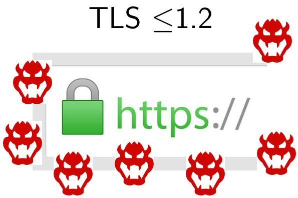
POR for Security Protocol Equivalences:
Beyond Action-Determinism

ESORICS'18

David Baelde, Stéphanie Delaune, Lucca Hirschi

September 3rd, 2018

Designing secure cryptographic protocols



Extremely complex setting

- insecure network
- active attacker 
- concurrent executions

Designing secure cryptographic protocols

Extremely complex setting

- insecure network
- active attacker
- concurrent executions

Formal methods & symbolic model

- mathematical & exhaustive analysis
- formal guarantees
- automated or automatic

Symbolic Model (Dolev-Yao)

Cryptographic primitives assumed perfect

- primitives modelled as function symbols & equational theory
- *e.g.* ,  $\longmapsto$ $\mathsf{enc}(\cdot,\cdot), \mathsf{dec}(\cdot,\cdot)$ & $\mathsf{dec}(\mathsf{enc}(m,k),k) = m$

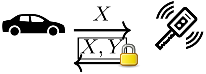
Symbolic Model (Dolev-Yao)

Cryptographic primitives assumed perfect

- primitives modelled as function symbols & equational theory
- *e.g.*  ,  $\longmapsto$ $\mathsf{enc}(\cdot,\cdot), \mathsf{dec}(\cdot,\cdot)$ & $\mathsf{dec}(\mathsf{enc}(m,k),k) = m$

Security protocols

- in a process algebra
- each party $\mapsto$ process



$\mapsto$

P =
$\mathsf{in}(X)$.
$\mathsf{new}\ Y$.
$\mathsf{out}(\mathsf{enc}((X,Y),k))$

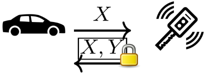
Symbolic Model (Dolev-Yao)

Cryptographic primitives assumed perfect

- primitives modelled as function symbols & equational theory
- *e.g.* , $\longmapsto$ $\mathsf{enc}(\cdot,\cdot), \mathsf{dec}(\cdot,\cdot)$ & $\mathsf{dec}(\mathsf{enc}(m,k),k) = m$

Security protocols

- in a process algebra
- each party $\mapsto$ process



$\mapsto$ P =
$\mathsf{in}(X)$.
$\mathsf{new}\ Y$.
$\mathsf{out}(\mathsf{enc}((X,Y),k))$

Attacker = network (worst case scenario:)

- eavesdrop: he learns all protocol outputs — $\mathsf{out}(c,u) \rightarrow (u)$
- injection: he chooses all protocol inputs — $\mathsf{in}(c,v) \leftarrow (v)$

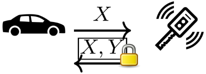
Symbolic Model (Dolev-Yao)

Cryptographic primitives assumed perfect

- primitives modelled as function symbols & equational theory
- *e.g.* ,  $\longmapsto$ $\mathsf{enc}(\cdot,\cdot), \mathsf{dec}(\cdot,\cdot)$ & $\mathsf{dec}(\mathsf{enc}(m,k),k) = m$

Security protocols

- in a process algebra
- each party $\mapsto$ process



$\mapsto$ $P =$
$\mathsf{in}(X).$
$\mathsf{new}\ Y.$
$\mathsf{out}(\mathsf{enc}((X,Y),k))$

Attacker = network (worst case scenario:)

- eavesdrop: he learns all protocol outputs — $\mathsf{out}(c,u) \rightarrow (u)$
- injection: he chooses all protocol inputs — $\mathsf{in}(c,v) \leftarrow (v)$

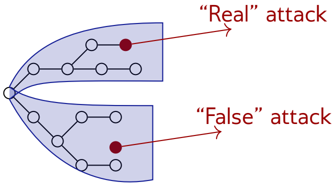
Automated verification

- secrecy, authentication, ... $\rightsquigarrow$ reachability (not this talk)
- privacy, ... $\rightsquigarrow$ **equivalence** between configurations (this talk)

Symbolic Verification of Equivalence: State-of-the-Art

Equivalence: $\forall A, \; P|A \approx Q|A$ Undecidable ☹

Semi-decision for ∞ sessions



- over-approximations of & semantics
- strong form of $\approx$ (*i.e.* diff-equivalence)
- tools: Tamarin, ProVerif, Maude-NPA

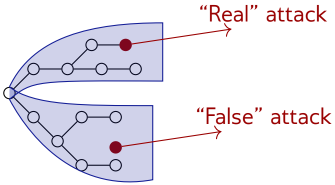
Lack of precision

e.g. too imprecise for untraceability

Symbolic Verification of Equivalence: State-of-the-Art

Equivalence: $\forall A, \; P|A \approx Q|A$ Undecidable ☹

Semi-decision for ∞ sessions

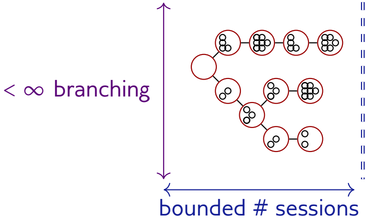


- over-approximations of [adversary] & semantics
- strong form of ≈ (*i.e.* diff-equivalence)
- tools: Tamarin, ProVerif, Maude-NPA

Lack of precision

e.g. too imprecise for untraceability

Decision for < ∞ sessions

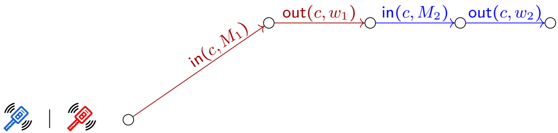


- bound number of sessions
- symbolic semantics
- exhaustive exploration of symbolic executions
- tools: DeepSec, Apte, Akiss, Spec

