# Enforcement of Opacity Security Properties Using Insertion Functions

Yi-Chin Wu and Stéphane Lafortune

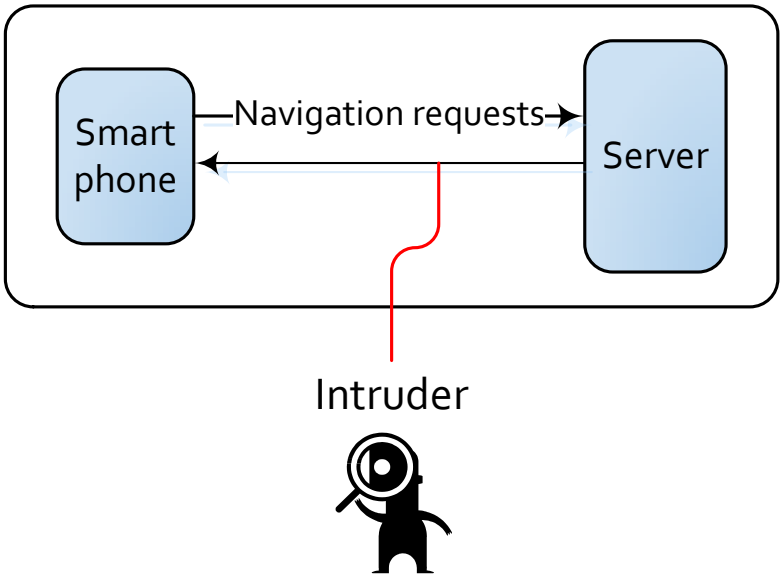
EECS Department, University of Michigan

CDC 12'

Dec. 13th, 2012

# Motivation

- Security and privacy concerns in online services
- "Opacity": whether the secret information of the system can be inferred by outside observers
- Example: Location-based services

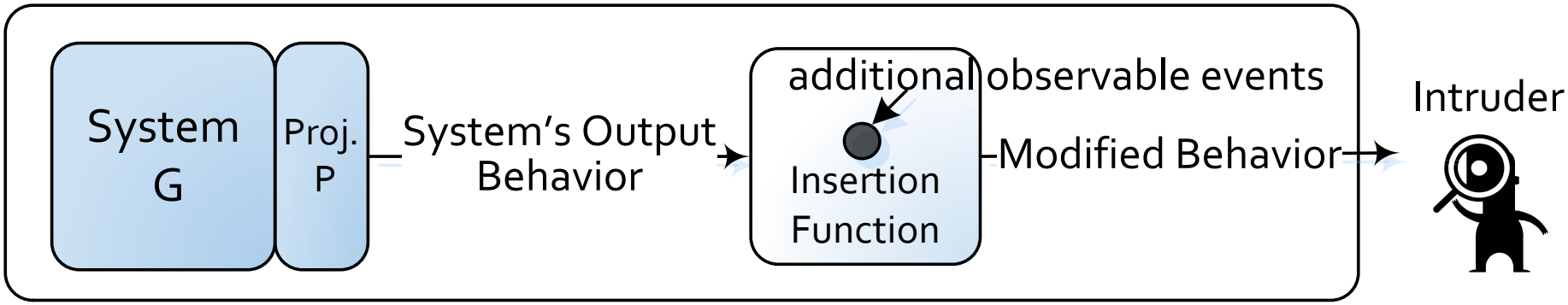


# Related Work

- Notions of opacity
  - [Mazaré et al 04 ], [Bryans et al. 05], [Saboori et al. 07], [Bryans et al. 08]
- Verification of opacity
  - [Saboori et al. 08, 09], [Cassez et al. 09], [Lin 11]
- Enforcement of opacity
  - [Dubreil et al. 08], [Saboori et al. 08], [Cassez et al. 09]

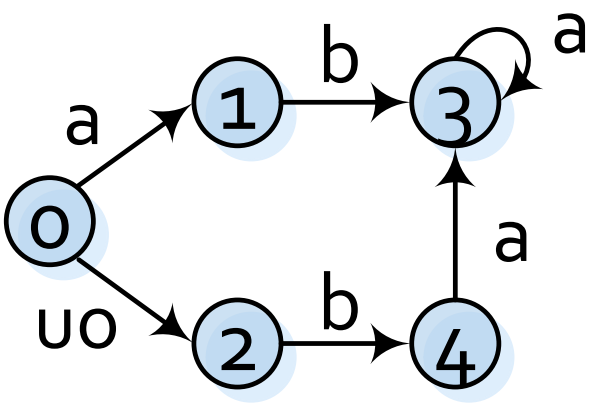
# Contribution

- Enforce opacity using insertion functions




  - "i-enforcing" property
  - "All Insertion Structure" (AIS)
  - Synthesis of i-enforcing insertion functions

# Automaton Model




$E_O=\{a,b\}$

- $X_0$
- $L(G,X_0) := \{s\in E^*: (\exists i\in X_0)\ [f(i,s) \text{ is defined}]\}$
- $E = E_o \cup E_{uo}$
- $P(e)=e$ if $e\in E_o$; $P(e)=\varepsilon$ if $e\in E_{uo}\cup\{\varepsilon\}$

# What Is The Opacity Problem?

- The system is partially observable
- The system has a secret
  - initial state, current state, sublanguage, initial-and-final state
- The intruder knows the system structure

# What Is The Opacity Problem?

- The system is partially observable
- The system has a secret
  - initial state, current state, sublanguage, initial-and-final state
- The intruder knows the system structure

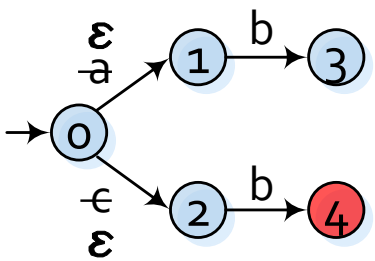
The secret is opaque if for every secret behavior, there is a nonsecret behavior that is observationally-equivalent

# Current State Opacity

## Definition (Current-State Opacity)

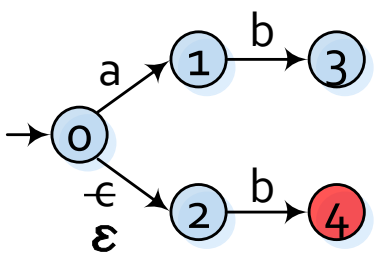
Given $G = (X,E,f,X_0)$, a set of secret states $X_S$, and a set of non-secret states $X_{NS}$, the automaton is current-state opaque if $\forall i \in X_0$, $\forall t \in L(G,i)$ such that $f(i,t) \in X_S$, $\exists j \in X_0$, $\exists t' \in L(G,j)$ such that (i) $f(j,t') \in X_{NS}$, (ii) $P(t)=P(t')$.

The system is opaque




$E_O=\{b\}$

The system is not opaque

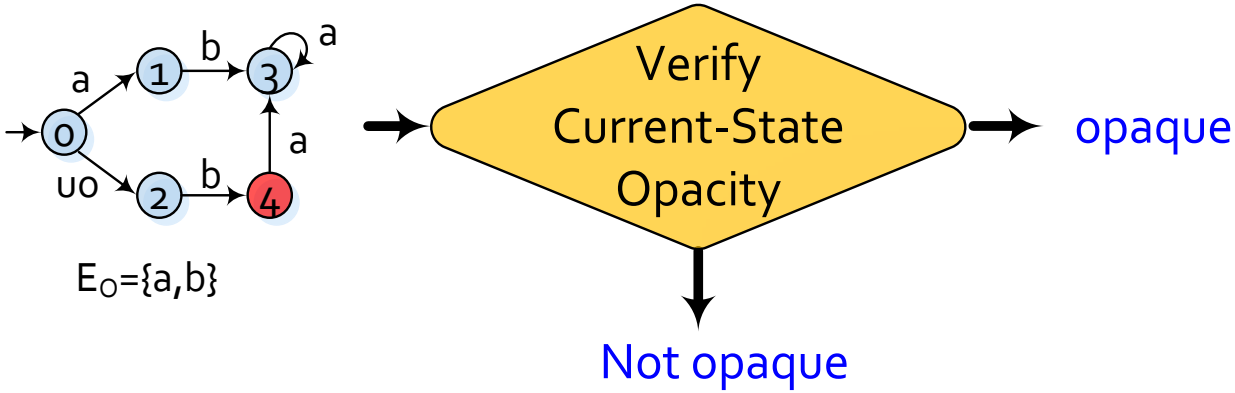



$E_O=\{a,b\}$

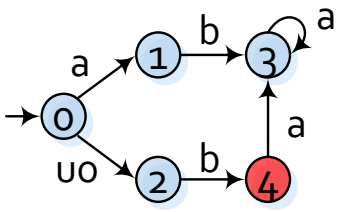
Secret states

Nonsecret states

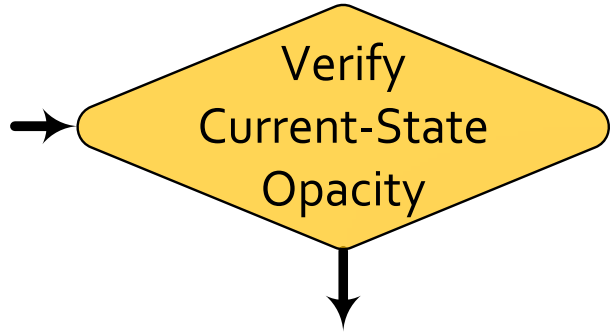
System G:


a
b
a
a
uo
b
a
0
1
2
3
4
$E_O$={a,b}
Verify
Current-State
Opacity
opaque
Not opaque

