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

- **TLS ≤ 1.2**
- **https://**
- **Formal verification**
- **TLS 1.3**
- **https://**

**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. ,  \( \iff \) enc(\cdot, \cdot), dec(\cdot, \cdot) & dec(enc(m, k), k) = m
Symbolic Model (Dolev-Yao)

Cryptographic primitives assumed perfect
- primitives modelled as function symbols & equational theory
- e.g. $\text{enc}(:, :)$, $\text{dec}(:, :)$ & $\text{dec}(\text{enc}(m, k), k) = m$

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

Automated verification
- secrecy, authentication, …; reachability (not this talk)
- privacy, …; equivalence between configurations (this talk)
Symbolic Model (Dolev-Yao)

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

Security protocols
- in a **process algebra**
- each party $\mapsto$ process

Attacker $\mathbb{A}$ = **network** (worst case scenario:)
- eavesdrop: he learns all protocol outputs
- injection: he chooses all protocol inputs

$P_A = \text{in}(X).$ new $Y.$\[\text{out(}\text{enc((}}X, Y\text{)}, k\text{))}\]

$\text{out}(c, u) \rightarrow \mathbb{A}(u)$
\[\text{in}(c, v) \leftarrow \mathbb{A}(v)\]
Symbolic Model (Dolev-Yao)

Cryptographic primitives assumed perfect

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

Security protocols

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

Attacker $\text{Network} = \text{network (worst case scenario:)}$

- eavesdrop: he learns all protocol outputs
- injection: he chooses all protocol inputs

Automated verification

- secrecy, authentication, ... $\leadsto$ reachability
- privacy, ... $\leadsto$ equivalence between configurations
Symbolic Verification of Equivalence: State-of-the-Art

Equivalence: \( \forall A, P|A \approx Q|A \) Undecidable 😞

Semi-decision for \( \infty \) 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 \( \infty \) sessions**
- "Real" attack
- "False" attack
- over-approximations of \( \infty \) & semantics
- strong form of \( \approx \) \( (i.e. \) diff-equivalence)
- tools: Tamarin, ProVerif, Maude-NPA

**Lack of precision**
e.g. too imprecise for untraceability

**Decision for \( < \infty \) 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** $\sim$ state space explosion
A concurrency issue

Problem: concurrency $\leadsto$ 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

Lucca Hirschi
A concurrency issue

Problem: *concurrency* \(\sim\) state space explosion

Partial Order Reductions (POR) (Model-Checking)

Leverage *independencies of actions* to avoid *redundant* interleavings
A concurrency issue

Problem: concurrency $\sim$ state space explosion

Partial Order Reductions (POR) (Model-Checking)

Leverage independencies of actions to avoid redundant interleavings
A concurrency issue

Problem: **concurrency** \(\sim\) state space explosion

Partial Order Reductions (POR)

Leverage *independencies of actions* to avoid *redundant* interleavings
A concurrency issue

Problem: **concurrency** \(\sim\) 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 \(\sim\) 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.

\[ A \approx B \quad \text{Encoding} \quad \text{Reach}(\text{LTS}) \quad \text{Classical POR} \]

\[ \text{Reach}(\text{Red}(\text{LTS})) \]

(Contributions in red)
Our Contributions

This Work

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

A ≈ B  Encoding  Reach(LTS)  Symbolic Approximation  Reach(LTS\textsuperscript{s})  Symbolic POR  Reach(\textit{Red}\textsuperscript{s}(LTS\textsuperscript{s}))

Classical POR (persistent & sleep sets)  Thm:  Reach(Red(LTS))  \(\cap\)  Thm:  Reach(Red\textsuperscript{s}(LTS\textsuperscript{s}))  (persistent + sleep sets + optims.)

