Symbolic Verification of Cryptographic Protocols

Protocol Equivalences

David Baelde

LSV, ENS Paris-Saclay

2018–2019
Static equivalence

The first equivalence does not involve process executions, but only sequences of messages.

When are two sequences of messages distinguishable?

Examples

- \langle u, v, v \rangle \sim \langle v, u, v \rangle ?
Static equivalence

The first equivalence does not involve process executions, but only sequences of messages.

When are two sequences of messages distinguishable?

Examples

- $\langle u, v, v \rangle \sim \langle v, u, v \rangle$?
- $\langle n \rangle \sim \langle n' \rangle$? $\langle \langle n, m \rangle \rangle \sim \langle \langle n', n' \rangle \rangle$?
- $\langle senc(u, k) \rangle \sim \langle n' \rangle$?
- $\langle senc(u, k) \rangle \sim \langle senc(v, k) \rangle$?
- $\langle aenc(u, pk), u, pk \rangle \sim \langle aenc(v, pk), u, pk \rangle$?
Static equivalence

The first equivalence does not involve process executions, but only sequences of messages.

When are two sequences of messages distinguishable?

Examples

- \(\langle u, v, v \rangle \sim \langle v, u, v \rangle\) ?
- \(\langle n \rangle \sim \langle n' \rangle\) ? \(\langle\langle n, m \rangle \rangle \sim \langle\langle n', n' \rangle \rangle\) ?
- \(\langle\langle u, v \rangle \rangle \sim \langle n' \rangle\) ? \(\langle\text{senc}(u, k) \rangle \sim \langle n' \rangle\) ?
Static equivalence

The first equivalence does not involve process executions, but only sequences of messages.

When are two sequences of messages distinguishable?

Examples

- \(\langle u, v, v \rangle \sim \langle v, u, v \rangle?\)
- \(\langle n \rangle \sim \langle n' \rangle? \langle \langle n, m \rangle \rangle \sim \langle \langle n', n' \rangle \rangle?\)
- \(\langle \langle u, v \rangle \rangle \sim \langle n' \rangle? \langle \text{senc}(u, k) \rangle \sim \langle n' \rangle?\)
- \(\langle \text{senc}(u, k) \rangle \sim \langle \text{senc}(v, k) \rangle? \langle \text{senc}(u, k) \rangle \sim \langle \text{senc}(u, k') \rangle?\)
Static equivalence

The first equivalence does not involve process executions, but only sequences of messages.

When are two sequences of messages distinguishable?

Examples

- $\langle u, v, v \rangle \sim \langle v, u, v \rangle$?
- $\langle n \rangle \sim \langle n' \rangle$? $\langle \langle n, m \rangle \rangle \sim \langle \langle n', n' \rangle \rangle$?
- $\langle \langle u, v \rangle \rangle \sim \langle n' \rangle$? $\langle \text{senc}(u, k) \rangle \sim \langle n' \rangle$?
- $\langle \text{senc}(u, k) \rangle \sim \langle \text{senc}(v, k) \rangle$? $\langle \text{senc}(u, k) \rangle \sim \langle \text{senc}(u, k') \rangle$?
- $\langle \text{aenc}(u, pk), u, pk \rangle \sim \langle \text{aenc}(v, pk), u, pk \rangle$?
Static equivalence

As before, consider frames in $N^* \times (W \rightarrow T_c(N))$:
- 1st component = bound/private names, noted $bn(\Phi)$;
- 2nd component = intruder’s knowledge, addressed via handles of $\text{dom}(\Phi)$.

Definition

Two frames $\Phi_1$ and $\Phi_2$ are statically equivalent when
- they have the same domain: $\text{dom}(\Phi_1) = \text{dom}(\Phi_2)$;
- for all $M \in T(W \cup N \setminus bn(\Phi_1, \Phi_2))$, $M\Phi_1 \Downarrow$ iff $M\Phi_2 \Downarrow$;
- for all $M, N \in T(W \cup N \setminus bn(\Phi_1, \Phi_2))$,
  $M\Phi_1 \Downarrow =_E N\Phi_1 \Downarrow$ iff $M\Phi_2 \Downarrow =_E N\Phi_2 \Downarrow$.