...but **scalability issue**

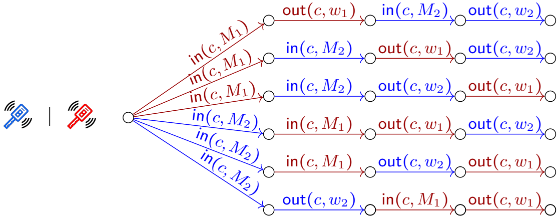
A concurrency issue

Problem: **concurrency** $\rightsquigarrow$ state space explosion



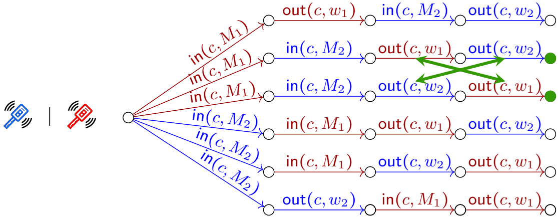
A concurrency issue

Problem: **concurrency** $\rightsquigarrow$ state space explosion



A concurrency issue

Problem: **concurrency** $\rightsquigarrow$ state space explosion

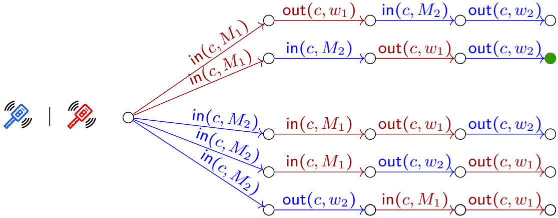


Partial Order Reductions (POR) (Model-Checking)

Leverage independencies of actions to avoid redundant interleavings

A concurrency issue

Problem: **concurrency** $\rightsquigarrow$ state space explosion



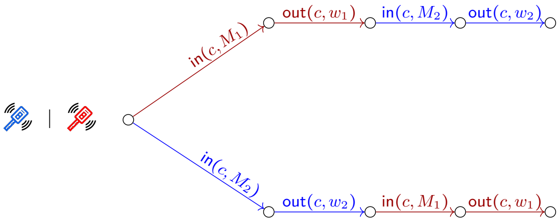
Partial Order Reductions (POR)

(Model-Checking)

Leverage independencies of actions to avoid redundant interleavings

A concurrency issue

Problem: **concurrency** $\rightsquigarrow$ state space explosion

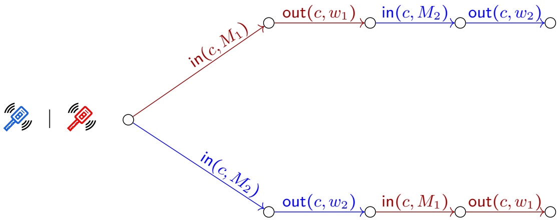


Partial Order Reductions (POR) (Model-Checking)

Leverage independencies of actions to avoid redundant interleavings

A concurrency issue

Problem: **concurrency** $\rightsquigarrow$ state space explosion



Partial Order Reductions (POR) (Model-Checking)

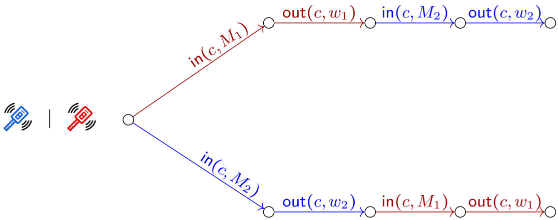
Leverage independencies of actions to avoid redundant interleavings

State-of-the-art of POR for $\approx$ in security: [Baelde, Delaune, Hirschi,'14,'15,'17]

- brings significant speedups in all tools
- but only for restricted class of protocols/properties

A concurrency issue

Problem: **concurrency** $\rightsquigarrow$ state space explosion



Partial Order Reductions (POR)

(Model-Checking)

Leverage independencies of actions to avoid redundant interleavings

State-of-the-art of POR for $\approx$ in security: [Baelde, Delaune, Hirschi,'14,'15,'17]

- brings significant speedups in all tools
- but only for restricted class of protocols/properties

Restriction is problematic: excludes important properties *e.g.* untraceability

Our Contributions

This Work

New POR obtained through a different approach removes this restriction and deals with the general case

(Contributions in red)

Our Contributions

This Work

New POR obtained through a different approach removes this restriction and deals with the general case

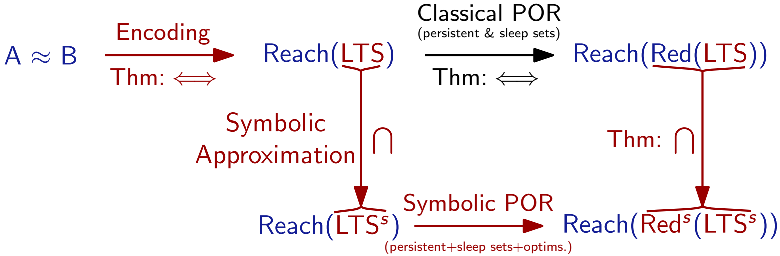
$A \approx B$ —Encoding (Thm: $\Longleftrightarrow$)→ Reach(LTS) —Classical POR (persistent & sleep sets) (Thm: $\Longleftrightarrow$)→ Reach(Red(LTS))

(Contributions in red)

Our Contributions

This Work

New POR obtained through a different approach removes this restriction and deals with the general case

