The Opacity of Real-time Automata

Lingtai Wang[1,2] Naijun Zhan[1,2] **Jie An**[3]

State Key Lab. of Comp. Sci., Institute of Software, Chinese Academy of Sciences, China

University of Chinese Academy of Sciences, Beijing, China

School of Software Engineering, Tongji University, Shanghai, China

November 1, 2018

Overview

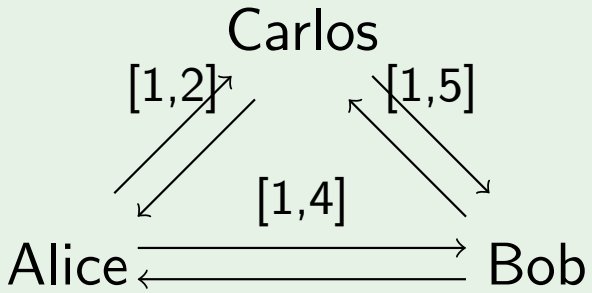
Opacity

Opacity is an information flow property aiming at keeping the secret of a system *opaque* to the intruder with partial observability over its behaviours.

Opacity

Opacity is an information flow property aiming at keeping the secret of a system *opaque* to the intruder with partial observability over its behaviours.

Example

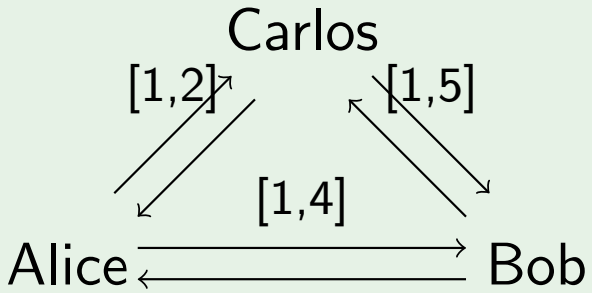


Eve (the intruder)
only knows Alice's and Bob's behaviours

Opacity

Opacity is an information flow property aiming at keeping the secret of a system *opaque* to the intruder with partial observability over its behaviours.

Example



Eve (the intruder)
only knows Alice's and Bob's behaviours

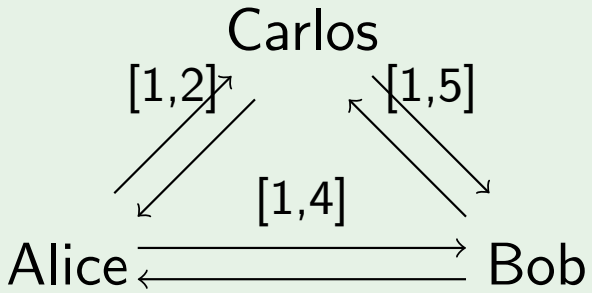
Can Eve make sure that Alice $\longrightarrow$ Bob has happened if she's observed ...

1. $\text{Alice} \rightarrow\ ;\ \rightarrow \text{Bob}$

Opacity

Opacity is an information flow property aiming at keeping the secret of a system *opaque* to the intruder with partial observability over its behaviours.

Example



Eve (the intruder)
only knows Alice's and Bob's behaviours

Can Eve make sure that Alice $\longrightarrow$ Bob has happened if she's observed ...

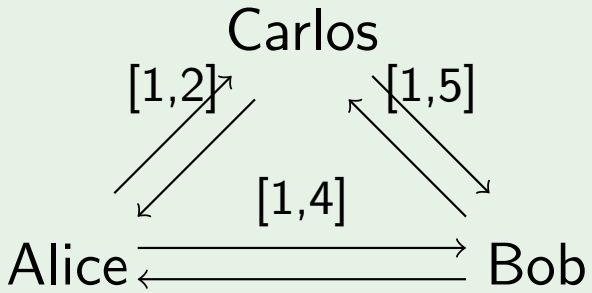
1. $\mathrm{Alice} \rightarrow\ ;\ \rightarrow \mathrm{Bob}$

(and if it takes more than 3 min for a message to transmit...)

Opacity

Opacity is an information flow property aiming at keeping the secret of a system *opaque* to the intruder with partial observability over its behaviours.

Example



Eve (the intruder)
only knows Alice's and Bob's behaviours

Can Eve make sure that Alice $\longrightarrow$ Bob has happened if she's observed ...

1. $\mathrm{Alice} \rightarrow\ ;\ \rightarrow \mathrm{Bob}$

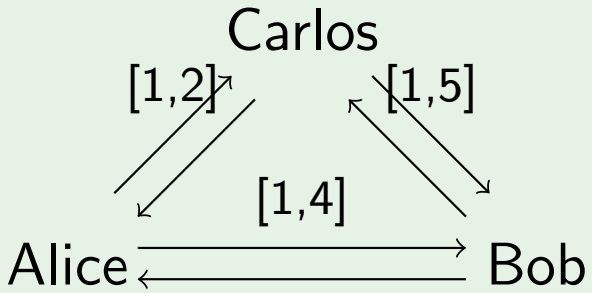
(and if it takes more than 3 min for a message to transmit...)

2.1 $\mathrm{Alice} \rightarrow\ @\ 10{:}00; \rightarrow \mathrm{Bob}\ @\ 10{:}03$

Opacity

Opacity is an information flow property aiming at keeping the secret of a system *opaque* to the intruder with partial observability over its behaviours.

Example



Eve (the intruder)
only knows Alice's and Bob's behaviours

Can Eve make sure that Alice $\longrightarrow$ Bob has happened if she's observed ...

1. $\mathrm{Alice} \rightarrow\ ;\ \rightarrow \mathrm{Bob}$

(and if it takes more than 3 min for a message to transmit...)

2.1 $\mathrm{Alice} \rightarrow\ @\ 10{:}00; \rightarrow \mathrm{Bob}\ @\ 10{:}03$

2.2 $\mathrm{Alice} \rightarrow\ @\ 10{:}00; \rightarrow \mathrm{Bob}\ @\ 10{:}06$

Opacity

The **system** has a secret: a subset of its states, a language, etc. As a result, some of its behaviours are secret.

Opacity

The **system** has a secret: a subset of its states, a language, etc. As a result, some of its behaviours are secret.
The **intruder** has partial observability, that is, projection of the behaviours.

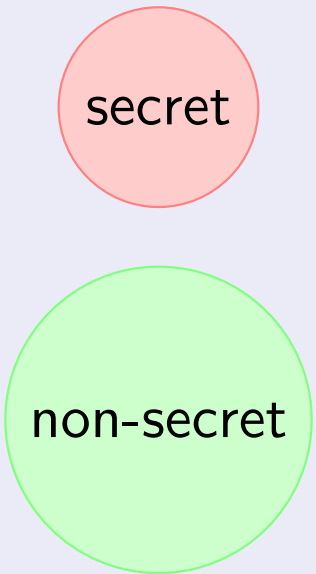
Opacity

The **system** has a secret: a subset of its states, a language, etc. As a result, some of its behaviours are secret.
The **intruder** has partial observability, that is, projection of the behaviours.
Can the intruder make sure the current behaviour is secret according to what he has seen?

Opacity

The **system** has a secret: a subset of its states, a language, etc. As a result, some of its behaviours are secret.
The **intruder** has partial observability, that is, projection of the behaviours.
Can the intruder make sure the current behaviour is secret according to what he has seen?