Proposition

Static equivalence is an equivalence. It is stable by bijective renaming.
It does not compose: $\Phi_1 \sim \Phi'_1$ and $\Phi_2 \sim \Phi'_2 \not\Rightarrow \Phi_1 \uplus \Phi_2 \sim \Phi'_1 \uplus \Phi'_2$. 
Static equivalence: examples

Suppose we have only constructors and the standard equations for pairs and (a)symmetric encryption.

Examples (bis)

- \(\{\, w_1 \mapsto u, \, w_2 \mapsto v, \, w_3 \mapsto v \,\} \sim \{\, w_1 \mapsto v, \, w_2 \mapsto u, \, w_3 \mapsto v \,\}\) ?
- \(\{\, w \mapsto n \,\} \sim \{\, w \mapsto n' \,\} \) ? \(\{\, w \mapsto \langle n, m \rangle \,\} \sim \{\, w \mapsto \langle n', n' \rangle \,\} \) ?
- \(\{\, w \mapsto \langle u, v \rangle \,\} \sim \{\, w \mapsto n' \,\} \) ? \(\{\, w \mapsto \text{senc}(u, k) \,\} \sim \{\, w \mapsto n' \,\} \) ?
- \(\{\, w \mapsto \text{senc}(u, k) \,\} \sim \{\, w \mapsto \text{senc}(v, k) \,\} \) ?
- \(\{\, w \mapsto \text{senc}(u, k) \,\} \sim \{\, w \mapsto \text{senc}(u, k') \,\} \) ?
- \(\{\, w \mapsto \text{aenc}(u, pk), \, w' \mapsto u, \, w'' \mapsto pk \,\} \sim \{\, w \mapsto \text{aenc}(v, pk), \, w' \mapsto u, \, w'' \mapsto pk \,\} \) ?
Application: guessing attacks

We usually assume that secrets cannot be guessed: no brute force attacks.

That is not reasonable for low/fixed entropy secrets, such as PIN, passwords, one-time verification code, etc.

Offline guessing attacks

A protocol is resistant against offline guessing attacks on some name $d$ when any reachable frame $\Phi$ is such that

$$\Phi \cup \{w \mapsto d\} \sim \Phi \cup \{w \mapsto d'\}$$

for $w, d'$ fresh.

This notion is meaningful even with a passive adversary.
Assume public-key encryption but no PKI (public keys ≠ identities). A and B only share a weak password $p$, want to authenticate.

1. $A \to B : \text{ senc}(\text{pub}(k), p)$
2. $B \to A : \text{ senc}(\text{aenc}(r, \text{pub}(k)), p)$
3. $A \to B : \text{ senc}(n_a, r)$
4. $B \to A : \text{ senc}(\langle n_a, n_b \rangle, r)$
5. $A \to B : \text{ senc}(n_b, r)$
Application: EKE

Assume public-key encryption but no PKI (public keys ≠ identities). A and B only share a weak password $p$, want to authenticate.

1. $A \rightarrow B : \text{senc}(\text{pub}(k), p)$
2. $B \rightarrow A : \text{senc}(\text{aenc}(r, \text{pub}(k)), p)$
3. $A \rightarrow B : \text{senc}(n_a, r)$
4. $B \rightarrow A : \text{senc}(\langle n_a, n_b \rangle, r)$
5. $A \rightarrow B : \text{senc}(n_b, r)$

Let $\Phi = \{ w_1 \mapsto \text{senc}(\text{pub}(k), p), \ldots, w_5 \mapsto \text{senc}(n_b, r) \}$. Can $p$ be guessed offline, that is

$$\Phi \cup \{ w \mapsto p \} \sim \Phi \cup \{ w \mapsto p' \} ?$$
Assume public-key encryption but no PKI (public keys ≠ identities). A and B only share a weak password $p$, want to authenticate.