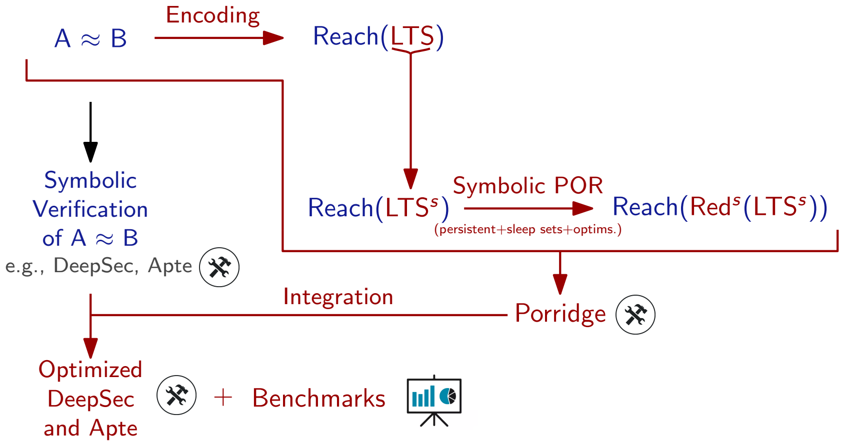


(Contributions in red)

Our Contributions

This Work

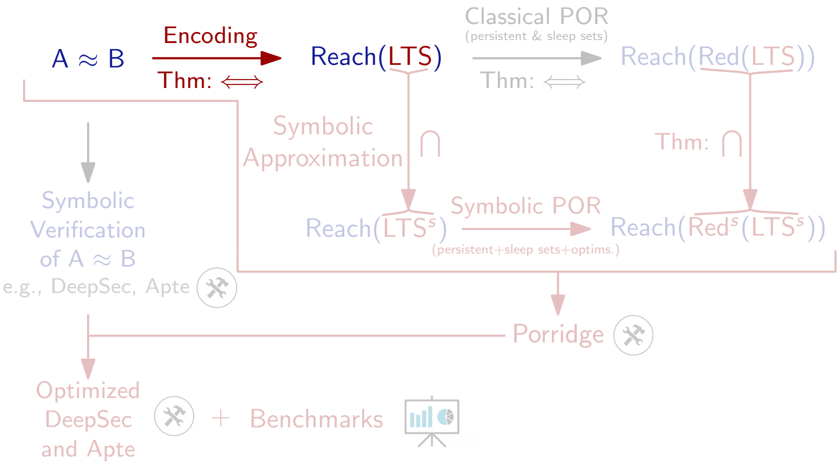
New POR obtained through a different approach removes this restriction and deals with the general case



(Contributions in red)

Outline

From $A \approx B$ to Reach(LTS)



Applied-π Calculus

- Process:

P, Q	:=	$\mathsf{in}(c, x).P$	input
	$\mid$	$\mathsf{out}(c, m).P$	output
	$\mid$	$P \mid Q$	parallel
	$\mid$	$P + Q$	choice
	$\mid$	$\texttt{if}\ Test\ \texttt{then}\ P\ \texttt{else}\ Q$	conditional
	$\mid$	$\mathsf{new}\ X.P$	creation of name
	$\mid$	0	null

Applied-π Calculus

- **Process**:

P, Q	$:=$	$\mathsf{in}(c,x).P$	input
	$\mid$	$\mathsf{out}(c,m).P$	output
	$\mid$	$P \mid Q$	parallel
	$\mid$	$P + Q$	choice
	$\mid$	$\texttt{if}\ Test\ \texttt{then}\ P\ \texttt{else}\ Q$	conditional
	$\mid$	$\mathsf{new}\ X.P$	creation of name
	$\mid$	0	null

- **Frame** (ϕ): the set of messages revealed to  ('s knowledge)

$$\phi = \{ \underbrace{w_{c,1}}_{\text{variable}} \mapsto \underbrace{\mathsf{enc}(m,k)}_{\text{out. message}}, w_{d,1} \mapsto k \}$$

- **Configuration**: $A = (\mathcal{P}; \phi)$ for $\mathcal{P}$ multiset of processes and frame ϕ

Applied-π Calculus

- **Process**: $P, Q \;:=\; \mathsf{in}(c,x).P$ — input
  - $\mid\; \mathsf{out}(c,m).P$ — output
  - $\mid\; P \mid Q$ — parallel
  - $\mid\; P + Q$ — choice
  - $\mid\; \texttt{if}\; Test\; \texttt{then}\; P\; \texttt{else}\; Q$ — conditional
  - $\mid\; \mathsf{new}\; X.P$ — creation of name
  - $\mid\; 0$ — null
- **Frame** (ϕ): the set of messages revealed to the attacker (attacker's knowledge)

$$\phi = \{\underbrace{w_{c,1}}_{\text{variable}} \mapsto \underbrace{\mathsf{enc}(m,k)}_{\text{out. message}}, w_{d,1} \mapsto k\}$$

- **Configuration**: $A = (\mathcal{P};\phi)$ for $\mathcal{P}$ multiset of processes and frame ϕ
- **Semantics**:

$$(\{P, \mathsf{out}(c,u).Q\};\ \phi) \xrightarrow{\mathsf{out}(c,\mathsf{w}_{c,i})} (\{P,Q\};\ \phi \cup \{\mathsf{w}_{c,i} \mapsto u\}) \text{ when } i = |\phi|_c$$

$$(\{P, \mathsf{in}(c,x).Q\};\ \phi) \xrightarrow{\mathsf{in}(c,M)} (\{P, Q\{x \mapsto M\phi\}\};\ \phi)$$

when M is a term over ϕ

Encoding Requirements

Trace Equivalence

$A \approx B$ when: $\forall A \xrightarrow{t}{}^{*} A', \exists B \xrightarrow{t}{}^{*} B'$ such that $\phi(A') \sim \phi(B')$ (and conversely)