System G:

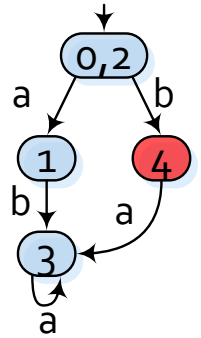



$E_O$={a,b}

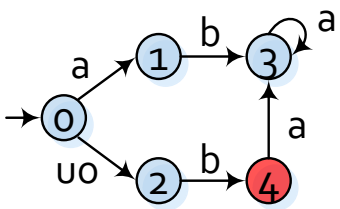



Not opaque

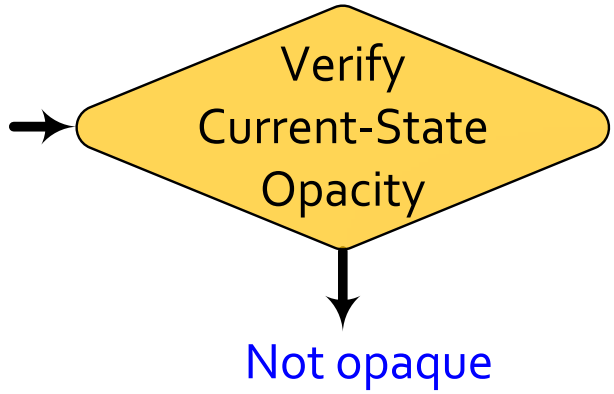
Current-State Estimator:



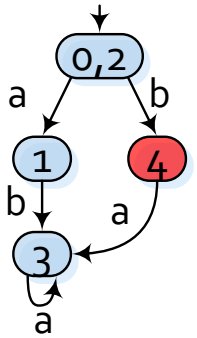
System G:




$E_O$={a,b}




Current-State Estimator:





# Opacity Enforcement Problem

How can we enforce the secret to be opaque?

# Existing Opacity Enforcement Mechanisms

- Supervisory control
  - Only partial system behavior is allowed
  - [Dubreil et al. 10][Saboori et al. 12]
- Dynamic observer
  - Create new observable behavior
  - [Cassez et al. 09]

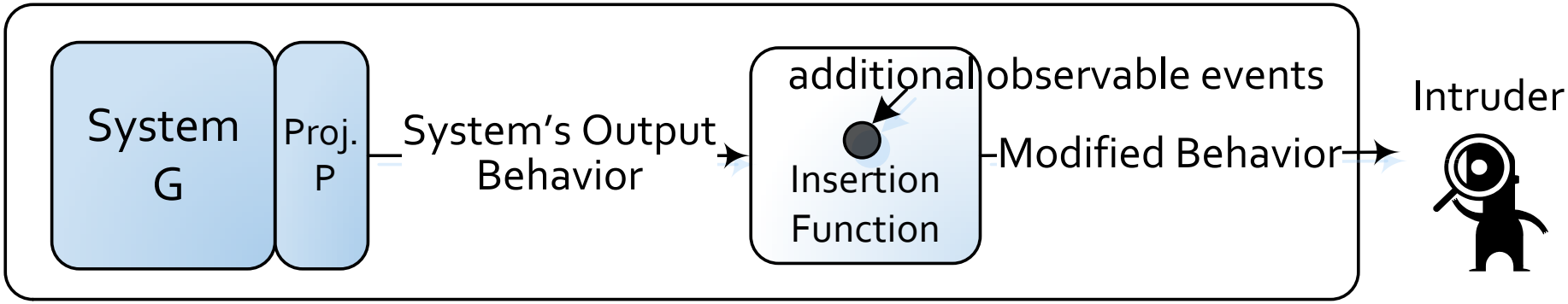
# Existing Opacity Enforcement Mechanisms

- Supervisory control
  - Only partial system behavior is allowed
  - [Dubreil et al. 10][Saboori et al. 12]
- Dynamic observer
  - Create new observable behavior
  - [Cassez et al. 09]

We enforce opacity such that

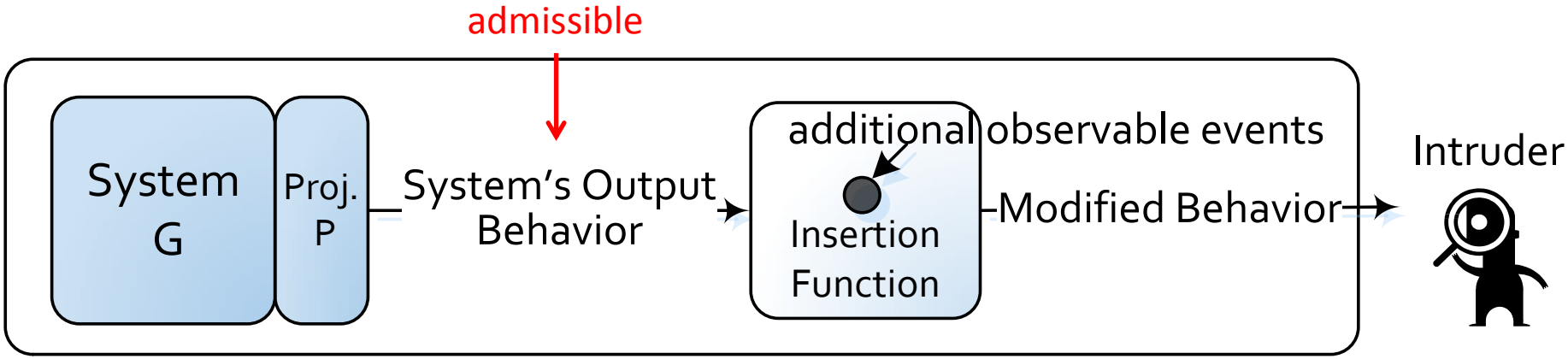
- All system behavior is allowed to occur
- No new observable behavior is created

# Our Approach: Insertion Functions



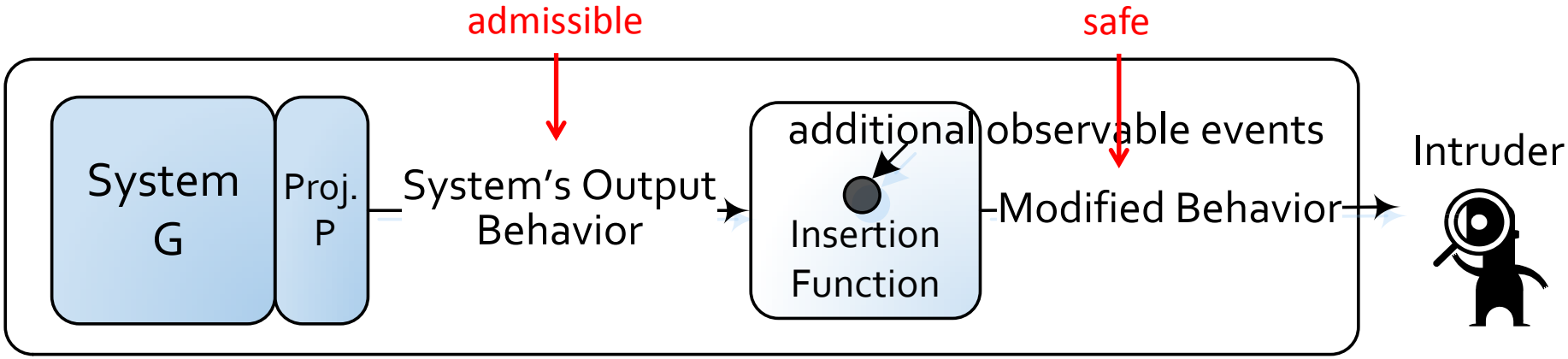

- A monitoring interface
- Insert extra observable events
  - Intruder cannot distinguish between inserted events and system's observable events