1. $A \rightarrow B : \text{senc}(\text{pub}(k), p)$
2. $B \rightarrow A : \text{senc}(\text{aenc}(r, \text{pub}(k)), p)$
3. $A \rightarrow B : \text{senc}(n_a, r)$
4. $B \rightarrow A : \text{senc}(\langle n_a, n_b \rangle, r)$
5. $A \rightarrow B : \text{senc}(n_b, r)$

Let $\Phi = \{w_1 \mapsto \text{senc}(\text{pub}(k), p), \ldots, w_5 \mapsto \text{senc}(n_b, r)\}$. Can $p$ be guessed offline, that is

$$\Phi \cup \{w \mapsto p\} \sim \Phi \cup \{w \mapsto p'\}$$

Only if $\text{senc}(\text{sdec}(x, y), y) = x \ldots$ and no getkey primitive for $\text{aenc}$. 

---

David Baelde (ENS Paris-Saclay)  
Protocol Equivalences  
2018–2019 6 / 23
May testing

The reduction semantics (cf. previous lectures) provide a first natural definition of when two processes can be distinguished.

Definition

A test is a process with no free name and in which a special channel $T$ may occur. A process $P$ may pass a test $T$, written $P \models T$ if

$$P \mid T \rightsquigarrow^* \text{out}(T, u) \mid Q$$

for some $u$ and $Q$.

Let $T(P) := \{ \; T \mid P \models T \; \}$. Processes $P$ and $Q$ are in may-testing equivalence when $T(P) = T(Q)$. 

Quite natural, but may not model all desired aspects, e.g. probabilities, must testing, asynchronicity. As such, may testing equivalence is hard to verify!
May testing

The reduction semantics (cf. previous lectures) provide a first natural definition of when two processes can be distinguished.

**Definition**

A **test** is a process with no free name and in which a special channel $T$ may occur. A process $P$ **may pass** a test $T$, written $P \models T$ if

$$P \parallel T \leadsto^* \text{out}(T, u) \parallel Q$$

for some $u$ and $Q$.

Let $T(P) := \{ T \mid P \models T \}$.

Processes $P$ and $Q$ are in **may-testing equivalence** when $T(P) = T(Q)$.

Quite natural, but may not model all desired aspects, e.g. probabilities, must testing, asynchronicity.
May testing

The reduction semantics (cf. previous lectures) provide a first natural definition of when two processes can be distinguished.

Definition

A test is a process with no free name and in which a special channel $T$ may occur. A process $P$ may pass a test $T$, written $P \models T$ if

$$P \mid T \rightsquigarrow^* \text{out}(T, u) \mid Q$$

for some $u$ and $Q$.

Let $T(P) := \{ T \mid P \models T \}$.

Processes $P$ and $Q$ are in may-testing equivalence when $T(P) = T(Q)$.

Quite natural, but may not model all desired aspects, e.g. probabilities, must testing, asynchronicity. As such, may testing equivalence is hard to verify!
Trace equivalence

Weak labelled transitions

We write $A \xrightarrow{\text{tr}} B$ when:
- $\text{tr}$ only contains input and output actions (no $\tau$);
- there exists $\text{tr}'$ obtained from $\text{tr}$ by adding $\tau$s such that $A \xrightarrow{\text{tr}'} B$.

Definition

Given a configuration $A = (P, \Phi)$, define

$$\text{Tr}(A) := \{ (\text{tr}, \Phi') \mid A \xrightarrow{\text{tr}} (_, \Phi') \}.$$ 

We say that $A$ and $B$ are trace equivalent, noted $A \approx B$, iff

for all $(\text{tr}, \Phi') \in \text{Tr}(A)$ there exists $(\text{tr}, \Psi') \in \text{Tr}(B)$. $\Phi' \sim \Psi'$

and conversely.
Alternative definition

Proposition

*Close Tr(·) under static equivalence:*

\[
Tr'(P, \Phi) := \{ (\text{tr}, \Phi') \mid (P, \Phi) \xrightarrow{\text{tr}} (P', \Phi''), \Phi'' \sim \Phi' \}
\]

Then we have \(A \approx B\) iff \(\text{Tr}'(A) = \text{Tr}'(B)\).