$\downarrow$

Reachability in a LTS

- An LTS is given by:
  - a set of *states* S,
  - a set of *transitions* T, and
  - a *partial function* $\delta : S \times T \mapsto S$ (deterministic!). $s \xrightarrow{\alpha} s'$ when $s' = \delta(s, \alpha)$
- A state $s \in S$ is *final* when $s \notin \mathrm{dom}(\delta)$
- Given: initial state s_0 and $S_{\text{bad}} \subseteq S$ (bad states),
  Reach($\cdot$): no final, bad state in S_{bad} is reachable from s_0

Desired property:

$$\forall A, B.\ \ A \approx B \iff \text{Reach}(\text{LTS}(A, B))$$

Encoding

First attempt

- T=transitions of configurations
- S=pairs of sets of configurations (noted $\langle| \mathbf{A} \approx \mathbf{B} |\rangle$) $\quad s_0 = \langle| \{A\} \approx \{B\} |\rangle$
- Transition function δ:

$$\langle| \mathbf{A} \approx \mathbf{B} |\rangle \xrightarrow{\alpha} \langle| \mathbf{A}_\alpha \approx \mathbf{B}_\alpha |\rangle \text{ with } \mathbf{X}_\alpha = \{C' \;:\; C \in \mathbf{X},\; C \xrightarrow{\alpha} C'\}$$

- $\langle| \mathbf{A} \approx \mathbf{B} |\rangle \in S_{\text{bad}}$ if $\phi(\mathbf{A}) \not\sim \phi(\mathbf{B})$

Encoding

First attempt

- T=transitions of configurations
- S=pairs of sets of configurations (noted $\langle\!|\mathbf{A} \approx \mathbf{B}|\!\rangle$) $\quad s_0 = \langle\!|\{A\} \approx \{B\}|\!\rangle$
- Transition function δ:

$$\langle\!|\mathbf{A} \approx \mathbf{B}|\!\rangle \xrightarrow{\alpha} \langle\!|\mathbf{A}_\alpha \approx \mathbf{B}_\alpha|\!\rangle \text{ with } \mathbf{X}_\alpha = \{C' \,:\, C \in \mathbf{X},\ C \xrightarrow{\alpha} C'\}$$

- $\langle\!|\mathbf{A} \approx \mathbf{B}|\!\rangle \in S_{\mathsf{bad}}$ if $\phi(\mathbf{A}) \not\sim \phi(\mathbf{B})$

Unsound! ...because witnesses can be lost

$\exists\langle\!|\mathbf{A} \approx \mathbf{B}|\!\rangle \xrightarrow{\alpha} \langle\!|\mathbf{A}' \approx \mathbf{B}'|\!\rangle$ such that $\phi(\mathbf{A}) \not\sim \phi(\mathbf{B})$ but $\phi(\mathbf{A}') \sim \phi(\mathbf{B}')$!

$$\forall A, B.\ \ A \approx B \not\Leftarrow \mathsf{Reach}(\mathsf{LTS}(A, B))$$

Example:

$$\langle\!|\{\mathsf{out}(0) + \mathsf{out}(k).\alpha\} \approx \{\mathsf{out}(1) + \mathsf{out}(k).\alpha\}|\!\rangle \xrightarrow{\mathsf{out}(w)} \langle\!|\mathbf{A} \approx \mathbf{B}|\!\rangle \xrightarrow{\alpha} \langle\!|\mathbf{A}' \approx \mathbf{B}'|\!\rangle$$

Encoding

Definition: LTS(A, B)

- S=pairs of sets of configurations or *ghost* configurations $(\bot_i; \phi)$
  keep track of "dead" witnesses
- ...

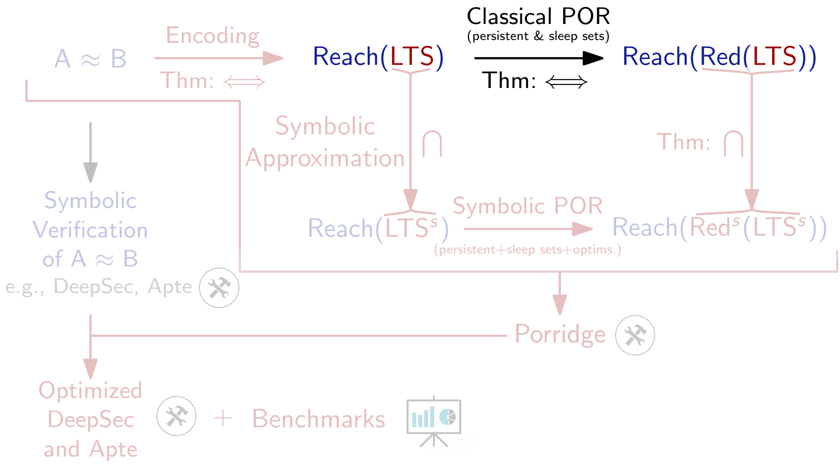
Property: If $\phi(\mathbf{A}) \not\sim \phi(\mathbf{B})$ and $\langle|\mathbf{A} \approx \mathbf{B}|\rangle \xrightarrow{t} \langle|\mathbf{A}' \approx \mathbf{B}'|\rangle$ then $\phi(\mathbf{A}') \not\sim \phi(\mathbf{B}')$

Theorem

$$\forall A, B.\ \ A \approx B \Longleftrightarrow \text{Reach}(\text{LTS}(A, B))$$

Outline

POR over Reach(LTS)



Dependencies

Independence relation between transitions ($\alpha, \beta \in T$, $s \in S$):