# I-Enforcing Property for Insertion Functions



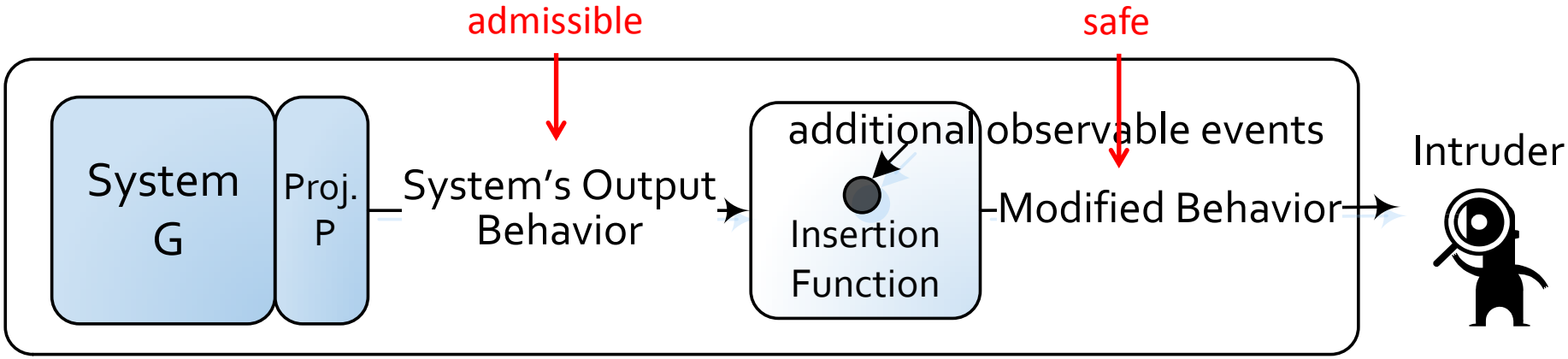

- Admissible: allows all system's output behavior

# I-Enforcing Property for Insertion Functions




- Admissible: allows all system's output behavior
- Safe: behavior after insertion must look like existing non-secret strings

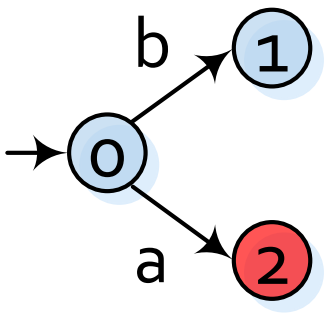
# I-Enforcing Property for Insertion Functions




- Admissible: allows all system's output behavior
- Safe: behavior after insertion must look like existing non-secret strings

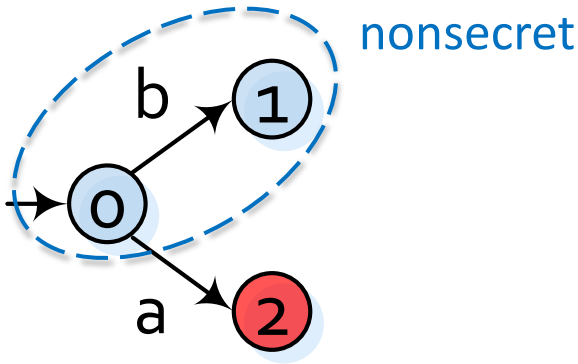
- i-enforcing = admissible + safe
- i-enforceable opacity property

# Is Opacity Always i-enforceable?



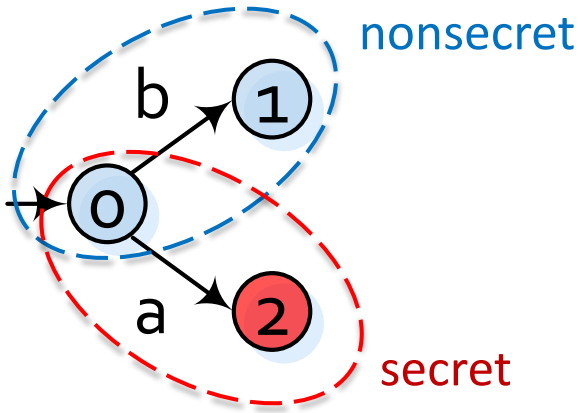

$E_O=\{a,b\}$

# Is Opacity Always i-enforceable?



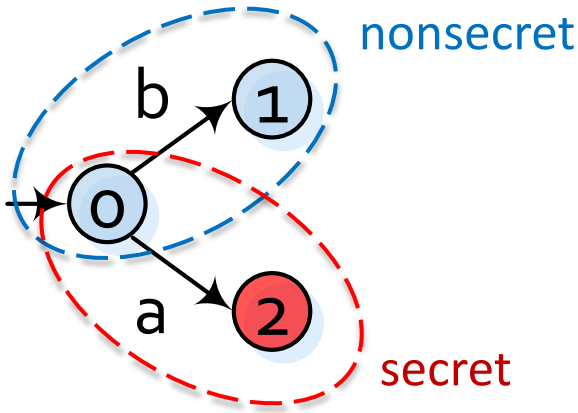

$E_O=\{a,b\}$

# Is Opacity Always i-enforceable?




$E_O=\{a,b\}$

# Is Opacity Always i-enforceable?

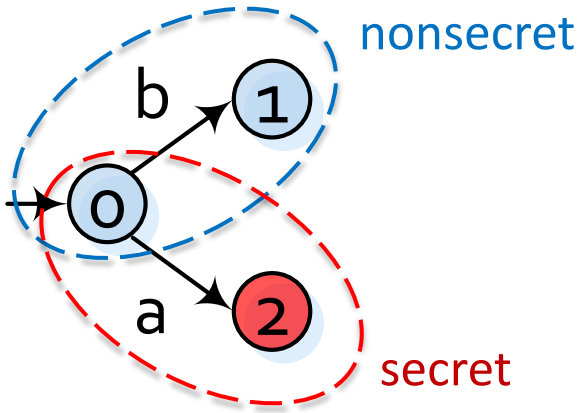



$E_O=\{a,b\}$

**I-enforceability Verification Problem**

Is opacity i-enforceable?

# Is Opacity Always i-enforceable?




$E_O=\{a,b\}$

**I-enforceability Verification Problem**

Is opacity i-enforceable?

**Insertion Function Synthesis Problem**

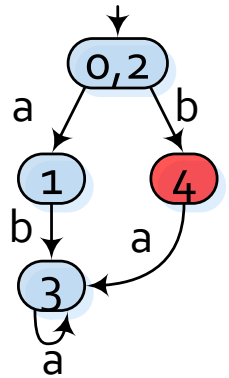
How to synthesize an i-enforcing insertion function?

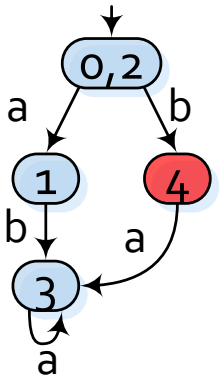
# The All-Insertion Structure (AIS)

## Enumerate all i-enforcing insertion functions

### [Step 1] i-Verifier

Current-State Estimator:



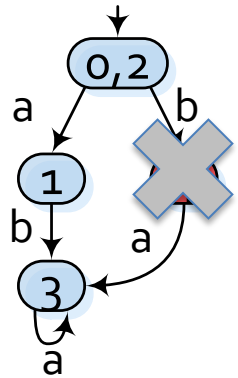



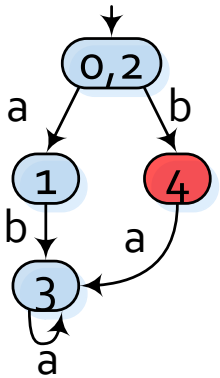
# The All-Insertion Structure (AIS)

## Enumerate all i-enforcing insertion functions

### [Step 1] i-Verifier

Current-State Estimator:



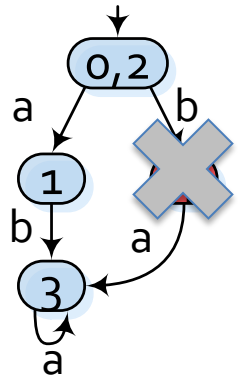



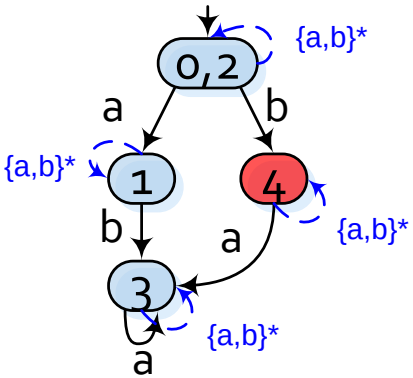
# The All-Insertion Structure (AIS)

## Enumerate all i-enforcing insertion functions

### [Step 1] i-Verifier

Current-State Estimator:



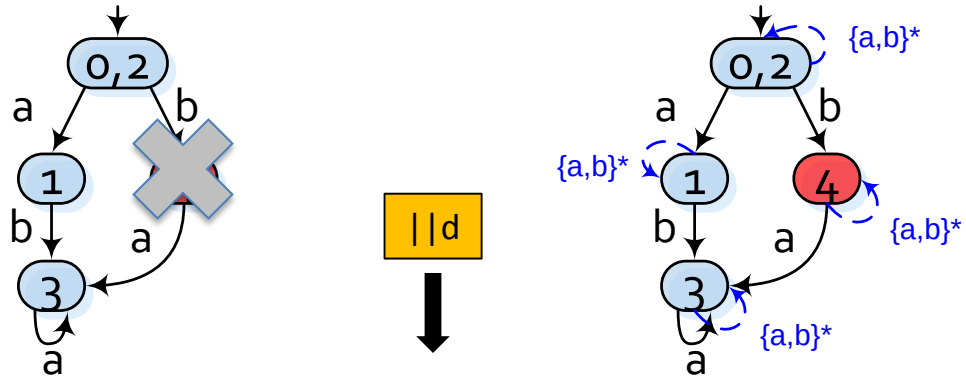



# The All-Insertion Structure (AIS)

## Enumerate all i-enforcing insertion functions

### [Step 1] i-Verifier

Current-State Estimator:

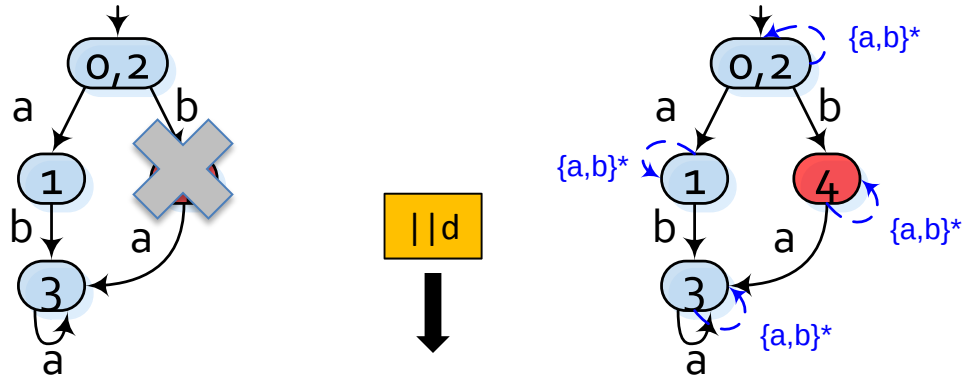


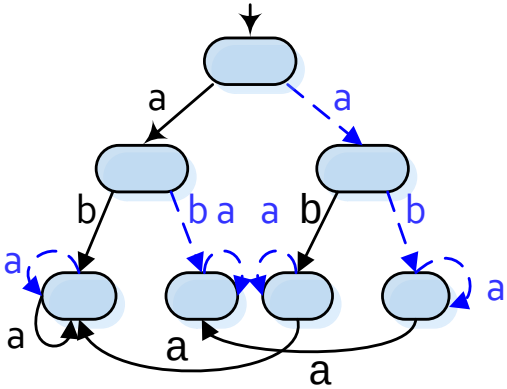
# The All-Insertion Structure (AIS)

## Enumerate all i-enforcing insertion functions

### [Step 1] i-Verifier

Current-State Estimator:

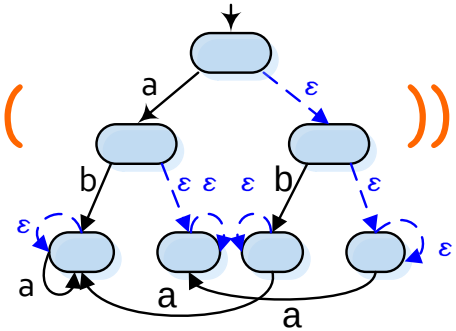



Nondeterministic, safe

# The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Test 1] L(  )) = P(L(G)) ? *

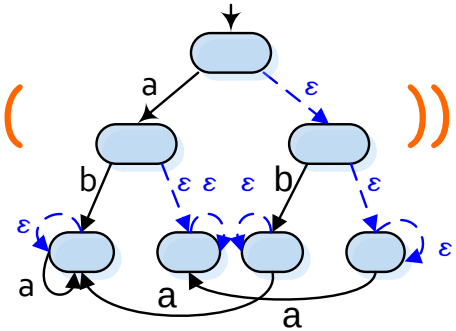


- "No": No i-enforcing insertion function exists. Stop.
- "Yes": Go to Step 2 and complete the AIS

*Should be included in the paper

# The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

[Test 1] L( 

)) = P(L(G)) ? *

- "Yes": Go to Step 2 and complete the AIS

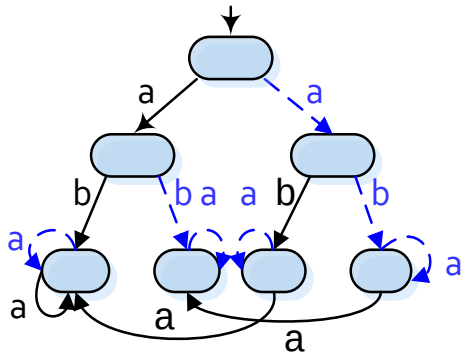
*Should be included in the paper

# The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

## [Step 2] Unfolded i-Verifier $V_u$

- determinization and unfolding



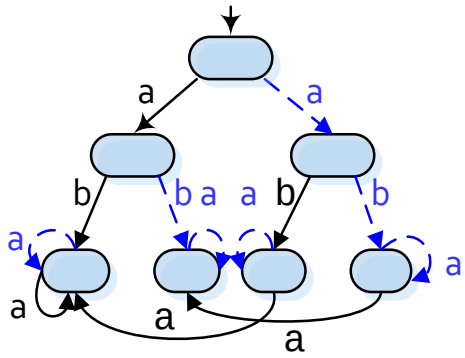

i-verifier

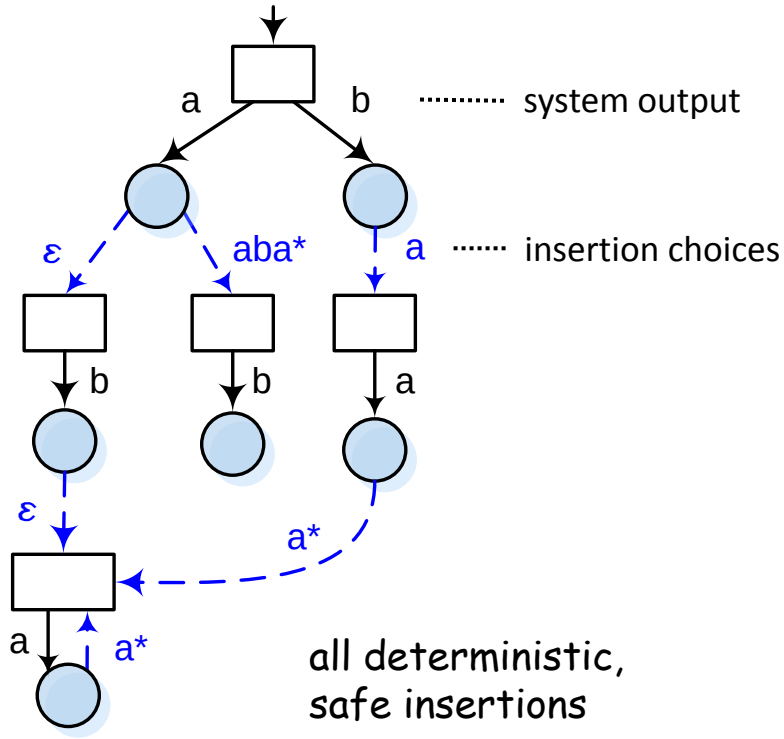
# The All-Insertion Structure (AIS)

