Dynamic Observers for the Synthesis of Opaque Systems

Franck Cassez[1], Jérémy Dubreil[2] and Hervé Marchand[2]

[1] NICTA & CNRS
Sydney
Australia

[2] INRIA/IRISA
Rennes Bretagne Atlantique
France

ATVA'09, Macau SAR
October 13–16, 2009

Context

Need for Security in Transactional Systems

- Web-services: e-banking, online transactions
- Id documents: biometric passport, Medicare Card
- E-voting systems

Different Types of Security

- Integrity: illegal actions cannot be performed by an unauthorized user
  Bank account management cannot be managed by a third party
- Availability: some actions must be available
  Withdrawing money from your bank account
- Privacy: information should remain hidden from some users
  PIN code

Opacity was introduced in [Mazaré, 2004, Bryans et al., 2008]

Context

Need for Security in Transactional Systems

- Web-services: e-banking, online transactions
- Id documents: biometric passport, Medicare Card
- E-voting systems

Different Types of Security

- Integrity: illegal actions cannot be performed by an unauthorized user
  Bank account management cannot be managed by a third party
- Availability: some actions must be available
  Withdrawing money from your bank account
- Privacy: information should remain hidden from some users
  PIN code

In this paper we consider opacity (privacy)

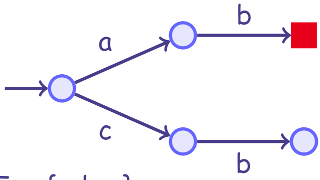
Opacity was introduced in [Mazaré, 2004, Bryans et al., 2008]

Outline of the Talk

Outline

What is Opacity?

System S



$\Sigma = \{a, b, c\}$

Secret F

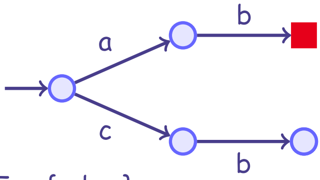
Secret = set of states ■

Events in $\Sigma_o \subseteq \Sigma$ are observable

Example: $\Sigma_o = \{b\}$

What is Opacity?

System S



$\Sigma = \{a, b, c\}$

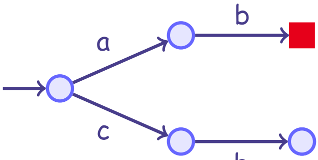
Secret F

Secret = set of states ■
Events in $\Sigma_o \subseteq \Sigma$ are observable
Example: $\Sigma_o = \{b\}$

Opacity: an external observer should never know F-states

What is Opacity?

System S



$\Sigma = \{a, b, c\}$

Secret F

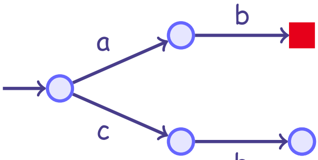
Secret = set of states ■
Events in $\Sigma_o \subseteq \Sigma$ are observable
Example: $\Sigma_o = \{b\}$

Secret F is opaque

Opacity: an external observer should never know F-states

What is Opacity?

System S



$\Sigma = \{a, b, c\}$

Secret F

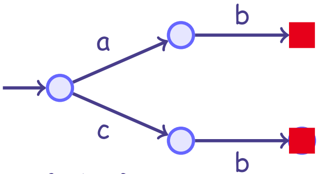
Secret = set of states ■
Events in $\Sigma_o \subseteq \Sigma$ are observable
Example: $\Sigma_o = \{a, b\}$

Secret F is not opaque

Opacity: an external observer should never know F-states

What is Opacity?

System S



$\Sigma = \{a, b, c\}$

Secret F

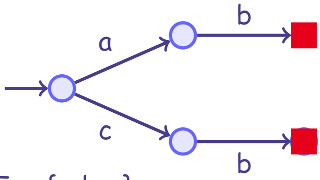
Secret = set of states ■
Events in $\Sigma_o \subseteq \Sigma$ are observable
Example: $\Sigma_o = \{b\}$

Secret F is not opaque

Opacity: an external observer should never know F-states

What is Opacity?

System S



$\Sigma = \{a, b, c\}$

Secret F

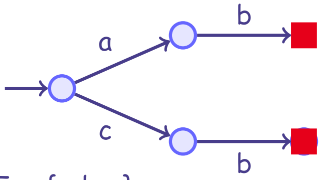
Secret = set of states ■
Events in $\Sigma_o \subseteq \Sigma$ are observable
Example: $\Sigma_o = \{b\}$

Opacity Verification Problem: Is F opaque w.r.t. (S, Σ_o) ?

What is Opacity?

System S

Secret F



$\Sigma = \{a, b, c\}$

Secret = set of states ■
Events in $\Sigma_o \subseteq \Sigma$ are observable
Example: $\Sigma_o = \{b\}$

Opacity Verification Problem: Is F opaque w.r.t. (S, Σ_o) ?

To check opacity: use your favorite Formal Method:

- Model-checking
- Theorem proving
- Tools to support automatic analysis of systems

Opacity for Non-Deterministic Automata

$A = (Q, q_0, \Sigma, \delta, F)$ a NDA

$\Sigma_o \subseteq \Sigma$ set of observable events

F set of secret states

NDA A $\xrightarrow{u \in \Sigma^*}$ Projection P $\xrightarrow{v = P(u) \in \Sigma_o^*}$ Attacker U

Assumptions

- Attacker knows A and the projection P/alphabet Σ_o
- $K_{\Sigma_o}(v)$: knowledge set (of states) of the attacker after observing v

Definition (Opacity)

F is opaque w.r.t. (A, Σ_o) if $\forall v \in P(Tr(A)), K_{\Sigma_o}(v) \not\subseteq F$ $(K_{\Sigma_o}(v) \cap (Q \setminus F) \neq \emptyset)$.

Opacity Problem

Input: A NDA A, F secret set of states, Σ_o set of observable events.
Problem: Is F opaque w.r.t. (A, Σ_o) ?

Opacity for Non-Deterministic Automata

$A = (Q, q_0, \Sigma, \delta, F)$ a NDA
$\Sigma_o \subseteq \Sigma$ set of observable events

F set of secret states

$$\begin{tikzcd} \boxed{\text{NDA } A} \arrow[r, "u \in \Sigma^*"] & \boxed{\text{Projection } P} \arrow[r, "v = P(u) \in \Sigma_o^*"] & \boxed{\text{Attacker } U} \end{tikzcd}$$

Assumptions

- Attacker knows A and the projection P/alphabet Σ_o
- $K_{\Sigma_o}(v)$: knowledge set (of states) of the attacker after observing v

Definition (Opacity)

F is opaque w.r.t. (A, Σ_o) if $\forall v \in P(Tr(A)), K_{\Sigma_o}(v) \not\subseteq F$ $(K_{\Sigma_o}(v) \cap (Q \setminus F) \neq \emptyset)$.

Opacity Problem

Input: A NDA A, F secret set of states, Σ_o set of observable events.
Problem: Is F opaque w.r.t. (A, Σ_o) ?

Opacity for Non-Deterministic Automata

$A = (Q, q_0, \Sigma, \delta, F)$ a NDA

$\Sigma_o \subseteq \Sigma$ set of observable events

F set of secret states

$$\boxed{\text{NDA } A} \xrightarrow{u \in \Sigma^*} \boxed{\text{Projection } P} \xrightarrow{v = P(u) \in \Sigma_o^*} \boxed{\text{Attacker } U}$$

Assumptions

- Attacker knows A and the projection P/alphabet Σ_o
- $K_{\Sigma_o}(v)$: knowledge set (of states) of the attacker after observing v

Definition (Opacity)

F is opaque w.r.t. (A, Σ_o) if $\forall v \in P(Tr(A)), K_{\Sigma_o}(v) \not\subseteq F$ ($K_{\Sigma_o}(v) \cap (Q \setminus F) \neq \emptyset$).

Opacity Problem

Input: A NDA A, F secret set of states, Σ_o set of observable events.
Problem: Is F opaque w.r.t. (A, Σ_o) ?