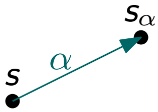
$\alpha \leftrightarrow_s \beta$: s •

Intuitively: $\alpha \leftrightarrow_s \beta$ when α and β commute from s

Dependencies

Independence relation between transitions ($\alpha, \beta \in T$, $s \in S$):

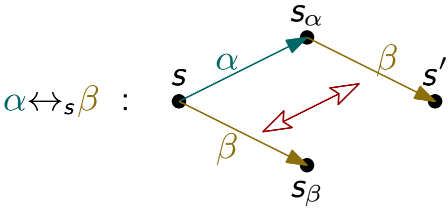
$\alpha \leftrightarrow_s \beta$:



Intuitively: $\alpha \leftrightarrow_s \beta$ when α and β commute from s

Dependencies

Independence relation between transitions ($\alpha, \beta \in T$, $s \in S$):

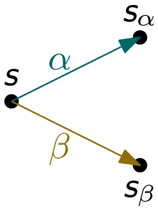


Intuitively: $\alpha \leftrightarrow_s \beta$ when α and β commute from s

Dependencies

Independence relation between transitions ($\alpha, \beta \in T$, $s \in S$):

$\alpha \leftrightarrow_s \beta$:

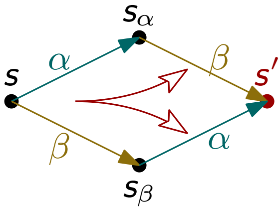


Intuitively: $\alpha \leftrightarrow_s \beta$ when α and β commute from s

Dependencies

Independence relation between transitions ($\alpha, \beta \in T$, $s \in S$):

$\alpha \leftrightarrow_s \beta$:



Intuitively: $\alpha \leftrightarrow_s \beta$ when α and β commute from s

Persistent Sets

Goal:

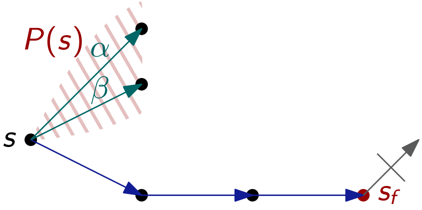
- Define $P : S \mapsto 2^T$
- From s, only explore transitions in $P(s)$ $\rightsquigarrow$ persistent traces

Persistent Sets

Goal:

- Define $P : S \mapsto 2^T$
- From s, only explore transitions in $P(s)$ $\rightsquigarrow$ persistent traces

Property achieved by persistent sets:



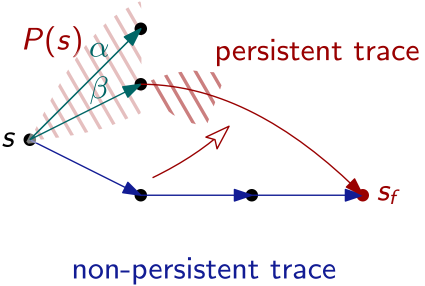
non-persistent trace

Persistent Sets

Goal:

- Define $P : S \mapsto 2^T$
- From s, only explore transitions in $P(s)$ $\rightsquigarrow$ persistent traces

Property achieved by persistent sets:



Theorem: Reach(LTS(A, B)) $\Longleftrightarrow$ Reach(Red(LTS(A, B)))

POR over Reach(LTS)

Persistent sets:

- can be computed as fixed points via forward explorations (stubborn sets)
- standard model-checking: syntactical analyses but not on LTS(A, B)

POR over Reach(LTS)

Reach(LTS) $\xrightarrow[\text{Thm: } \Longleftrightarrow]{\text{Classical POR}}$ Reach(Red(LTS))

Persistent sets:

- can be computed as fixed points via forward explorations (stubborn sets)
- standard model-checking: syntactical analyses but not on LTS(A, B)

We also leverage: sleep sets based on backward analysis

POR over Reach(LTS)

Reach(LTS) $\xrightarrow[\text{Thm: } \iff]{\text{Classical POR}}$ Reach(Red(LTS))

Persistent sets:

- can be computed as fixed points via forward explorations (stubborn sets)
- standard model-checking: syntactical analyses but not on LTS(A, B)

We also leverage: sleep sets based on backward analysis

Theorem: Reach(LTS(A, B)) $\iff$ Reach(Red(LTS(A, B)))

Are we done?

POR over Reach(LTS)

Persistent sets:

- can be computed as fixed points via forward explorations (stubborn sets)
- standard model-checking: syntactical analyses but not on LTS(A, B)

We also leverage: sleep sets based on backward analysis

Theorem: Reach(LTS(A, B)) $\Longleftrightarrow$ Reach(Red(LTS(A, B)))

Are we done?

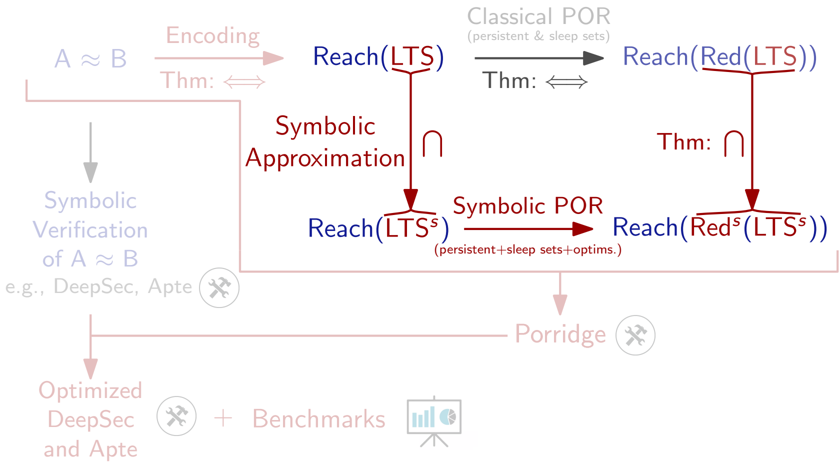
No ☹: How to efficiently compute those sets? (∞ branching)
How to combine POR with symbolic explorations? (verifiers)

