Deciding equivalence properties in security protocols

Itsaka Rakotonirina

joint work with

Vincent Cheval, Steve Kremer

INRIA Nancy Grand-Est, LORIA

Security protocols

Google SSO

BAC (e-passport)

Helios (e-voting)

TLS 1.3 (prior ver.)

WPA2 (wifi)

Security protocols

Google SSO

☠ Armando *et al.* (2008)

BAC (e-passport)

☠ Chothia and Smirnov (2010)

Helios (e-voting)

☠ Cortier and Smyth (2011)

TLS 1.3 (prior ver.)

☠ Cremers *et al.* (2016)

WPA2 (wifi)

☠ Vanhoef and Piessens (2017)

Security protocols

The attacker…

- Reads / Writes
- Intercepts

But they do not need to…

- Break cryptography
- Use side channels

Security protocols

The attacker…

- Reads / Writes
- Intercepts

But they do not need to…

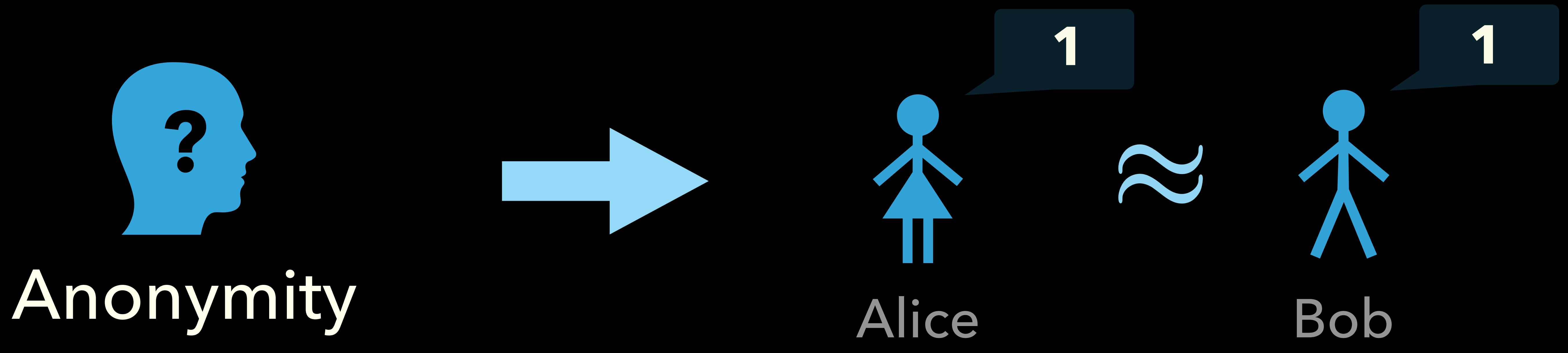
- Break cryptography
- Use side channels