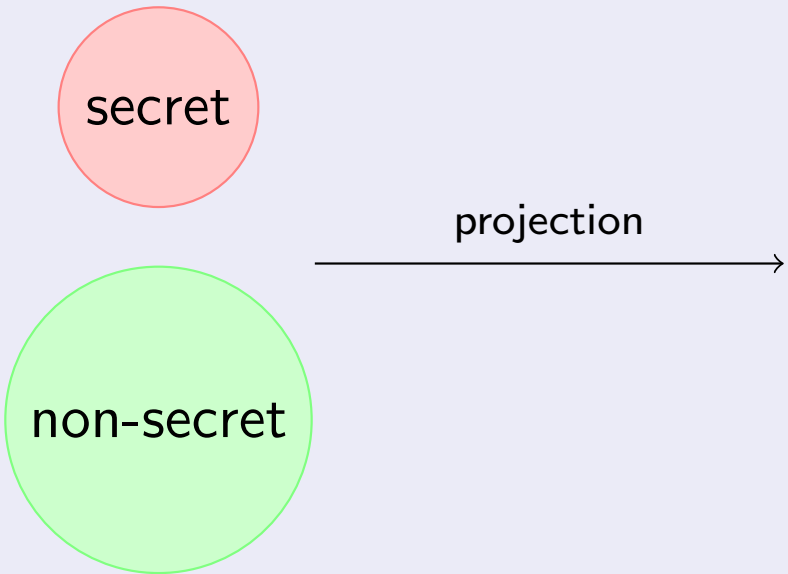
System's behaviours:



Opacity

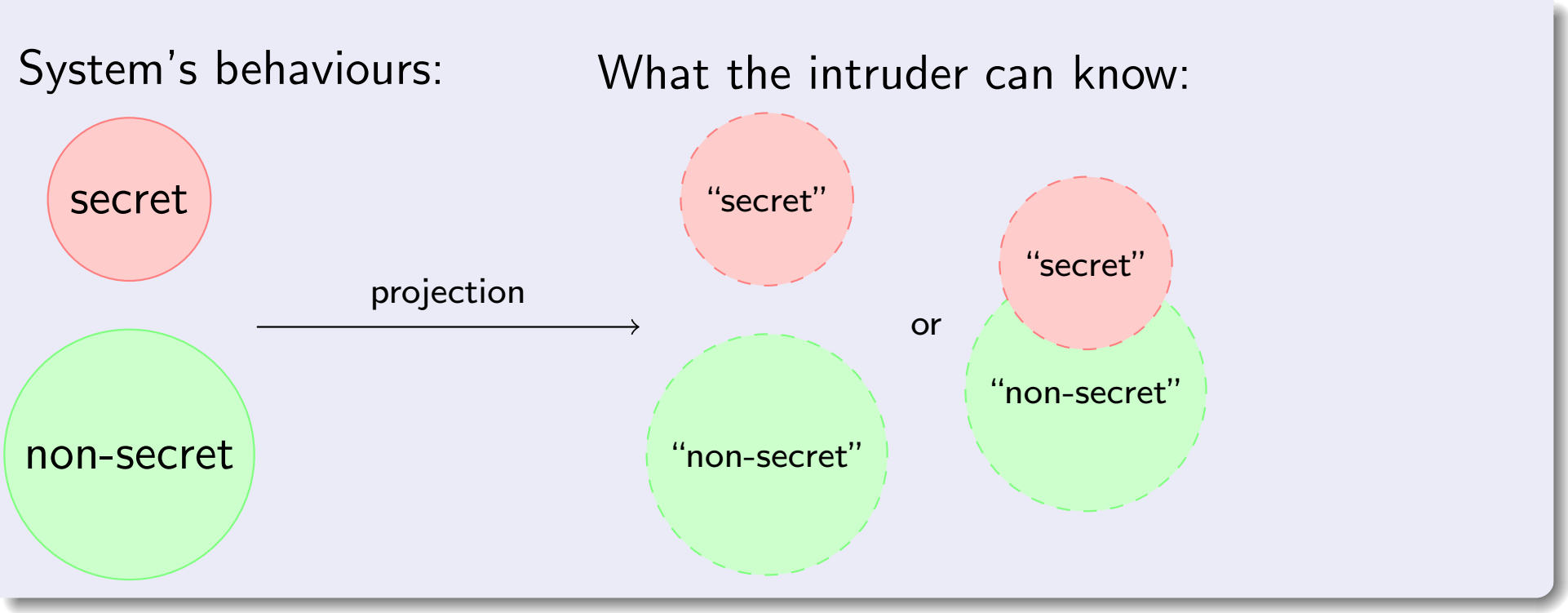
The **system** has a secret: a subset of its states, a language, etc. As a result, some of its behaviours are secret.
The **intruder** has partial observability, that is, projection of the behaviours.
Can the intruder make sure the current behaviour is secret according to what he has seen?

System's behaviours: What the intruder can know:



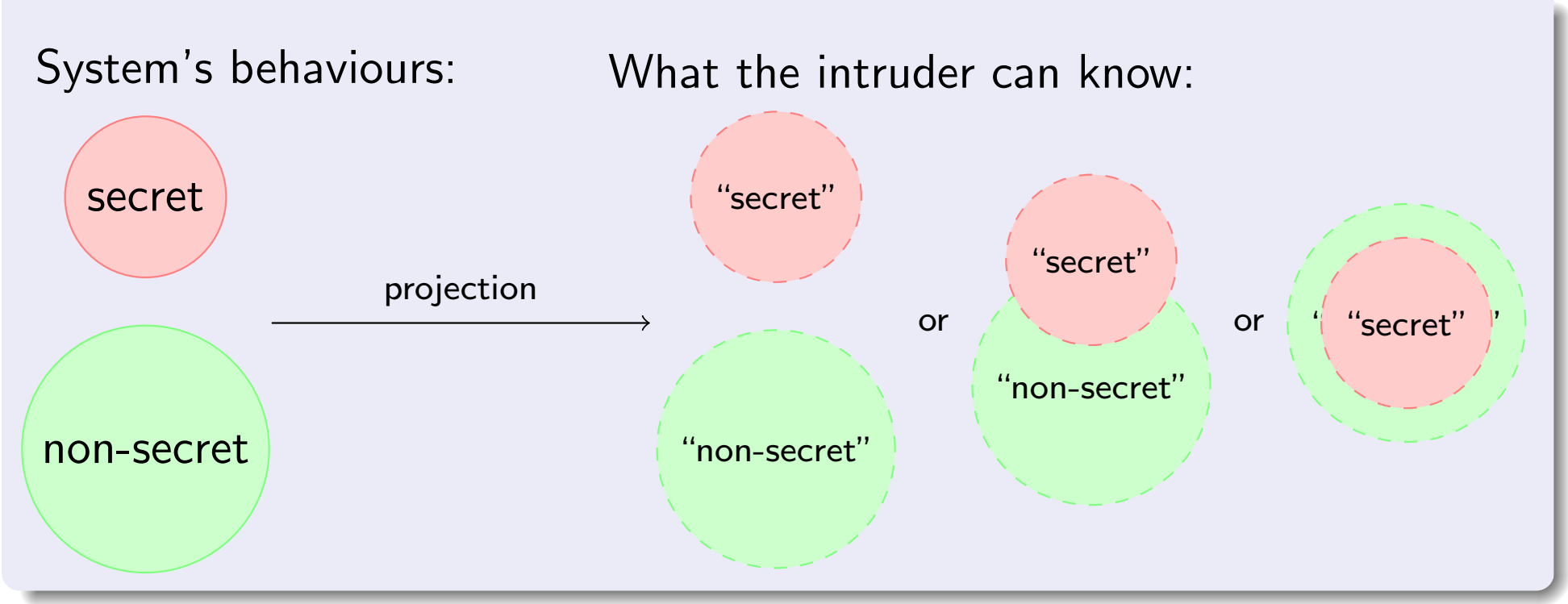
Opacity

The **system** has a secret: a subset of its states, a language, etc. As a result, some of its behaviours are secret.
The **intruder** has partial observability, that is, projection of the behaviours.
Can the intruder make sure the current behaviour is secret according to what he has seen?