Solution to both problems: symbolic approximations

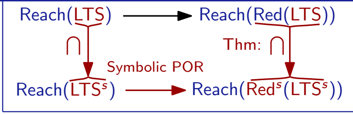
- over-approximation of LTS(A, B)
- over-approximation of persistent and sleep sets

Outline

Symbolic POR over Reach(LTS^s)



Symbolic Abstraction of LTS



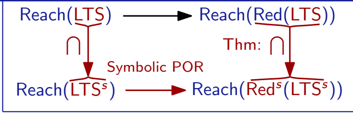
Symbolic LTS and semantics use symbolic states

$$\langle \mathbf{A}^s \approx \mathbf{B}^s \rangle_{\mathcal{C}}$$

where $\mathcal{C}$ is a set of (dis)equality *constraints* and $C^s \in \mathbf{A}^s \cup \mathbf{B}^s$ contains *input variables*.

- Input messages are replaced by variables
- Transitions branch on conditionals + extend $\mathcal{C}$

Symbolic Abstraction of LTS



Symbolic LTS and semantics use symbolic states

$$\langle \mathbf{A}^s \approx \mathbf{B}^s \rangle_{\mathcal{C}}$$

where $\mathcal{C}$ is a set of (dis)equality *constraints* and
$C^s \in \mathbf{A}^s \cup \mathbf{B}^s$ contains *input variables*.

- Input messages are replaced by variables
- Transitions branch on conditionals + extend $\mathcal{C}$
- $\mathcal{C}$ induces $\mathcal{S}ol$: *Only need to detect immediate syntactic contradictions*
- S is an *abstraction* of s when $s = S\theta$ for $\theta \in \mathcal{S}ol(\mathcal{C})$

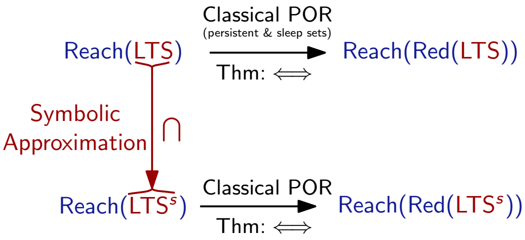
Results:

- *Completeness*: concrete transitions mimicked by symbolic ones
- *No soundness*: $\mathcal{S}\theta$ might be unreachable
- *Weak soundness*: S and s have the same enabled actions $(E(\cdot))$

Symbolic POR

POR(LTS^s)?

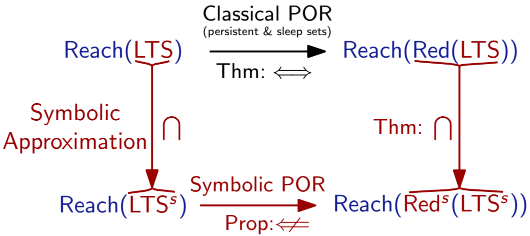
- POR on symbolic LTS Red(LTS^s) $\rightsquigarrow$ extremely poor reduction ☹



Symbolic POR

POR(LTS^s)?

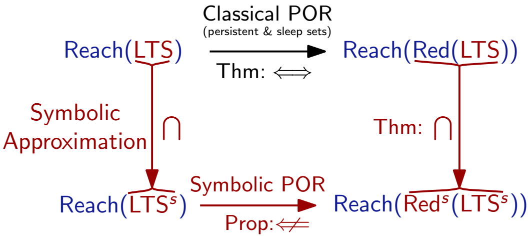
- POR on symbolic LTS Red(LTS^s) $\rightsquigarrow$ extremely poor reduction ☹
- LTS^s used to over-approximate Red(LTS): $Red^s(LTS^s)$ $\rightsquigarrow$ good reduction ☺



Symbolic POR

POR(LTS^s)?

- POR on symbolic LTS Red(LTS^s) $\rightsquigarrow$ extremely poor reduction ☹
- LTS^s used to over-approximate Red(LTS): Red^s(LTS^s) $\rightsquigarrow$ good reduction ☺



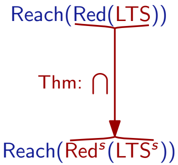
Symbolic POR (Red$^s(\cdot)$) is based on:

- $\Leftrightarrow_S$ (for symbolic state S): a *sound abstraction of* $\leftrightarrow_s$ for any $s = S\theta$

  $\Leftrightarrow \neq \leftrightarrow$ (LTS^s) !

- symbolic, persistent and sleep sets computed with $\Leftrightarrow$ on “plausible” symbolic executions

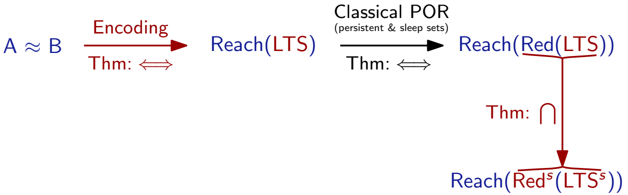
Putting Everything Together



Theorem:

$\exists$ concrete reduced execution in LTS(A, B) reaching a bad state
$\iff$ $\exists$ symbolic reduced execution in LTS$^s(A, B)$ which abstracts a concrete execution reaching a bad state