(Contributions in red)
Our Contributions

This Work

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

Diagram:

- A \approx B
- Encoding
- Reach(LTS)
- Symbolic Verification of A \approx B (e.g., DeepSec, Apte)
- Reach(LTS^s)
- Symbolic POR (persistent+sleep sets+optims.)
- Reach(\text{Red}^s(LTS^s))
- Integration
- Porridge
- Optimized DeepSec and Apte + Benchmarks

(Contributions in red)
Outline

From $A \approx B$ to $\text{Reach}(\text{LTS})$

1. **Encoding**
   - $A \approx B \xrightarrow{\text{Thm: } \iff} \text{Reach}(\text{LTS})$

2. **Symbolic Verification**
   - $A \approx B$
     - e.g., DeepSec, Apte

3. **Symbolic Approximation**
   - $\bigcap \text{Symbolic POR}$

4. **Optimized DeepSec and Apte**

5. **Porridge**

6. **Classical POR**
   - $(\text{persistent & sleep sets})$
   - $\text{Reach}(\text{Red}(\text{LTS}))$
   - $\text{Thm: } \iff$

7. **Symbolic POR**
   - $(\text{persistent & sleep sets + optims.})$
   - $\text{Reach}(\text{Red}^s(\text{LTS}^s))$
   - $\text{Thm: } \bigcap$

8. **Benchmarks**
   - A $\approx B$
Applied-$\pi$ Calculus

- Process: $P, Q ::= \begin{align*}
in(c, x).P & \quad \text{input} \\
out(c, m).P & \quad \text{output} \\
P & \quad \text{null} \\
\ | & \quad \text{parallel} \\
\ | & \quad \text{choice} \\
\ | & \quad \text{conditional} \\
\ | & \quad \text{creation of name}
\end{align*}$

\[ A \approx B \quad \text{Encoding} \quad \Rightarrow \quad \text{Reach(LTS)} \]
Applied-$\pi$ Calculus

- **Process**: $P, Q := \begin{align*}
in(c, x).P & \quad \text{input} \\
out(c, m).P & \quad \text{output} \\
P & \quad \text{parallel} \\
P & \quad \text{choice} \\
if \text{Test} \text{ then } P & \quad \text{conditional} \\
new X.P & \quad \text{creation of name} \\
0 & \quad \text{null}
\end{align*}$

- **Frame** ($\phi$): the set of messages revealed to $\mathfrak{M}$'s knowledge

  $\phi = \{ w_{c,1} \rightarrow \text{enc}(m, k), w_{d,1} \rightarrow k \}$

  \[
  \begin{align*}
  \phi &= \{ \text{variable} \rightarrow \text{out. message} \} \\
  &= \{ w_{c,1} \rightarrow \text{enc}(m, k), w_{d,1} \rightarrow k \}
  \end{align*}
  \]

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

- **Process:** $P, Q := \begin{align*} &\text{in}(c, x).P & \text{input} \\ &\text{out}(c, m).P & \text{output} \\ &P | Q & \text{parallel} \\ &P + Q & \text{choice} \\ &\text{if } \text{Test} \text{ then } P \text{ else } Q & \text{conditional} \\ &\text{new } X.P & \text{creation of name} \\ &0 & \text{null} \end{align*}$

- **Frame** ($\phi$): the set of messages revealed to (user’s knowledge)
  \[ \phi = \{ w_{c,1} \mapsto \text{enc}(m, k), w_{d,1} \mapsto k \} \]
  
  - variable
  - out. message

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

- **Semantics:**
  \[ \begin{align*} 
  (\{ P, \text{out}(c, u).Q \}; \phi) & \xrightarrow{\text{out}(c, w_{c,i})} (\{ P, Q \}; \phi \cup \{ w_{c,i} \mapsto u \}) \text{ when } i = |\phi|_c \\
  (\{ P, \text{in}(c, x).Q \}; \phi) & \xrightarrow{\text{in}(c, M)} (\{ P, Q\{ x \mapsto M\phi \} \}; \phi) \text{ when } M \text{ is a term over } \phi 
  \end{align*} \]
Trace Equivalence

\[ A \approx B \text{ when: } \forall A \stackrel{t}{\rightarrow} A', \exists B \stackrel{t}{\rightarrow} B' \text{ such that } \phi(A') \sim \phi(B') \] (and conversely)

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!).

  \[ s \xrightarrow{\alpha} s' \text{ when } s' = \delta(s, \alpha) \]

- A state \( s \in S \) is final when \( s \notin \text{dom}(\delta) \)

- Given: initial state \( s_0 \) and \( S_{\text{bad}} \subseteq S \) (bad states),
  Reach(\cdot): no final, bad state in \( S_{\text{bad}} \) is reachable from \( s_0 \)