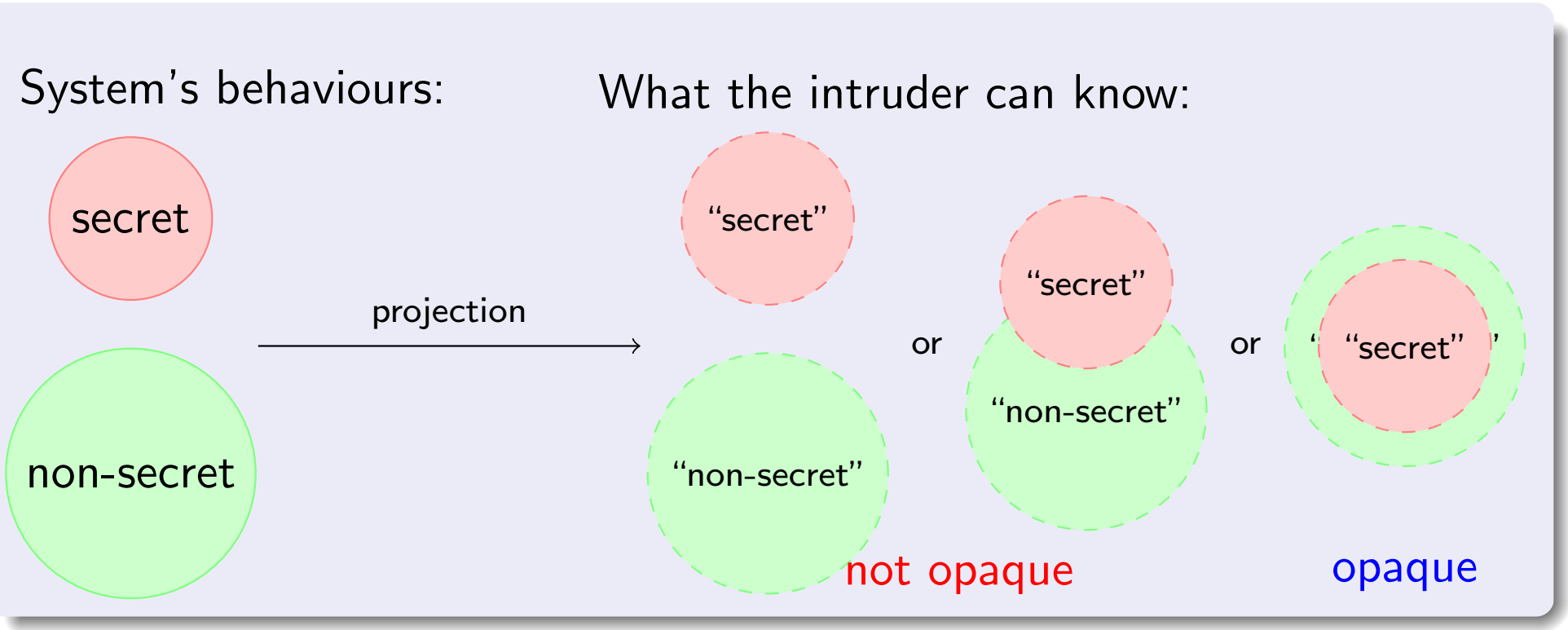
Opacity

The **system** has a secret: a subset of its states, a language, etc. As a result, some of its behaviours are secret.
The **intruder** has partial observability, that is, projection of the behaviours.
Can the intruder make sure the current behaviour is secret according to what he has seen?



Opacity

The **system** has a secret: a subset of its states, a language, etc. As a result, some of its behaviours are secret.
The **intruder** has partial observability, that is, projection of the behaviours.
Can the intruder make sure the current behaviour is secret according to what he has seen?



Real-time Automata (RTA)

Definition (Real-time Automata (RTA))

An RTA is a six-tuple $\mathcal{A} = (S, \Sigma, \Delta, \mathit{Init}, F, \mu)$ where

- S is a finite set of states; $\mathit{Init} \subset S$ is initial states; $F \subset S$ is accepting states;
- Σ is a finite alphabet;
- $\Delta \subset S \times \Sigma \times S$ is the transition relation;
- $\mu : \Delta \to 2^{\mathbb{R}_{\geq 0}} \setminus \{\emptyset\}$ is the time labeling function.

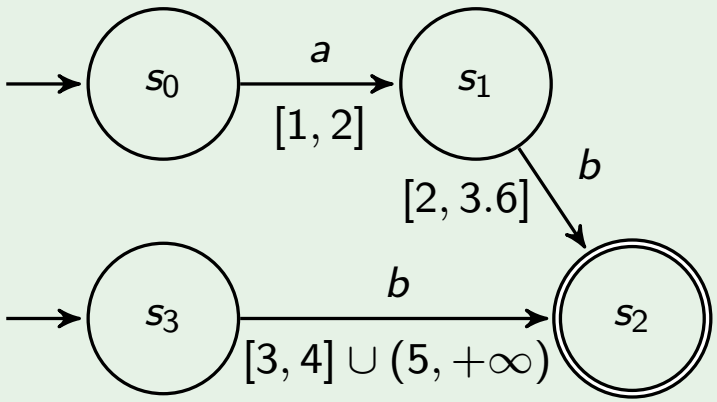
Real-time Automata (RTA)

Definition (Real-time Automata (RTA))

An RTA is a six-tuple $\mathcal{A} = (S, \Sigma, \Delta, \mathit{Init}, F, \mu)$ where

- S is a finite set of states; $\mathit{Init} \subset S$ is initial states; $F \subset S$ is accepting states;
- Σ is a finite alphabet;
- $\Delta \subset S \times \Sigma \times S$ is the transition relation;
- $\mu : \Delta \to 2^{\mathbb{R}_{\geq 0}} \setminus \{\emptyset\}$ is the time labeling function.

Example (RTA)

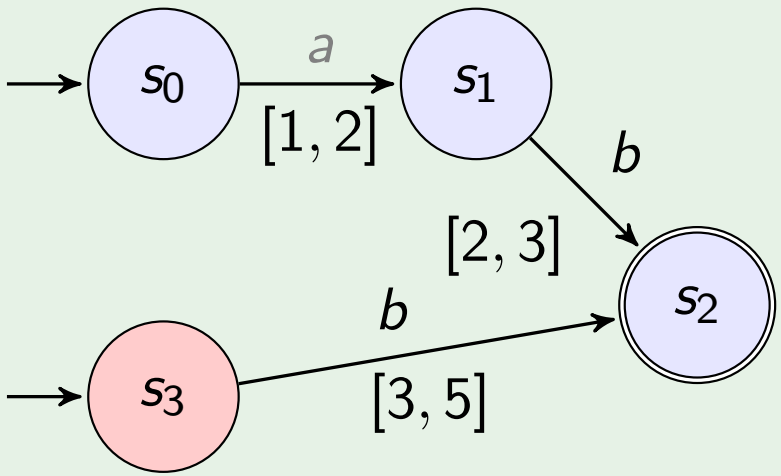


Initial-state Opacity & Language-Opacity of RTA

- **Initial-state Opacity Problem**: The secret is a *subset of its initial-states* of the RTA. Can the intruder never judge whether the run starts from a secret initial-state?

Example

- Let $\{b\}$ be the observable set and s_3 be the "secret" initial state.



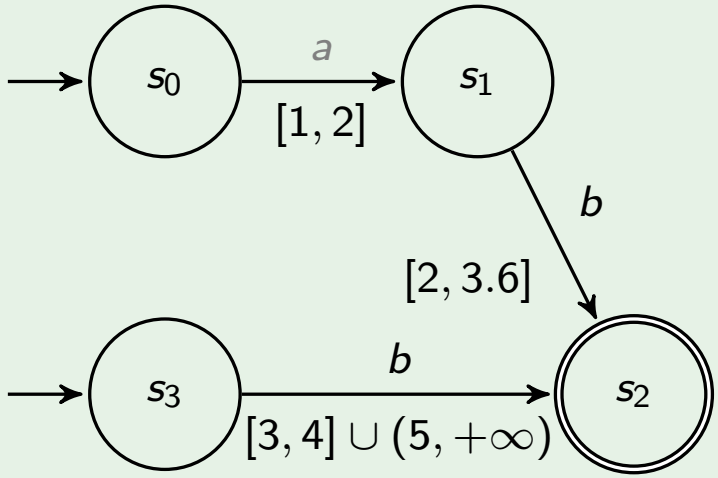
- If (b,t) is observed, possible runs include not only $\rho_0 = s_3 \xrightarrow[t]{b} s_2$, but also $\rho_1 = s_0 \xrightarrow[1]{a} s_1 \xrightarrow[t-1]{b} s_2$, $\cdots$, $\rho_2 = s_0 \xrightarrow[2]{a} s_1 \xrightarrow[t-2]{b} s_2$. So the RTA is initial-state opaque with respect to $\{s_3\}$ and $\{b\}$ in this example.