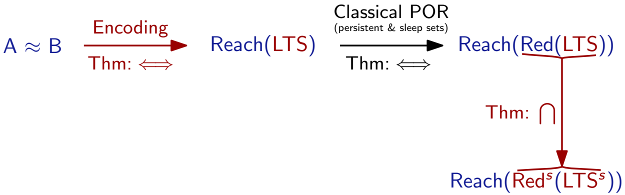
Putting Everything Together



Theorem:

$A \not\approx B$
$\iff$ a bad state can be reached from $s_0 = \langle|\{A\} \approx \{B\}|\rangle$ in LTS(A, B)
$\iff$ $\exists$ concrete reduced execution in LTS(A, B) reaching a bad state
$\iff$ $\exists$ symbolic reduced execution in LTS$^s(A, B)$ which abstracts a concrete execution reaching a bad state

Putting Everything Together



Theorem:

$A \not\approx B$

$\iff$ a bad state can be reached from $s_0 = \langle | \{A\} \approx \{B\} | \rangle$ in LTS(A, B)

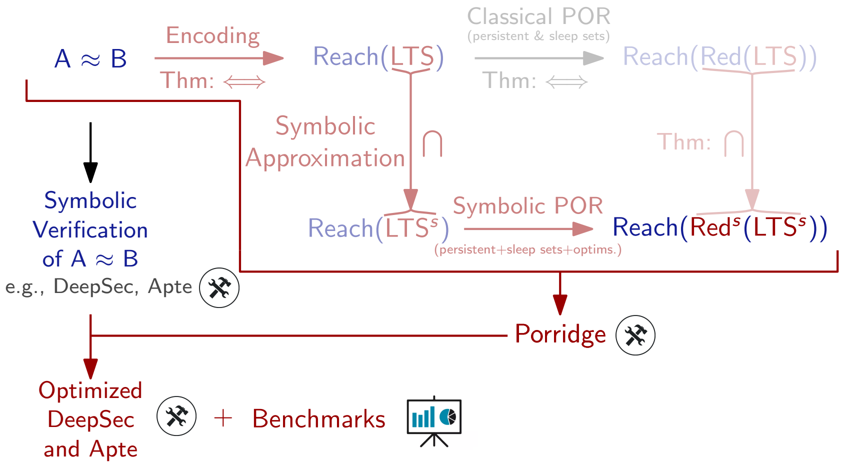
$\iff$ $\exists$ concrete reduced execution in LTS(A, B) reaching a bad state

$\iff$ $\exists$ symbolic reduced execution in LTS$^s(A, B)$ which abstracts a concrete execution reaching a bad state

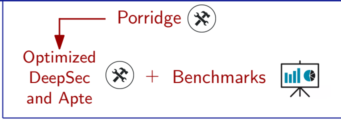
- The set of symbolic reduced executions in LTS$^s(A, B)$ can be computed !
- State-of-the-art verifiers can decide if a symbolic execution has a concretization reaching a bad state

Outline

Implementation and Benchmarks



Implementation and Integration



PORridge

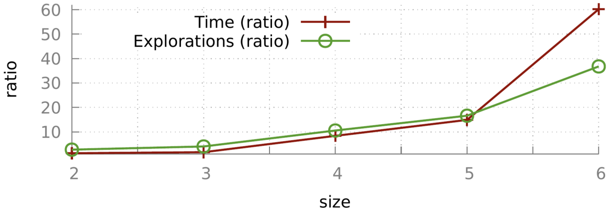
- Standalone OCaml library performing symbolic POR computations
- Heavily relies on hash-consing, not yet on multiple cores

Integration in Apte/DeepSec

- Compute reduced set of symbolic traces using Porridge
- Restrict explorations to the given reduced set

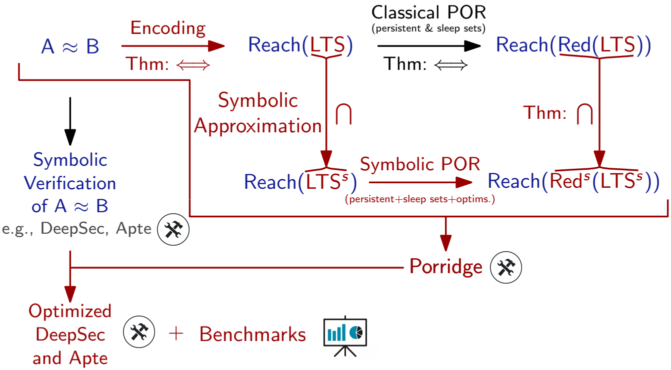
Benchmarks using DeepSec

- Various case studies: BAC, PA (ePassport), Feldhofer (RFID), etc.
- Speedups up to 60 (PA ANO, 6 sessions):



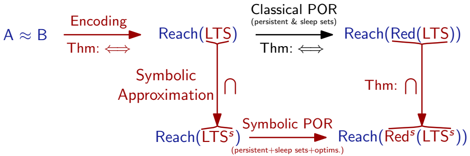
Conclusion

Summary



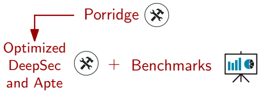
- First POR techniques for ≈ and a large class of protocols
- State-of-the-art verifiers already benefit from our techniques
- Unlock traditional POR techniques (Model-Checking) for ≈ (Security)

Future work



Theory:

- traditional POR techniques (Model-Checking) $\rightsquigarrow$ $\approx$ (Security)
- handle more dependencies based on data (dependency constraint)
- avoid interleaving $\rightsquigarrow$ true concurrency semantics? (event structures)



Implementation & Integration:

- Porridge: multicore, better trade-off precision/pre-computation cost
- interactive integration

Backup Slides

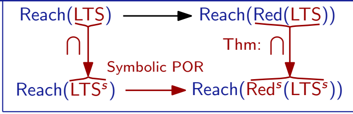
POR & Trace Equivalence

What about trace equivalence ($\approx_c$) ?

e.g., $(\mathsf{in}(c_1, x) \;\mid\; \mathsf{out}(c_2, m)) \not\approx (\mathsf{out}(c_2, m).\mathsf{in}(c_1, x))$

- $\rightsquigarrow$ same swaps are possible (≡ same sequential dependencies)
- Lemma: A, B action-det, $A \approx B \Rightarrow$ same sequential dependencies

Symbolic Dependencies



Compute on S a *sound abstraction of* $\leftrightarrow_s$ for any $s = S\theta$.