Dolev-Yao models

Concurrent systems where dishonest parties have control over communications

but cryptography is idealised

Privacy = trace equivalence



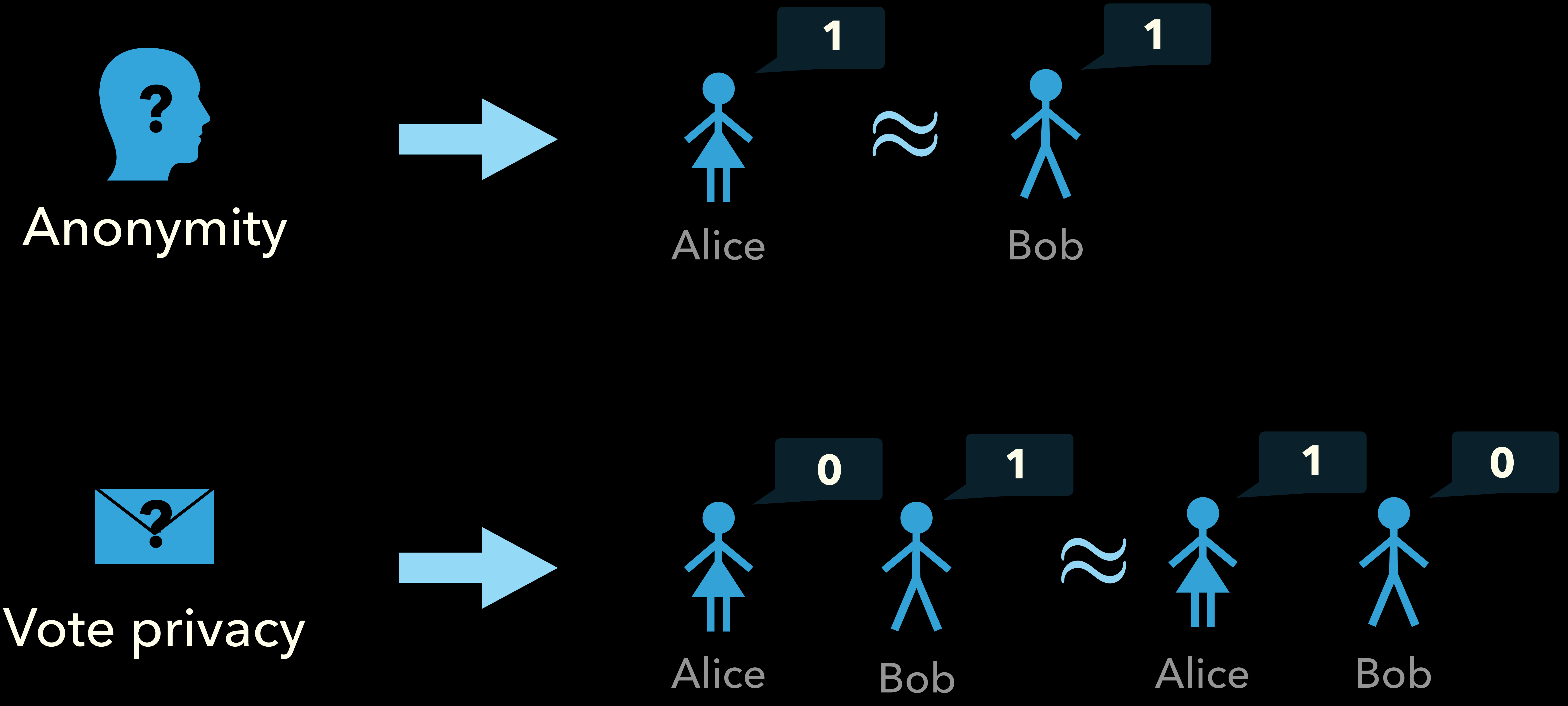
Anonymity

1

1

Alice ≈ Bob

Privacy = trace equivalence


Anonymity

Vote privacy

Proving equivalence

for a fixed number of protocol sessions

Proving equivalence

for a fixed number of protocol sessions

🙂

Decidable

for subterm convergent crypto

☹

coNEXP-complete

in the size of crypto equations + processes

Proving equivalence

for a fixed number of protocol sessions

🙂

Decidable

for subterm convergent crypto

Huge optimisations

for determinate processes

☹

coNEXP-complete

in the size of crypto equations + processes

Problems of scalability

for non-determinate processes

Contributions

- A refinement of trace equivalence

  for processes with structural similarities

- Lifting the optim. of determinate processes

  to any process for this new equivalence

- Reductions by symmetry

Refining trace equivalence

Performances (DEEPSEC)

Determinate

	#Agents	Time
Wide-Mouth Frog (strong secrecy)	10	<1s ✓
	23	3s ✓
Denning-Sacco (strong secrecy)	7	<1s ✓
	29	6s ✓

Non-determinate

	#Agents	Time
Helios Vanilla	6	<1s ⚡
Helios ZKP revote	11	2h 42min ✓
BAC (unlinkability)	4	1s ⚡
	6	>12h ⏰

✓ security property verified

⚡ security property violated

⏰ timeout

Why this gap?

Determinate process. (simplified)

Parallel subprocess operate on different communication channels

→

Partial-order reductions

Commutativity of independent actions

Why this gap?