Initial-state Opacity & Language-Opacity of RTA

- **Language-Opacity Problem**: The secret is a *set of timed words*. Can the intruder never judge whether the run is in the secret set?

Example

The secret is $L_{secret} = \{(a, t_a) \cdot (b, t_b) \mid 1 \leq t_a \leq 3 \wedge 2 \leq t_b \leq 5 \wedge t_a \leq t_b\}$. Again a is unobservable and b is observable.



If $(b, 5)$ is observed, all possible runs are of the form $\rho = s_0 \xrightarrow[t_a]{a} s_1 \xrightarrow[5-t_a]{b} s_2$, where $1 \leq t_a \leq 2 \wedge 2 \leq 5 - t_a \leq 3.6$, i.e., the corresponding traces are $\{(a, t_a) \cdot (b, 5) \mid 1 \leq t_a \leq 1.4\} \subset L_{secret}$.

So the RTA is not language-opaque in this case.

Initial-state Opacity & Language-Opacity of RTA

Initial-state opacity of RTA

$\mathcal{A}$ is initial-state opaque with respect to S_{secret} and Σ_o iff $P_{\Sigma_o,t}(Traces\,(Init \cap S_{secret}))) \subseteq P_{\Sigma_o,t}(Traces\,(Init \setminus S_{secret}))$, equivalently, iff $P_{\Sigma_o,t}(L(\mathcal{A})) \subseteq P_{\Sigma_o,t}(Traces\,(Init \setminus S_{secret}))$.

Initial-state Opacity & Language-Opacity of RTA

Initial-state opacity of RTA

$\mathcal{A}$ is initial-state opaque with respect to S_{secret} and Σ_o iff $P_{\Sigma_o,t}(Traces\,(Init \cap S_{secret}))) \subseteq P_{\Sigma_o,t}(Traces\,(Init \setminus S_{secret}))$, equivalently, iff $P_{\Sigma_o,t}(L(\mathcal{A})) \subseteq P_{\Sigma_o,t}(Traces\,(Init \setminus S_{secret}))$.

Language-opacity of RTA

$\mathcal{A}$ is language-opaque with respect to L_{secret} and Σ_o iff $P_{\Sigma_o,t}(L(\mathcal{A}) \cap L_{secret}) \subseteq P_{\Sigma_o,t}(L(\mathcal{A}) \setminus L_{secret})$, equivalently, iff $P_{\Sigma_o,t}(L(\mathcal{A})) \subseteq P_{\Sigma_o,t}(L(\mathcal{A}) \setminus L_{secret})$.

Initial-state Opacity & Language-Opacity of RTA

Initial-state opacity of RTA

$\mathcal{A}$ is initial-state opaque with respect to S_{secret} and Σ_o iff $P_{\Sigma_o,t}(Traces\,(Init \cap S_{secret}))) \subseteq P_{\Sigma_o,t}(Traces\,(Init \setminus S_{secret}))$, equivalently, iff $P_{\Sigma_o,t}(L(\mathcal{A})) \subseteq P_{\Sigma_o,t}(Traces\,(Init \setminus S_{secret}))$.

Language-opacity of RTA

$\mathcal{A}$ is language-opaque with respect to L_{secret} and Σ_o iff $P_{\Sigma_o,t}(L(\mathcal{A}) \cap L_{secret}) \subseteq P_{\Sigma_o,t}(L(\mathcal{A}) \setminus L_{secret})$, equivalently, iff $P_{\Sigma_o,t}(L(\mathcal{A})) \subseteq P_{\Sigma_o,t}(L(\mathcal{A}) \setminus L_{secret})$.

Theorem (1)

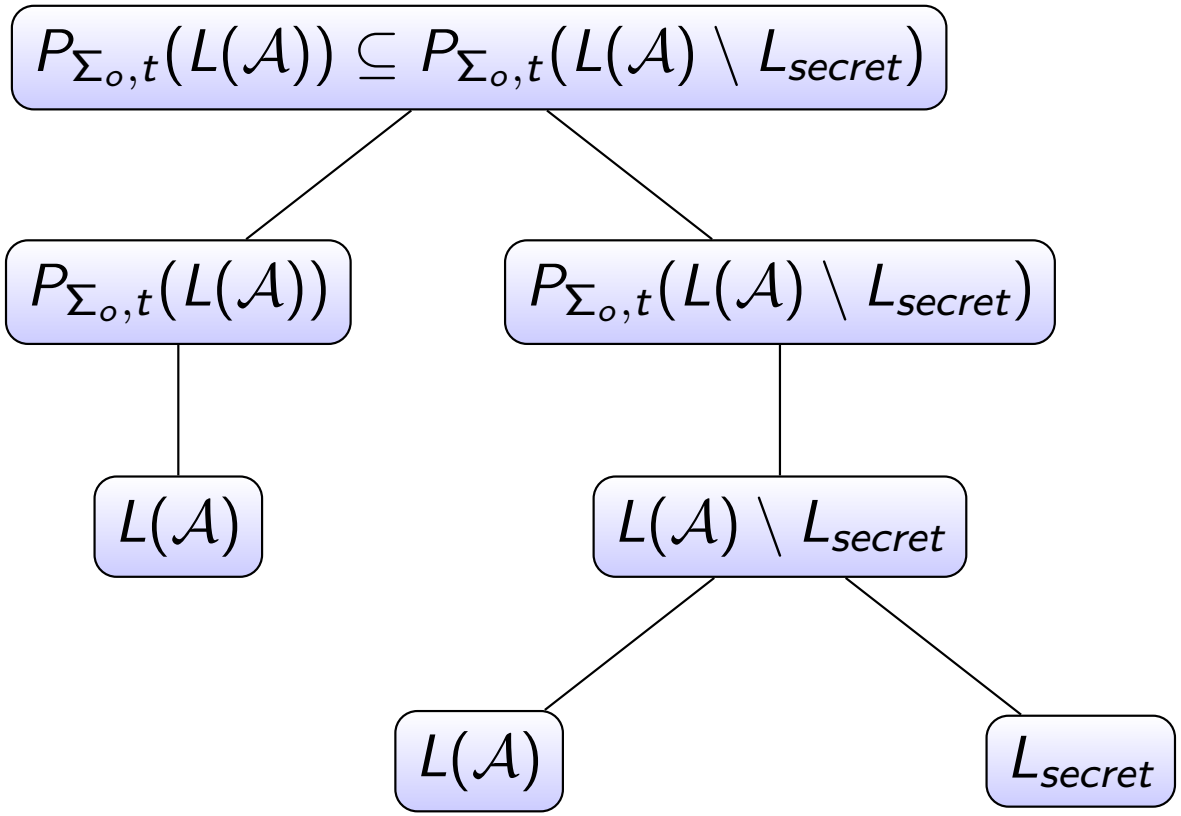
An RTA $\mathcal{A}$ is initial-state opaque with respect to S_{secret} and Σ_o iff it's language-opaque with respect to $L(\mathcal{A}) \setminus Traces\,(Init \setminus S_{secret})$ and Σ_o