Remarks

- \(A \approx B\) imposes \(\Phi(A) \sim \Phi(B)\), but not \(\Phi(A) = \Phi(B)\).
- The definition really makes sense only when \(\text{bn}(\Phi(A)) = \text{bn}(\Phi(B))\).
- In general we do not have that \(\Phi \sim \Psi\) implies \((P, \Phi) \approx (P, \Psi)\).
Examples

1. \( \text{in}(c, x).\text{out}(c, \text{ok}) \approx? \ \text{in}(c, x) | \text{out}(c, \text{ok}) \)

2. \( \text{in}(c, x).\text{out}(c, \text{ok}) \approx? \ \text{in}(c, x).\text{out}(c, x) \)

3. \( \text{new } n, m. \text{out}(c, n) | \text{out}(c, m) \approx? \ \text{new } n, m. \text{out}(c, n).\text{out}(c, m) \)

4. \( \text{new } n, m. \text{out}(c, n) | \text{out}(c, \text{hash}(m)) \approx? \)
   \( \text{new } n. \text{out}(c, n).\text{out}(c, \text{hash}(n)) \)

5. \( \text{out}(c, u_1).\ldots.\text{out}(c, u_n).\text{in}(c, x)\text{.if } x = v \text{ then } \text{out}(c, \text{ok}) \approx? \)
   \( \text{out}(c, u_1).\ldots.\text{out}(c, u_n).\text{in}(c, x).0 \)
Trace equivalence $\subseteq$ may-testing ?

**Proposition**

If $(P, \emptyset) \approx (Q, \emptyset)$ then they are in may-testing equivalence...
Trace equivalence \(\subseteq\) may-testing?

**Proposition**

*If \((P, \emptyset) \approx (Q, \emptyset)\) then they are in may-testing equivalence... provided computation is deterministic, i.e. for all \(t, u\) and \(v\) such that \(t \Downarrow u\), we have \(t \Downarrow v\) iff \(u =_{E} v\).*
Proposition

If \((P, \emptyset) \approx (Q, \emptyset)\) then they are in may-testing equivalence...

provided computation is deterministic, i.e.

for all \(t, u\) and \(v\) such that \(t \Downarrow u\), we have \(t \Downarrow v\) iff \(u =_E v\).

Proof idea.

Decompose \(P \upharpoonright T \rightsquigarrow^* \text{out}(\mathbb{T}, _) \upharpoonright _\) into internal reductions of \(P\) and \(T\), and communications between the two. This yields a trace of \(P\), which \(Q\) can simulate. Compose this with the reductions of \(T\) to obtain \(Q \upharpoonright T \rightsquigarrow^* \text{out}(\mathbb{T}, _) \upharpoonright _\).

Devil is in the details! there is a counter-example when computation is non-deterministic because traces do not keep track of how recipes are evaluated.
Proposition

If $P$ and $Q$ are may-testing equivalent then $P \approx Q$, ...
May testing ⊆ trace equivalence?

**Proposition**

If $P$ and $Q$ are may-testing equivalent then $P \approx Q$, provided the processes are image-finite:

$$\text{for any } \text{tr, } \{ \Phi \mid (\text{tr}, \Phi) \in \text{Tr}'(P, \emptyset) \} \text{ is finite up to } \sim$$

and similarly for $Q$. 

Example

$P := \text{new } c. (\text{out } (c, \text{ok}) | !\text{in } (c, x) \text{. out } (c, h(x)) | \text{in } (c, x) \text{. out } (a, x))$

$Q := P | \text{new } n. \text{out } (a, n)$

We have $P \not\approx Q$ but $P$ and $Q$ are in may-testing equivalence. This is "only" pathological!
May testing $\subseteq$ trace equivalence?

**Proposition**

If $P$ and $Q$ are may-testing equivalent then $P \approx Q$, provided the processes are image-finite:

For any $tr$, \( \{ \Phi \mid (tr, \Phi) \in Tr'(P, \emptyset) \} \) is finite up to $\sim$

and similarly for $Q$.

**Example**

\[
P := \text{new } c. (\text{out}(c, \text{ok}) \mid !\text{in}(c, x).\text{out}(c, h(x)) \mid \text{in}(c, x).\text{out}(a, x))
\]

\[
Q := P \mid \text{new } n. \text{out}(a, n)
\]

We have $P \not\approx Q$ but $P$ and $Q$ are in may-testing equivalence.
May testing $\subseteq$ trace equivalence?