Desired property:

\[ \forall A, B. \ A \approx B \iff \text{Reach(LTS}(A, B)) \]
First attempt

- \( T = \) transitions of configurations
- \( S = \) pairs of sets of configurations (noted \( \langle |A \approx B| \rangle \))
  \[ s_0 = \langle \{A\} \approx \{B\} \rangle \]
- Transition function \( \delta \):
  \[ \langle |A \approx B| \rangle \xrightarrow{\alpha} \langle |A_\alpha \approx B_\alpha| \rangle \text{ with } X_\alpha = \{C' : C \in X, C \xrightarrow{\alpha} C'\} \]
- \( \langle |A \approx B| \rangle \in S_{\text{bad}} \) if \( \phi(A) \not\equiv \phi(B) \)
First attempt

- **T** = transitions of configurations
- **S** = pairs of sets of configurations (noted $\langle |A \approx B| \rangle$)
- **s_0** = $\langle |\{A\} \approx \{B\}| \rangle$
- Transition function $\delta$:
  
  $\langle |A \approx B| \rangle \xrightarrow{\alpha} \langle |A_\alpha \approx B_\alpha| \rangle$ with $X_\alpha = \{C' : C \in X, C \xrightarrow{\alpha} C'\}$

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

Unsound! ...because witnesses can be lost

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

$\forall A, B. A \approx B \not\iff \text{Reach}(\text{LTS}(A, B))$

Example:

$\langle |\{\text{out}(0) + \text{out}(k).\alpha\} \approx \{\text{out}(1) + \text{out}(k).\alpha\}| \rangle \xrightarrow{\text{out}(w)} \langle |A \approx B| \rangle \xrightarrow{\alpha} \langle |A' \approx B'| \rangle$
Definition: LTS($A,B$)

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

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

Theorem

\[ \forall A, B. \ A \approx B \iff \text{Reach}(\text{LTS}(A, B)) \]
Outline

POR over Reach(LTS)

A ≈ B

Encoding

Reach(LTS)

Classical POR
(persistent & sleep sets)

Reach(Red(LTS))

Symbolic POR
(persistent + sleep sets + optims.)

Reach(LTS\textsuperscript{s})

Symbolic Approximation

Reach(Red\textsuperscript{s}(LTS\textsuperscript{s}))

Thm: ⇐⇒

Porridge

Optimized DeepSec and Apte

Benchmarks

e.g., DeepSec, Apte

Lucca Hirschi

POR for Security Protocol Equivalences
Independence relation between transitions \((\alpha, \beta \in T, s \in S)\):

\[ \alpha \leftrightarrow_s \beta : s \]

Intuitively: \(\alpha \leftrightarrow_s \beta\) when \(\alpha\) and \(\beta\) commute from \(s\)
Independence relation between transitions \((\alpha, \beta \in T, s \in S)\):

\[\alpha \leftrightarrow_{s} \beta\]

Intuitively: \(\alpha \leftrightarrow_{s} \beta\) when \(\alpha\) and \(\beta\) commute from \(s\)
Independence relation between transitions \((\alpha, \beta \in T, s \in S)\):

\[
\alpha \leftrightarrow_{s} \beta :
\]

Intuitively: \(\alpha \leftrightarrow_{s} \beta\) when \(\alpha\) and \(\beta\) commute from \(s\)
Independence relation between transitions $(\alpha, \beta \in T, s \in S)$:

Intuitively: $\alpha \leftrightarrow_s \beta$ when $\alpha$ and $\beta$ commute from $s$.
Independence relation between transitions \((\alpha, \beta \in T, s \in S)\):

\[ \alpha \leftrightarrow_s \beta \]

Intuitively: \(\alpha \leftrightarrow_s \beta\) when \(\alpha\) and \(\beta\) commute from \(s\)
Goal:

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

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

Property achieved by persistent sets:

Theorem: \( \text{Reach}(\text{LTS}(A, B)) \iff \text{Reach}(\text{Red}(\text{LTS}(A, B))) \)
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)

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)**

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)) ⇐⇒ 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)