Knowledge Set of the Attacker

- $Tr(A)$ = set of words generated by A
- P is the projection over $\Sigma_o \subseteq \Sigma$
- $P^{-1}(w)$ = set of words which project onto w $\qquad P^{-1} : \Sigma_o^* \longrightarrow 2^{\Sigma^*}$
- $Pre(\varepsilon) = \{\varepsilon\}$ and $Pre(u.\lambda) = P^{-1}(u).\lambda \cap Tr(A)$
- Knowledge set of U: $K_{\Sigma_o}(u) = \delta(q_0, Pre(u))$

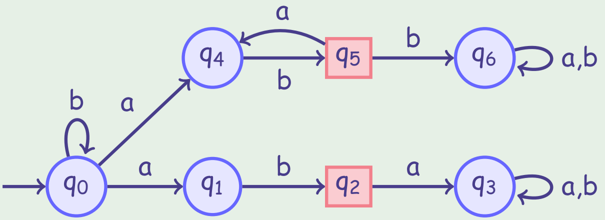
Consider knowledge set right after each observation of the attacker

Knowledge Set of the Attacker

- $Tr(A)$ = set of words generated by A
- P is the projection over $\Sigma_o \subseteq \Sigma$
- $P^{-1}(w)$ = set of words which project onto w $\qquad P^{-1} : \Sigma_o^* \longrightarrow 2^{\Sigma^*}$
- $Pre(\varepsilon) = \{\varepsilon\}$ and $Pre(u.\lambda) = P^{-1}(u).\lambda \cap Tr(A)$
- Knowledge set of U: $K_{\Sigma_o}(u) = \delta(q_0, Pre(u))$

Example

$\Sigma_o = \{b\}$
$P(b.b.a.b.a) = b.b.b$
$P^{-1}(b) = a^*.b.a^*$
$Pre(b) = \{b, a.b\}$
$K_{\Sigma_o}(b) = \{q_0, q_5, q_2\}$



Knowledge Set of the Attacker

- $Tr(A)$ = set of words generated by A
- P is the projection over $\Sigma_o \subseteq \Sigma$
- $P^{-1}(w)$ = set of words which project onto w $\qquad P^{-1} : \Sigma_o^* \longrightarrow 2^{\Sigma^*}$
- $Pre(\varepsilon) = \{\varepsilon\}$ and $Pre(u.\lambda) = P^{-1}(u).\lambda \cap Tr(A)$
- Knowledge set of U: $K_{\Sigma_o}(u) = \delta(q_0, Pre(u))$

Problem 1: Checking opacity with Static Filters

Input: a NDA A, F secret set of states, Σ_o set of observable events.
Problem: Is F opaque w.r.t. (A, Σ_o) ?

Theorem

Problem 1 is PSPACE-complete.

Algorithms for Checking Opacity

Proof.

Reduction of universality problem for non-deterministic automaton.
Given A over Σ with accepting states F, the universality problem is:

$$\text{decide whether } L_F(A) = \Sigma^*.$$

Assume A is complete i.e. $Tr(A) = \Sigma^*$.
Reduction:

$$A \text{ is universal iff } Q \setminus F \text{ is opaque for } (A, \Sigma).$$

□

Algorithm to Check Opacity

1. Subset construction
2. check whether a subset $S \subseteq F$ is reachable

What if the system is NOT opaque ?

Algorithms for Checking Opacity

Proof.

Reduction of universality problem for non-deterministic automaton.
Given A over Σ with accepting states F, the universality problem is:

$$\text{decide whether } L_F(A) = \Sigma^*.$$

Assume A is complete i.e. $Tr(A) = \Sigma^*$.
Reduction:

$$A \text{ is universal iff } Q \setminus F \text{ is opaque for } (A, \Sigma).$$

□

Algorithm to Check Opacity

1. Subset construction
2. check whether a subset $S \subseteq F$ is reachable

What if the system is NOT opaque ?

Algorithms for Checking Opacity

Proof.

Reduction of universality problem for non-deterministic automaton.
Given A over Σ with accepting states F, the universality problem is:

$$\text{decide whether } L_F(A) = \Sigma^*.$$

Assume A is complete i.e. $Tr(A) = \Sigma^*$.
Reduction:

$$A \text{ is universal iff } Q \setminus F \text{ is opaque for } (A, \Sigma).$$

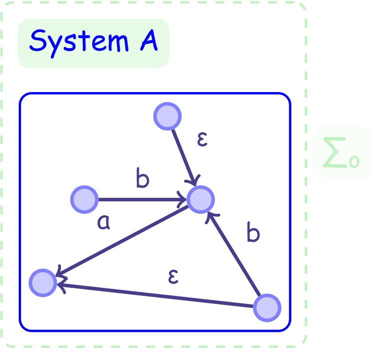
$\square$

Algorithm to Check Opacity

1. Subset construction
2. check whether a subset $S \subseteq F$ is reachable

What if the system is NOT opaque ?

Enforcing Opacity

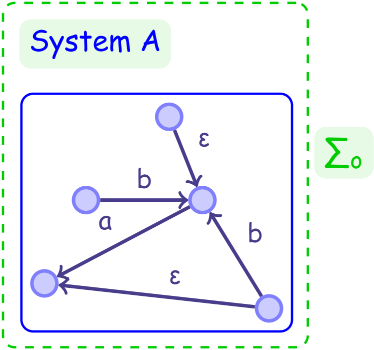


- Either restrict set of behaviours: add a controller C

  [Dubreil et al., 2008]

- Or modify the set of observable events to $\Sigma_o' \neq \Sigma_o$

Enforcing Opacity



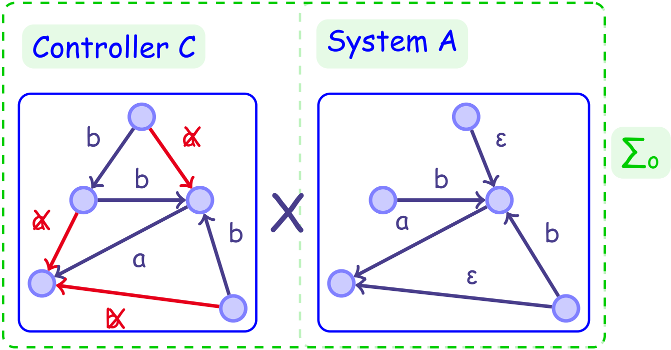
- Either restrict set of behaviours: add a controller C

[Dubreil et al., 2008]

- Or modify the set of observable events to $\Sigma_o' \neq \Sigma_o$

(A, Σ_o) is NOT opaque

Enforcing Opacity

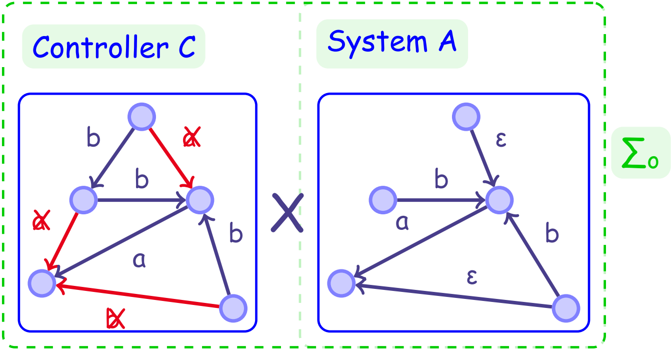


- Either restrict set of behaviours: add a controller C

[Dubreil et al., 2008]

- Or modify the set of observable events to $\Sigma'_o \neq \Sigma_o$

Enforcing Opacity



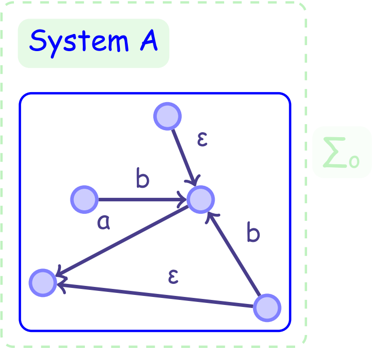
- Either restrict set of behaviours: add a controller C

  [Dubreil et al., 2008]

- Or modify the set of observable events to $\Sigma_o' \neq \Sigma_o$

Ensure $(C \times A, \Sigma_o)$ is opaque

Enforcing Opacity

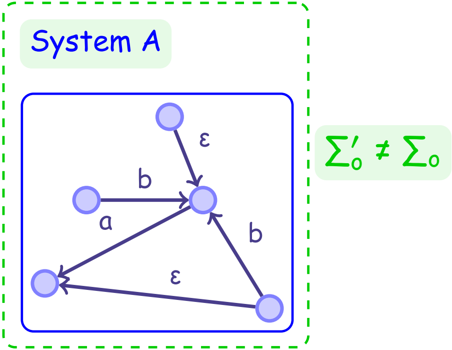


- Either restrict set of behaviours: add a controller C

  [Dubreil et al., 2008]

- Or modify the set of observable events to $\Sigma_o' \neq \Sigma_o$

Enforcing Opacity



- Either restrict set of behaviours: add a controller C

[Dubreil et al., 2008]

- Or modify the set of observable events to $\Sigma_o' \neq \Sigma_o$

Ensure (G, Σ_o') is opaque

Outline

Minimization of the Set of Hidden Events

- Events = services provided to (external) users
- Hiding events = restricting services

Goal: ensure opacity while preserving services

Problem 2: Static Minimization Problem

Input: $A = (Q, q_0, \Sigma, \delta, F)$ a NDA, F secret set of states and $n \in \mathbb{N}$.
Problem: Is there any $\Sigma_o \subseteq \Sigma$ with $|\Sigma_o| \geq n$ s.t. F is opaque w.r.t. (A, Σ_o) ?