Deciding the opacity of RTA

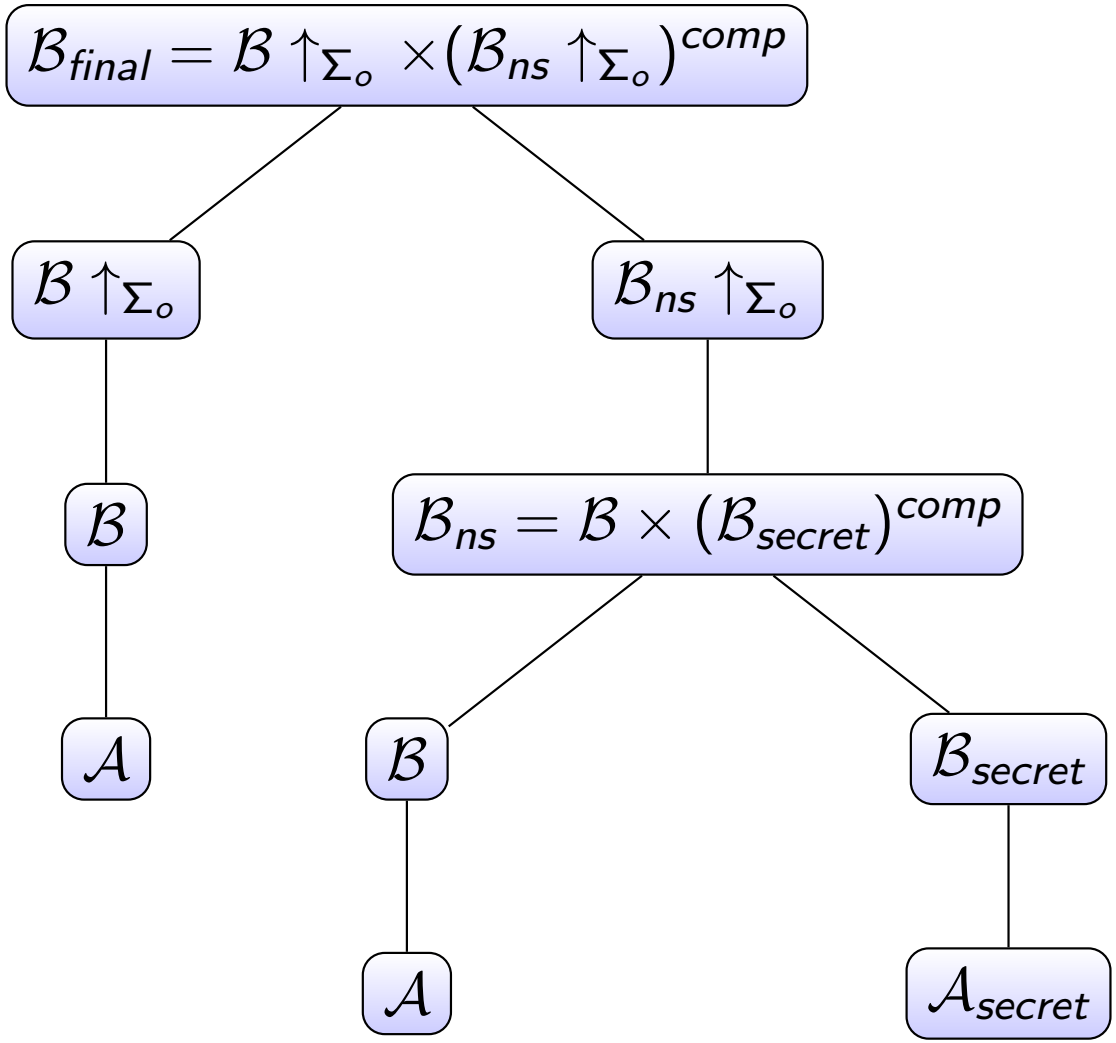
- With the **Theorem (1)**, we focus on language-opacity in the following.

Deciding the opacity of RTA

- With the **Theorem (1)**, we focus on language-opacity in the following.



The Whole Progress



Main methods

Translating RTA into FA

- the relation of trace-equivalence
- the partitioned alphabet and partitioned language

Computing projection of FA

- summation of successive unobservable durations
- merging unobservable durations into observable transitions' time labels

The Relation of Trace-Equivalence — $L_{FA} \approx_{tr} L_{RTA}$

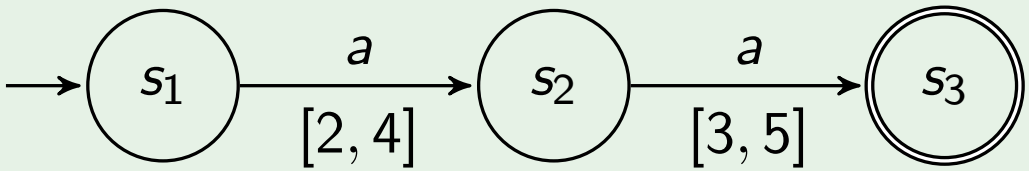
- The languages of the newly-obtained FA $\mathcal{B}$ and of the original RTA $\mathcal{A}$ should represent each other.

The Relation of Trace-Equivalence — $L_{FA} \approx_{tr} L_{RTA}$

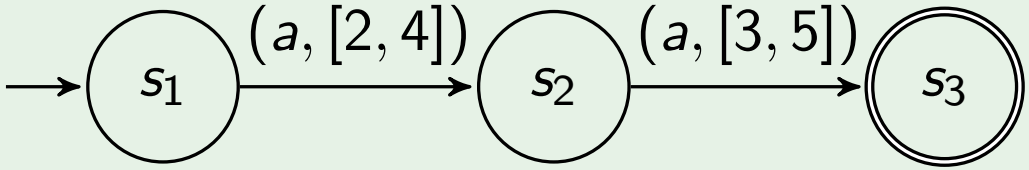
- The languages of the newly-obtained FA $\mathcal{B}$ and of the original RTA $\mathcal{A}$ should represent each other.

Example

$L_f(\mathcal{A}) = \{(a, t_1) \cdot (a, t_2) \mid t_1 \in [2, 4] \wedge t_2 - t_1 \in [3, 5]\}$



$L_f(\mathcal{B}) = \{(a, [2, 4]) \cdot (a, [3, 5])\}$



$L_f(\mathcal{B}) \approx_{tr} L_f(\mathcal{A})$

- However, the relation of trace-equivalence is not preserved when it comes to the complement and product operations of FA.

- However, the relation of trace-equivalence is not preserved when it comes to the complement and product operations of FA.

Example (Ctd.)

The complement of $L_f(\mathcal{A})$ is

$$\{\varepsilon\} \cup \{(a,t) \mid t \in [0,+\infty)\} \cup$$
$$\{(a,t_1)\cdot(a,t_2) \mid t_1 \notin [2,4] \vee t_2 - t_1 \notin [3,5]\} \cup \ldots$$

while the complement of $L_f(\mathcal{B})$ is

$$\{\varepsilon\} \cup \{(a,[2,4]),(a,[3,5])\} \cup$$
$$\{(a,[2,4])\cdot(a,[2,4]),(a,[3,5])\cdot(a,[3,5]),(a,[3,5])\cdot(a,[2,4])\} \cup \ldots$$

Not trace-equivalent at all!

Put restrictions on the timed alphabet

Therefore, some restrictions should be put on the timed alphabet:

Partitioned alphabets and partitioned languages

Let the timed alphabet be $\Sigma_t = \bigcup_{\sigma \in \Sigma} \{\sigma\} \times T_\sigma$.
An alphabeta Σ_t is called a **partitioned alphabet** if for each σ, T_σ is a partition of $\mathbb{R}_{\geq 0}$.
And language over partitioned alphabets are called **partitioned langauges**.

Put restrictions on the timed alphabet

Therefore, some restrictions should be put on the timed alphabet:

Partitioned alphabets and partitioned languages

Let the timed alphabet be $\Sigma_t = \bigcup_{\sigma \in \Sigma} \{\sigma\} \times T_\sigma$.
An alphabeta Σ_t is called a **partitioned alphabet** if for each σ, T_σ is a partition of $\mathbb{R}_{\geq 0}$.
And language over partitioned alphabets are called **partitioned langauges**.

Lemma (Trace-equivalence preserved under complement)

$(\Sigma_t^* \setminus L_2) \approx_{tr} (TW^*(\Sigma) \setminus L_1)$ if L_1 is a timed language over Σ, L_2 is a partitioned language over Σ_t with $L_2 \approx_{tr} L_1$.

Lemma (Trace-equivalence preserved under intersection)

$(L_{21} \cap L_{22}) \approx_{tr} (L_{11} \cap L_{12})$ if L_{11} and L_{12} are timed languages over Σ, L_{21} and L_{22} are partitioned languages over Σ_t, $L_{21} \approx_{tr} L_{11}$ and $L_{22} \approx_{tr} L_{12}$.