Determinate process. (simplified)

Parallel subprocess operate on
different communication channels

→

Partial-order reductions

Commutativity of
independent actions

IDEA. Enforce independence natively, in the operational semantics

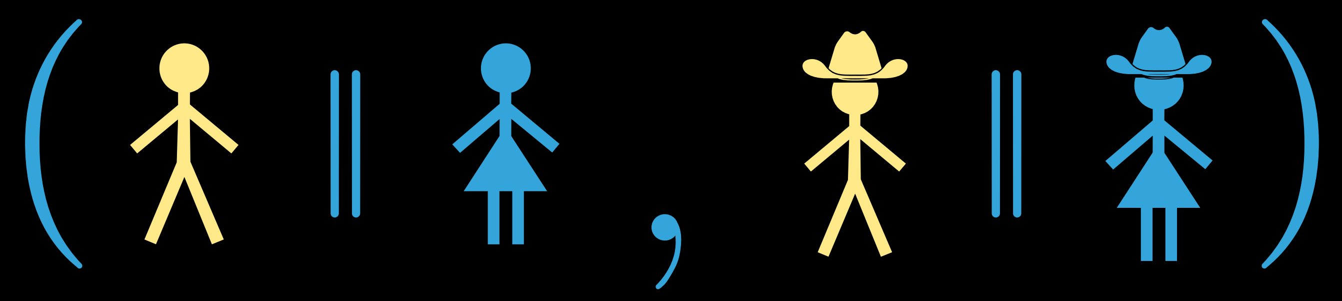
Why this gap?

Determinate process. (simplified)
Parallel subprocess operate on
different communication channels

→

Partial-order reductions
Commutativity of
independent actions

Idea. Enforce independence natively, in the operational semantics



Why this gap?

Determinate process. (simplified)

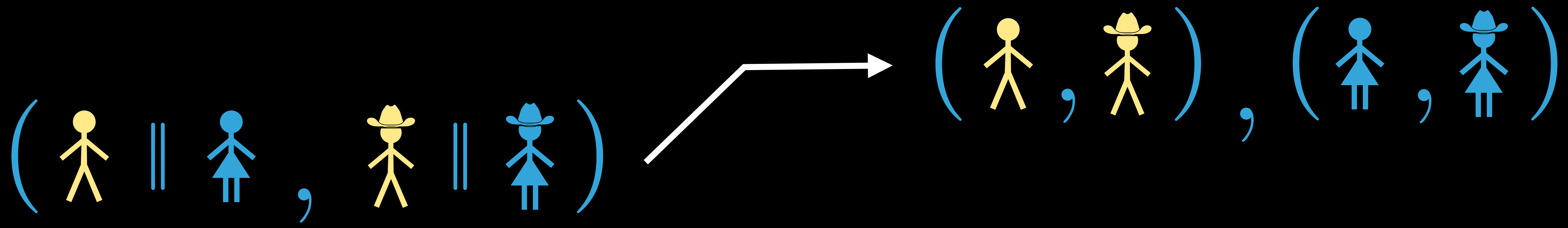
Parallel subprocess operate on different communication channels

→

Partial-order reductions

Commutativity of independent actions

Idea. Enforce independence natively, in the operational semantics



Why this gap?

Determinate process. (simplified)

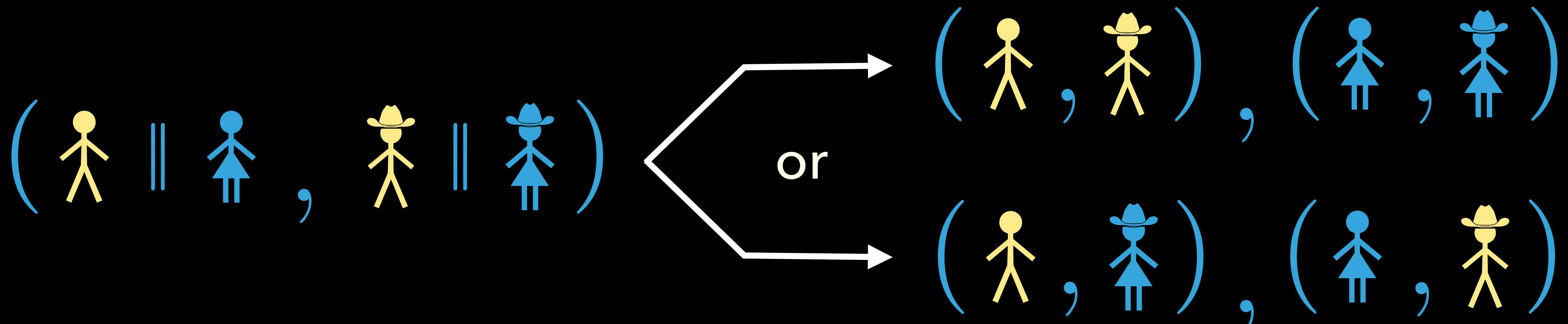
Parallel subprocess operate on different communication channels