## Enumerate all i-enforcing insertion functions

### [Step 2] Unfolded i-Verifier $V_u$

- determinization and unfolding



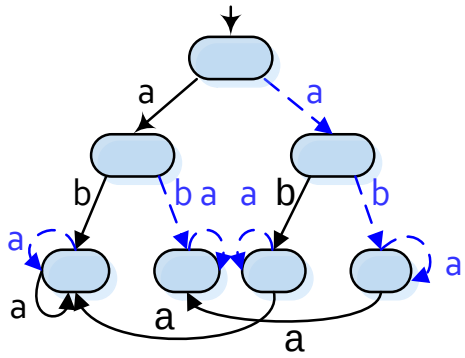



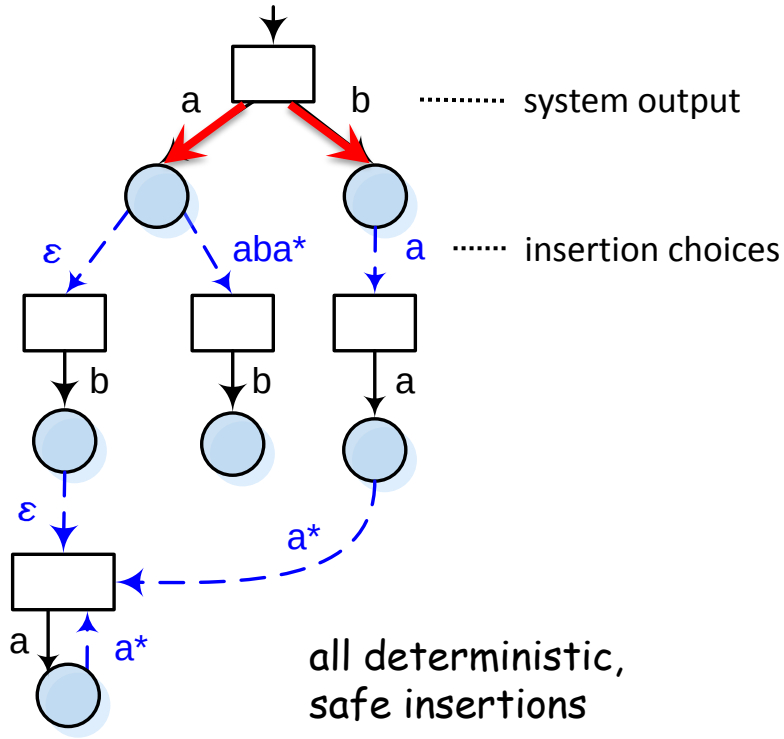
# The All-Insertion Structure (AIS)

## Enumerate all i-enforcing insertion functions

### [Step 2] Unfolded i-Verifier $V_u$

- determinization and unfolding



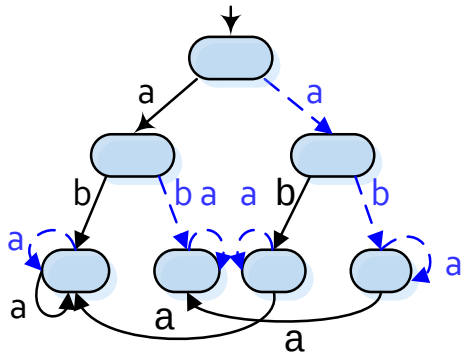



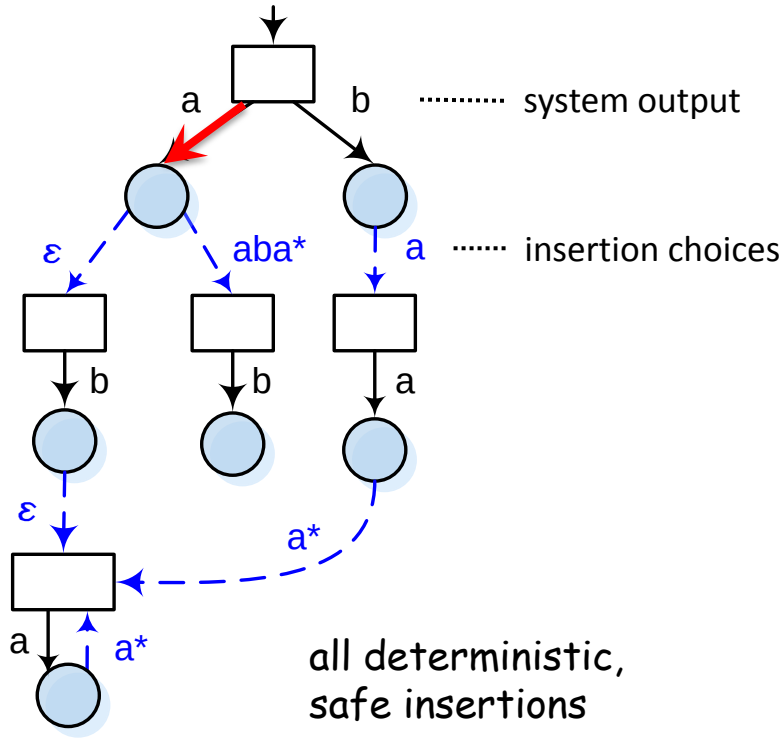
# The All-Insertion Structure (AIS)

## Enumerate all i-enforcing insertion functions

### [Step 2] Unfolded i-Verifier $V_u$

- determinization and unfolding



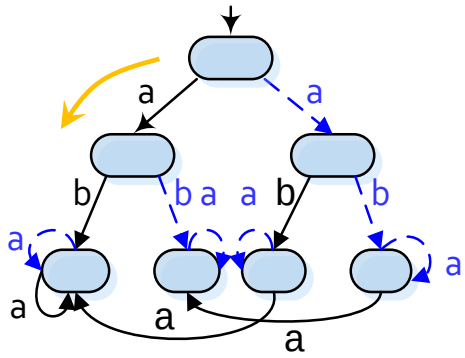



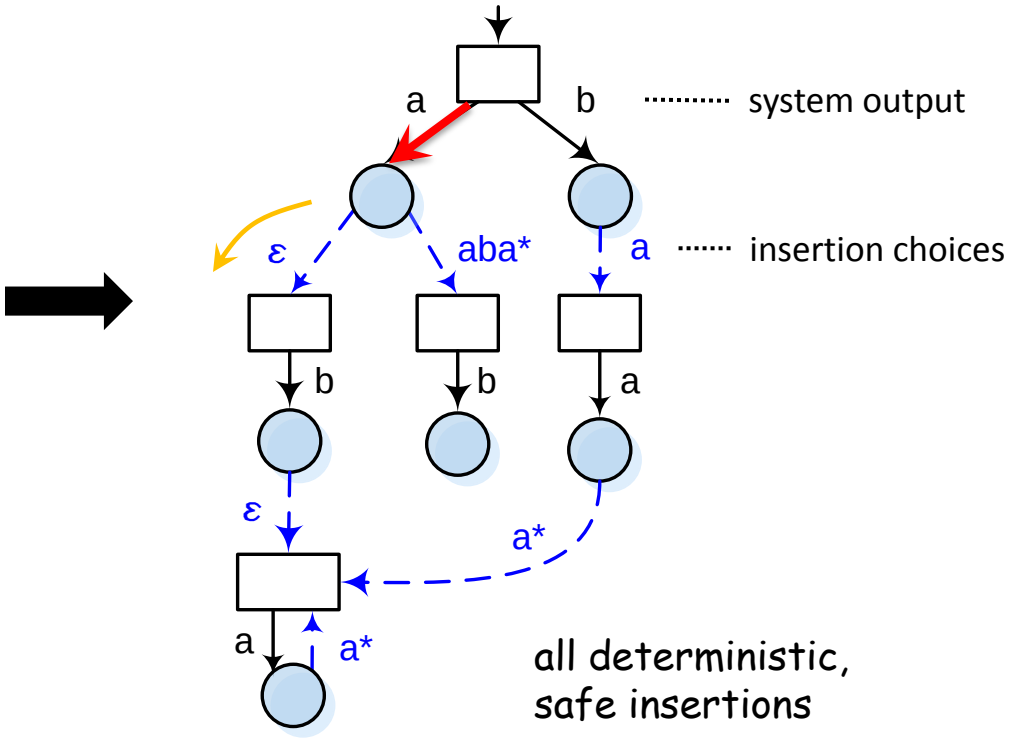
# The All-Insertion Structure (AIS)