Theorem

Problem 2 is PSPACE-complete.

Computing the maximum n is also PSPACE-complete.

Can we do any better ?

Minimization of the Set of Hidden Events

- Events = services provided to (external) users
- Hiding events = restricting services

Goal: ensure opacity while preserving services

Problem 2: Static Minimization Problem

Input: $A = (Q, q_0, \Sigma, \delta, F)$ a NDA, F secret set of states and $n \in \mathbb{N}$.
Problem: Is there any $\Sigma_o \subseteq \Sigma$ with $|\Sigma_o| \geq n$ s.t. F is opaque w.r.t. (A, Σ_o) ?

Theorem

Problem 2 is PSPACE-complete.

Computing the maximum n is also PSPACE-complete.

Can we do any better ?

Minimization of the Set of Hidden Events

- Events = services provided to (external) users
- Hiding events = restricting services

Goal: ensure opacity while preserving services

Problem 2: Static Minimization Problem

Input: $A = (Q, q_0, \Sigma, \delta, F)$ a NDA, F secret set of states and $n \in \mathbb{N}$.
Problem: Is there any $\Sigma_o \subseteq \Sigma$ with $|\Sigma_o| \geq n$ s.t. F is opaque w.r.t. (A, Σ_o) ?

Theorem

Problem 2 is PSPACE-complete.

Computing the maximum n is also PSPACE-complete.

Can we do any better ?

Minimization of the Set of Hidden Events

- Events = services provided to (external) users
- Hiding events = restricting services

Goal: ensure opacity while preserving services

Problem 2: Static Minimization Problem

Input: $A = (Q, q_0, \Sigma, \delta, F)$ a NDA, F secret set of states and $n \in \mathbb{N}$.
Problem: Is there any $\Sigma_o \subseteq \Sigma$ with $|\Sigma_o| \geq n$ s.t. F is opaque w.r.t. (A, Σ_o) ?

Theorem

Problem 2 is PSPACE-complete.

Computing the maximum n is also PSPACE-complete.

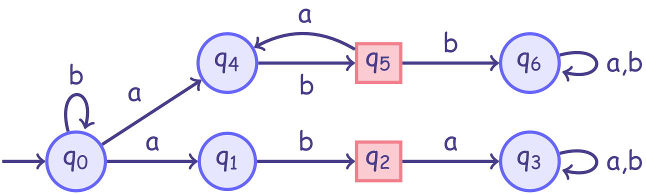
Can we do any better ?

Using Dynamic Filters

- **Static Filter:** $\Sigma_o = \{a\}$ or $\Sigma_o = \{b\} \Rightarrow F$ is opaque
  Must hide at least one event
- **Dynamic Filter:** Hide b after the observation of an a and let everything be observable after the observation of a second a

Result: Events are more often visible

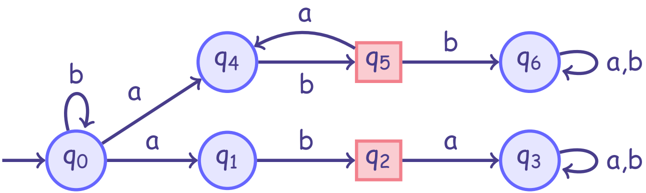
Using Dynamic Filters



- **Static Filter:** $\Sigma_o = \{a\}$ or $\Sigma_o = \{b\} \Rightarrow$ F is opaque
  Must hide at least one event
- **Dynamic Filter:** Hide b after the observation of an a and let everything be observable after the observation of a second a

Result: Events are more often visible

Using Dynamic Filters



- **Static Filter**: $\Sigma_o = \{a\}$ or $\Sigma_o = \{b\} \Rightarrow$ F is opaque
  Must hide at least one event
- **Dynamic Filter**: Hide b after the observation of an a and let everything be observable after the observation of a second a

Result: Events are more often visible

Outline

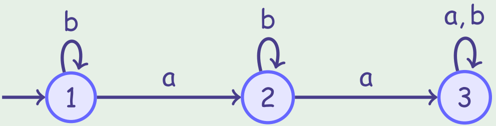
Dynamic Filters

Definition (Dynamic Filter)

A dynamic filter is a complete (infinite) deterministic labeled transition system $\Phi = (X, x_0, \Sigma, \delta_o, L)$ where

- $L : X \longrightarrow 2^{\Sigma}$ is a labeling function that specifies the set of events that are observable at state x;
- For all $x \in X$ and for all $\lambda \in \Sigma$, if $\lambda \notin L(x)$, then $\delta_o(x, \lambda) = x$.

Example (Finite State Filter)



$L(1) = \{a, b\}$ $\quad$ $L(2) = \{a\}$ $\quad$ $L(3) = \{a, b\}$

Φ is also a transducer

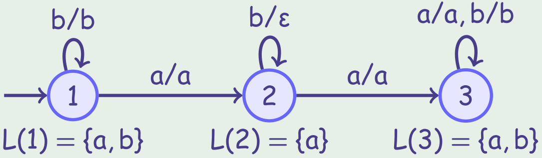
Dynamic Filters

Definition (Dynamic Filter)

A **dynamic filter** is a complete (infinite) deterministic labeled transition system $\Phi = (X, x_0, \Sigma, \delta_o, L)$ where

- $L : X \longrightarrow 2^{\Sigma}$ is a labeling function that specifies the set of events that are observable at state x;
- For all $x \in X$ and for all $\lambda \in \Sigma$, if $\lambda \notin L(x)$, then $\delta_o(x, \lambda) = x$.

Example (Finite State Filter)



Φ is also a **transducer**

Opacity with Dynamic Filters

- $\Phi^{-1}(w)$ = set of words u that filter onto w (s.t. $\Phi(u) = w$)
- $Pre(\varepsilon) = \{\varepsilon\}$ and $Pre(u.\lambda) = \Phi^{-1}(u).\lambda \cap Tr(A)$
- Knowledge set of attacker: $K_\Phi(u) = \delta(q_0, Pre(u))$

Definition (Opacity)

F is opaque w.r.t. (A, Φ) if $\forall u \in \Phi(Tr(A)), K_\Phi(u) \not\subseteq F$.

Problem 3: Opacity Problem with Dynamic Filters

Input: A a NDA, F secret set of states, Φ a filter.
Problem: Is F opaque w.r.t. (A, Φ) ?

Issues

- How to check opacity with a dynamic filter?
- How to compare dynamic filters?
- How to synthesize optimal dynamic filters?

Opacity with Dynamic Filters

- $\Phi^{-1}(w)$ = set of words u that filter onto w (s.t. $\Phi(u) = w$)
- $Pre(\varepsilon) = \{\varepsilon\}$ and $Pre(u.\lambda) = \Phi^{-1}(u).\lambda \cap Tr(A)$
- Knowledge set of attacker: $K_\Phi(u) = \delta(q_0, Pre(u))$

Definition (Opacity)

F is opaque w.r.t. (A, Φ) if $\forall u \in \Phi(Tr(A)), K_\Phi(u) \not\subseteq F$.

Problem 3: Opacity Problem with Dynamic Filters

Input: A a NDA, F secret set of states, Φ a filter.
Problem: Is F opaque w.r.t. (A, Φ) ?

Issues

- How to **check** opacity with a dynamic filter?
- How to **compare** dynamic filters?
- How to **synthesize** optimal dynamic filters?

Opacity with Dynamic Filters

- $\Phi^{-1}(w)$ = set of words u that filter onto w (s.t. $\Phi(u) = w$)
- $Pre(\varepsilon) = \{\varepsilon\}$ and $Pre(u.\lambda) = \Phi^{-1}(u).\lambda \cap Tr(A)$
- Knowledge set of attacker: $K_\Phi(u) = \delta(q_0, Pre(u))$

Definition (Opacity)

F is opaque w.r.t. (A, Φ) if $\forall u \in \Phi(Tr(A)), K_\Phi(u) \not\subseteq F$.

Problem 3: Opacity Problem with Dynamic Filters

Input: A a NDA, F secret set of states, Φ a filter.
Problem: Is F opaque w.r.t. (A, Φ) ?