Put restrictions on the timed alphabet

Example

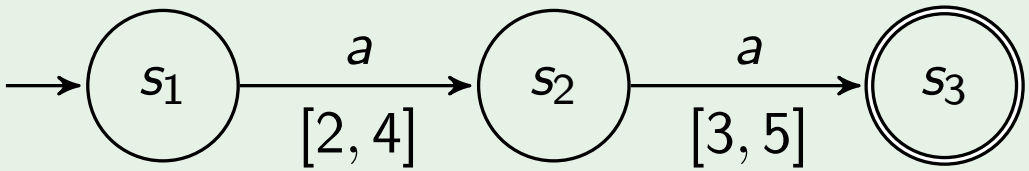
Let $\Sigma_t = \{(a,[2,3)),(a,[3,4]),(a,(4,5]),(a,[0,2)\cup(5,+\inf))\}$.
(Σ_t is partitioned.)

$$L_{21} = \{(a,[3,4]),(a,(4,5])\} \approx_{tr} \{(a,t) \mid t \in [3,5]\} = L_{11}$$
$$L_{22} = \{(a,[2,3)),(a,[3,4])\} \approx_{tr} \{(a,t) \mid t \in [2,4]\} = L_{12}$$
$$L_{21} \cap L_{22} = \{(a,[3,4])\} \approx_{tr} \{(a,t) \mid t \in [3,4]\} = L_{11} \cap L_{12}$$
$$L_{21}^c \approx_{tr} L_{11}^c$$

Summary: Translating RTA into FA

1. Moving time labels into actions to obtain a new FA;
2. Splitting transitions such that the alphabet of the FA is partitioned.

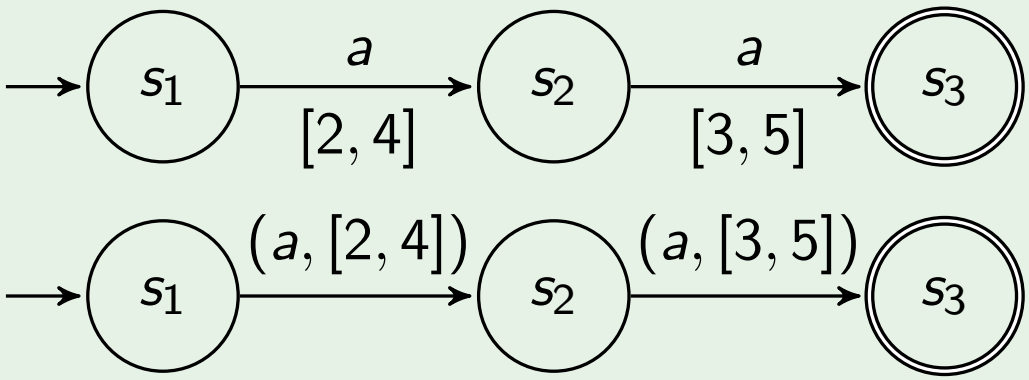
Example



Summary: Translating RTA into FA

1. Moving time labels into actions to obtain a new FA;
2. Splitting transitions such that the alphabet of the FA is partitioned.

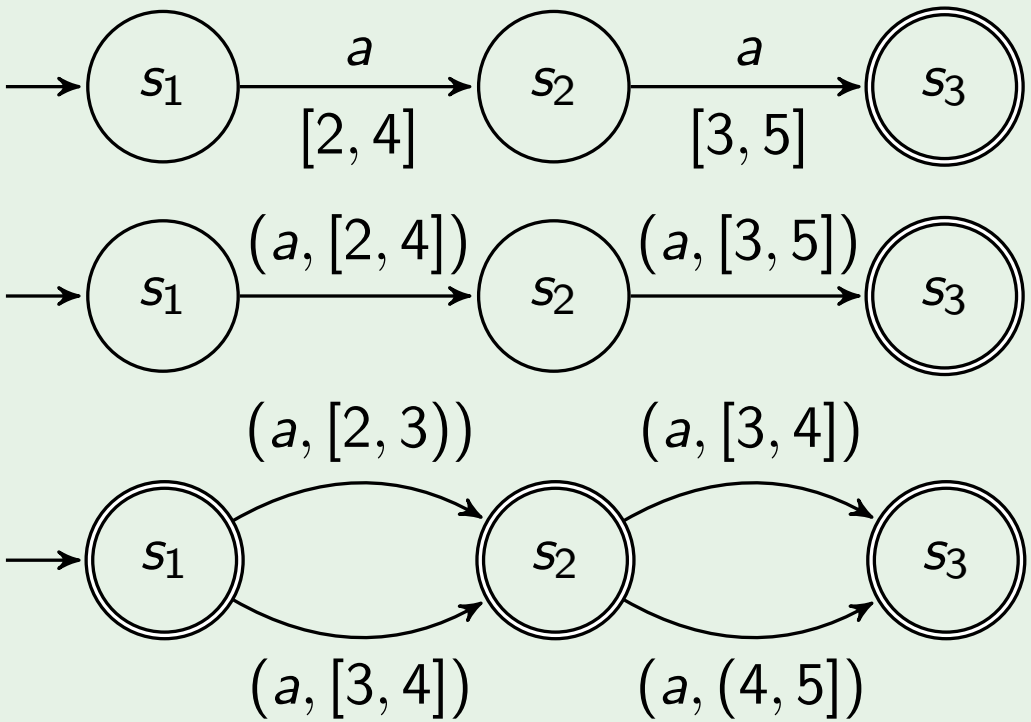
Example



Summary: Translating RTA into FA

1. Moving time labels into actions to obtain a new FA;
2. Splitting transitions such that the alphabet of the FA is partitioned.

Example



Projection of words and languages

Definition

$P_{\uparrow\Sigma_o}(w)$ is inductively defined based on the number of observable symbol-duration pairs in w:

$$P_{\uparrow\Sigma_o}(u^1) = \varepsilon;$$

$$P_{\uparrow\Sigma_o}(u^1 \cdot (a_1, \Lambda_1)) = \begin{cases} (a_1, \Lambda_1), & \text{if } u^1 = \varepsilon \\ (a_1, \sum_{k=1}^{m_1} \Lambda_{1k} + \Lambda_1), & \text{otherwise} \end{cases}$$

$$P_{\uparrow\Sigma_o}((u^1 \cdot (a_1, \Lambda_1)) \cdot w_{suffix}) = P_{\uparrow\Sigma_o}(u^1 \cdot (a_1, \Lambda_1)) \cdot P_{\uparrow\Sigma_o}(w_{suffix})$$

where u^1 is either ε or $(\tau, \Lambda_{11}) \ldots (\tau, \Lambda_{1m_1})$.
And $P_{\uparrow\Sigma_o}(L) = \{P_{\uparrow\Sigma_o}(w) \mid w \in L\}$.

Lemma (Trace-equivalence preserved under projection)

If $L_2 \approx_{tr} L_1$, then $P_{\uparrow\Sigma_o}(L_2) \approx_{tr} P_{\Sigma_o,t}(L_1)$.