## Enumerate all i-enforcing insertion functions

### [Step 2] Unfolded i-Verifier $V_u$

- determinization and unfolding



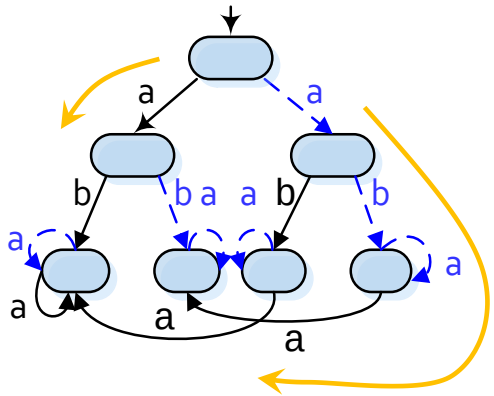



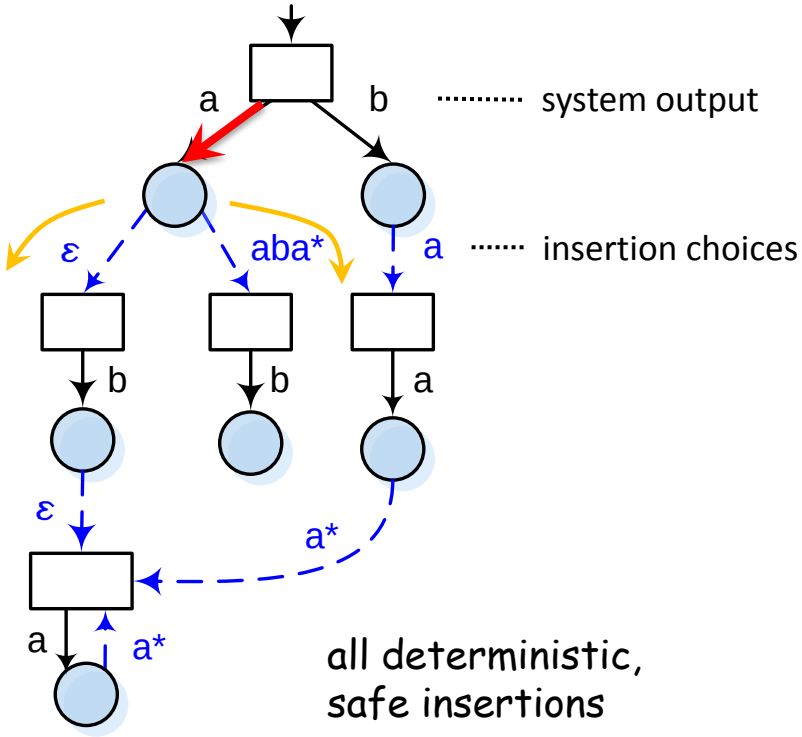
# The All-Insertion Structure (AIS)