Issues

- How to check opacity with a dynamic filter?
- How to compare dynamic filters?
- How to synthesize optimal dynamic filters?

Opacity with Dynamic Filters

- $\Phi^{-1}(w)$ = set of words u that filter onto w (s.t. $\Phi(u) = w$)
- $Pre(\varepsilon) = \{\varepsilon\}$ and $Pre(u.\lambda) = \Phi^{-1}(u).\lambda \cap Tr(A)$
- Knowledge set of attacker: $K_\Phi(u) = \delta(q_0, Pre(u))$

Definition (Opacity)

F is opaque w.r.t. (A, Φ) if $\forall u \in \Phi(Tr(A)), K_\Phi(u) \not\subseteq F$.

Problem 3: Opacity Problem with Dynamic Filters

Input: A a NDA, F secret set of states, Φ a filter.
Problem: Is F opaque w.r.t. (A, Φ) ?

Issues

- How to check opacity with a dynamic filter?
- How to compare dynamic filters?
- How to synthesize optimal dynamic filters?

Checking Opacity for Finite State Filters

Opacity for Finite State Filters

Input: A, F, Φ a finite state filter.
Problem: Is F opaque w.r.t. (A, Φ) ?

To check opacity, build a product $A \otimes \Phi$

1. initial state (q_0, x_0)
2. $(q, x) \xrightarrow{\lambda} (q', x')$ iff $q \xrightarrow{\lambda}_A q'$, $x \xrightarrow{\lambda}_\Phi x'$ and $\lambda \in L(x)$;
3. $(q, x) \xrightarrow{\varepsilon} (q', x)$ iff $q \xrightarrow{\lambda}_A q'$ and $\lambda \notin L(x)$.

Theorem

F is opaque w.r.t. (A, Φ) iff $F \times X$ is opaque w.r.t. to $(A \otimes \Phi, \Sigma)$.

Consequence: Problem 3 is PSPACE-complete.

How to compare dynamic filters ?

Checking Opacity for Finite State Filters

Opacity for Finite State Filters

Input: A, F, Φ a finite state filter.
Problem: Is F opaque w.r.t. (A, Φ) ?

To check opacity, build a product $A \otimes \Phi$

1. initial state (q_0, x_0)
2. $(q, x) \xrightarrow{\lambda} (q', x')$ iff $q \xrightarrow{\lambda}_A q'$, $x \xrightarrow{\lambda}_\Phi x'$ and $\lambda \in L(x)$;
3. $(q, x) \xrightarrow{\varepsilon} (q', x)$ iff $q \xrightarrow{\lambda}_A q'$ and $\lambda \not\in L(x)$.

Theorem

F is opaque w.r.t. (A, Φ) iff $F \times X$ is opaque w.r.t. to $(A \otimes \Phi, \Sigma)$.

Consequence: Problem 3 is PSPACE-complete.

How to compare dynamic filters ?

Checking Opacity for Finite State Filters

Opacity for Finite State Filters

Input: A, F, Φ a finite state filter.
Problem: Is F opaque w.r.t. (A, Φ) ?

To check opacity, build a product $A \otimes \Phi$

1. initial state (q_0, x_0)
2. $(q,x) \xrightarrow{\lambda} (q',x')$ iff $q \xrightarrow{\lambda}_A q'$, $x \xrightarrow{\lambda}_\Phi x'$ and $\lambda \in L(x)$;
3. $(q,x) \xrightarrow{\varepsilon} (q',x)$ iff $q \xrightarrow{\lambda}_A q'$ and $\lambda \not\in L(x)$.

Theorem

F is opaque w.r.t. (A, Φ) iff $F \times X$ is opaque w.r.t. to $(A \otimes \Phi, \Sigma)$.

Consequence: Problem 3 is PSPACE-complete.

How to compare dynamic filters ?

Checking Opacity for Finite State Filters

Opacity for Finite State Filters

Input: A, F, Φ a finite state filter.
Problem: Is F opaque w.r.t. (A, Φ) ?

To check opacity, build a product $A \otimes \Phi$

1. initial state (q_0, x_0)
2. $(q, x) \xrightarrow{\lambda} (q', x')$ iff $q \xrightarrow{\lambda}_A q'$, $x \xrightarrow{\lambda}_\Phi x'$ and $\lambda \in L(x)$;
3. $(q, x) \xrightarrow{\varepsilon} (q', x)$ iff $q \xrightarrow{\lambda}_A q'$ and $\lambda \notin L(x)$.

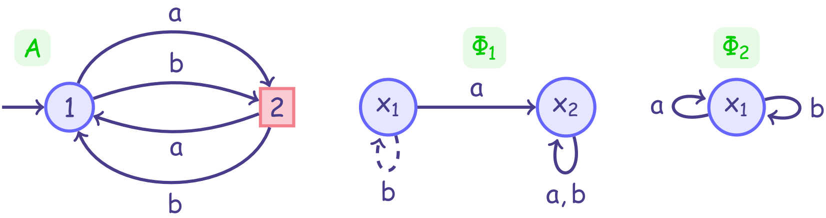
Theorem

F is opaque w.r.t. (A, Φ) iff $F \times X$ is opaque w.r.t. to $(A \otimes \Phi, \Sigma)$.

Consequence: Problem 3 is PSPACE-complete.

How to compare dynamic filters ?

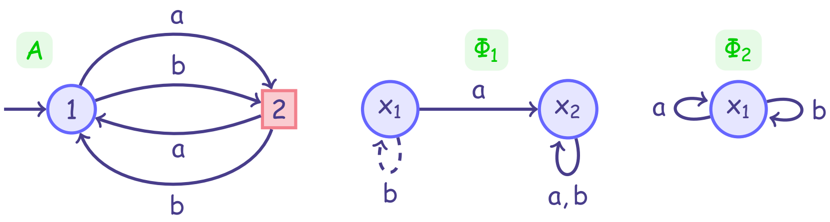
Comparison of Dynamic Filters



- Disabling/Hiding an action costs 1 per time unit (1 t.u. = step of A)
- On input word b^n
  - cost of Φ_1 is n
  - cost of Φ_2 is 0
- Φ_2 is **better** than Φ_1

Need to define a cost measure for dynamic filters

Comparison of Dynamic Filters



- Disabling/Hiding an action costs 1 per time unit (1 t.u. = step of A)
- On input word b^n
  - cost of Φ_1 is n
  - cost of Φ_2 is 0
- Φ_2 is better than Φ_1

Need to define a cost measure for dynamic filters

Computing the Cost of a Given Filter

A run of A: $\rho = q_0 \xrightarrow{\lambda_1} q_1 \quad \cdots \quad q_{n-1} \xrightarrow{\lambda_n} q_n$

$\Phi = (X, x_0, \Sigma, \delta_o, L)$ a filter, and $x_i = \delta_o(x_0, w_i)$ with $w_i = \lambda_1 \cdots \lambda_i$

$C : 2^\Sigma \longrightarrow \mathbb{N}$: Cost of hiding a subset of Σ

Average Cost on a run ρ

$$Cost(\rho, \Phi) = \frac{Cost(\rho)}{|\rho| + 1} = \frac{\sum_{i=0..n} C(\Sigma \setminus L(x_i))}{n+1}.$$

Maximal Cost on runs of A of length n

$$Cost(n, A, \Phi) = \max\{\, Cost(\rho, \Phi) \text{ for } \rho \in Runs^n(A) \,\}.$$

Cost of a pair (A, Φ)

$$Cost(A, \Phi) = \limsup_{n \to \infty} Cost(n, A, \Phi)$$

Theorem

For finite state filters, $Cost(A, \Phi)$ can be computed in PTIME.

Use Karp's Maximum Mean-weight Cycle Algorithm [Karp, 1978]

Computing the Cost of a Given Filter

A run of A: $\rho = q_0 \xrightarrow{\lambda_1} q_1 \quad \cdots \quad q_{n-1} \xrightarrow{\lambda_n} q_n$

$\Phi = (X, x_0, \Sigma, \delta_o, L)$ a filter, and $x_i = \delta_o(x_0, w_i)$ with $w_i = \lambda_1 \cdots \lambda_i$

$C : 2^\Sigma \longrightarrow \mathbb{N}$: Cost of hiding a subset of Σ

Average Cost on a run ρ

$$\mathrm{Cost}(\rho, \Phi) = \frac{\mathrm{Cost}(\rho)}{|\rho| + 1} = \frac{\sum_{i=0..n} C(\Sigma \setminus L(x_i))}{n+1}.$$

Maximal Cost on runs of A of length n

$$\mathrm{Cost}(n, A, \Phi) = \max\{\, \mathrm{Cost}(\rho, \Phi) \text{ for } \rho \in \mathrm{Runs}^n(A) \,\}.$$

Cost of a pair (A, Φ)

$$\mathrm{Cost}(A, \Phi) = \limsup_{n \to \infty} \mathrm{Cost}(n, A, \Phi)$$

Theorem

For finite state filters, $\mathrm{Cost}(A, \Phi)$ can be computed in PTIME.