Projection of words and languages

Example

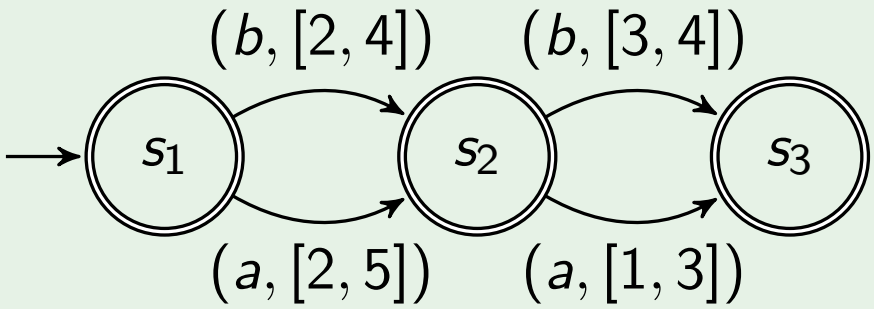
- τ represents all unobservable actions and $a_1, \cdots, a_n$ are observable.

$$
\begin{aligned}
w_t =& (\tau, 0.2)(\tau, 0.5)(\tau, 0.9) \cdot (a_1, 1) \cdot \\
& (\tau, 1.2)(\tau, 1.7) \cdot (a_2, 2) \ldots \\
& (\tau, n-0.5)(\tau, n-0.2) \cdot (a_n, n) \cdot \\
& (\tau, n+0.2) \\
P_{\Sigma_o, t}(w_t) =& (a_1, 1) \cdot (a_2, 2) \cdot \ldots \cdot (a_n, n) \\
w =& (\tau, [0, 1, 0.3])(\tau, [0, 0.4])(\tau, [0.2, 0.4]) \cdot (a_1, [0, 0.2]) \cdot \\
& (\tau, [0.1, 0.3])(\tau, [0.3.0.6]) \cdot (a_2, [0.2, 0.4]) \cdots \\
& (\tau, [0.3, 0.6])(\tau, [0.3, 0.6]) \cdot (a_n, [0, 0.2]) \cdot \\
& (\tau, [0.2, 0.4]) \\
P_{\uparrow \Sigma_o}(w) =& (a_1, [0.3, 1.3]) \cdot (a_2, [0.6, 1.3]) \cdots (a_n, [0.6, 1.4])
\end{aligned}
$$

Summing up Successive Unobservable Durations

Example

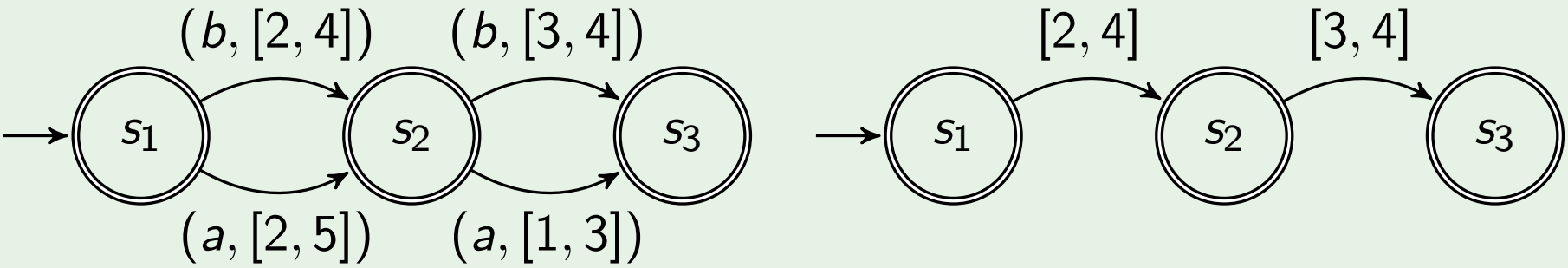
Let $\Sigma_o = \{a\}$.



Summing up Successive Unobservable Durations

Example

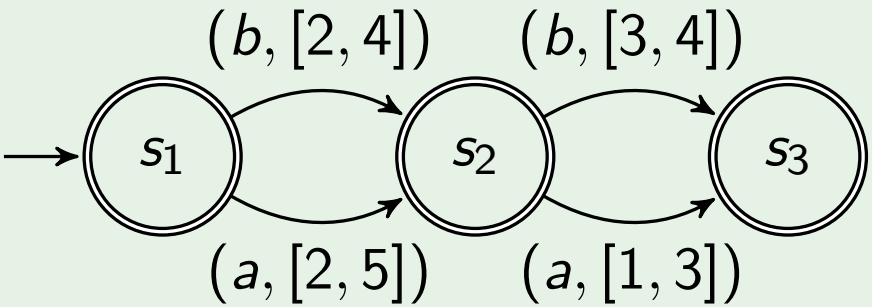
Let $\Sigma_o = \{a\}$.

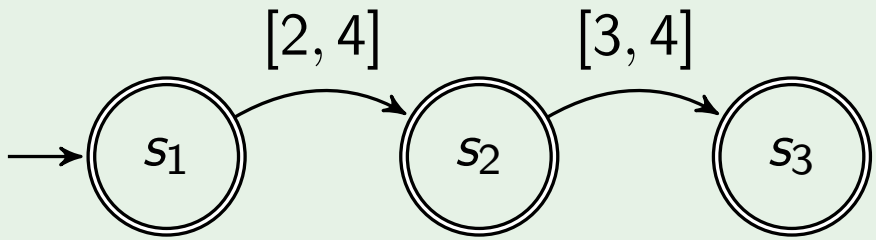


Summing up Successive Unobservable Durations

Example

Let $\Sigma_o = \{a\}$.





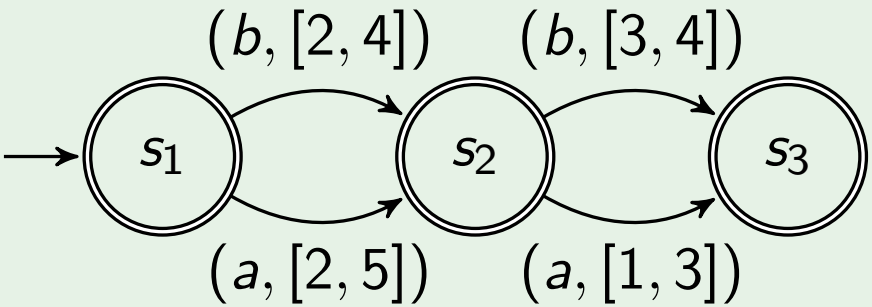
Regular expressions

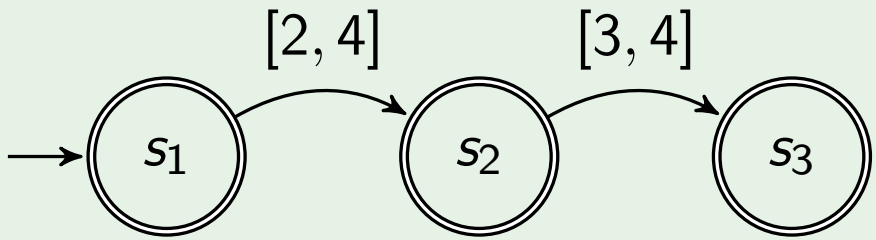
	s_1	s_2	s_3
s_1	ε	$[2,4]$	$[2.4] \cdot [3,4]$
s_2	$\emptyset$	ε	$[3,4]$
s_3	$\emptyset$	$\emptyset$	ε

Summing up Successive Unobservable Durations

Example

Let $\Sigma_o = \{a\}$.





Regular expressions

	s_1	s_2	s_3
s_1	ε	$[2, 4]$	$[2.4] \cdot [3, 4]$
s_2	$\emptyset$	ε	$[3, 4]$
s_3	$\emptyset$	$\emptyset$	ε

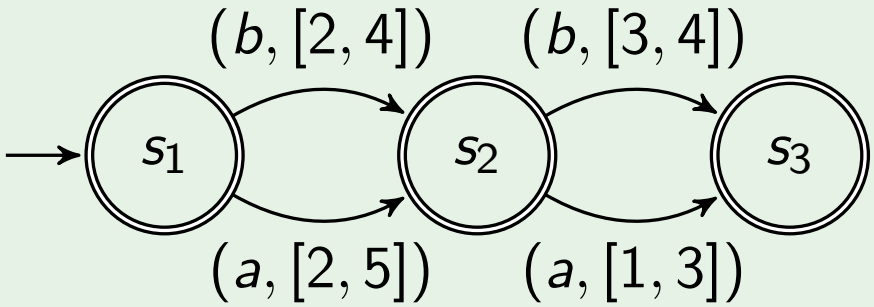
Durations

	s_1	s_2	s_3
s_1	$[0, 0]$	$[2, 4]$	$[6, 8]$
s_2	$\emptyset$	$[0, 0]$	$[3, 4]$
s_3	$\emptyset$	$\emptyset$	$[0, 0]$

