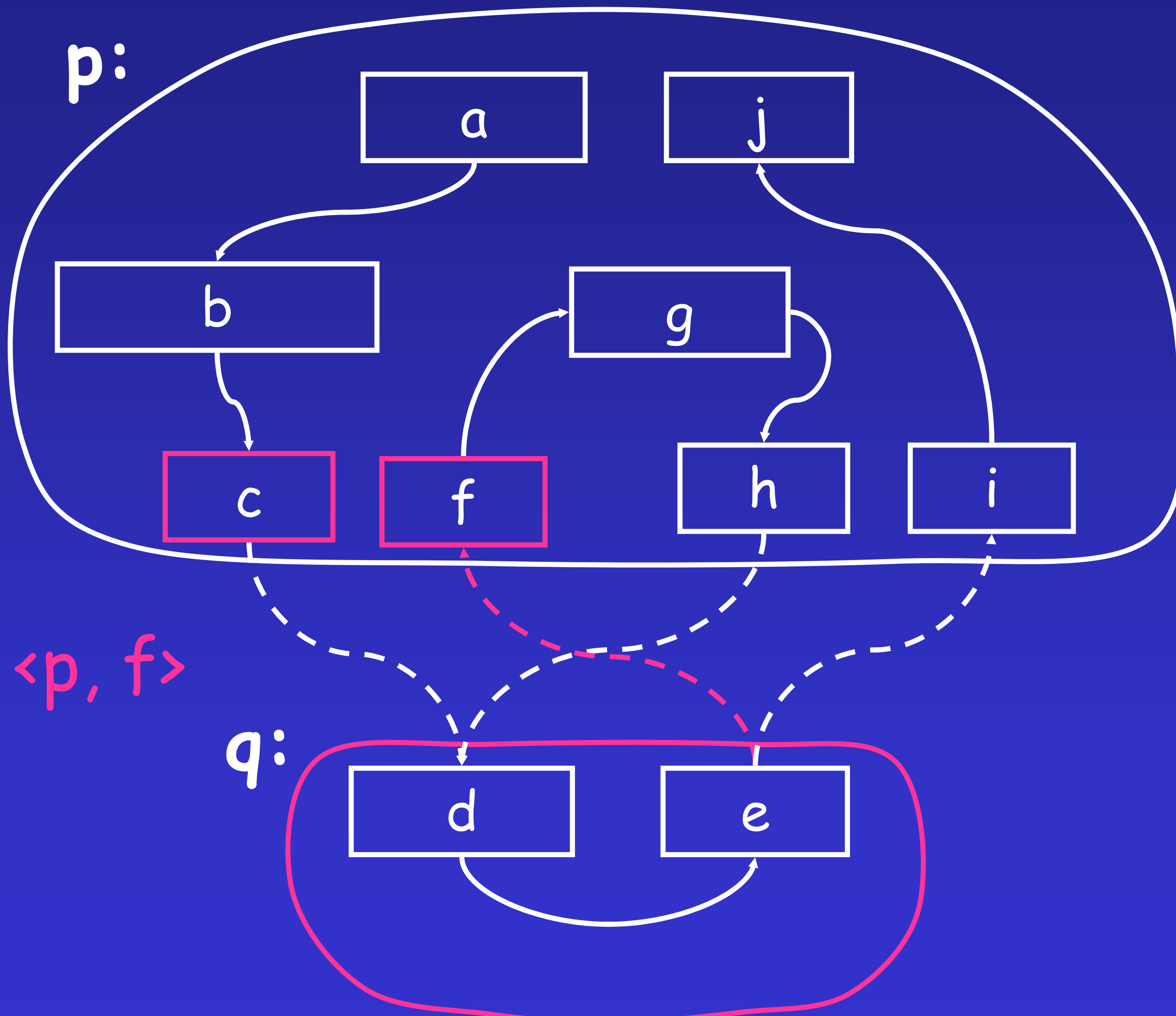
Hierarchical Graph = PDS



$\langle q, e\rangle \to \langle