→

Partial-order reductions

Commutativity of independent actions

Idea. Enforce independence natively, in the operational semantics



Equivalence by session

(Match)

$$(P_1 \parallel \ldots \parallel P_n,\ Q_1 \parallel \ldots \parallel Q_n) \longrightarrow (P_{\sigma(1)}, Q_1), \ldots, (P_{\sigma(n)}, Q_n)$$

σ permutation of $\{1,\ldots,n\}$

(Exec)

$$(P,Q) \xrightarrow{\alpha} (P',Q')$$

if $P \xrightarrow{\alpha} P'$ and $Q \xrightarrow{\alpha} Q'$

Equivalence by session

(Match)

$$(P_1 \parallel \ldots \parallel P_n,\ Q_1 \parallel \ldots \parallel Q_n) \longrightarrow (P_{\sigma(1)}, Q_1), \ldots, (P_{\sigma(n)}, Q_n)$$

σ permutation of $\{1,\ldots,n\}$

(Exec)

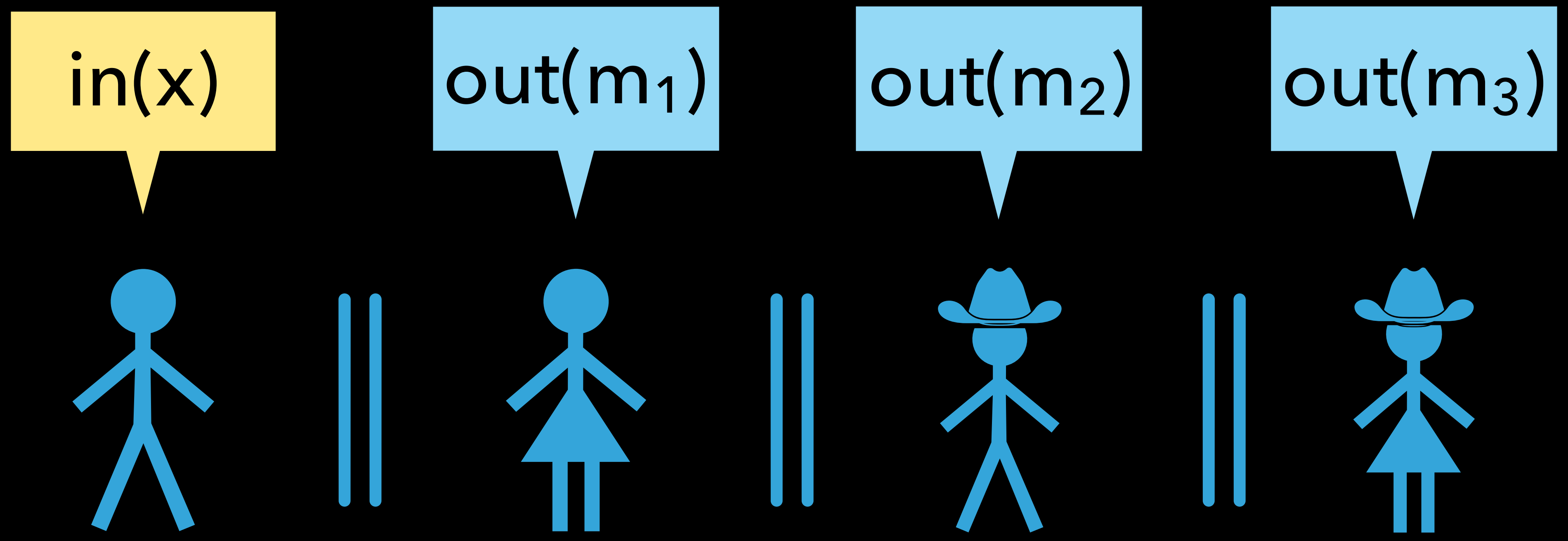
$$(P,Q) \xrightarrow{\alpha} (P',Q')$$

if $P \xrightarrow{\alpha} P'$ and $Q \xrightarrow{\alpha} Q'$

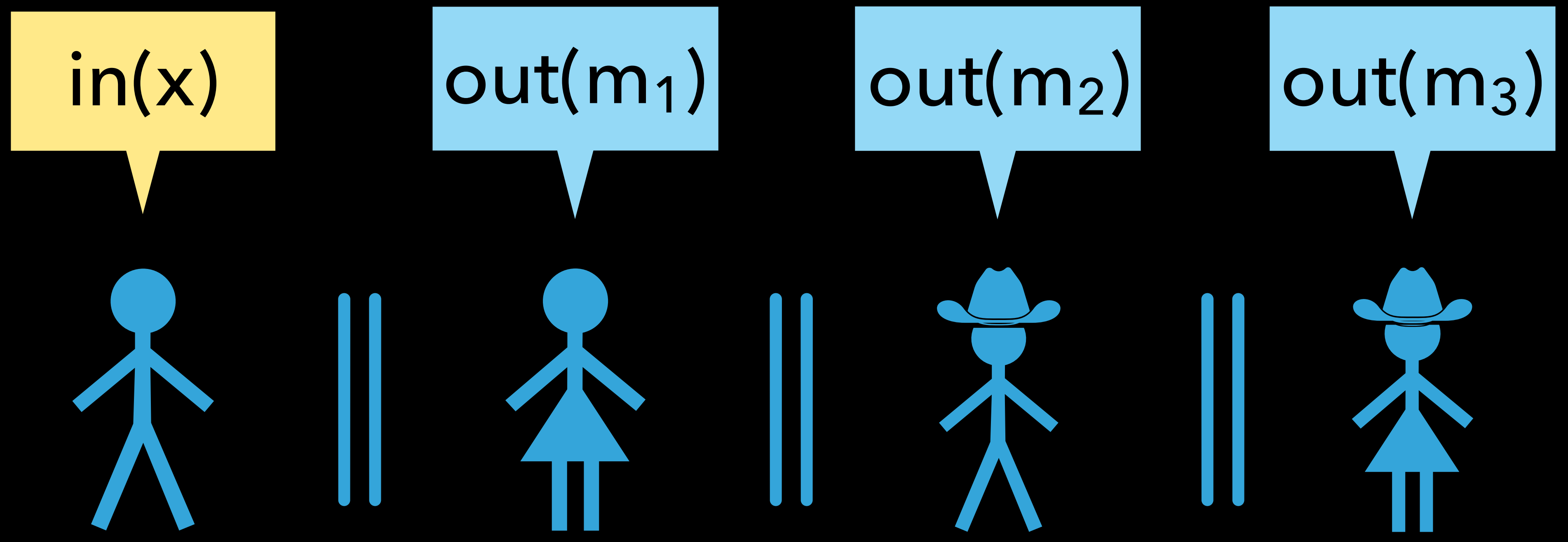
$P \approx_s Q$ ***iff*** Traces(P) ~ Traces(P,Q) ~ Traces(Q)