A \approx B \quad \text{Encoding} \quad \text{Reach}(LTS) \quad \text{Classical POR} \quad \text{Reach}(\text{Red}(LTS))

\text{Thm:} \iff 

\bigcap 

\text{Reach}(LTS^s) \quad \text{Symbolic POR} \quad \text{Reach}(\text{Red}^s(LTS^s))

\text{Thm:} \cap 

\text{Symbolic Verification of } A \approx B 

e.g., DeepSec, Apte

\text{Optimized DeepSec and Apte} \quad + \quad \text{Benchmarks}

\text{Porridge}
Symbolic Abstraction of LTS

Symbolic LTS and semantics use symbolic states

\[ \langle A^s \approx B^s \rangle_C \]

where \( C \) is a set of (dis)equality constraints and \( C^s \in A^s \cup B^s \) contains input variables.

- Input messages are replaced by variables
- Transitions branch on conditionals + extend \( C \)
Symbolic LTS and semantics use symbolic states

\[\langle A^s \approx B^s \rangle_C\]

where \(C\) is a set of (dis)equality constraints and \(C^s \in A^s \cup B^s\) contains input variables.

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

Results:

- **Completeness**: concrete transitions mimicked by symbolic ones
- **No soundness**: \(S\theta\) 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) \sim extremely poor reduction 😞

\[
\begin{align*}
\text{Reach}(\text{LTS}) & \quad \text{Classical POR} \\
\text{Symbolic Approximation} & \quad \text{(persistent & sleep sets)} \\
\text{Reach}(\text{LTS}^s) & \quad \text{Thm}: \iff \\
\text{Classical POR} & \quad \text{Reach}(\text{Red}(\text{LTS})) \\
\text{Thm}: \iff & \quad \text{Reach}(\text{Red}(\text{LTS}^s))
\end{align*}
\]
Symbolic POR

POR($\text{LTS}^s$)?

- POR on symbolic LTS $\text{Red}(\text{LTS}^s)$ $\leadsto$ extremely poor reduction 😞
- $\text{LTS}^s$ used to over-approximate $\text{Red}(\text{LTS})$: $\text{Red}^s(\text{LTS}^s) \leadsto$ good reduction 😊

\[ \text{Classical POR (persistent & sleep sets)} \]

$\text{Reach}(\text{LTS}) \overset{\text{Thm: } \Leftrightarrow}{\cap} \text{Symbolic POR}$

$\text{Reach}(\text{LTS}^s) \overset{\text{Prop: } \Leftrightarrow}{\cap} \text{Symbolic POR}$

$\text{Reach}(\text{Red}(\text{LTS})) \overset{\text{Thm: } \cap}{\cap} \text{Classical POR}$

$\text{Reach}(\text{Red}^s(\text{LTS}^s))$
Symbolic POR

POR($LTS^S$)?

- POR on symbolic LTS $\text{Red}(LTS^S)$ $\leadsto$ extremely poor reduction 😞
- $LTS^S$ used to over-approximate $\text{Red}(LTS)$: $\text{Red}^S(LTS^S) \leadsto$ good reduction 😊

Symbolic POR ($\text{Red}^S(\cdot)$) is based on:

- $\leftrightarrow^S$ (for symbolic state $S$): a sound abstraction of $\leftrightarrow^S$ for any $s = S\theta$

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

∃ concrete reduced execution in LTS($A, B$) reaching a bad state

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

\[ A \approx B \quad \xrightarrow{\text{Encoding}} \quad \text{Reach}(\text{LTS}) \quad \xrightarrow{\text{Classical POR (persistent & sleep sets)}} \quad \text{Reach}(\text{Red}(\text{LTS})) \]

\[ \text{Reach}(\text{Red}^s(\text{LTS}^s)) \]

**Theorem:**

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

\( A \not\approx B \)

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

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

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

- The set of symbolic reduced executions in \( \text{LTS}^s(A, B) \) 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

A \approx B

Symbolic Verification of A \approx B

e.g., DeepSec, Apte

Symbolic Approximation

Reach(LTS)

Reach(Red(LTS))

Classical POR
(persistent & sleep sets)

Thm: \iff

Reach(LTS^s)

Symbolic POR
(persistent + sleep sets + optims.)