**Proposition**

If $P$ and $Q$ are may-testing equivalent then $P \approx Q$, provided the processes are image-finite:

$$\{ \Phi \mid (\text{tr}, \Phi) \in \text{Tr}'(P, \emptyset) \}$$

is finite up to $\sim$

and similarly for $Q$.

**Example**

$P := \text{new } c. (\text{out}(c, \text{ok}) \mid ! \text{in}(c, x).\text{out}(c, h(x)) \mid \text{in}(c, x).\text{out}(a, x))$

$Q := P \mid \text{new } n. \text{out}(a, n)$

We have $P \not\approx Q$ but $P$ and $Q$ are in may-testing equivalence.

This is “only” pathological!
Application: strong secrecy

**Definition**

A protocol $P$ ensures the **strong secrecy** of some variables $\vec{x}$ if, for all (relevant) values $\vec{u}, \vec{v}$, $P[\vec{x} := \vec{u}] \approx P[\vec{x} := \vec{v}]$.

**Weak secrecy:** some value cannot be (fully) derived by the attacker.

**Strong secrecy:** the attacker has no information at all about the value.
Application: strong secrecy

**Definition**

A protocol $P$ ensures the **strong secrecy** of some variables $\vec{x}$ if, for all (relevant) values $\vec{u}$, $\vec{v}$, $P[\vec{x} := \vec{u}] \approx P[\vec{x} := \vec{v}]$.

**Weak secrecy**: some value cannot be (fully) derived by the attacker.

**Strong secrecy**: the attacker has no information at all about the value.

Blanchet’s key exchange protocol:

1. $A \rightarrow B$ : $\text{aenc}(\text{sign}(\langle pk_A, pk_B, k \rangle, sk_A), pk_B)$
2. $B \rightarrow A$ : $\text{senc}(x, k)$
3. $A \rightarrow B$ : $\text{senc}(y, k)$

**Scenario**: $A$ and $B$ honest. Is $x$ strongly secret? Are $x$, $y$ strongly secret?
Application: private authentication

Agents $A$ and $B$ want to authenticate, without revealing their identities.

\[
\begin{align*}
I(ska, pk_b) & \quad R(skb, pak) \\
\text{new } n_a. & \quad \text{new } n_b. \\
\text{let } pk_a = \text{pub}(sk_a) & \quad \text{let } pk_b = \text{pub}(sk_b) \text{ in} \\
\text{out}(c, aenc(\langle n_a, pk_a \rangle, pk_b)). & \quad \text{in}(c, x). \text{let } y = \text{adec}(x, sk_b) \text{ in} \\
\ldots & \quad \text{if } \text{proj}_2(y) = pk_a \text{ then} \\
 & \quad \text{out}(c, aenc(\langle \text{proj}_1(y), n_b, pk_b \rangle, pk_a))
\end{align*}
\]

Anonymity

\[
\begin{align*}
\text{new } sk_a, sk_b, sk_c. & \quad \text{out}(c, \langle \text{pub}(sk_a), \text{pub}(sk_b), \text{pub}(sk_c) \rangle). R(skb, \text{pub}(sk_a)) \\
\cong? & \\
\text{new } sk_a, sk_b, sk_c. & \quad \text{out}(c, \langle \text{pub}(sk_a), \text{pub}(sk_b), \text{pub}(sk_c) \rangle). R(skb, \text{pub}(sk_c))
\end{align*}
\]
Application: private authentication

Agents $A$ and $B$ want to authenticate, without revealing their identities.

\[
\begin{align*}
I(sk_a, pk_b) & \quad R(sk_b, pk_a) \\
\text{new } n_a. & \quad \text{new } n_b. \\
\text{let } pk_a = \text{pub}(sk_a) \text{ in} & \quad \text{let } pk_b = \text{pub}(sk_b) \text{ in} \\
\text{out}(c, \text{aenc}(\langle n_a, pk_a \rangle, pk_b)). & \quad \text{in}(c, x). \text{let } y = \text{adec}(x, sk_b) \text{ in} \\
\text{...} & \quad \text{if } \text{proj}_2(y) = pk_a \text{ then} \\
& \quad \text{out}(c, \text{aenc}(\langle \text{proj}_1(y), n_b, pk_b \rangle, pk_a)) \\
& \quad \text{else out}(c, \text{aenc}(n_b, pk_b)) \leftarrow \text{decoy!}
\end{align*}
\]