Reducing the trace space

Partial-order reductions

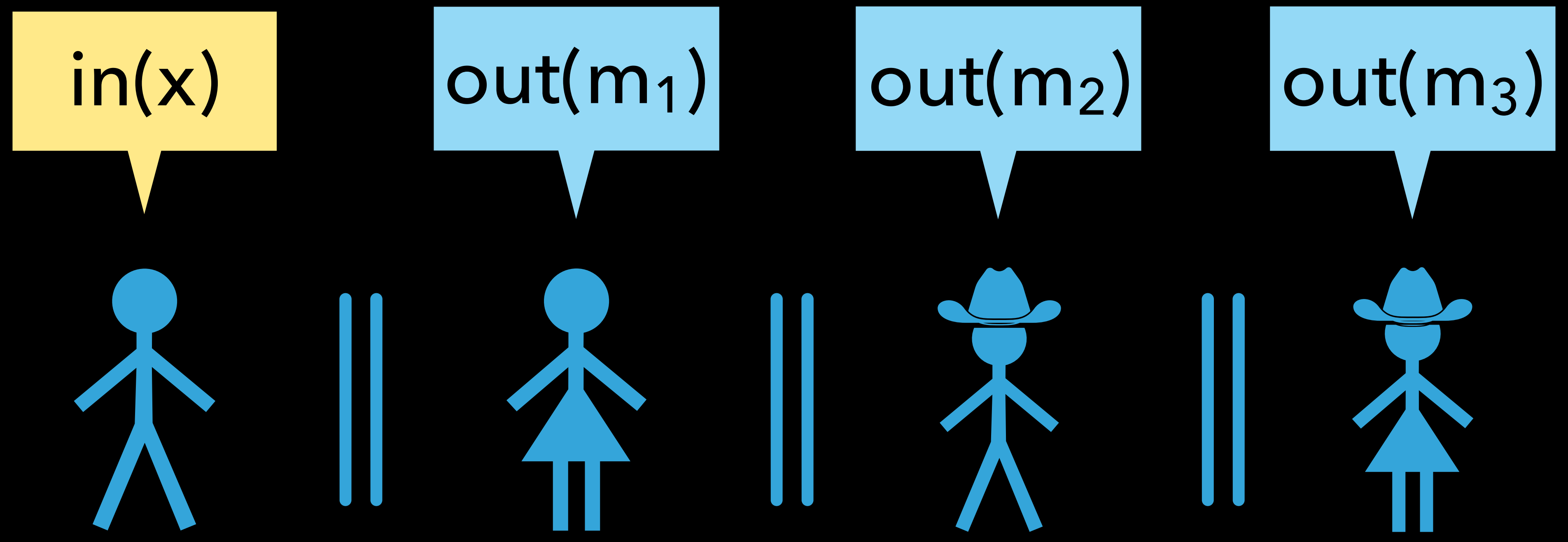


Partial-order reductions



Idea. Perform first (in any order) actions that increase the attacker's knowledge

Partial-order reductions



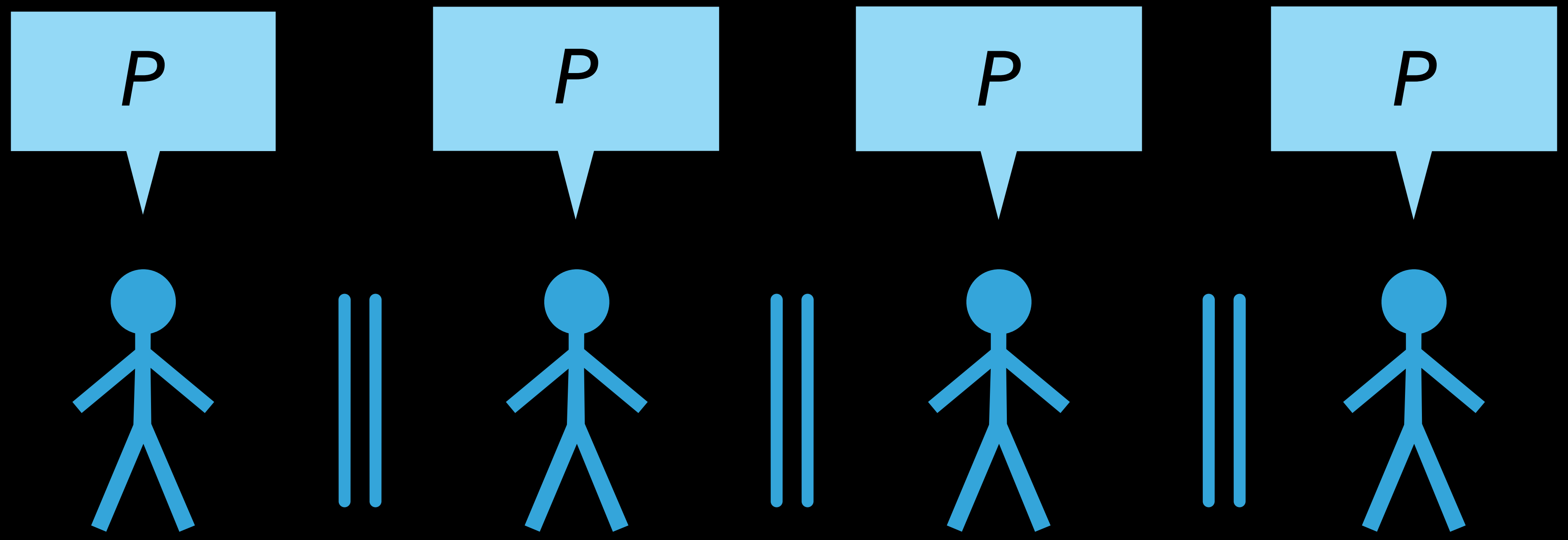
IDEA. Perform first (in any order) actions that increase the attacker's knowledge