Use Karp's Maximum Mean-weight Cycle Algorithm [Karp, 1978]

Computing the Cost of a Given Filter

A run of A: $\rho = q_0 \xrightarrow{\lambda_1} q_1 \quad \cdots \quad q_{n-1} \xrightarrow{\lambda_n} q_n$

$\Phi = (X, x_0, \Sigma, \delta_o, L)$ a filter, and $x_i = \delta_o(x_0, w_i)$ with $w_i = \lambda_1 \cdots \lambda_i$

$C : 2^\Sigma \longrightarrow \mathbb{N}$: Cost of hiding a subset of Σ

Average Cost on a run ρ

$$Cost(\rho, \Phi) = \frac{Cost(\rho)}{|\rho| + 1} = \frac{\sum_{i=0..n} C(\Sigma \setminus L(x_i))}{n+1}.$$

Maximal Cost on runs of A of length n

$$Cost(n, A, \Phi) = \max\{\, Cost(\rho, \Phi) \text{ for } \rho \in Runs^n(A) \,\}.$$

Cost of a pair (A, Φ)

$$Cost(A, \Phi) = \limsup_{n \to \infty} Cost(n, A, \Phi)$$

Theorem

For finite state filters, $Cost(A, \Phi)$ can be computed in PTIME.

Use Karp's Maximum Mean-weight Cycle Algorithm [Karp, 1978]

Computing the Cost of a Given Filter

A run of A: $\rho = q_0 \xrightarrow{\lambda_1} q_1 \quad \cdots \quad q_{n-1} \xrightarrow{\lambda_n} q_n$

$\Phi = (X, x_0, \Sigma, \delta_o, L)$ a filter, and $x_i = \delta_o(x_0, w_i)$ with $w_i = \lambda_1 \cdots \lambda_i$

$C : 2^\Sigma \longrightarrow \mathbb{N}$: Cost of hiding a subset of Σ

Average Cost on a run ρ

$$Cost(\rho, \Phi) = \frac{Cost(\rho)}{|\rho| + 1} = \frac{\sum_{i=0..n} C(\Sigma \setminus L(x_i))}{n + 1}.$$

Maximal Cost on runs of A of length n

$$Cost(n, A, \Phi) = \max\{\, Cost(\rho, \Phi) \text{ for } \rho \in Runs^n(A) \,\}.$$

Cost of a pair (A, Φ)

$$Cost(A, \Phi) = \limsup_{n \to \infty} Cost(n, A, \Phi)$$

Theorem

For finite state filters, $Cost(A, \Phi)$ can be computed in PTIME.

Use Karp's Maximum Mean-weight Cycle Algorithm [Karp, 1978]

Computing the Cost of a Given Filter

A run of A: $\rho = q_0 \xrightarrow{\lambda_1} q_1 \quad \cdots \quad q_{n-1} \xrightarrow{\lambda_n} q_n$

$\Phi = (X, x_0, \Sigma, \delta_o, L)$ a filter, and $x_i = \delta_o(x_0, w_i)$ with $w_i = \lambda_1 \cdots \lambda_i$

$C : 2^\Sigma \longrightarrow \mathbb{N}$: Cost of hiding a subset of Σ

Average Cost on a run ρ

$$Cost(\rho, \Phi) = \frac{Cost(\rho)}{|\rho| + 1} = \frac{\sum_{i=0..n} C(\Sigma \setminus L(x_i))}{n + 1}.$$

Maximal Cost on runs of A of length n

$$Cost(n, A, \Phi) = \max\{\, Cost(\rho, \Phi) \text{ for } \rho \in Runs^n(A) \,\}.$$

Cost of a pair (A, Φ)

$$Cost(A, \Phi) = \limsup_{n \longrightarrow \infty} Cost(n, A, \Phi)$$

Theorem

For finite state filters, $Cost(A, \Phi)$ can be computed in PTIME.

Use Karp's Maximum Mean-weight Cycle Algorithm [Karp, 1978]

Bounded Cost Filter

Problem 4: Bounded Cost Filter

Inputs: a NDA $A = (Q, q_0, \Sigma, \delta, F)$ and an integer $k \in \mathbb{N}$.
Problems: Assume F is opaque w.r.t. $(A, \emptyset)$.

(A) Is there any Φ s.t. F is opaque w.r.t. (A, Φ) and $Cost(A, \Phi) \leq k$?

(B) If the answer to (A) is "yes", compute a witness filter.

Steps to solve Problem 4

- Step 1: compute the most permissive filter MPΦ see Problem 5 next
- Step 2: check wether some filter in MPΦ costs less than k

Theorem

There is finite state most permissive filter (EXPTIME) for A.

Theorem

Problems 4.(A) and 4.(B) can be solved in EXPTIME.

Bounded Cost Filter

Problem 4: Bounded Cost Filter

Inputs: a NDA $A = (Q, q_0, \Sigma, \delta, F)$ and an integer $k \in \mathbb{N}$.
Problems: Assume F is opaque w.r.t. $(A, \varnothing)$.

(A) Is there any Φ s.t. F is opaque w.r.t. (A, Φ) and $Cost(A, \Phi) \leq k$?

(B) If the answer to (A) is "yes", compute a witness filter.

Steps to solve Problem 4

- Step 1: compute the most permissive filter $MP\Phi$ see Problem 5 next
- Step 2: check wether some filter in $MP\Phi$ costs less than k

Theorem

There is finite state most permissive filter (EXPTIME) for A.

Theorem

Problems 4.(A) and 4.(B) can be solved in EXPTIME.

Bounded Cost Filter

Problem 4: Bounded Cost Filter

Inputs: a NDA $A = (Q, q_0, \Sigma, \delta, F)$ and an integer $k \in \mathbb{N}$.
Problems: Assume F is opaque w.r.t. $(A, \varnothing)$.

(A) Is there any Φ s.t. F is opaque w.r.t. (A, Φ) and $Cost(A, \Phi) \leq k$?

(B) If the answer to (A) is "yes", compute a witness filter.

Steps to solve Problem 4

- Step 1: compute the most permissive filter $MP\Phi$ see Problem 5 next
- Step 2: check wether some filter in $MP\Phi$ costs less than k

Theorem

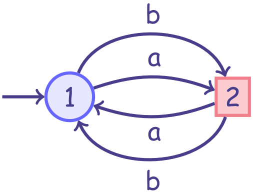
There is finite state most permissive filter (EXPTIME) for A.

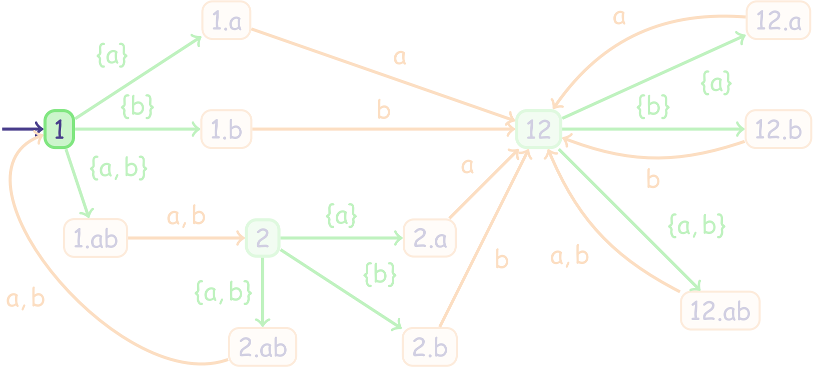
Theorem

Problems 4.(A) and 4.(B) can be solved in EXPTIME.