Anonymity

\[
\begin{align*}
\text{new } sk_a, sk_b, sk_c. \quad \text{out}(c, \langle \text{pub}(sk_a), \text{pub}(sk_b), \text{pub}(sk_c) \rangle). R(sk_b, \text{pub}(sk_a)) & \approx? \\
\text{new } sk_a, sk_b, sk_c. \quad \text{out}(c, \langle \text{pub}(sk_a), \text{pub}(sk_b), \text{pub}(sk_c) \rangle). R(sk_b, \text{pub}(sk_c))
\end{align*}
\]
The BAC e-passport protocol is used between a tag $T$ and a reader $R$. After $k_E$ and $k_M$ are derived from optical scan (shared secrets), a key is established as follows:

1. $T \rightarrow R : n_T$
2. $R \rightarrow T : \text{senc}(\langle n_R, n_T, k_R \rangle, k_E), \text{mac}(\text{senc}(\langle n_R, n_T, k_R \rangle, k_E), k_M)$
3. $T \rightarrow R : \text{senc}(\langle n_T, n_R, k_T \rangle, k_E), \text{mac}(\text{senc}(\langle n_T, n_R, k_T \rangle, k_E), k_M)$
Application: unlinkability

The BAC e-passport protocol is used between a tag $T$ and a reader $R$. After $k_E$ and $k_M$ are derived from optical scan (shared secrets), a key is established as follows:

1. $T \rightarrow R$: $n_T$
2. $R \rightarrow T$: $\text{senc}(\langle n_R, n_T, k_R \rangle, k_E), \text{mac}(\text{senc}(\langle n_R, n_T, k_R \rangle, k_E), k_M)$
3. $T \rightarrow R$: $\text{senc}(\langle n_T, n_R, k_T \rangle, k_E), \text{mac}(\text{senc}(\langle n_T, n_R, k_T \rangle, k_E), k_M)$

French implementation:

$$T(k_E, k_M) := \text{new } n_T, k_T. \text{ out}(c, n_T).\text{in}(c, x).$$

if $\text{mac}(\text{proj}_1(x), k_M) = \text{proj}_2(x)$ then
  if $n_T = \text{proj}_1(\text{sdec}(\text{proj}_1(x), k_E))$ then ... else
    out$(c, \text{ERR_nonce})$
else out$(c, \text{ERR_mac})$
The BAC e-passport protocol is used between a tag $T$ and a reader $R$. After $k_E$ and $k_M$ are derived from optical scan (shared secrets), a key is established as follows:

1. $T \rightarrow R : n_T$
2. $R \rightarrow T : \text{senc}(\langle n_R, n_T, k_R \rangle, k_E), \text{mac}(\text{senc}(\langle n_R, n_T, k_R \rangle, k_E), k_M)$
3. $T \rightarrow R : \text{senc}(\langle n_T, n_R, k_T \rangle, k_E), \text{mac}(\text{senc}(\langle n_T, n_R, k_T \rangle, k_E), k_M)$

French implementation:

$$T(k_E, k_M) := \text{new } n_T, k_T. \text{ out}(c, n_T).\text{ in}(c, x).$$

if \( \text{mac}(\text{proj}_1(x), k_M) = \text{proj}_2(x) \) then

if \( n_T = \text{proj}_1(\text{sdec}(\text{proj}_1(x), k_E)) \) then \ldots else

out\((c, \text{ERRNonce})\)
else
out\((c, \text{ERRMac})\)

Linkability issue:

new $k_E, k_M, k'_E, k'_M$. $T(k_E, k_M) \parallel R(k_E, k_M) \not\equiv T(k_E', k_M) \parallel R(k'_E, k'_M)$
Some general definitions

Let $I(\vec{k}, \vec{n})$ and $R(\vec{k}, \vec{n})$ be two roles of a protocol, where $\vec{k}$ represents identity parameters and $\vec{n}$ represent session parameters.