q, \varepsilon\rangle$

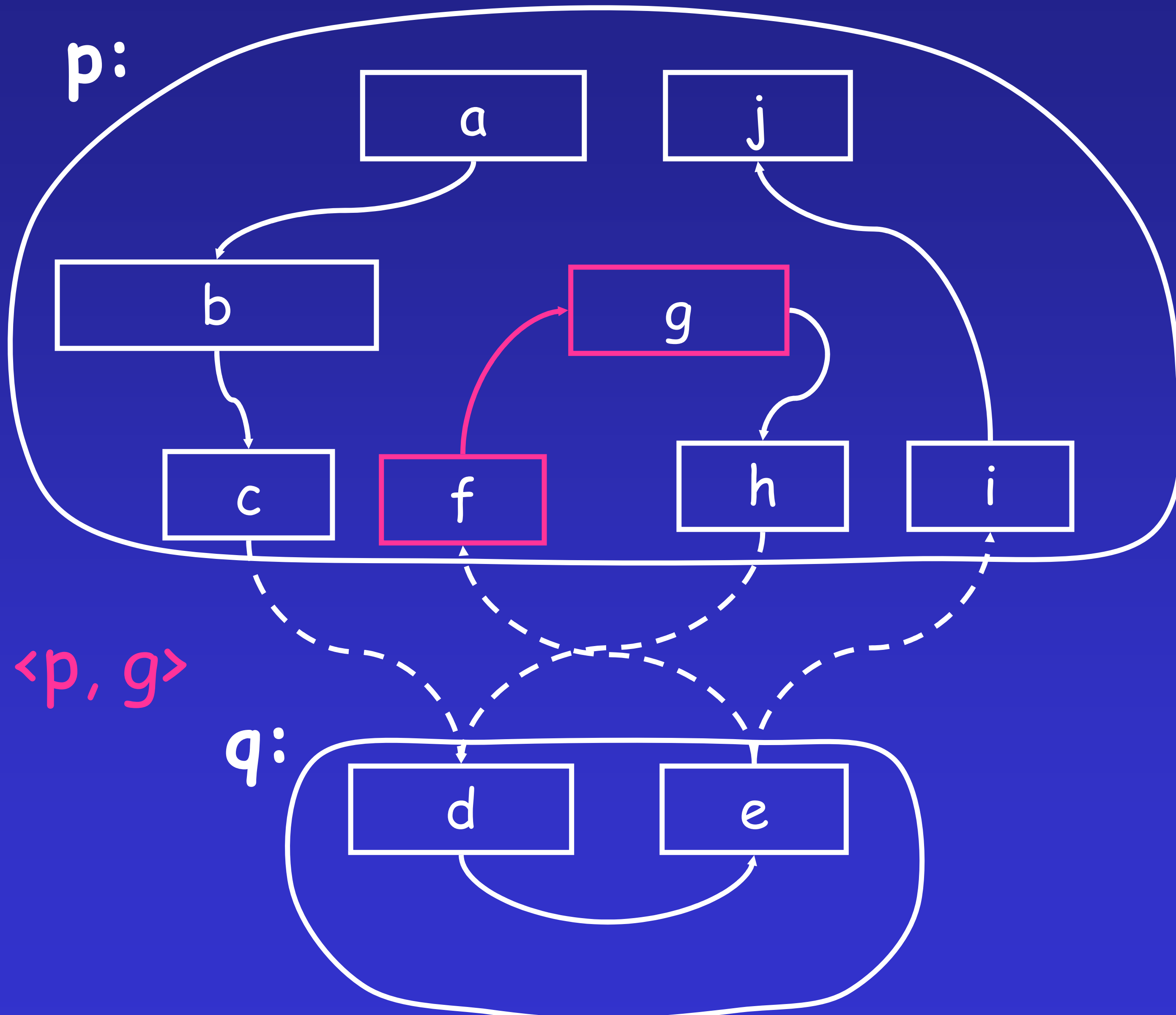
uncovers most recent call site

Hierarchical Graph = PDS