Merging Unobservable Durations into Observable Transitions

Example

According to the original FA and the durations calculated before,

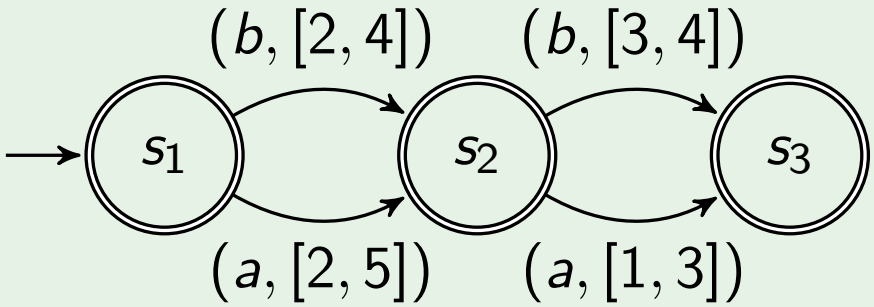


	s_1	s_2	s_3
s_1	$[0, 0]$	$[2, 4]$	$[6, 8]$
s_2	$\emptyset$	$[0, 0]$	$[3, 4]$
s_3	$\emptyset$	$\emptyset$	$[0, 0]$

Merging Unobservable Durations into Observable Transitions

Example

According to the original FA and the durations calculated before,



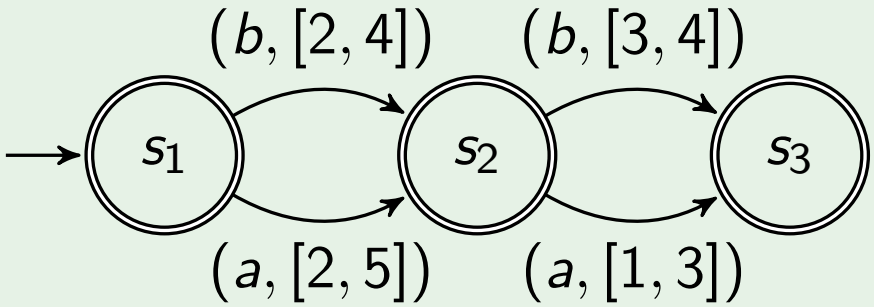
	s_1	s_2	s_3
s_1	$[0, 0]$	$[2, 4]$	$[6, 8]$
s_2	$\emptyset$	$[0, 0]$	$[3, 4]$
s_3	$\emptyset$	$\emptyset$	$[0, 0]$

the projection FA is as follows:

Merging Unobservable Durations into Observable Transitions

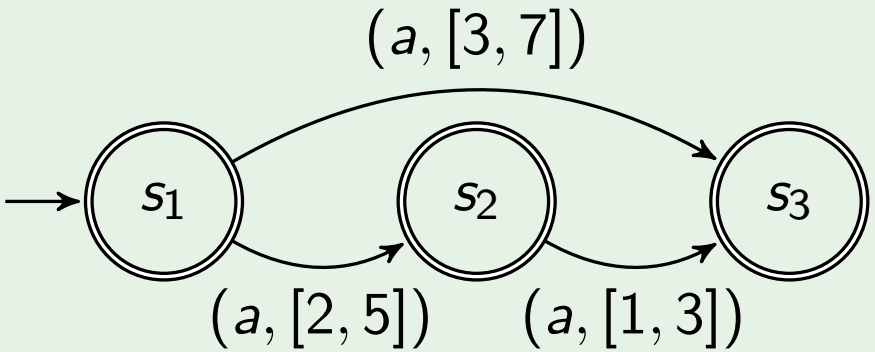
Example

According to the original FA and the durations calculated before,



	s_1	s_2	s_3
s_1	$[0, 0]$	$[2, 4]$	$[6, 8]$
s_2	$\emptyset$	$[0, 0]$	$[3, 4]$
s_3	$\emptyset$	$\emptyset$	$[0, 0]$

the projection FA is as follows:

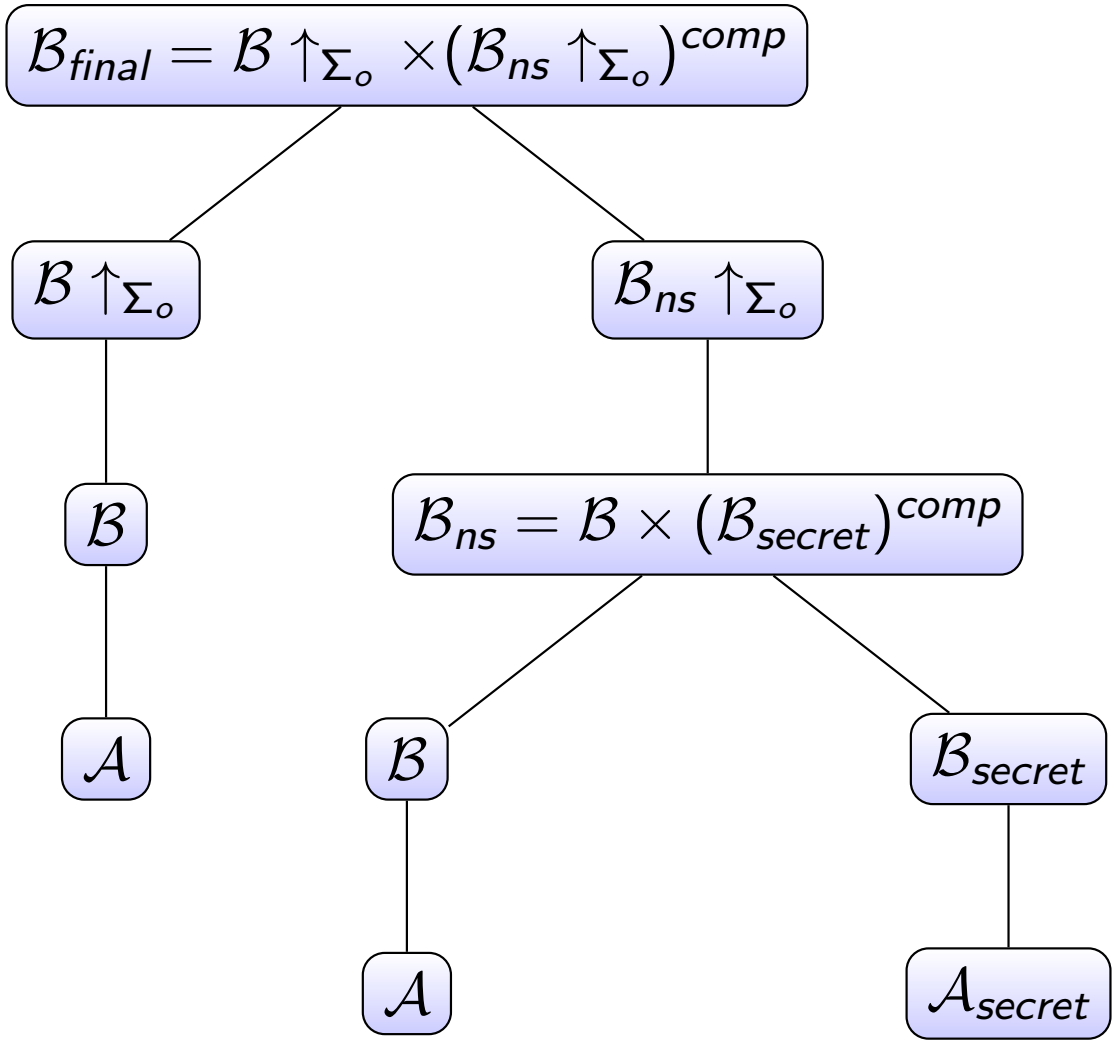


Summary: Computing Projection of FA

1. Summing up successive unobservable durations (using Dima's Normal Form[a] of intervals of Int)
2. Merging unobservable durations into observable transitions' time labels

[a] Dima, C: *Real-time automata.* Journal of Automata, Languages and Combinatorics 6(1), 3-24 (2001)

The Whole Progress



Decidability of Language-Opacity and Initial-State Opacity of RTA

Theorem (2)

The language-opacity problem of RTA is decidable.

Theorem (3)

The initial-state opacity problem of RTA is decidable.

The End

Thank you!