**Definition**

The protocol ensures **strong unlinkability** when:

$! \text{new } \vec{k}. \! \text{new } \vec{n}. \; I(\vec{k}, \vec{n}) \parallel R(\vec{k}, \vec{n}) \approx \! \text{new } \vec{k}. \! \text{new } \vec{n}. \; I(\vec{k}, \vec{n}) \parallel R(\vec{k}, \vec{n})$

**Definition**

The protocol ensures **anonymity** when:

$\mathcal{M} \approx \mathcal{M} \mid ! \text{new } \vec{n}. \; I(\vec{k}_0, \vec{n}) \parallel R(\vec{k}_0, \vec{n})$

where $\mathcal{M}$ is the left process on the previous equivalence.
Observational equivalence

We write $P \Downarrow c$ when $P$ can output on $c$ after internal reductions, i.e. $P \rightsquigarrow^* \text{out}(c, u).P' | P''$.

Definition

The binary relation $\mathcal{R}$ over closed processes is a **observational bisimulation** if it is symmetric and $P \mathcal{R} Q$ implies:

- for all $c$, $P \Downarrow c$ implies $Q \Downarrow c$;
- for all $P'$, $P \rightsquigarrow^* P'$ implies $Q \rightsquigarrow^* \mathcal{R} P'$;
- for all $R$, $(P | R) \mathcal{R} (Q | R)$.

**Observational equivalence** is the largest observational bisimulation.
Observational equivalence

We write $P \Downarrow c$ when $P$ can output on $c$ after internal reductions, i.e. $P \rightsquigarrow^* \text{out}(c, u).P' | P''$.

**Definition**

The binary relation $\mathcal{R}$ over closed processes is a *observational bisimulation* if it is symmetric and $P \mathcal{R} Q$ implies:

- For all $c$, $P \Downarrow c$ implies $Q \Downarrow c$;
- For all $P'$, $P \rightsquigarrow^* P'$ implies $Q \rightsquigarrow^* \mathcal{R} P'$;
- For all $R$, $(P | R) \mathcal{R} (Q | R)$.

*Observational equivalence* is the largest observational bisimulation.

The quantification over all contexts makes it hard to prove obs. equiv, both by hand and mechanically.
Labelled bisimulation

**Definition**

The binary relation $\mathcal{R}$ over configurations is a **bisimulation** if it is symmetric and $A \mathcal{R} B$ implies:

- $\Phi(A) \sim \Phi(B)$;
- $A \xrightarrow{\tau} A'$ implies $B \xrightarrow{\tau}^* \mathcal{R} A'$;
- $A \xrightarrow{\alpha} A'$ implies $B \xrightarrow{\alpha} \mathcal{R} A'$.

**Bisimilarity** is the largest bisimulation.

**Theorem (Abadí, Blanchet & Fournet 2001/2017)**

$P$ and $Q$ are observationally equivalent iff they are bisimilar.
Comparison with trace equivalence

Proposition

If A and B are bisimilar, then $A \approx B$. 

Trace equivalence is a linear-time property, bisimularity is branching-time: trace equivalence does not “see” choice points.

Example

Assume a choice operator $P_1 + P_2 \xrightarrow{\tau} P_i$ for $i \in \{1, 2\}$.

$\text{out}(a, \text{ok}) + \text{out}(b, \text{ok}) + \text{out}(c, \text{ok}) \approx \text{out}(a, \text{ok}) + \text{out}(b, \text{ok}) + \text{out}(a, \text{ok}) + \text{out}(c, \text{ok})$ but they are not bisimilar.

Example without choice (Pous & Madiot)

Without choice, take two observably distinct actions $\alpha$ and $\beta$.

Consider $P := \alpha. (\alpha. (\alpha. \beta. \alpha \mid \beta. \beta) \mid \beta. \alpha)$ and $Q := \alpha. \beta. \alpha \mid \alpha. (\alpha. \beta. (\alpha \mid \beta) \mid \beta)$.

We have $P \approx Q$ but $P \alpha. \beta. \alpha \xrightarrow{\tau} \alpha. \beta. \alpha \mid \beta. \beta \mid \alpha$ which cannot be matched by $Q$. 

David Baelde (ENS Paris-Saclay)
Comparison with trace equivalence

**Proposition**

If $A$ and $B$ are bisimilar, then $A \approx B$.

Trace equivalence is a linear-time property, bisimilarity is branching-time: trace equivalence does not “see” choice points.