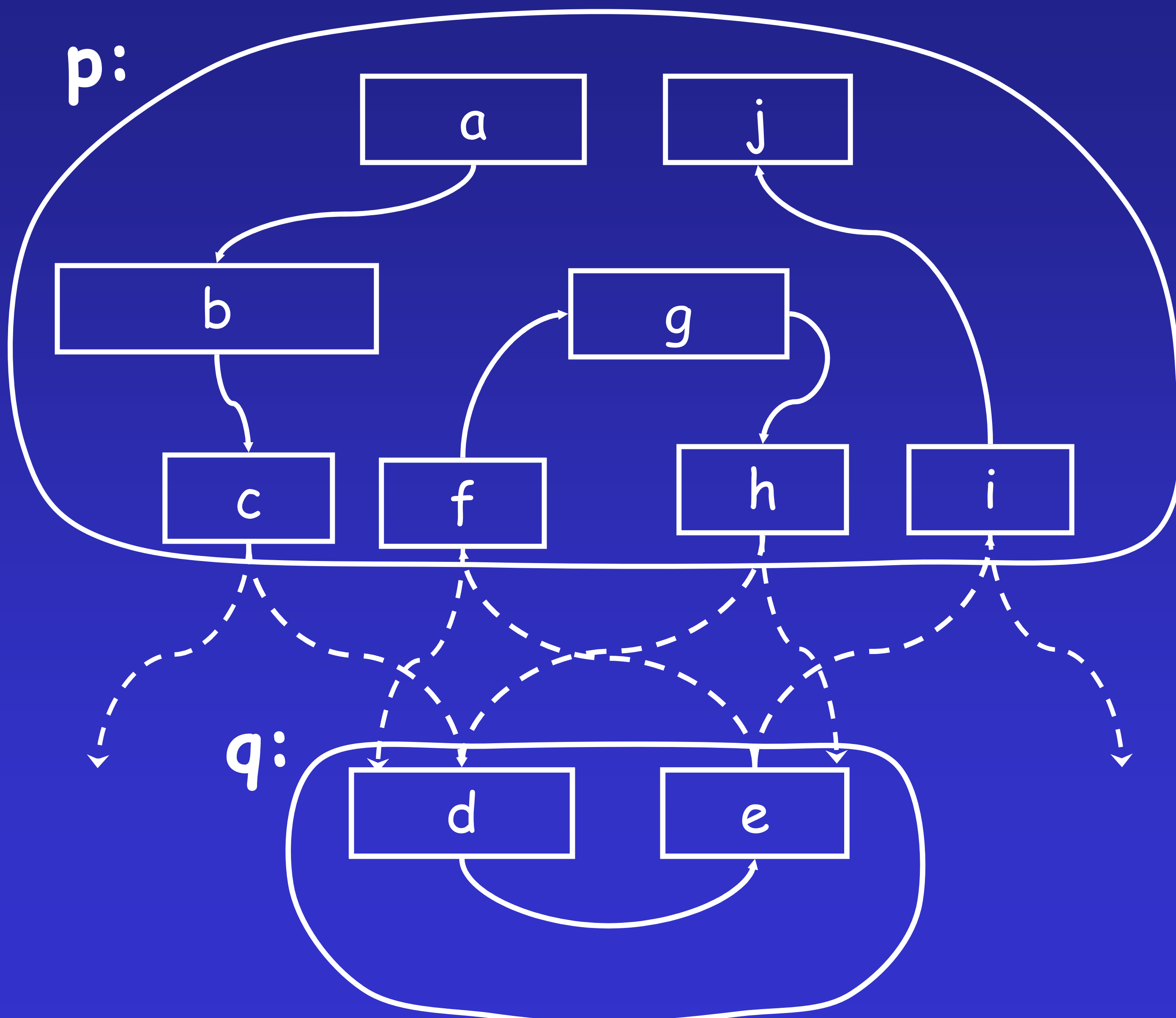
$\langle q, c\rangle \rightarrow \langle p, f\rangle$