## Enumerate all i-enforcing insertion functions

### [Step 2] Unfolded i-Verifier $V_u$

- determinization and unfolding

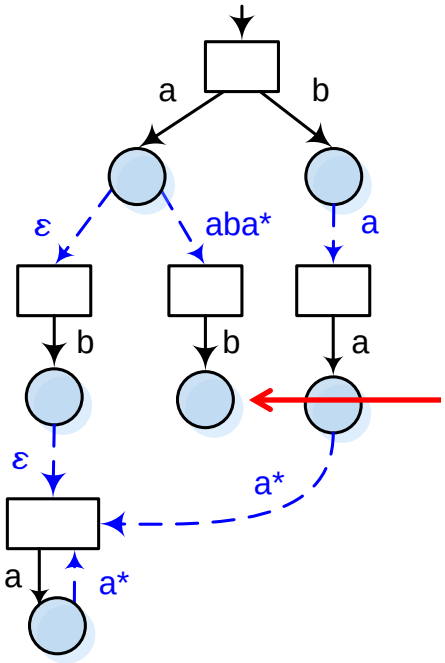


# The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

## [Step 3] The AIS

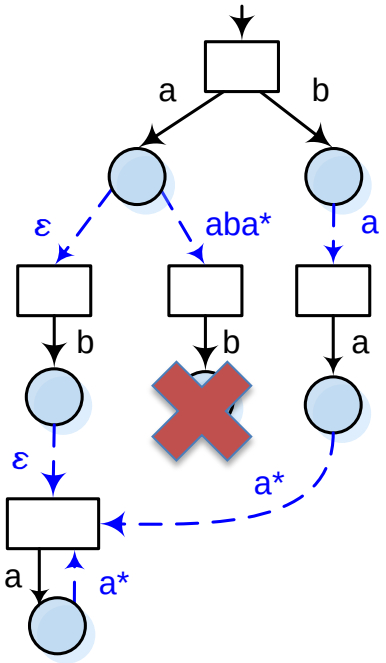



Unfolded i-Verifier

# The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

## [Step 3] The AIS

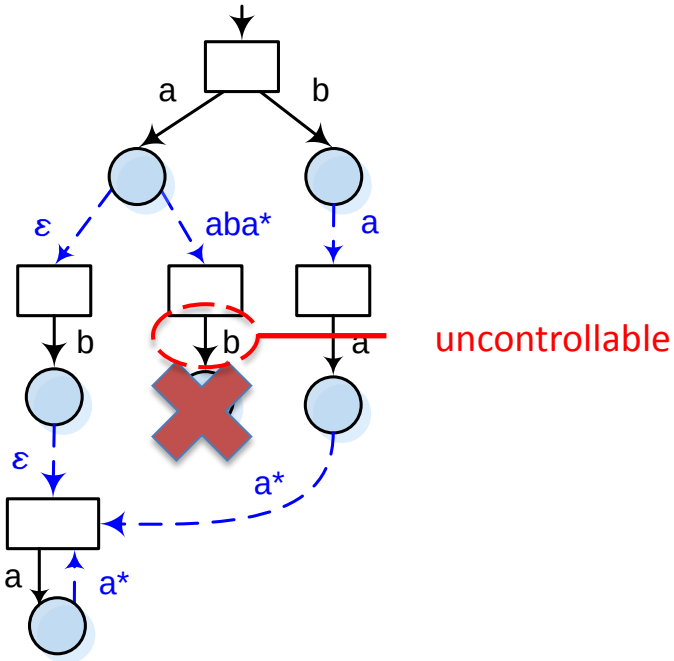



Unfolded i-Verifier

# The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

## [Step 3] The AIS

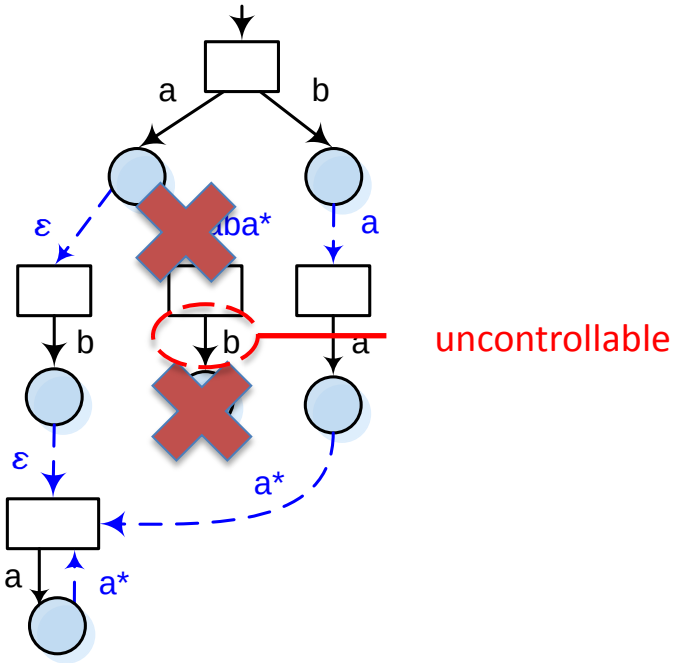



Unfolded i-Verifier

# The All-Insertion Structure (AIS)

Enumerate all i-enforcing insertion functions

## [Step 3] The AIS

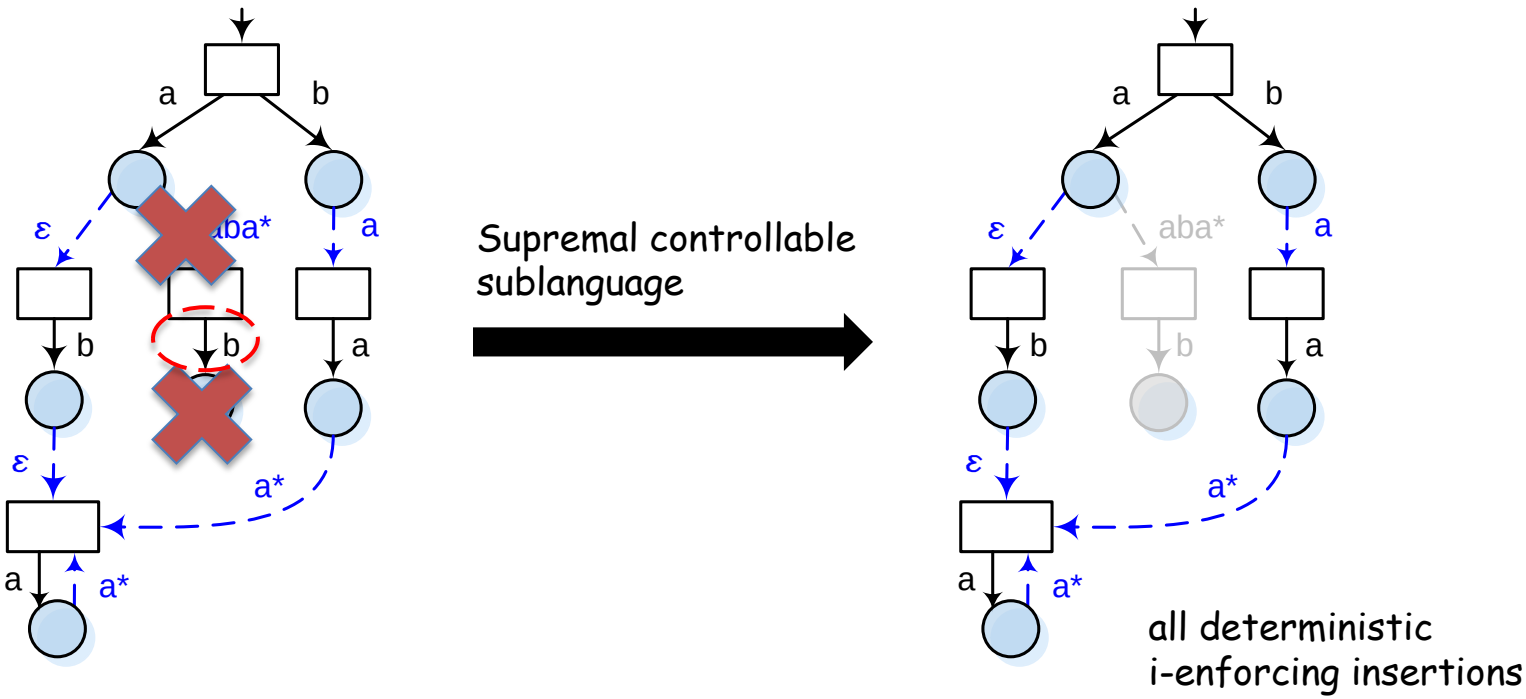



Unfolded i-Verifier

# The All-Insertion Structure (AIS)

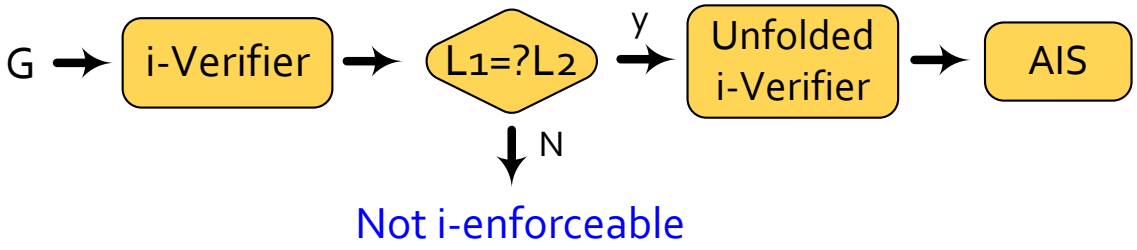
## Enumerate all i-enforcing insertion functions