Problem 5 as a Game Problem

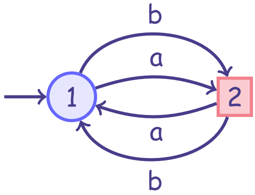
- Reduce Problem 5 to a safety 2-player game
- Player 1 chooses what to hide
- Player 2 tries to observe F

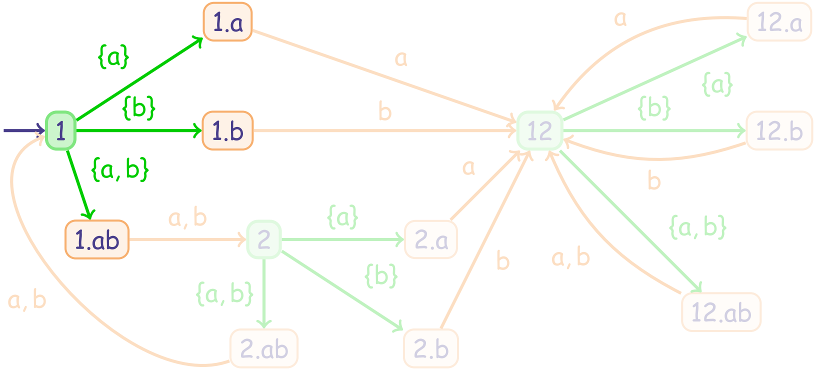




Problem 5 as a Game Problem

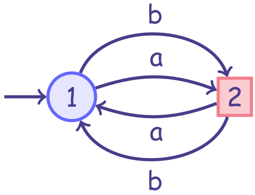
- Reduce Problem 5 to a safety 2-player game
- Player 1 chooses what to hide
- Player 2 tries to observe F

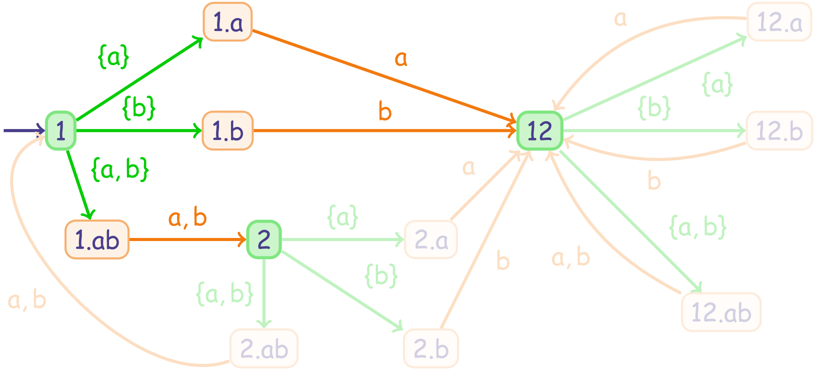




Problem 5 as a Game Problem

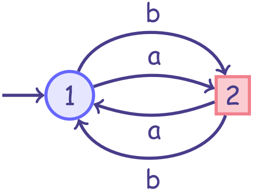
- Reduce Problem 5 to a safety 2-player game
- Player 1 chooses what to hide
- Player 2 tries to observe F

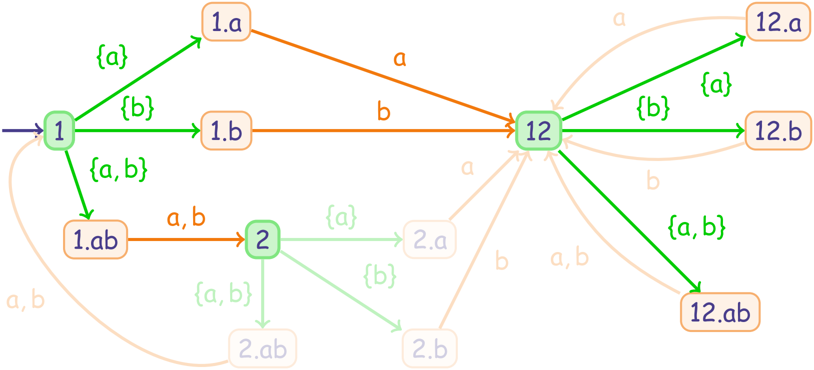




Problem 5 as a Game Problem

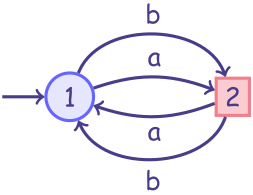
- Reduce Problem 5 to a safety 2-player game
- Player 1 chooses what to hide
- Player 2 tries to observe F

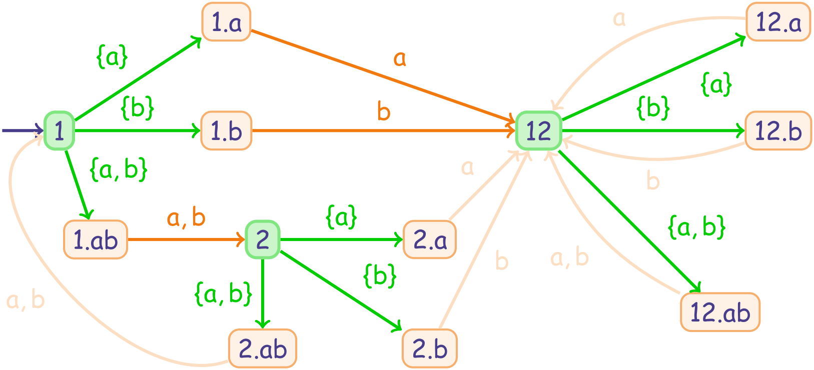




Problem 5 as a Game Problem

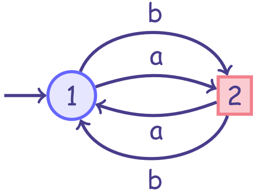
- Reduce Problem 5 to a safety 2-player game
- Player 1 chooses what to hide
- Player 2 tries to observe F

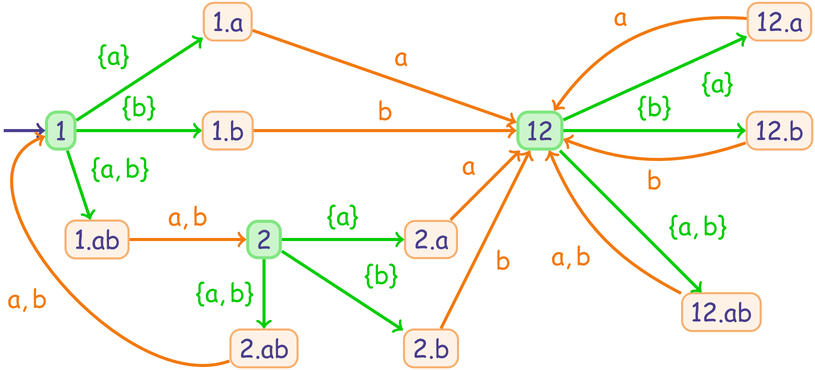




Problem 5 as a Game Problem

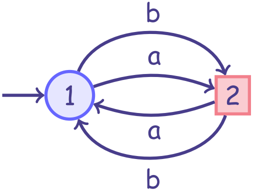
- Reduce Problem 5 to a safety 2-player game
- Player 1 chooses what to hide
- Player 2 tries to observe F

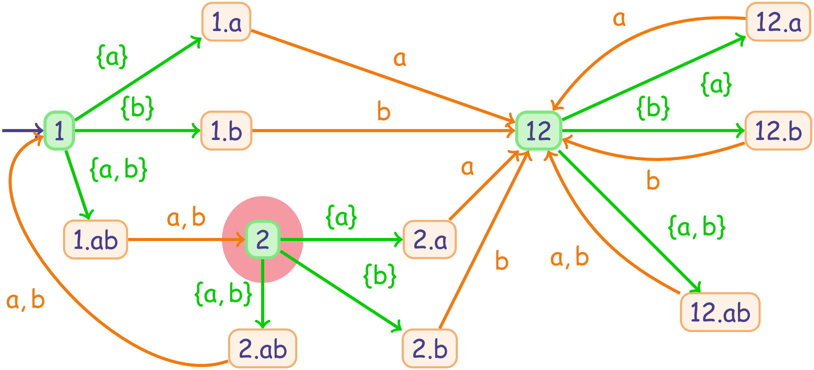




Problem 5 as a Game Problem

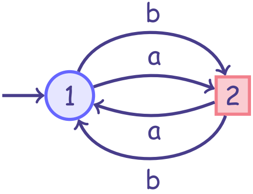
- Reduce Problem 5 to a safety 2-player game
- Player 1 chooses what to hide
- Player 2 tries to observe F