Hierarchical Graph = PDS

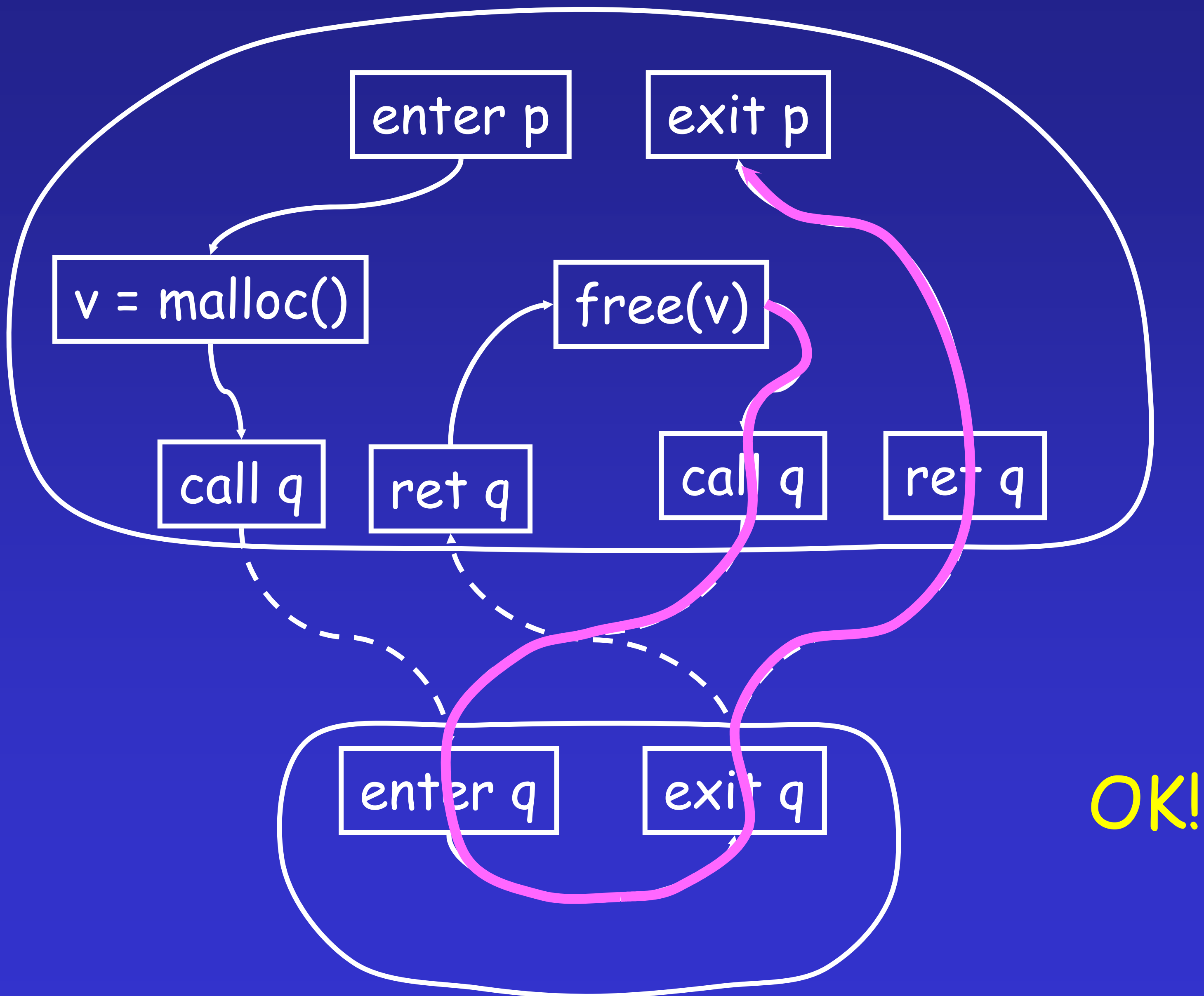


<p, f> → <p, g>

Transition System = Unrolled Program



The Need for Context Sensitivity



Hierarchical Graph + Weights = Context-Sensitive Dataflow Analysis

```
int x;

void main() {
  x = 5;
  p();    // main_calls_p
  return;
}
```

```
void p() {
  if (...) {
    x = x + 1;
    p();         // p_calls_p1
    x = x - 1;
  }
  else if (...) {
    x = x - 1;
    p();         // p_calls_p2
    x = x + 1;
  }
  return;
}
```

Hierarchical Graph + Weights = Context-Sensitive Dataflow Analysis

```
int x;

void main() {
  n1:  x = 5;
  n2: p(); //main_calls_p
  return;
}
```

```
void p() {
  n4: if (...) {
    n5: x = x + 1;
    n6: p();    // p_calls_p1
    n8: x = x - 1;
  }
  n9: else if (...) {
    n10: x = x - 1;
    n11:  p();  // p_calls_p2
    n13: x = x + 1;
  }
  n14: return;
}
```

Demo

<x@exit, p p_calls_p2 p_calls_p1 main_calls_p>

Topics

- Model checking of pushdown systems
- Context-sensitive dataflow analysis
- Authorization problems
- Authorization problems + privacy, recency, validity, and trust

Reachability
Reachability
+ a value

- Jha, S. and Reps, T., Analysis of SPKI/SDSI certificates using model checking. In *Proc. of the 15th IEEE Computer Security Foundations Workshop*, 2002

- Schwoon, S., Jha, S., Reps, T., and Stubblebine, S., On generalized authorization problems. Submitted to *16th IEEE Computer Security Foundations Workshop*, 2003.