Enabled actions

If $A \Leftrightarrow_S^{ee} B$, and $\alpha, \beta \in E(s)$, then $\alpha \leftrightarrow_s \beta$.

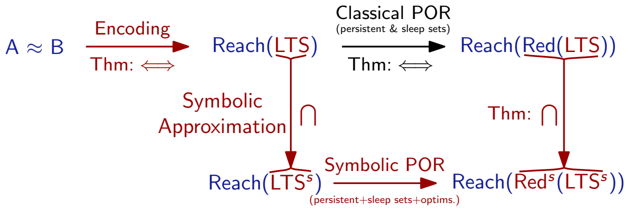
- Simply explore all transitions.
- Need to consider all cases for conditionals.

Disabled actions

If $A \Leftrightarrow_S^{de} B$, $\alpha \notin E(s)$ and $\beta \in E(s)$, then $\alpha \leftrightarrow_s \beta$.

- Either A is not executable in S' for any S' such that $S \overset{B}{\mapsto} S'$,
- or A is executable in S but A/B are not of the form in$(c, \mathsf{X}^{c,i}, W)$/out$(d, \mathsf{w}_{d,j})$ with $\mathsf{w}_{d,j} \in W$.

Optimization Handling Conditional Branching



$(P_1; \emptyset) \not\approx (P_2; \emptyset)$
$\iff$ a bad state can be reached from $s_0 = \langle | \{P\} \approx \{Q\} | \rangle$ in LTS(P, Q)
$\iff$ $\exists$ reduced execution in LTS(P, Q) reaching a bad state
$\iff$ $\exists$ symbolic reduced execution in LTS$^s(P, Q)$ whose concretization reaches a bad state

Final optimization

- branching due to conditional + non-det. $\rightsquigarrow \neq$ state space explosion ☹
- we address this explosion by soundly "collapsing" most of conditionals: Red^s(LTS^s) = Red^s(SimplCond(LTS^s)) but SimplCond(LTS^s) <<< LTS^s ☺

POR v1: Compressed Strategy

Compressed semantics $\rightarrow_c$

- Polarities: Negative: $\mathsf{out}().P, (P_1 \mid P_2), 0$ & Positive: $\mathsf{in}().P$
- Negative: explored greedily, in a given order *e.g.* $c_1 < c_2$
- Positive: explored only when $\nexists$ Negative,
  - chooses one and put it under focus
  - focus is released when becomes begative

Replication: $!^a_{\overline{c},\overline{n}} P$ is positive but releases the focus.

POR 1: Reduction

Reduced semantics (roughly)

- Priority order $<$ over independent blocks *e.g.* $\mathsf{IO}_{c_1} < \mathsf{IO}_{c_2}$
- Explore IO_c after $\mathsf{IO}_{c_1}.\dots.\mathsf{IO}_{c_n}$ only if any violation of $<$ is for "good reason" (*i.e.* data dependencies) "I need this w"

Theorem: $\approx_r = \approx$ [Baelde, Delaune, H.: POST'14, CONCUR'15]

Let A and B be two action-deterministic configurations.

$A \approx B$ if, and only if, $A \approx_r B$.

Encoding

First attempt

- T=transitions of configurations
- S=pairs of sets of configurations (noted $\langle\!|\mathbf{A} \approx \mathbf{B}|\!\rangle$) $s_0 = \langle\!|\{A\} \approx \{B\}|\!\rangle$
- Transition function δ:

$$\langle\!|\mathbf{A} \approx \mathbf{B}|\!\rangle \xrightarrow{\alpha} \langle\!|\mathbf{A}_\alpha \approx \mathbf{B}_\alpha|\!\rangle \text{ with } \mathbf{X}_\alpha = \{C' \ : \ C \in \mathbf{X},\ C \xrightarrow{\alpha} C'\}$$

- $\langle\!|\mathbf{A} \approx \mathbf{B}|\!\rangle \in S_{\mathsf{bad}}$ if $\mathbf{A} \not\sim \mathbf{B}$

Unsound! ...because witnesses can be lost

$\exists\langle\!|\mathbf{A} \approx \mathbf{B}|\!\rangle \xrightarrow{\alpha} \langle\!|\mathbf{A}' \approx \mathbf{B}'|\!\rangle$ such that $\mathbf{A} \not\sim \mathbf{B}$ but $\mathbf{A}' \sim \mathbf{B}'$!

$$\forall A, B.\ \ A \approx B \not\Leftarrow \mathsf{Reach}(\mathsf{LTS}(A, B))$$

Example:

$$\langle\!|\{\mathsf{out}(0) + \mathsf{out}(k).\alpha\} \approx \{\mathsf{out}(1) + \mathsf{out}(k).\alpha\}|\!\rangle \xrightarrow{\mathsf{out}(w)} \langle\!|\mathbf{A} \approx \mathbf{B}|\!\rangle \xrightarrow{\alpha} \langle\!|\mathbf{A}' \approx \mathbf{B}'|\!\rangle$$