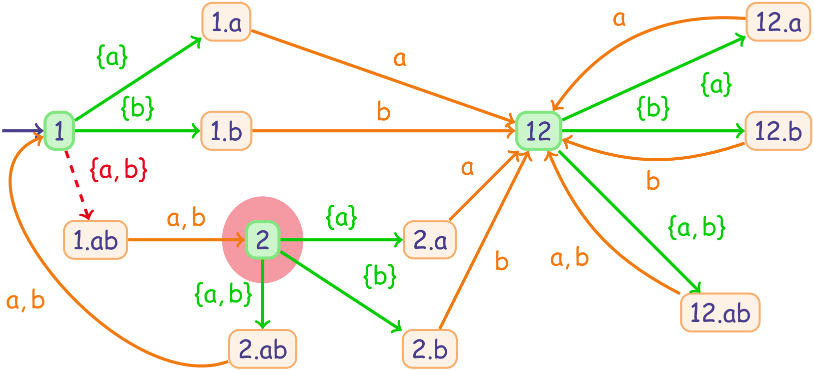




Problem 5 as a Game Problem

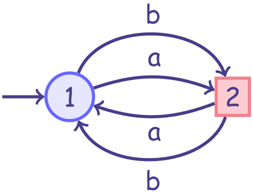
- Reduce Problem 5 to a safety 2-player game
- Player 1 chooses what to hide
- Player 2 tries to observe F

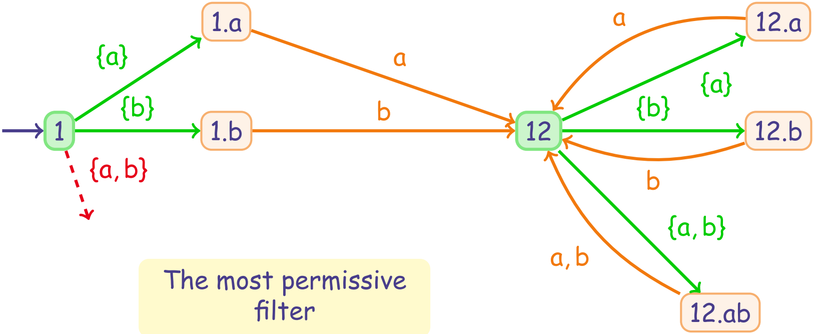




Problem 5 as a Game Problem

- Reduce Problem 5 to a safety 2-player game
- Player 1 chooses what to hide
- Player 2 tries to observe F





The most permissive filter

Results for Problem 5

Let $G(A, \Sigma)$ be the game defined previously.

Theorem

- if Φ is a filter s.t. F is opaque w.r.t. (A, Φ) then there is a corresponding winning strategy $f(\Phi)$ for Player 1 in $G(A, \Sigma)$
- if f is a winning strategy for Player 1 in $G(A, \Sigma)$, there is a corresponding filter $\Phi(f)$ s.t. F is opaque w.r.t. $(A, \Phi(f))$

Known Result:
There is a memoryless most permissive strategy for any safety game.

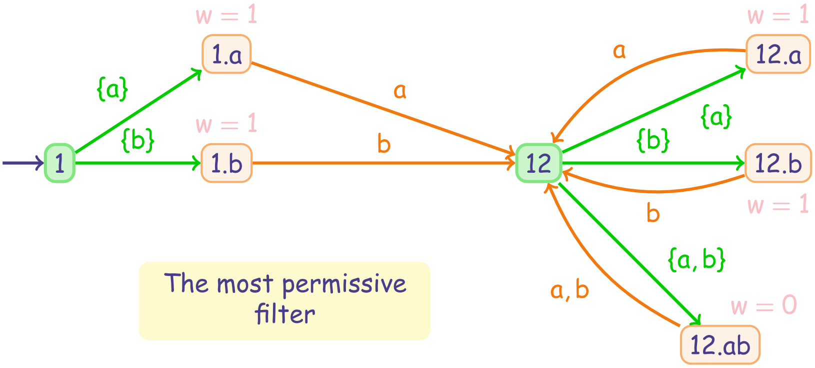
Theorem

There is a finite memory (EXPTIME) most permissive filter $MP\Phi$ for A.

Proof.

$G(A, \Sigma)$ has size exponential in A, Σ.
Solving safety games can be done in linear time. □

Optimal Dynamic Filter

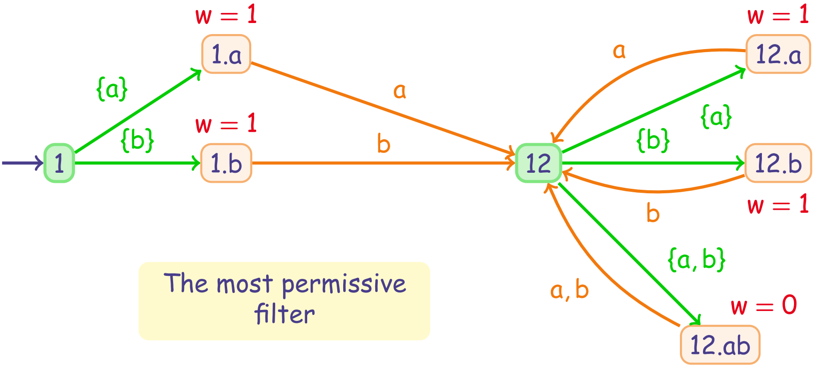


- Player 1 chooses what to hide: strategy f
- Player 2 chooses an action
- Add weight on Player 1's choices
- Player 1 playing f and Player 2 produce weighted runs $w(\rho)$ and

$$\mathbf{Cost}(\rho, f) = w(\rho)/(|\rho| + 1)$$

- Goal for Player 1: minimize $\limsup_{n\to\infty}\{\, w(\rho)/(|\rho| + 1) \mid \rho \in Runs^n(A)\,\}$

Optimal Dynamic Filter

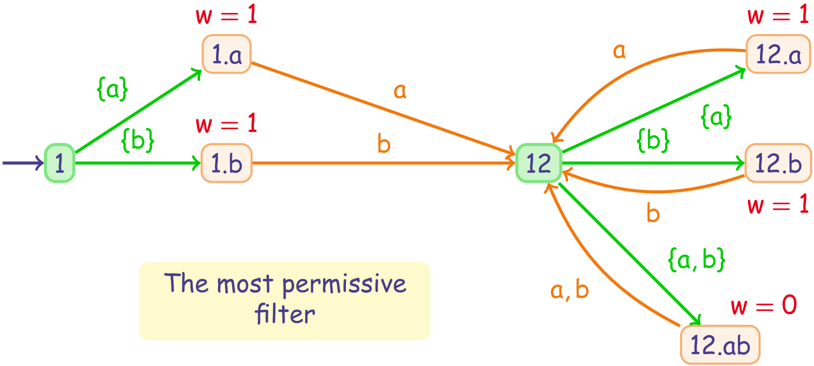


- Player 1 chooses what to hide: strategy f
- Player 2 chooses an action
- Add weight on Player 1's choices
- Player 1 playing f and Player 2 produce weighted runs $w(\rho)$ and

$$\mathbf{Cost}(\rho, f) = w(\rho)/(|\rho| + 1)$$

- Goal for Player 1: minimize $\limsup_{n \to \infty} \{ w(\rho)/(|\rho| + 1) \mid \rho \in Runs^n(A) \}$

Optimal Dynamic Filter

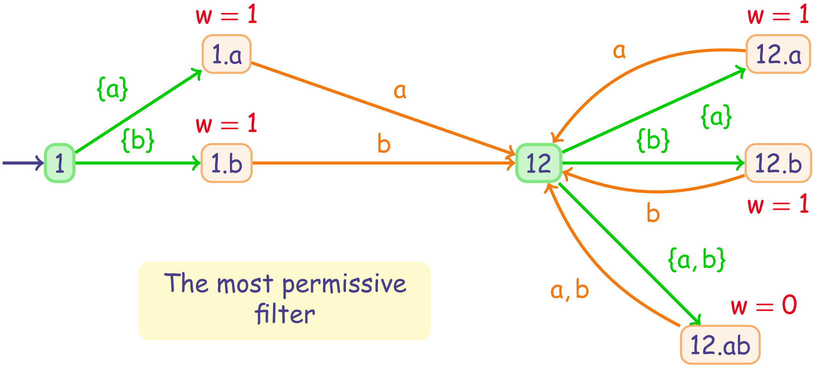


- Player 1 chooses what to hide: strategy f
- Player 2 chooses an action
- Add weight on Player 1's choices
- Player 1 playing f and Player 2 produce weighted runs $w(\rho)$ and

$$\mathbf{Cost}(\rho, f) = w(\rho)/(|\rho| + 1)$$

- Goal for Player 1: minimize $\limsup_{n\to\infty}\{ w(\rho)/(|\rho| + 1) \mid \rho \in Runs^n(A) \}$