### [Step 3] The AIS



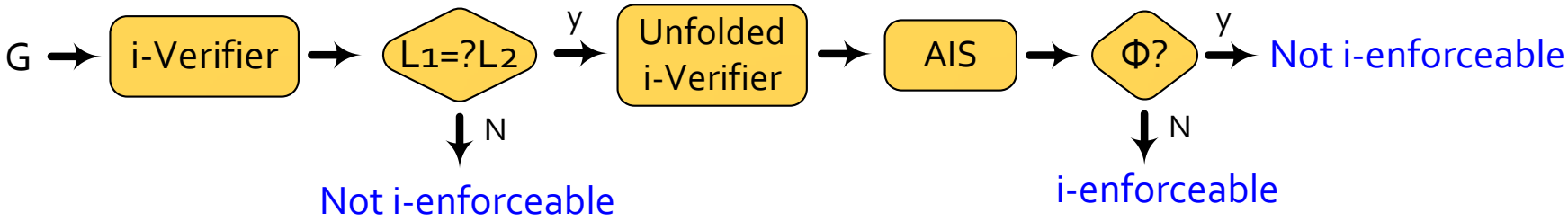

Unfolded i-Verifier

# The AIS Construction Workflow




The AIS enumerates all i-enforcing insertion functions

# The AIS Construction Workflow

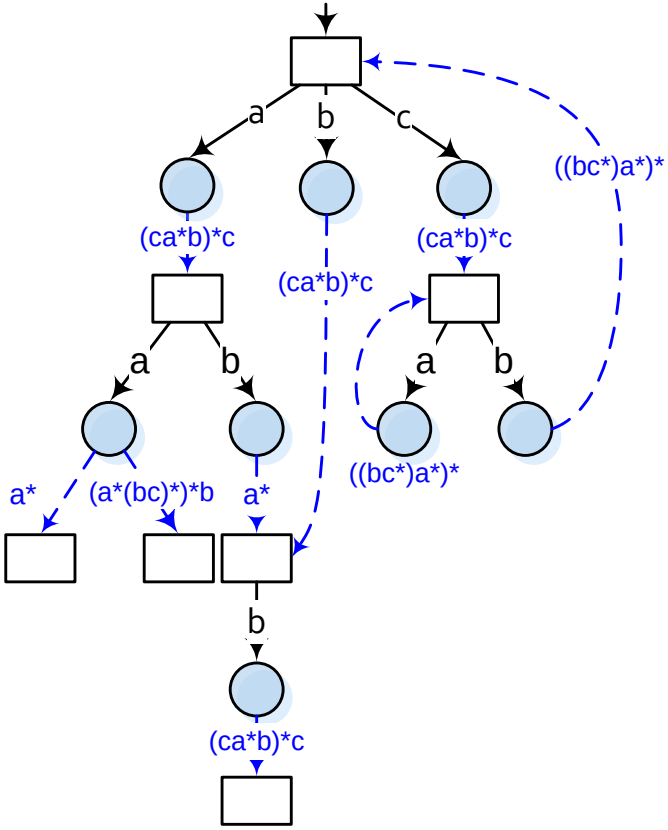



The AIS enumerates all i-enforcing insertion functions

**Theorem (I-Enforceability)**
An opacity property is i-enforceable iff the AIS is not the empty automaton

# Synthesis of I-Enforcing Insertion Functions

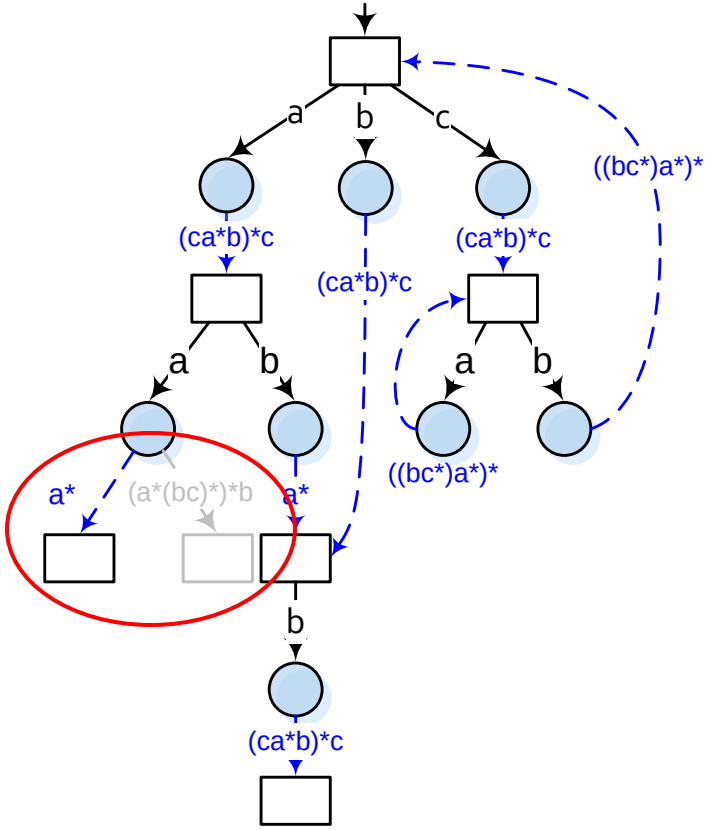
## Given the AIS



# Synthesis of I-Enforcing Insertion Functions

## Given the AIS

- Select one insertion at every circle state

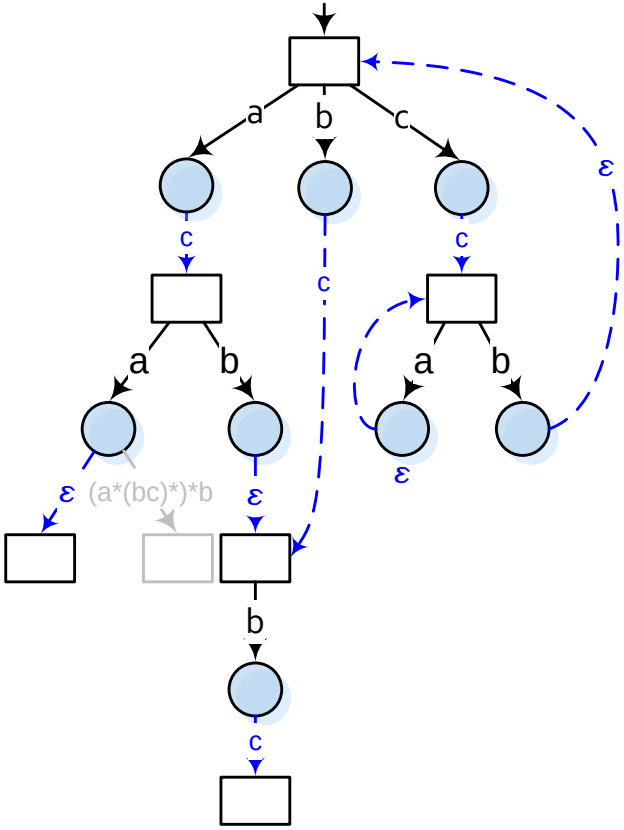



- One insertion choice

# Synthesis of I-Enforcing Insertion Functions

## Given the AIS

- Select one insertion at every circle state

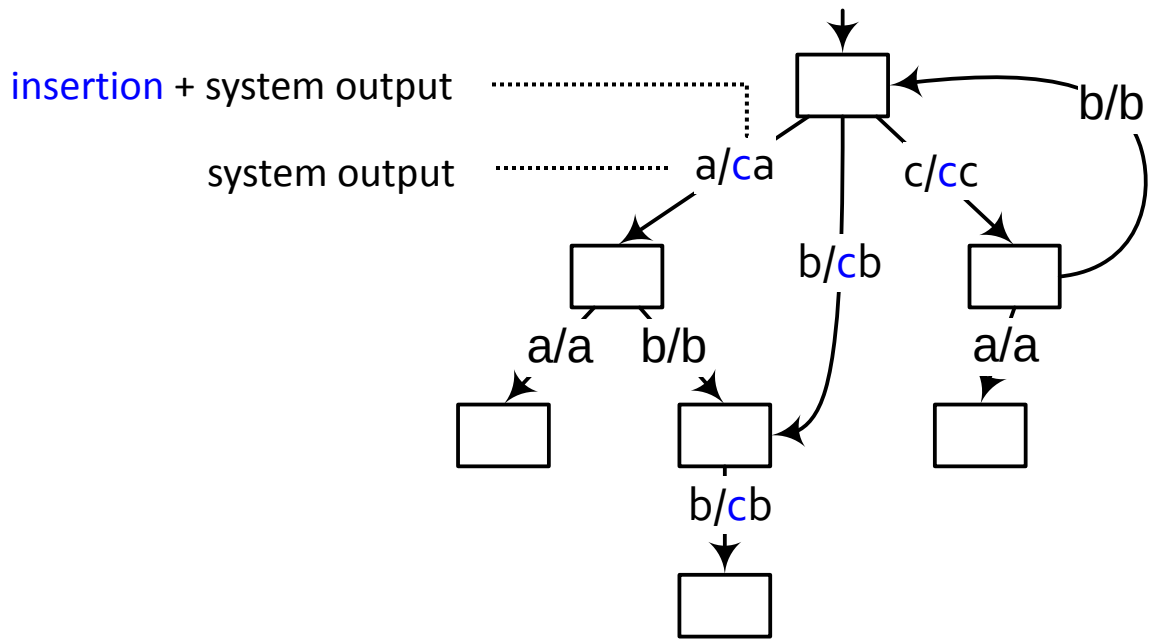



- One insertion choice
- One insertion string

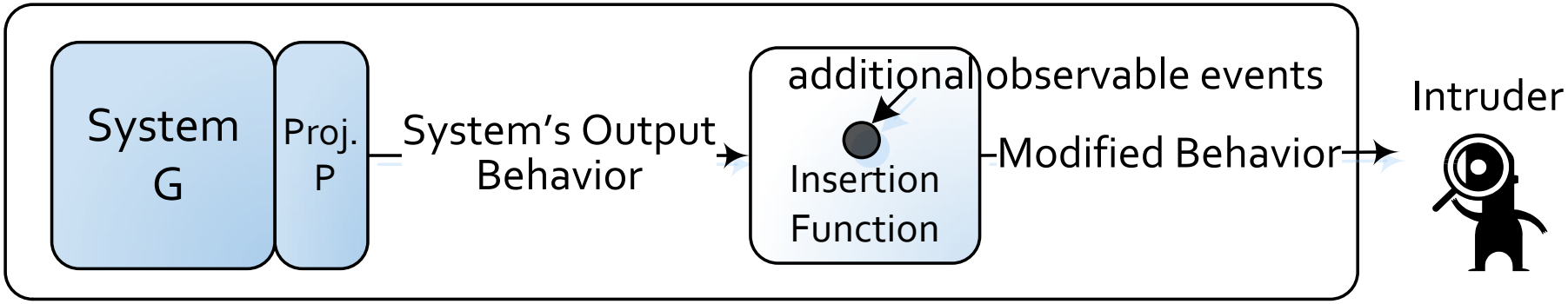
# Synthesis of I-Enforcing Insertion Functions

## Given the AIS

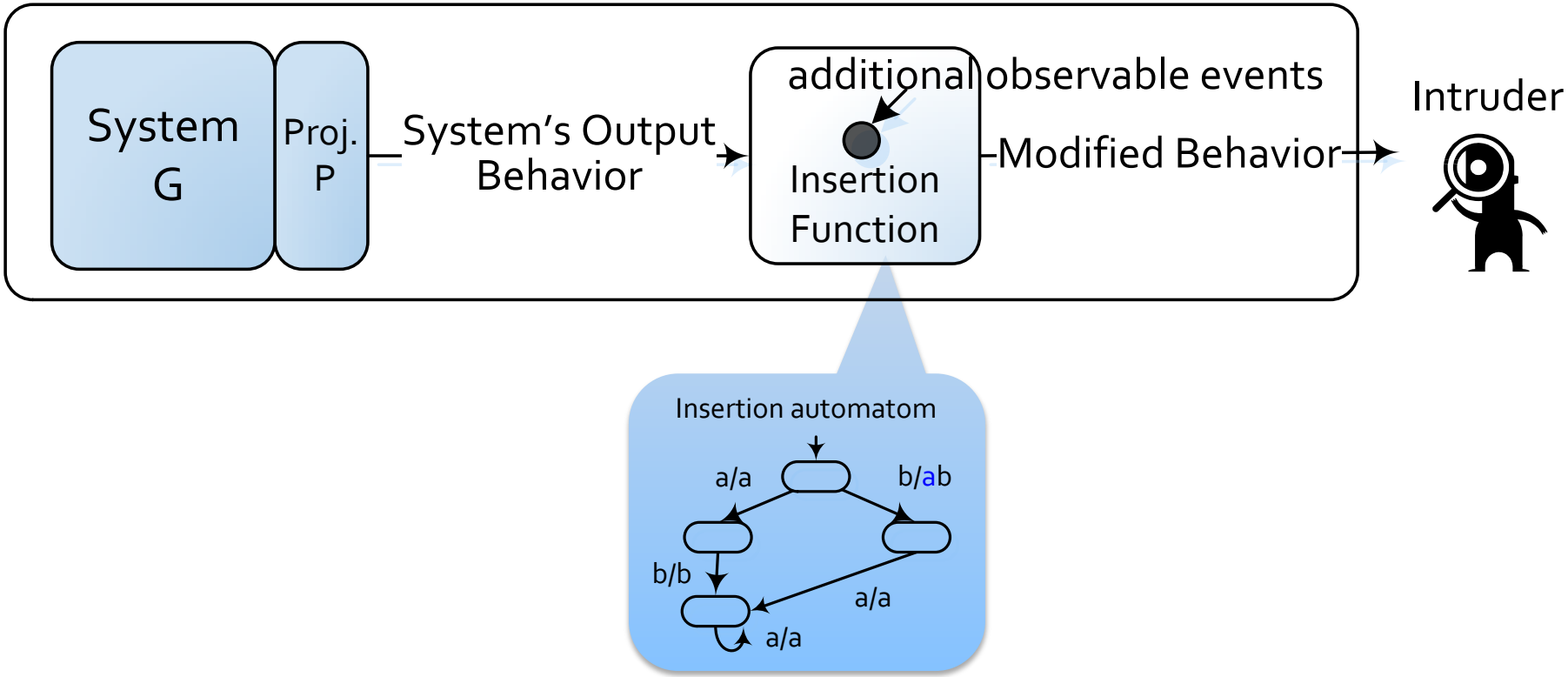
- Select one insertion at every circle state
- Translate into the insertion automaton



# Insertion Enforcement Mechanism



# Insertion Enforcement Mechanism




- A finite encoding of an insertion function

# Conclusion

- A new opacity enforcement mechanism using insertion functions
- Characterization of the "i-enforceability" property
- An algorithmic procedure to check i-enforceability
- An algorithmic procedure to synthesize i-enforcing insertion functions

# Future work

- Improve the algorithm for AIS construction
- Optimal insertion function
- Location-based service applications