**Example**

Assume a choice operator $P_1 + P_2 \overset{\tau}{\rightarrow} P_i$ for $i \in \{1, 2\}$.

\[
\text{out}(a, \text{ok}).(\text{out}(b, \text{ok}) + \text{out}(c, \text{ok})) \approx \\
\text{out}(a, \text{ok}).\text{out}(b, \text{ok}) + \text{out}(a, \text{ok}).\text{out}(c, \text{ok})
\]

but they are not bisimilar.
Comparison with trace equivalence

Proposition

If $A$ and $B$ are bisimilar, then $A \approx B$.

Trace equivalence is a linear-time property, bisimilarity is branching-time: trace equivalence does not “see” choice points.

Example

Assume a choice operator $P_1 + P_2 \xrightarrow{\tau} P_i$ for $i \in \{1, 2\}$. 

\[
\text{out}(a, \text{ok}).(\text{out}(b, \text{ok}) + \text{out}(c, \text{ok})) \approx \\
\text{out}(a, \text{ok}).\text{out}(b, \text{ok}) + \text{out}(a, \text{ok}).\text{out}(c, \text{ok})
\]

but they are not bisimilar.

Example without choice (Pous & Madiot)

Without choice, take two observably distinct actions $\alpha$ and $\beta$. Consider $P := \alpha.(\alpha.\beta.\alpha|\beta.\beta)|\beta.\alpha$ and $Q := \alpha.\beta.\alpha|\alpha.(\alpha.\beta.(\alpha|\beta)|\beta)$.

We have $P \approx Q$ but $P \xrightarrow{\alpha.\beta.\alpha} \alpha.\beta.\alpha|\beta.\beta|\alpha$ which cannot be matched by $Q$. 
Comparison with trace equivalence

**Proposition**

If $A$ and $B$ are determinate, and $A \approx B$, then $A$ and $B$ are bisimilar.

**Definition (A possible definition of determinacy)**

$A$ is determinate if, for all $A \xrightarrow{\text{tr}} A'$: $A'$ does not have two inputs (resp. outputs) on the same $c$ at toplevel.
Bisimilarity in practice

The gap between bisim and trace equivalence (determinacy) may or may not matter depending on applications.

Bisimilarity is generally easier to prove than trace equivalence:
- by hand: bisimulation proof technique;
- mechanically: incrementally find matching processes.

In verification, even more constraining forms of equivalences are considered, e.g. diff-equivalence where the two processes must have the same structure and differ only in the terms that they use.

Tools

- diff-equivalence: proverif, tamarin (unbounded sessions)
- bisimilarity: SPEC (bounded sessions)
- trace equivalence: Apte/DeepSec, Akiss (bounded sessions)
Equivalence examples

**Diff-equivalence successes**

- Strong secrecy: $P[x := u] \ vs \ P[x := 0]$ ?
- Anonymity: $P[x := A] \ vs \ P[x := B]$ ?

**Unlinkability: gray zone**

- Not bisimilar in general, trace equiv. needed:

  \[ \begin{align*}
    \& \mid \text{! new } k \ \text{! new } n, m. \ l(k, n) | R(k, m) \\
    \& \mid \text{! new } k \ \text{new } n, m. \ l(k, n) | R(k, m)
  \end{align*} \]

- Often diff-equivalent when no shared identity:

  \[ \begin{align*}
    \& \mid \text{! new } k \ \text{! new } k' \ \text{new } n, m. \ l(k, n) | R(m) \\
    \& \mid \text{! new } k \ \text{! new } k' \ \text{new } n, m. \ l(k', n) | R(m)
  \end{align*} \]
# Summary

## Static equivalence
- Indistinguishable sequences of messages
- Depends on equational theory, destructors vs. constructors

## May testing & trace equivalence
- May testing: there exists an adversary (in the same model)
- Trace equivalence: the same traces can be observed
- Trace equivalence is a good approximation of may testing, often used in practice for verification.

## Obs. equiv., bisimulation and diff-equiv.
- Obs. equiv = bisimulation = strongest “reasonable” equivalence
- Good properties: compositional, congruence, easier to check
- Common approximation for verification: diff-equivalence