Reach(Red^s(LTS^s))

Porridge

Optimized DeepSec and Apte + Benchmarks

Lucca Hirschi

POR for Security Protocol Equivalences
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):

![Graph showing time and explorations ratios with size as the x-axis and ratio as the y-axis. The graph has two lines: one for time (ratio) and another for explorations (ratio). The time (ratio) line starts at a lower point and has a steeper incline compared to the explorations (ratio) line, indicating faster computation times. The explorations (ratio) line has a gentler incline.]
Conclusion
Summary

- **First POR techniques** for $\approx$ and a large class of protocols
- State-of-the-art verifiers already **benefit from our techniques**
- Unlock traditional POR techniques (Model-Checking) for $\approx$ (Security)
Future work

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

Implementation & Integration:
- Porridge: multicore, better trade-off precision/pre-computation cost
- interactive integration
Backup Slides
What about trace equivalence ($\approx_c$)?

E.g., $(\text{in}(c_1, x) \mid \text{out}(c_2, m)) \not\approx (\text{out}(c_2, m).\text{in}(c_1, x))$

- $\sim$ same swaps are possible ($\equiv$ 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 $S \Rightarrow B S'$,
- or $A$ is executable in $S$ but $A/B$ are not of the form $\text{in}(c, X_{c,i}, W)/\text{out}(d, w_{d,j})$ with $w_{d,j} \in W$. 

Lucca Hirschi

POR for Security Protocol Equivalences
Optimization Handling Conditional Branching

\( A \approx B \)\[\text{Encoding}\] \rightarrow \text{Reach}(LTS) \quad \text{Classical POR (persistent & sleep sets)} \rightarrow \text{Reach}(\text{Red}(LTS))

\text{Symbolic Approximation} \quad \bigcap \quad \text{Symbolic POR (persistent+sleep sets+optims.)} \rightarrow \text{Reach}(\text{Red}^s(LTS^s))

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

Final optimization

- branching due to conditional + non-det. \(\not\sim\) state space explosion 😞
- we address this explosion by soundly “collapsing” most of conditionals: \(\text{Red}^s(LTS^s) = \text{Red}^s(\text{SimplCond}(LTS^s))\) but \(\text{SimplCond}(LTS^s) \ll\ll \text{LTS}^s\) 😊
Compressed semantics $\rightarrow_c$

- **Polarities:**
  - **Negative:** out().$P$, ($P_1 | P_2$), 0
  - **Positive:** in().$P$

- **Negative:** explored greedily, in a given order
  - e.g. $c_1 < c_2$

- **Positive:** explored only when $\not\exists$ Negative,
  - chooses one and put it under focus
  - focus is released when becomes negative

**Replication:** $!a_{c,n}P$ is positive but releases the focus.
POR 1: Reduction

Reduced semantics (roughly)

- Priority order $<_{\text{over independent blocks}}$ e.g. $\text{IO}_{c1} < \text{IO}_{c2}$
- Explore $\text{IO}_c$ after $\text{IO}_{c1} \ldots \text{IO}_{cn}$ 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.

$$A \approx B \text{ if, and, only if, } A \approx_r B.$$
Encoding

First attempt

- $T =$ transitions of configurations
- $S =$ pairs of sets of configurations (noted $\langle |A \approx B| \rangle$)
  \[ s_0 = \langle \{|A\| \approx \{B\}| \rangle \]
- Transition function $\delta$:
  \[ \langle |A \approx B| \rangle \xrightarrow{\alpha} \langle |A_\alpha \approx B_\alpha| \rangle \text{ with } X_\alpha = \{C' : C \in X, C \xrightarrow{\alpha} C' \} \]
- $\langle |A \approx B| \rangle \in S_{\text{bad}}$ if $A \not\sim B$

Unsound! ...because witnesses can be lost

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

\[ \forall A, B. \ A \approx B \nRightarrow \text{ Reach}(\text{LTS}(A, B)) \]

Example:

\[ \langle \{|\text{out}(0) + \text{out}(k).\alpha\| \approx \{|\text{out}(1) + \text{out}(k).\alpha\}|\rangle \xrightarrow{\text{out}(w)} \langle |A \approx B| \rangle \xrightarrow{\alpha} \langle |A' \approx B'|\rangle \]