Optimal Dynamic Filter



- Player 1 chooses what to hide: strategy f
- Player 2 chooses an action
- Add weight on Player 1's choices
- Player 1 playing f and Player 2 produce weighted runs $w(\rho)$ and

$$\mathbf{Cost}(\rho, f) = w(\rho)/(|\rho| + 1)$$

- Goal for Player 1: minimize $\limsup_{n\to\infty}\{\, w(\rho)/(|\rho| + 1) \mid \rho \in \mathrm{Runs}^n(A) \,\}$

Mean Payoff Games

[Zwick & Paterson, 1996]

Weighted two-player games

- Each state s has a weight $w(s)$
  alternatively: weight on edges
- Turn-based game
- Goal of the Players:
  - Player 1: minimize $l_0 = \limsup w(\rho)/(|\rho| + 1)$
  - Player 2: maximize $l_1 = \liminf w(\rho)/(|\rho| + 1)$

Results for weighted two-player games

[Zwick & Paterson, 1996]

- There is a value $v \in \mathbb{Q}$ s.t. each player has a memoryless strategy to ensure $l_0 \leq v$ and $l_1 \geq v$
- v can be effectively computed (PTIME)
- Memoryless strategies for both players can be effectively computed

v-Winning Strategy for Player 1 = Optimal filter

Mean Payoff Games [Zwick & Paterson, 1996]

Weighted two-player games

- Each state s has a **weight** $w(s)$
  alternatively: weight on edges
- **Turn-based** game
- Goal of the Players:
  - Player 1: **minimize** $l_0 = \limsup w(\rho)/(|\rho| + 1)$
  - Player 2: **maximize** $l_1 = \liminf w(\rho)/(|\rho| + 1)$

Results for weighted two-player games [Zwick & Paterson, 1996]

- There is a value $v \in \mathbb{Q}$ s.t. each player has a **memoryless** strategy to **ensure** $l_0 \leq v$ and $l_1 \geq v$
- v can be **effectively** computed (PTIME)
- **Memoryless** strategies for both players can be **effectively** computed

v-Winning Strategy for Player 1 = Optimal filter

Mean Payoff Games [Zwick & Paterson, 1996]

Weighted two-player games

- Each state s has a **weight** $w(s)$
  alternatively: weight on edges
- **Turn-based** game
- Goal of the Players:
  - Player 1: **minimize** $l_0 = \limsup w(\rho)/(|\rho| + 1)$
  - Player 2: **maximize** $l_1 = \liminf w(\rho)/(|\rho| + 1)$

Results for weighted two-player games [Zwick & Paterson, 1996]

- There is a value $v \in \mathbb{Q}$ s.t. each player has a **memoryless** strategy to **ensure** $l_0 \leq v$ and $l_1 \geq v$
- v can be **effectively** computed (PTIME)
- **Memoryless** strategies for both players can be **effectively** computed

v-Winning Strategy for Player 1 = Optimal filter

Results for Problem 4

Problem 4: Bounded Cost Filter

Inputs: a NDA $A = (Q, q_0, \Sigma, \delta, F)$ and an integer $k \in \mathbb{N}$.
Problems: Assume F is opaque w.r.t. $(A, \varnothing)$.

(A) Is there any Φ s.t. F is opaque w.r.t. (A, Φ) and $Cost(A, \Phi) \leq k$?

(B) If the answer to (A) is "yes", compute a witness filter.

Solution for Problem 4

1. Compute the most permissive filter $MP\Phi$
2. Build a weighted graph game: $MP\Phi \times A$
3. Use Zwick & Paterson's algorithm to compute the value of the game
4. Compare k to the value of the game

Theorem

Problem 4 can be solved in EXPTIME.
An optimal filter can be computed in EXPTIME.

Results for Problem 4

Problem 4: Bounded Cost Filter

Inputs: a NDA $A = (Q, q_0, \Sigma, \delta, F)$ and an integer $k \in \mathbb{N}$.
Problems: Assume F is opaque w.r.t. $(A, \varnothing)$.

(A) Is there any Φ s.t. F is opaque w.r.t. (A, Φ) and $\mathrm{Cost}(A, \Phi) \leq k$?

(B) If the answer to (A) is "yes", compute a witness filter.

Solution for Problem 4

1. Compute the most permissive filter $MP\Phi$
2. Build a weighted graph game: $MP\Phi \times A$
3. Use Zwick & Paterson's algorithm to compute the value of the game
4. Compare k to the value of the game

Theorem

Problem 4 can be solved in EXPTIME.
An optimal filter can be computed in EXPTIME.

Results for Problem 4

Problem 4: Bounded Cost Filter

Inputs: a NDA $A = (Q, q_0, \Sigma, \delta, F)$ and an integer $k \in \mathbb{N}$.
Problems: Assume F is opaque w.r.t. $(A, \varnothing)$.

(A) Is there any Φ s.t. F is opaque w.r.t. (A, Φ) and $Cost(A, \Phi) \leq k$?

(B) If the answer to (A) is "yes", compute a witness filter.

Solution for Problem 4

1. Compute the most permissive filter $MP\Phi$
2. Build a weighted graph game: $MP\Phi \times A$
3. Use Zwick & Paterson's algorithm to compute the value of the game
4. Compare k to the value of the game

Theorem

Problem 4 can be solved in EXPTIME.
An optimal filter can be computed in EXPTIME.

Results for Problem 4

Problem 4: Bounded Cost Filter

Inputs: a NDA $A = (Q, q_0, \Sigma, \delta, F)$ and an integer $k \in \mathbb{N}$.
Problems: Assume F is opaque w.r.t. $(A, \varnothing)$.

(A) Is there any Φ s.t. F is opaque w.r.t. (A, Φ) and $Cost(A, \Phi) \leq k$?

(B) If the answer to (A) is "yes", compute a witness filter.

Solution for Problem 4

1. Compute the most permissive filter $MP\Phi$
2. Build a weighted graph game: $MP\Phi \times A$
3. Use Zwick & Paterson's algorithm to compute the value of the game
4. Compare k to the value of the game

Theorem

Problem 4 can be solved in EXPTIME.
An optimal filter can be computed in EXPTIME.

Outline

Results and Future Work

Summary of the Results

- Opacity with dynamic filters
  Secret can also be given by a regular language
- Cost & computation of the cost of a dynamic filter
- Existence & computation of the most permissive filter
- Existence of a finite optimal dynamic observer
- Effective computation of the optimal dynamic observer
- Extended version in [CDM, Tech. Rep., 2009]

Future Work

- Exact complexity of Problem 4 (EXPTIME-hardness)
- Extend to masks (renaming of events)
- Add new constraints to increase the Quality of Services
  e.g. availability properties
- Implement the algorithms

Results and Future Work

Summary of the Results

- Opacity with dynamic filters
  Secret can also be given by a regular language
- Cost & computation of the cost of a dynamic filter
- Existence & computation of the most permissive filter
- Existence of a finite optimal dynamic observer
- Effective computation of the optimal dynamic observer
- Extended version in [CDM, Tech. Rep., 2009]

Future Work

- Exact complexity of Problem 4 (EXPTIME-hardness)
- Extend to masks (renaming of events)
- Add new constraints to increase the Quality of Services
  e.g. availability properties
- Implement the algorithms

Some References

[Bryans et al., 2008] Bryans, J., Koutny, M., Mazaré, L., & Ryan, P. 2008.
Opacity generalised to transition systems.
International Journal of Information Security, 7(6), 421–435.

[CDM, Tech. Rep., 2009] Cassez, Franck, Dubreil, Jérémy, & Marchand, Hervé. 2009 (May).
Dynamic Observers for the Synthesis of Opaque Systems.
Tech. rept. 1930. IRISA.
available at http://www.irisa.fr/prive/hmarchand/rr-observer.pdf.

[Dubreil et al., 2008] Dubreil, Jérémy, Darondeau, Philippe, & Marchand, Hervé. 2008 (May).
Opacity Enforcing Control Synthesis.
Pages 28–35 of: Proceedings of the 9th International Workshop on Discrete Event Systems (WODES'08).

[Karp, 1978] Karp, Richard M. 1978.
A characterization of the minimum mean cycle in a digraph.
Discrete Mathematics, 23, 309–311.

[Mazaré, 2004] Mazaré, Laurent. 2004.
Using Unification for Opacity Properties.
Pages 165–176 of: Proceedings of the 4th IFIP WG1.7 Workshop on Issues in the Theory of Security (WITS'04).

[Zwick & Paterson, 1996] Zwick, U., & Paterson, M. 1996.
The complexity of mean payoff games on graphs.
Theoretical Computer Science, 158(1–2), 343–359.