In practice:

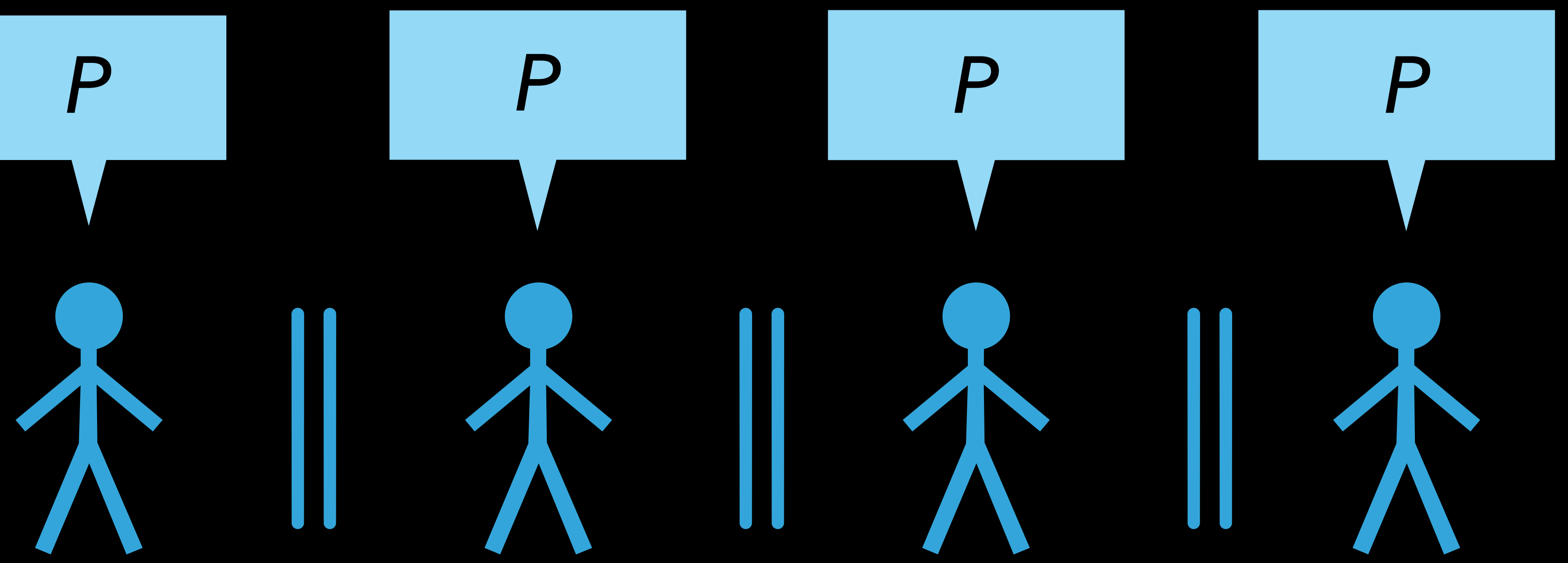
Only consider traces that alternate between

1. deterministic execution of all outputs
2. non-deterministic execution of 1 input

Symmetries

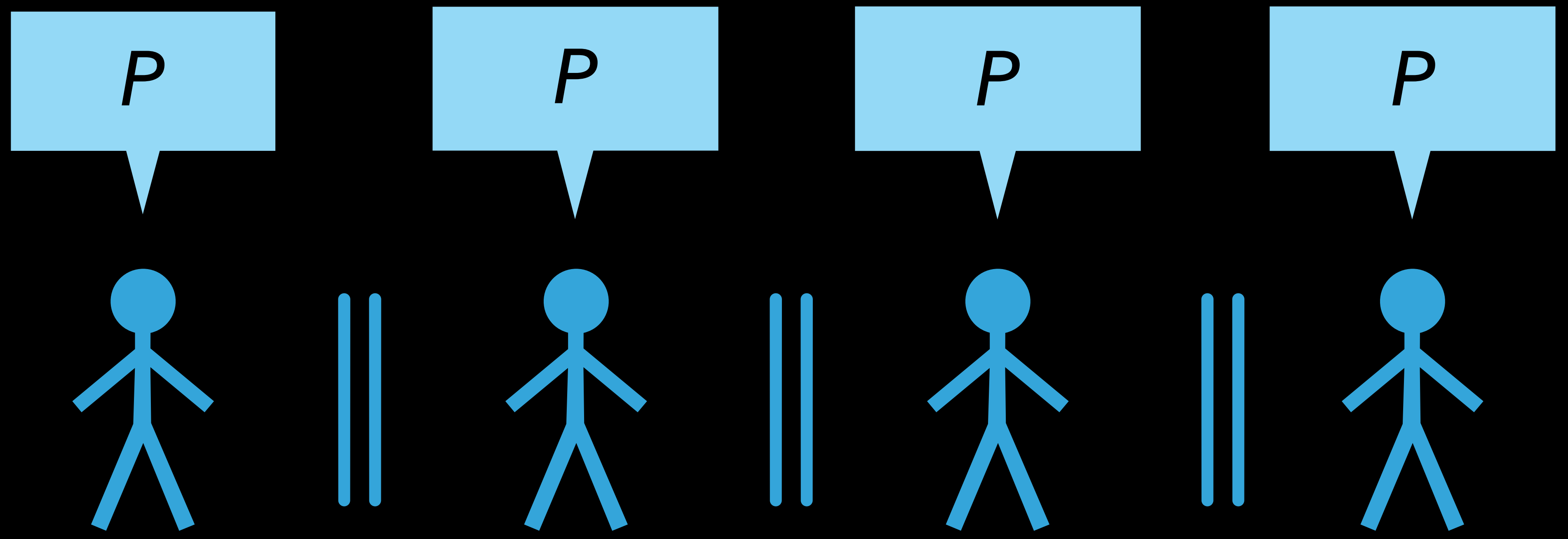


Symmetries



IDEA. Starting the trace with any of the copies of *P* makes no difference

Symmetries



Idea. Starting the trace with any of the copies of *P* makes no difference

In practice:

In transitions $(P_1 \parallel \ldots \parallel P_n, Q_1 \parallel \ldots \parallel Q_n) \to (P_{\sigma(1)}, Q_1), \ldots, (P_{\sigma(n)}, Q_n)$ only consider permutations σ up to the equivalence relation:

$$\sigma \sim \sigma' \quad \textit{iff} \quad \exists u,v.\ \sigma' = u\sigma v \ \text{ and } \forall i.\ P_{u(i)} = P_i,\ Q_{v(i)} = Q_i$$

Results

Experimental results

Work in progress

Conclusion

efficient detection of logical flaws in security protocols

Conclusion

efficient detection of logical flaws in security protocols

DONE

- ⊕ A refinement of trace equivalence
  for lighter proofs in practical scenarios
- ⊕ Partial-order reductions
  as a built-in mechanism of the new equivalence

FUTURE

- ➡ Implementation
  in the DEEPSEC prover
- ➡ Catch false negatives